A data processing firmware for an upgrade of the Outer Tracker detector at the LHCb experiment

Dissertation zur Erlangung des Grades
Dr. rer. nat.

vorgelegt von

Dipl.-Phys. Stefan Swientek

Fakultät Physik
Technische Universität Dortmund

Dortmund

Januar 2015

Erstgutachter: Prof. Dr. Bernhard Spaan

Zweitgutachter: Priv.-Doz. Dr. Reiner Klingenberg

Abstract

This thesis describes the data processing software for an Outer Tracker upgrade at the LHCb experiment. The 2018/19 intended upgrade for the LHCb detector will introduce new readout electronics on the front end of the detector as well as on the back end. The read out electronics on the back end of the detector will use a common board equipped with a Field Programmable Gate Array (FPGA). To ensure a correct data processing for each subdetector, the firmware used on the FPGA will contain subdetector specific data processing parts. The data processing part for the Outer Tracker includes receiving the data, sorting incoming data streams, merging hit pattern and drift times of three consecutive bunch crossings, clustering and formatting the data for an accurate output. In addition, a mezzanine card, called the SantaLuz board, is presented which can be used to extend existing FPGA boards with up to eight optical transceivers.

Zusammenfassung

Diese Arbeit beschreibt die Datenverarbeitungssoftware für ein Outer Tracker Upgrade am LHCb Experiment. Das für 2018/19 vorgesehene Upgrade des LHCb Detektors wird sowohl am Detektor als auch abseits dessen neue Ausleseelektronik einführen. Die Elektronik abseits des Detektors wird aus einer mit einem FPGA bestückten Platine bestehen. Um eine korrekte Datenverarbeitung zu gewährleisten, wird es für jeden Subdetektor eine eigene Datenverarbeitungssoftware auf der Firmware des FPGA geben. Die Datenverarbeitung für den Outer Tracker empfängt die Daten, sortiert die eingehenden Datenleitungen, fasst die Trefferliste und die Driftzeiten aus drei aufeinanderfolgenden Bunch Crossings zusammen, sucht nach Gruppierungen von Treffern und formatiert die Daten zur korrekten Ausgabe. Zusätzlich wird das SantaLuz board vorgestellt, eine Erweiterungskarte für bereits vorhandene FPGA Karten, die bis zu acht optische Transceiver bereit stellen kann.

Contents

List of Figures

1 Introduction

Physicists never stop asking questions. Why does the earth turn around the sun? Why does the apple fall to the ground? What is an atom built of? In modern particle physics these questions are far more detailed. Where does the mass of elementary particles come from? What happened during and directly after the Big Bang?

To answer these and other questions particle physicists are trying to reproduce similar conditions as in the beginning of our universe. Therefore, protons and other particles are collided with high energy in large particle colliders. The currently largest particle collider is the Large Hadron Collider (LHC) at CERN. Particles were colliding from 2010 to 2012 at a centre-of-mass energy of up to 8 TeV and the LHC will continue with colliding particles in 2015. To reconstruct the events of colliding particles large experiments are built with different purposes. One of the four big experiments at the LHC is the LHCb experiment, which is specialised to observe particles with a small production angle to the beam axis. This experiment was successfully observing B-meson decays for the whole first run period of the LHC, taking and processing a large amount of data.

An upgrade of the LHCb detector is intended for the years 2018/19 to increase the rate with which LHCb is capable of taking data. The hardware trigger shall be removed and the whole detector upgraded. With this change of the trigger method, LHCb will be able to specialise their triggering and therefore be able to measure decays more precisely, or adapt the triggers to new physics. This would also increase the data sent from the detector to the back end electronics, where it has to be processed before sending it to the Data Acquisition (DAQ). To process that amount of data, special electronics called the AMC40 is developed and working with an FPGA. A firmware for this FPGA has to be programmed to perform all necessary tasks needed for a fully functioning data taking.

The goal of this thesis is to develop a firmware for the data processing part of the AMC40 firmware for an upgrade of the Outer Tracker, a gaseous straw tube detector, of the LHCb experiment. This firmware has to perform without failure during high data flow and various running conditions.

First a short introductory overview of the LHC and the three large experiments ATLAS, CMS and ALICE is given in Chapter 2. Afterwards a motivation for LHCb, the fourth large experiment at the LHC, is given, including an overview of the detector in Chapter 3. A requirement for the development of a data processing firmware for a specific detector type is to understand its working procedure, which is explained for the Outer Tracker in Chapter 4, including the plans for the upgrade. Since the data processing firmware will be used on specific electronics, the design of the upgraded back end electronics is summarised in Chapter 5. Chapter 6 covers the basics of FPGAs, e.g. their structure and working procedures. The developed data processing firmware is explained in detail in Chapter 7, containing every step in data processing and explaining each function and algorithm. A comparison in logic usage on the FPGA is also given for different possibilities of workload on the back end electronics. In Chapter 8 two test setups are introduced, which are able to test a finished firmware on hardware. Finally, conclusions are given in Chapter 9.

2 The Large Hadron Collider

The LHC at CERN[I] near Geneva is the largest proton-proton-collider in the world. It is a circular proton-proton-collider with a circumference of 26.7 km, also capable of accelerating and colliding heavy ions. Built in a depth of about 100 m, the LHC is designed to collide protons with a centre of mass energy of $\sqrt{s}$=14 TeV [1], whereat during LHC data taking from 2010 to 2012 (called LHC Run I) the energy was set to a maximum of 8 TeV [2]. The protons are collided at four collision points, where the four large experiments of the LHC are located, called ATLAS, CMS, ALICE and LHCb (see Figure 2.1). A short overview of each detector will be given in the following, while the LHCb detector will be explained in detail in Chapter 3.

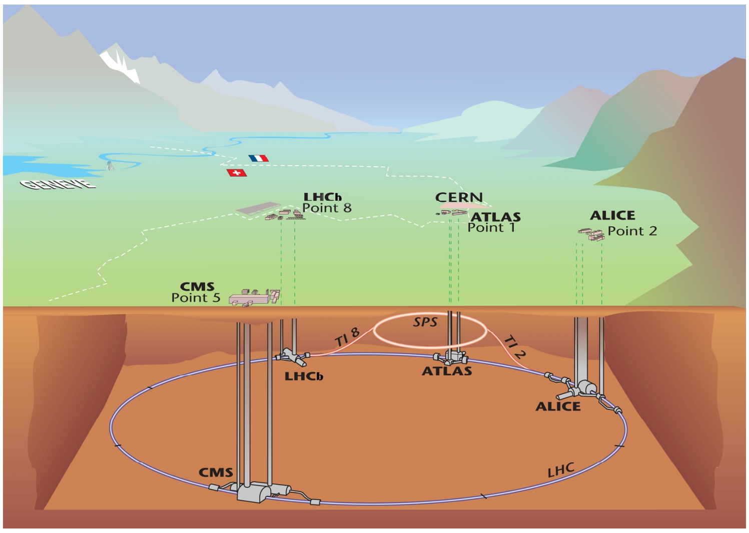
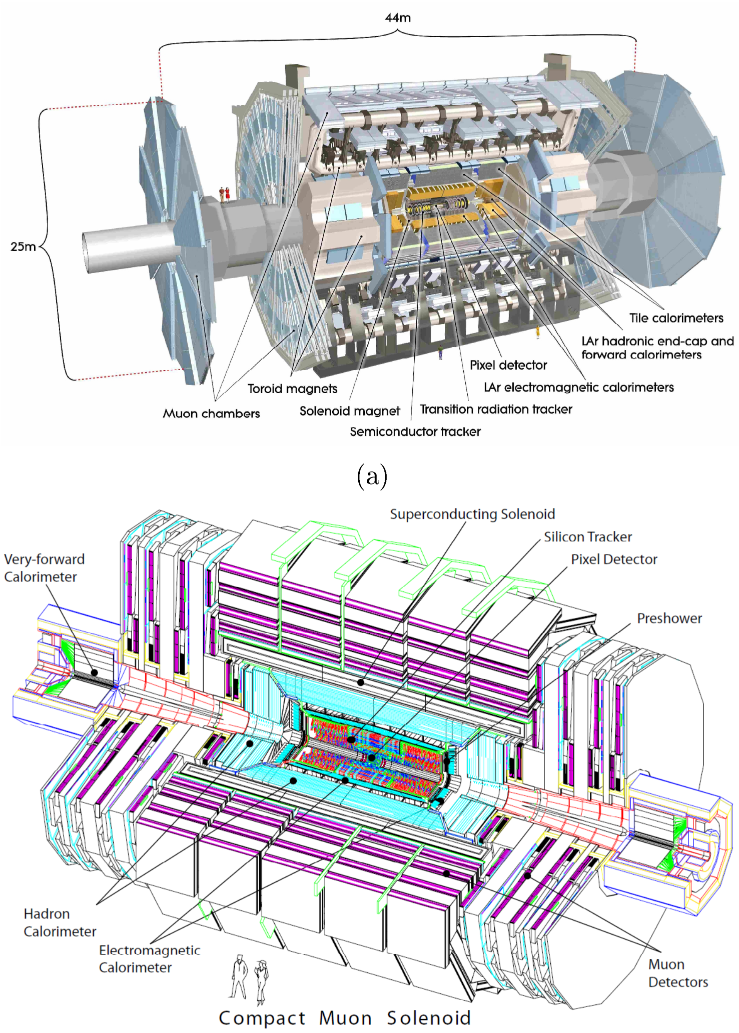


Figure 2.1: The LHC at CERN, with the four large experiments ATLAS, CMS, ALICE and LHCb [3].

[I] European Organization for Nuclear Research.

2.1 ATLAS and CMS

ATLAS (A Toroidal LHC ApparatuS) and CMS (Compact Muon Solenoid) are the two largest experiments at CERN. Their main goal is to search for new physics. For example, ATLAS and CMS try to discover particles which can be assigned to supersymmetric theories [4]. Both detectors cover the whole area around the interaction point. Figures 2.2a and 2.2b show both detectors in a schematic view.



(a)

(b)

Figure 2.2: Figure (a) shows a schematic view of the ATLAS detector [5], Figure (b) the CMS detector [6].

Such detectors are built with a tracking detector near the collision point, followed by a detector for energy measurements, like calorimeters. Finally, a muon spectrometer is placed at the outermost part of the whole detector.

2.2 ALICE

ALICE (A Large Ion Collider Experiment) is another detector at LHC covering the whole area around the interaction point, but specialised in analysing heavy ion collisions. Figure 2.3 shows a schematic of the ALICE detector, which is equipped with various tracking detectors. Such large amount of tracking detectors is needed to observe a high multiplicity of tracks, which occur at heavy ion collisions. The purpose of ALICE is to study the quark-gluon-plasma in this heavy ion collisions [7]. The quark-gluon-plasma is a state in which all matter should have existed a few milliseconds after the big bang. Thus, the results of this experiment could contribute to the understanding of Quantum Chromodynamics (QCD) [8].

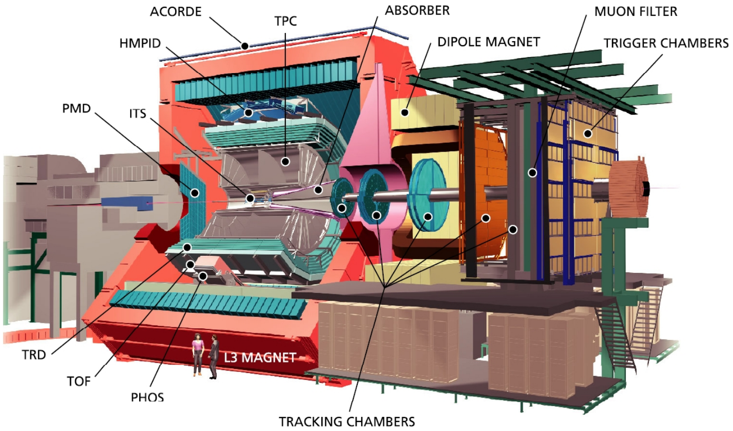


Figure 2.3: The ALICE detector [7]. A large amount of tracking detectors is used to capture the enormous number of tracks (labelled: ITS, TPC, TRD).

3 The LHCb experiment

The LHCb (Large Hadron Collider beauty) experiment is dedicated to search for indirect evidences of new physics in Charge-Parity (CP) violating processes and rare decays of bottom[I] and charm hadrons. Thus, its complete setup in hardware as well as software is specialized for b and c physics.

3.1 Physical motivation for the LHCb experiment

To introduce into the physical motivation of the LHCb experiment, a short introduction to the Standard Model of particle physics (SM) and LHCb specific searches is given before explaining the design of the LHCb detector.

The Standard Model of particle physics

The Standard Model of particle physics is a theory that describes the interactions of elementary particles. It covers the fundamental forces, which are represented through their spin 1 gauge bosons in Figure 3.1. These are the photon (γ), representing the gauge boson of the electromagnetic force, the W^+-, the W^-- and the Z^0- boson, mediating the weak interaction, and the gluons for the strong interaction. The photon is a massless electrically neutral boson and its own antiparticle [9]. In contrast to the other gauge bosons the gauge bosons of the weak interaction are massive[II] and the $\mathrm{W}^\pm$ are the only gauge bosons carrying an electric charge. The $\mathrm{W}^\pm$, Z^0 and γ are unified within the electroweak interaction [9]. The gluons are eight massless gauge bosons mediating the strong interaction and carry a charge called "colour". Three charges of colour (red, green, blue) combine up to eight gluons with different colour-anticolour combination [8].

The SM also identifies the quarks and leptons (both fermions with a spin of $1/2\,\hbar$ as fundamental fermions, shown in Figure 3.1. Additionally, there is an antiparticle for each fermion: same particles with all charge-like quantum numbers reversed. The quarks are the building blocks for all known particles. They are categorised into three families, each consisting of an up-type quark (up or u, charm or c, top or t)

[I] The bottom quark is also often referred to as the beauty quark.

[II] $\mathrm{W}^\pm$: 80.4 GeV/c^2, Z: 91.2 GeV/c^2.

and a down-type quark (down or d, strange or s, bottom or b). Hereby, each up-type quark carries an electric charge of $^2\!/_3\,\mathrm{e}$ and each down-type quark an electric charge of $-^1\!/_3\,\mathrm{e}$. The quarks and antiquarks can be combined to hadrons, represented by mesons as a quark-antiquark combination and baryons as a combination of three quarks or three antiquarks. Well known baryons are the proton, consisting of two up and one down quark (uud) or the neutron built of one up and two down quarks (udd).

The leptons are also categorized into three families, each consisting of an electrically charged particle (electron or e, muon or μ, tau or τ) and an electrically neutral neutrino (ν_e, ν_μ, ν_τ). The properties of the charged leptons are well measured, whereas especially the masses of the neutrinos are not known [10]. Since neutrinos are weakly interacting particles, which are nearly undetectable in a standard particle detector, their characteristics are only determined with low statistics in large experiments or through indirect measurements[III].

Another part of the SM is the Higgs boson (H). This spinless boson with a mass of about $126\,\mathrm{GeV}/c^2$ is the boson of the Higgs field. This field explains the mass of the other particles in the SM by using the so-called Higgs-mechanism postulated in 1964 [11, 12]. In 2012 a particle was found at CERN [13, 14], which could be the Higgs boson and therefore strengthen this theory.

However, the SM is incomplete. Gravitation is not included and besides other open questions neutrinos are assumed to be massless in the SM, although measurements of neutrino oscillations have already shown the opposite.

[III] An indirect measurement of neutrinos can be done e.g., by constraining the missing energy in a fully reconstructed event.

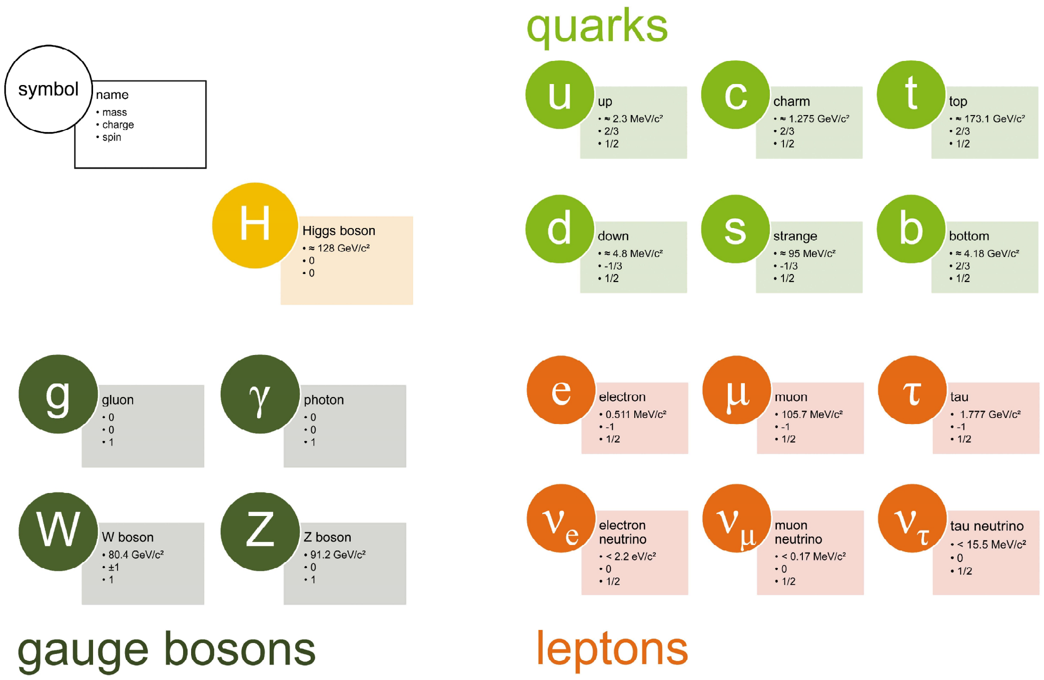


Figure 3.1: An illustration of the elementary particles included in the Standard Model of particle physics, with its quarks (green), leptons (red), gauge bosons (dark green) and the Higgs boson (yellow).

Basics of LHCb specific searches

As already mentioned, LHCb searches for indirect evidences of new physics in CP violating processes. These processes could help explaining the excess in the production rate of matter over antimatter. CP violation is described through the complex phase of the Cabibbo-Kobayashi-Maskawa matrix [15, 16], which indicates the strength of flavour-changing weak decays:

$$V_{CKM} \equiv V_L^u V_L^{d\dagger} = \begin{bmatrix} V_{ud} & V_{us} & V_{ub} \\ V_{cd} & V_{cs} & V_{cb} \\ V_{td} & V_{ts} & V_{tb} \end{bmatrix} .$$

The unitarity of this matrix is required in the theory of electroweak interactions, but it still has to be proven in experiment. Six unitarity conditions of the CKM matrix can be illustrated in six different unitarity triangles. The unitarity triangle corresponding to the unitarity relation $V_{qd}V_{qb}^{\star}$ is shown in Figure 3.2. The decay $B^0 \to J/\psi K_S^0$ is one particle decay used to study this relation at LHCb [17].

With the information gathered from these studies, indirect measurements of the angles and distributions of the unitarity triangles can be obtained (e.g. $\sin 2\beta$ in Figure 3.2). The combination of all results can lead to a perfectly closed triangle and thus prove the CKM-matrix, or if the triangle does not fit (loose ends or missing endpoints of sides) disagree with the CKM-matrix and lead to new pyhsics.

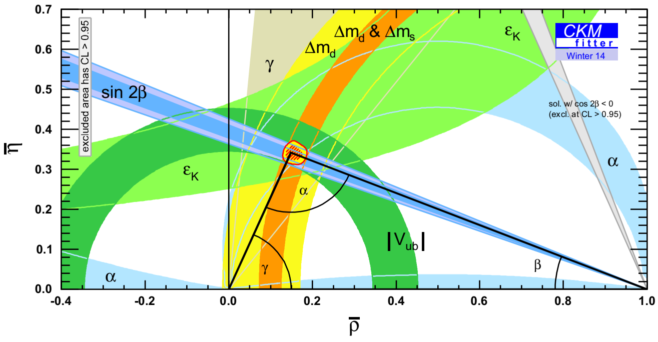


Figure 3.2: The unitarity triangle corresponding to the unitarity relation $V_{qd}V^{\star}_{qb}$ [18]. The colours indicate results of different measurements.

3.2 The LHCb detector

The production angle of $\mathrm{b\bar{b}}$ pairs in pp-collisions is a distinct distribution as shown in Figure 3.3 for LHCb. To ensure a precise measurement of decaying e.g. B mesons

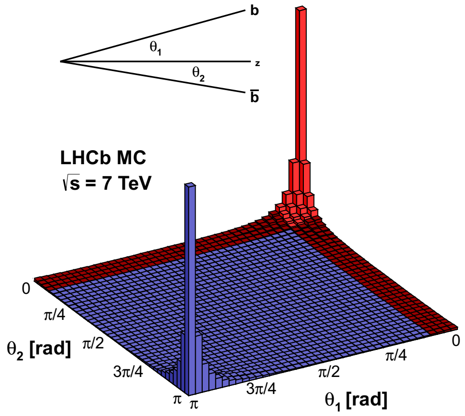


Figure 3.3: The $\mathrm{b\bar{b}}$ distribution in pp-collisions [19]. $b\bar{b}$ pairs are produced in a small angle corresponding to the beam axis. The figure shows the distribution in LHCb at $\sqrt{s}$=7 TeV with clear peaks at low angles.

the LHCb detector is built as a one-armed forward spectrometer specialised to detect particles decaying in low angles to the beam axis. The subdetectors, as shown in Figure 3.4, will be explained in the following, starting with the tracking detectors in front of the magnet, the Vertex Locator (VELO) and the Tracker Turicensis (TT). Afterwards, both Ring Imaging Cherenkov (RICH) detectors are explained, before introducing the tracking stations T1-T3 behind the magnet. Then the calorimeters are described, finishing with the muon system.

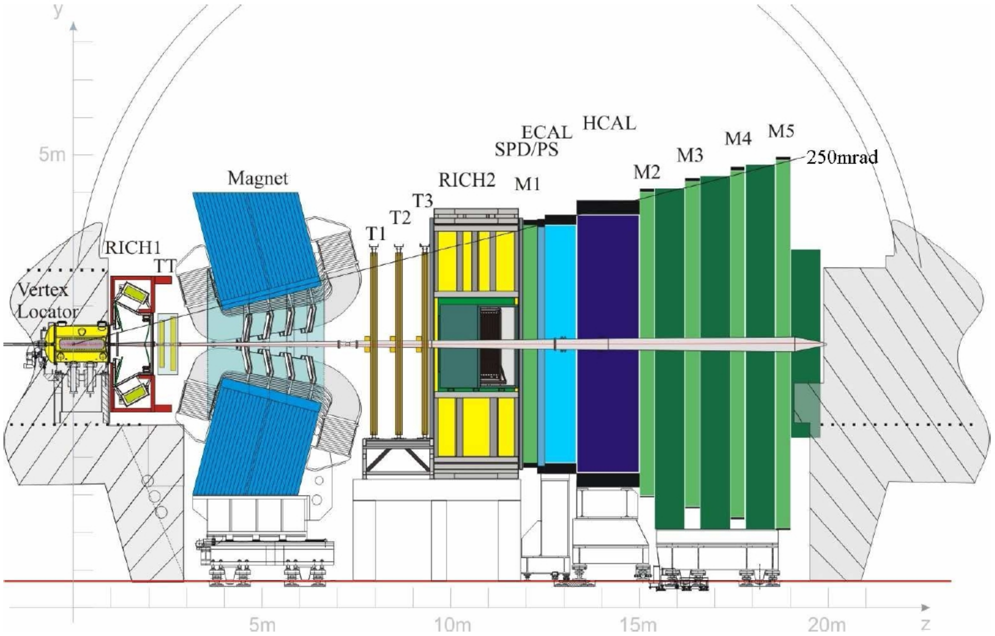


Figure 3.4: The LHCb detector with its subdetectors [1]. Built as a one-armed forward spectrometer the collision point for the two proton beams is located inside the VELO on the left side of the figure.

The Vertex Locator

The Vertex Locator is the first component of LHCb's tracking system. The detector has a special layout to reach the area as close to the interaction point as possible. With this feature it is capable of measuring secondary vertices, which are striking for b and c hadron decays. As shown in Figure 3.5, the VELO is split in two halves, which can be moved to widen or narrow the gap between them. Each half consists of 21 VELO-tracking-modules, with two layers of silicon strips: One layer to determine the distance to the beam axis ($|R|$ sensors; red in Figure 3.5) and another to determine the polar angle of the particle track (ϕ sensors; blue in Figure 3.5). The combination of both sensors leads to an explicit point in the LHCb coordinates.

With this information, the particle tracks can be observed and reconstructed close to the interaction point [1].

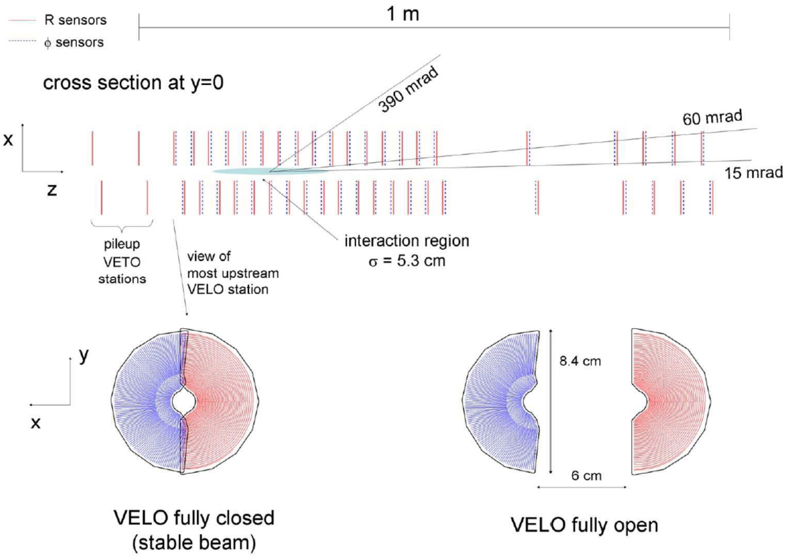


Figure 3.5: The Vertex Locator at LHCb [1]. On the top a cross section through the LHCb plane is shown. The 21 VELO stations can be seen as they are arranged around the interaction point of LHCb. On the bottom a closed and a fully opened VELO station is shown with an example of an $|R|$ sensor in red and a ϕ sensor in blue.

The Tracker Turicensis

The Tracker Turicensis is the second part of the tracking system and the last detector in front of the magnet. The TT is a silicon strip detector and covers an area of 1300 mm × 1500 mm. The tracker consists of 4 planes of silicon strips, with the inner two tilted by ±5° in the LHCb x-y-plane[IV]. This so-called X-U-V-X structure will be explained in detail in Chapter 4. For more information on semiconductor detectors see [20] and [21]. With the information gathered from the TT and the VELO the first part of track recognition can be performed before the particles pass the magnet [1].

RICH1 and RICH2

The RICH detectors are two detectors used for particle identification. As the name implies the detectors use the Cherenkov effect [22] to provide particle identification

[IV] The LHCb coordinate system can be seen in Figure 3.4.

of charged particles, by measuring the diameter of the emitted cone of light. Hereby, RICH1 is positioned right after the VELO whereas RICH2 is located behind the tracking station T3. As Figure 3.6 displays, both detectors use a similar setup with a gas filled body and photon detectors at their sides. They differ just in details, such as the used gases. While RICH2 is completely filled with CF4 (Tetrafluoromethane) RICH1 uses two different media, a small block of Aerogel and C4F10 (Perfluorobutane) for the rest of the body [23]. With a complex alignment of mirrors the Cherenkov emitted photons are guided to the frame of the RICHes, where the photon detectors are positioned. With the measured intensity and position of the triggered photon detectors the speed of the particles passing through the RICHes can be determined and used for particle identification [1].

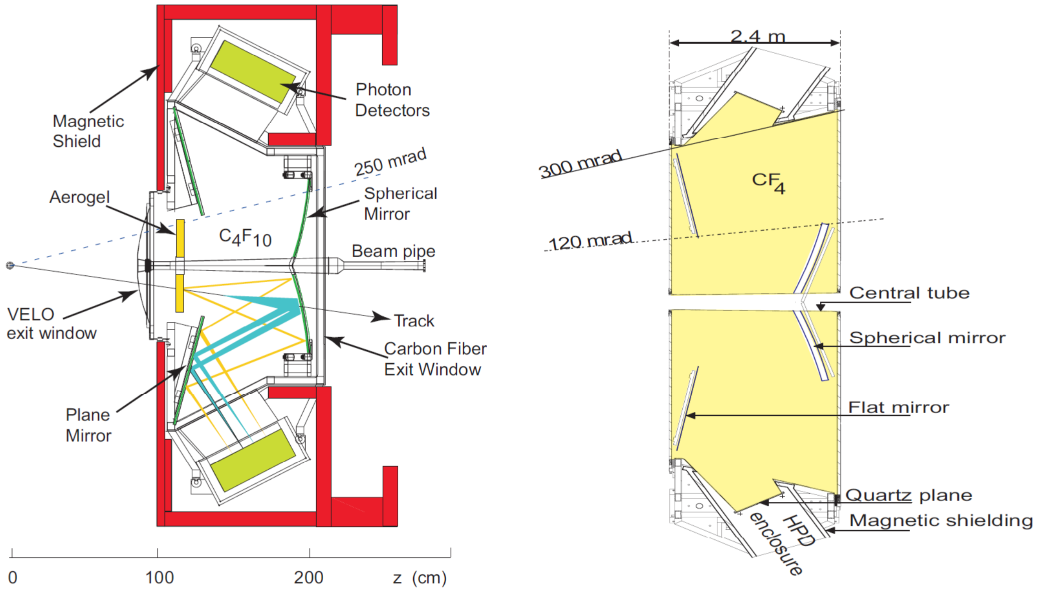


Figure 3.6: RICH1 (left) and RICH2 (right) [1]. These two detectors use the Cherenkov effect to determine the speed of passing particles by measuring the diameter of the emitted cone of light. The emitted light is then guided to photon detectors (HPD) by mirrors. The differences between these detectors are the position (see Figure 3.4) and the used medium.

The Tracking Stations T1 - T3

The three tracking stations T1 - T3 are located behind the magnet. As Figure 3.7 shows, each T-Station is built of an Outer Tracker (OT) and an Inner Tracker (IT). The IT uses the same technology as the TT. Its silicon strip modules cover an area of 1256 mm width and 414 mm height set in four boxes arranged as a cross (see Figure 3.8). With this positioning the innermost region of the T-Stations is covered

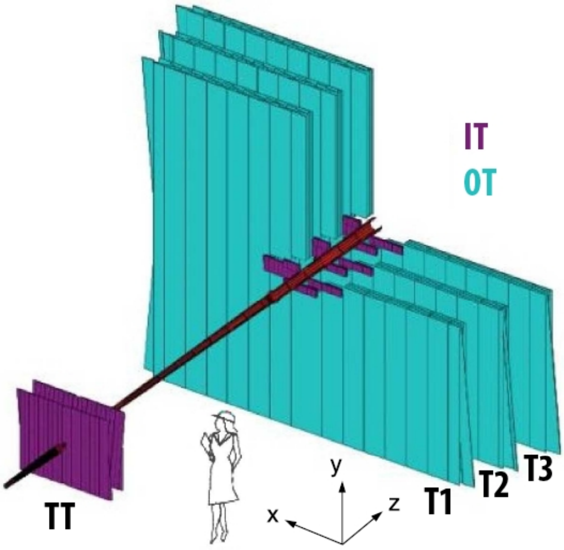
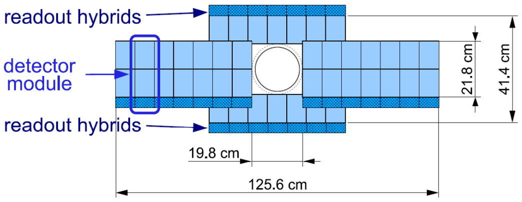


Figure 3.7: The figure shows the tracking stations T1-T3 and the TT [1]. The silicon detectors (TT and IT) are coloured purple and the OT (a gaseous detector) is shown surrounding the IT. Each station consists of four layers of each detector technology, with the the inner two layers tilted by ±5° in the LHXb x-y-plane.

with 200 µm wide silicon strips and a resulting track resolution of about 50 µm [1]. The modules inside the boxes are arranged in four planes in the already mentioned X-U-V-X structure to gain resolution in the LHCb y-axis. The Outer Tracker is a gaseous straw tube detector, which covers an area of $5 \times 6\,\mathrm{m}^2$ outside the IT, and will be covered in detail in Chapter 4 [1].



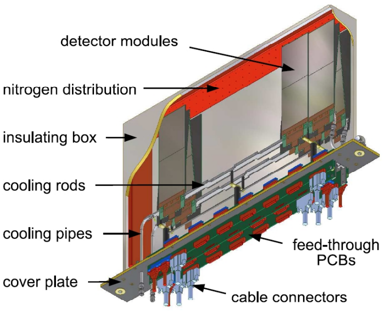


Figure 3.8: Schematics of the IT box set up within the tracking stations are shown on the left. The right figure illustrates the inside of an IT box with the four layers of silicon strip modules as well as the tilted modules [1].

The Calorimeter System

The LHCb calorimeter system consists of three different parts. The Silicon Pad Detector/PreShower (SPD/PS), the Electromagnetic Calorimeter (ECAL) and the Hadronic Calorimeter (HCAL) are lined up in this order behind RICH2 and the first muon chamber (M1). With the help of this system the energy of particles can be measured. This information is crucial for a successful particle identification. The SPD/PS consists of a lead converter, that is sandwiched by scintillating pads. Its main purpose is to separate electrons from pions. Both particles would shower in the ECAL and separating them beforehand gives a great benefit in particle identification.

The ECAL is used to measure electromagnetically interacting particles, like electrons and photons, and is built in a so-called shashlik calorimeter technology of alternating layers of lead and scintillators. The lead reduces the energy of passing particles, while the scintillators indicate how far the particle has intruded into the detector and how much light it produced. The scintillators are then read out and the penetration depth of the particle in combination with the amount of light can then be used to determine the deposited particle energy.

Finally, the HCAL determines the energy of strong interacting particles, hadrons as the name indicates. It consists of alternating layers of iron and scintillators and operates according to the same principle as the ECAL, as shown in Figure 3.9. The LHCb HCAL is oriented parallel to the LHC beam axis, thereby the length of the steel parts correspond to one interaction length in steel. The length of the HCAL is set to prevent most particles from leaving it, except for muons which are able to pass through the calorimeter system.

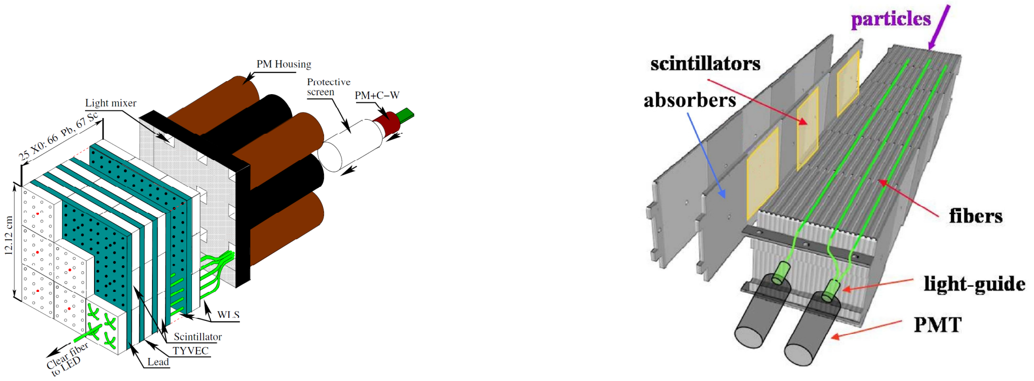


Figure 3.9: The LHCb calorimeter system. On the left a construction view of the ECAL is shown [24]. The shashlik building technique can be seen with alternating layers of lead and scintillators. On the right a schematic view of the HCAL is shown. In contrast to the ECAL the shashlik building technique is arranged along the particle beam direction [1].

The Muon System

The muon system is a crucial part of particle identification, since many final states in B decays include muons. It consists of five stations M1 to M5 positioned at the end of the detector behind the HCAL, except for M1 which is positioned between RICH2 and the calorimeters. The muon stations M2 to M5 are interleaved with iron absorbers to select muons and exclude other particles (see Figure 3.10). M1 is used to improve the p_T measurement for the trigger. Each muon station is split into four regions and equipped with Multi-Wire Proportional Chambers, which are gas detector, except for the innermost region of M1, which is built of Gas Electron Multipliers (GEMs) due to the high radiation environment. Both detectors work with the principle of ionising gases and measuring the drift time of the produced electrons. The principle of a gaseous detector is explained in detail in Chapter 4.

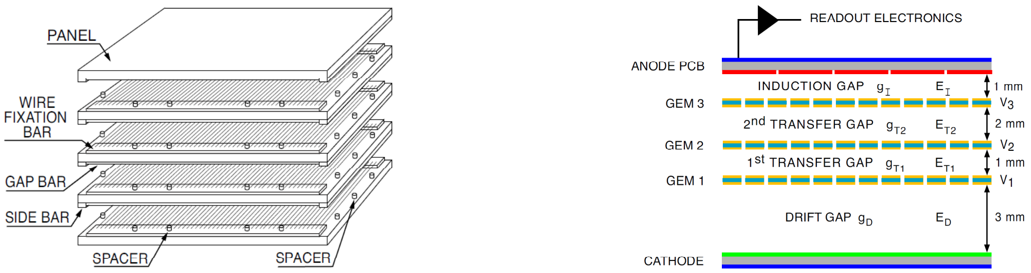


Figure 3.10: The muon system of LHCb is built of two different technologies. On the left a schematic view of the mostly used multi-wire proportional chambers is shown. On the right a schematic cross section of a triple-GEM detector is shown, which is used in the central region of M1 [1].

3.3 The LHCb upgrade

The whole LHCb experiment will experience an upgrade to improve its performance and be able to handle new circumstances. The instantaneous luminosity in LHCb will be raised from up to $5 \times 10^{32}\,\mathrm{cm^{-2}\,s^{-1}}$ [2] to $10^{33}\,\mathrm{cm^{-2}\,s^{-1}}$ [25]. Furthermore, the spacing between two consecutive proton bunches will be shortened to 25 ns [25] instead of the 50 ns mainly used during the first data taking period [2].

This will increase the experiments sensitivity in the b- and c-hadron decay studies, which will lead to higher precision measurements. These could surpass the precision in a lot of already published measurements, such as $B_s^0 \to \mu^+\mu^-$ [26] and $B^0 \to K^{*0}\mu^+\mu^-$ [27] in the flavour physics programme or the measurement of the weak phase in B_s^0 [28] oscillation in studies of CP violation. With the increased sensitivity even further measurements can be obtained, including rare decays, CP violating observables and fundamental parameters of the CKM unitarity triangles.

But not only precision can be improved, the upgraded detector can serve as a multi-purpose detector in the forward direction. With a flexible trigger and the high resolution in the forward region, LHCb can easily be adapted to new interesting physics fields. The search for exotic particles or measurements of the electroweak sector are possible, too.

To perform this task all subdetectors will be upgraded, for specific details see [25]. The electronics upgrade, which will apply each subdetector, is essential to improve the data acquisition rate. A detailed view on the front end electronics upgrade for the OT will be given in Chapter 4 and for the back end electronics in Chapter 5. Besides the electronics upgrade the event trigger will be modified. Since not every event contains interesting particle decays, the detector response is scanned for specific behaviour and selected by various trigger levels.

The current data reduction through trigger decisions is performed in several consecutive bandwidth reducing levels as illustrated in Figure 3.11. The data taken by the subdetectors with the LHC bunch crossing rate[V] is then buffered in the front end electronics. At this point the so-called Level-0 (L0) trigger reduces the output rate of the front end electronics to 1.1 MHz, by using direct hit information from the muon and calorimeter systems [29]. Afterwards, the data is processed in the back

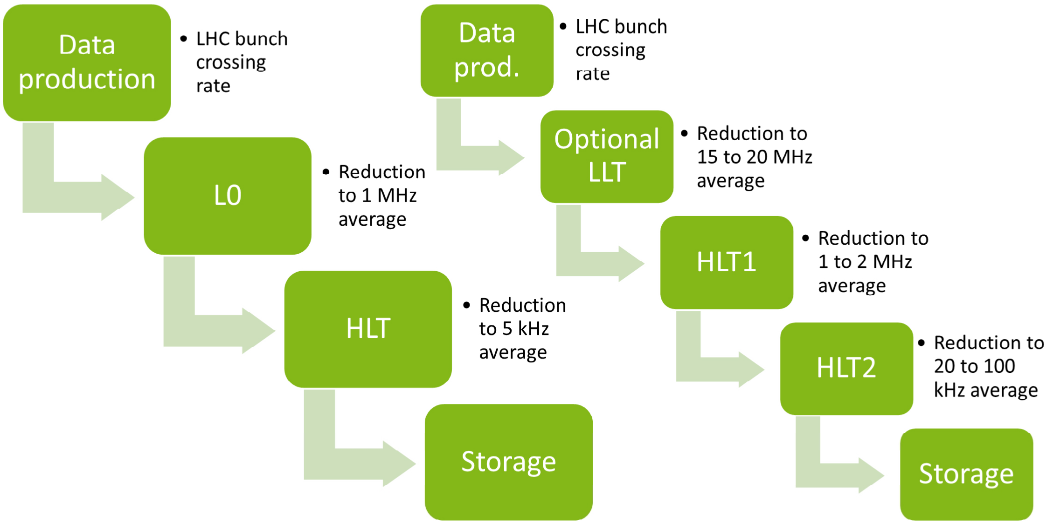

Data production
• LHC bunch crossing rate
L0
• Reduction to 1 MHz average
HLT
• Reduction to 5 kHz average
Storage
Data prod.
• LHC bunch crossing rate
Optional LLT
• Reduction to 15 to 20 MHz average
HLT1
• Reduction to 1 to 2 MHz average
HLT2
• Reduction to 20 to 100 kHz average
Storage

Figure 3.11: The DAQ at LHCb. On the left the current acquisition is shown with all rate reducing trigger levels. On the right a plan of the upgraded trigger steps is given. Here the data is reduced with more information than in the current triggering, where the hardware based L0 trigger uses just simple detector information.

[V] The LHC bunch crossing rate was designed to be 40 MHz. However the bunch crossing rate during LHC Run I was set to 10 MHz–15 MHz [2].

end electronics and further reduced by the High-Level-Trigger (HLT) to an average rate of 5 kHz, through detailed cuts on particle masses and pulses. With this average rate, the data is then stored in the data storage.

For the upgrade, LHCb targets a triggerless readout of the subdetectors. So each front end electronic has to be upgraded to be able to read out the particular subdetector in an 25 ns bunch crossing spacing. With a 25 ns bunch crossing spacing the full detector has to be read out with a rate of 40 MHz, to avoid a buffer overflow on the front end side. The data is then sent to the back end electronics, in which a first processing of the data occurs and an optional Low-Level-Trigger (LLT) can reduce the output rate down to 15 MHz–30 MHz. Afterwards another trigger level, the first part of the HLT, reduces the rate to 1 MHz–2 MHz. In contrast to the actual trigger the first reduction after the upgrade is done with a software trigger, which uses a full track reconstruction instead of just muon and calorimeter information (as the current L0) [30, 31]. At the end of the data acquisition process the second part of the HLT reduces the rate to 20 kHz–100 kHz. With this rate the data is then written into the storage. A schematic view of the upgraded DAQ is shown on the right of Figure 3.11.

4 The Outer Tracker and its upgrade

4.1 The Outer Tracker detector

As already mentioned in Chapter 3 the Outer Tracker (OT) is part of the tracking stations T1 to T3. It is a gaseous straw tube detector covering an area of $5 \times 6\,\mathrm{m}^2$. Each station is equipped with four layers of 22 modules. The layers are arranged back to back with a so called X-U-V-X geometry. That means that, the modules in the X-layers are oriented along the LHCb y-axis (see Figure 4.1), whereas the modules in the U- and V-layers are rotated by $+5°$ and $-5°$ respectively in the LHCb x-y-plane to gain a hit resolution in the LHCb y-direction. Each layer consists of 14 long and 8 short modules, in which 7 long and 4 short modules per layer are attached to one C-frame[I]. The long modules contain two times 128 2.5 m long channels, arranged in a double layer structure of straw tubes, which are read out at the top and the bottom

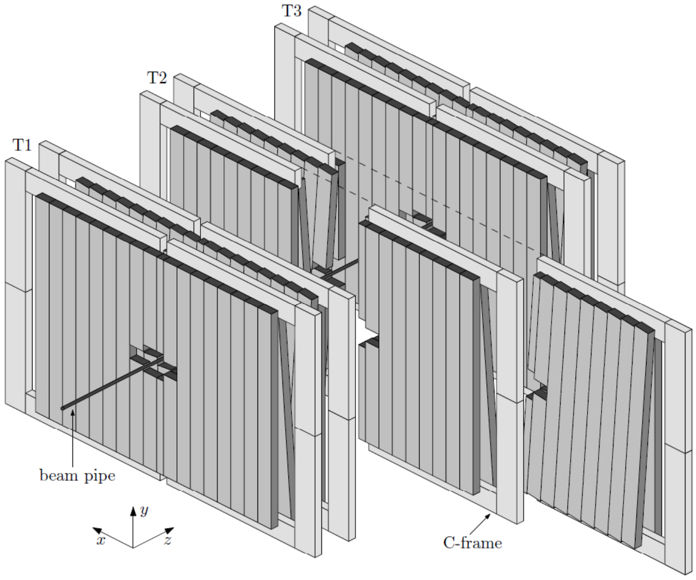


Figure 4.1: Scheme of the OT in the tracking stations T1 to T3 [32]. The X-U-V-X structure within each station can be observed just as the C-frames and the hole for the IT surrounding the beam pipe.

[I] A C-frame is a mechanical support structure, which can be pulled away from the beam pipe.

of each module. The short modules are equipped with 128 shorter straw tubes due to the adjacent IT. With this setting a total of 53,760[II] single straw tube channels are read out [32].

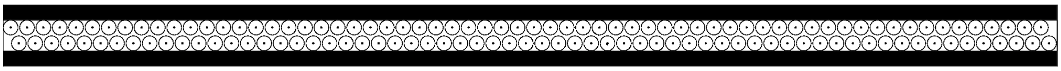


Figure 4.2: Close up of an OT module, with a two-rowed 128 channel layout.

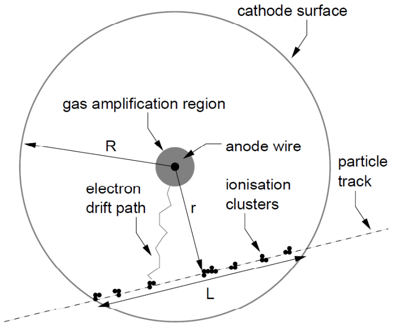


Figure 4.3: Functionality of a straw tube detector like it is used in the Outer Tracker [32]. A charged particle ionises the gas within the tube and the electrons drift towards the anode wire due to the applied high voltage. The distance to drift time relation can be seen in Figure 4.9 in Section 4.3

The double layer architecture of an OT module is shown in Figure 4.2. 128 channels are arranged in two lines, with 64 straw tubes each. An OT straw tube is up to 2.5 m long with an inner diameter of 4.9 mm and continuously flushed with a gas mixture of 70 % Ar, 28.5 % CO_2 and 1.5 % O_2. The anode wire is made of gold-plated tungsten of 25 µm diameter and operated at a high voltage of +1550 V, whereas the cathode is a sandwich of two Kapton-XC[III] foils with a layer of aluminium in between. When a charged particle passes through the straw tube it ionises the gas mixture and electrons are set free in this ionisation clusters (see Figure 4.3). These electrons are then attracted to the positive high voltage of the anode and the time of arrival is measured in relation to the LHC clock. With the measured drift time a distance drift time relation can be used to get the minimal distance the particle passed the anode. Thus, an accuracy of about 210 µm [32] could be obtained during LHC Run I. If two adjacent straws are hit and the distance of the track to the anode is known, this arrangement reduces the number of possible tracks through a module to four. Figure 4.4 shows the possible track reconstruction. To reduce the number of possible tracks even further, more consecutive modules are needed, and used in the OT. To determine the distance of the track to the anode a drift time is measured.

[II] Two short modules (one above and one below the beam pipe) in each layer are equipped with 64 channels only and read out with one front end box for two module layers.

[III] Kapton® is a polyimide film developed by DuPont.

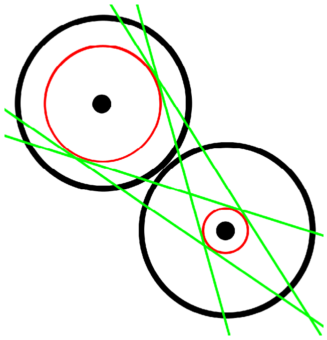


Figure 4.4: Possible track trajectories in a single OT module. Since only the hit distance (red) to the anode is known, four trajectories (green) are possible within a single module. By adding more consecutive modules these can be reduced.

4.2 The Outer Tracker electronics

Data acquisition from the straw tubes, especially the drift time, and keeping them functioning are the tasks of the Outer Tracker Front End (FE) electronics. Therefore, special electronics have been built and combined in the so called Front End Box. As Figure 4.5 illustrates, the FE Box consists of separate boards, being the HV board, the ASDBLR board, the OTIS board and the GOL board. Each will be explained in the following.

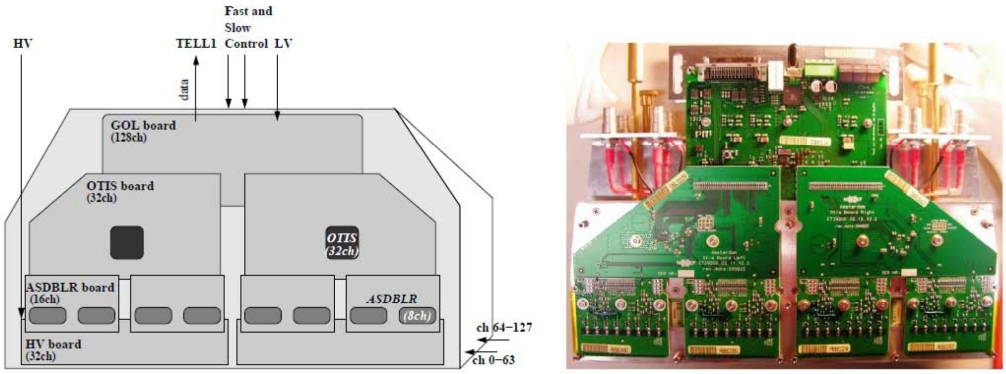


Figure 4.5: Schematics of the OT FE Box with its different electronic boards (left) and a picture of an actual Front End Box [32].

The High Voltage (HV) board is a simple supply unit, which delivers the used high voltage of +1550 V to each straw tube channel. One HV board handles 32 channels, thus four of them are needed in one FE Box.

The Amplifier, Shaper, Discriminator and BaseLine Restorer (ASDBLR) board works as the link between the straw tubes and the Outer Tracker Time Information System (OTIS) board. Each ASDBLR board hosts two ASDBLR chips, with eight channels each. These chips contain the whole analogue signal processing chain. The power of the signal coming from the straw tubes is increased with the amplifier. Afterwards, the signal is shaped to a format the Time to Digital Converter (TDC) can interpret, with the help of the shaper and discriminator. Finally, a baseline restorer stabilises the signal. Two ASDBLR boards are then connected to one OTIS board of which four are arranged in one FE Box.

An OTIS[IV] board holds one radiation-hard OTIS TDC chip. This TDC chip is capable of digitising the incoming signals of the ASDBLR chips in steps of 0.4 ns and thereby creates a digital drift time value for each channel. The determined data is stored in a pipeline memory capable of storing the data for 4.1 µs. On a positive Level-0 trigger drift times from three consecutive bunch crossings are merged and transferred to a derandomising buffer.

Data from the buffers of four OTIS boards are then sent to the back end electronics, called the Trigger Electronics Level 1 (TELL1), via the Gigabit Optical Link (GOL) auxiliary board. Besides providing the data connection to the off detector electronics, the GOL board handles the power connection and the interfaces to the fast and to the slow control[V].

4.3 The Outer Tracker performance

The Outer Tracker showed an excellent performance during LHC run I. The modules had an average occupancy of 3 % to 15 % as seen in the black curve in Figure 4.6, representing a 50 ns bunch crossing theme in the LHC. This occupancy is shown for typical run conditions in 2011 and 2012. After the upgrade, the occupancy for the 25 ns bunch spacing could be exceeded due to a higher multiplicity in LHCb events. But the shown occupancy can give an assumption to the estimated detector occupancy [32] after the upgrade.

Another crucial aspect in detector performance is the hit efficiency. The vast majority of the OT modules have a hit efficiency between 98.5 % and 99.7 %, for hits within a radius smaller than 1.25 mm around the anode. The hit efficiency within a straw tube can be seen in Figure 4.7 and is located above 50 % inside the whole straw tube.

Since the OT is a tracking detector and used for the reconstruction of particle tracks, the hit resolution has to be well known. The design specification for the hit resolution of the OT was 200 µm. As the residual plot in Figure 4.8 shows, a hit resolution of

[IV] The OTIS works on a Low Voltage Differential Signaling (LVDS) basis. LVDS is a voltage standard as described in ANSI/TIA/EIA-644-A.

[V] The fast control provides the electronics with continuous signals, like the LHC clock, whereas the slow control writes and reads registers to control the electronics.

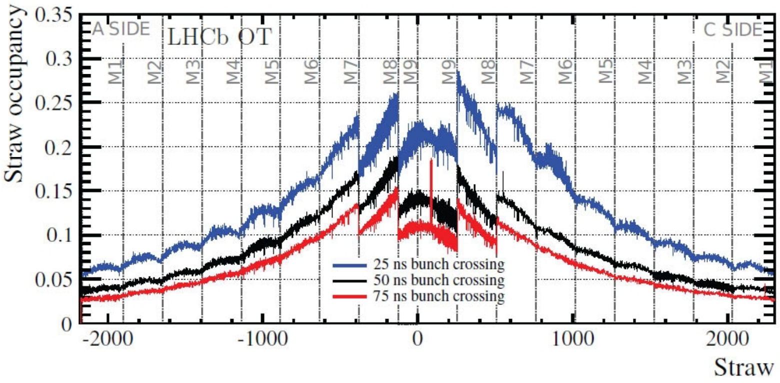


Figure 4.6: Straw occupancy versus straws for 75 ns (red), 50 ns (black) and 25 ns (blue) bunch crossing spacing in LHCb [32]. The lower occupancy in modules M8 and M9 is explained by shorter modules due to the IT.

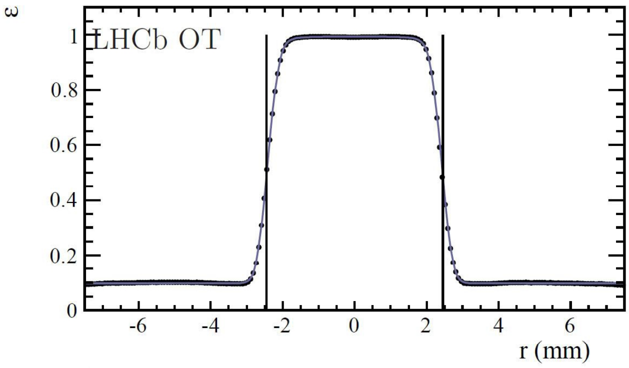


Figure 4.7: Hit efficiency (ε) versus radius (r) [32]. The bars indicate the border of the straw tubes.

205 µm was achieved in the 2011/2012 data taking period. This resolution is due to internal construction accuracy and subsequent offline alignment procedure. Hereby, errors occur from the straw accuracy in the modules (±50 µm), the accuracy with which the modules are hung inside the C-frame (±50 µm), the frame positioning (±1 mm) and the error on the optical survey (±0.2 mm) [32].

To get a proper distance out of the measured drift times a distance to drift time relation is needed. Figure 4.9 shows the distance to drift time relation with the shape of a second order polynomial distribution. The drift time calibration is done four times a year and the current calibration is found to be:

$$t_{\text{drift}}(r) = 20.5\,\text{ns} \times \frac{|r|}{R} + 15.85\,\text{ns} \times \frac{r^2}{R^2},$$

with r as the distance of the hit from the anode and R = 2.5 mm as the tube radius.

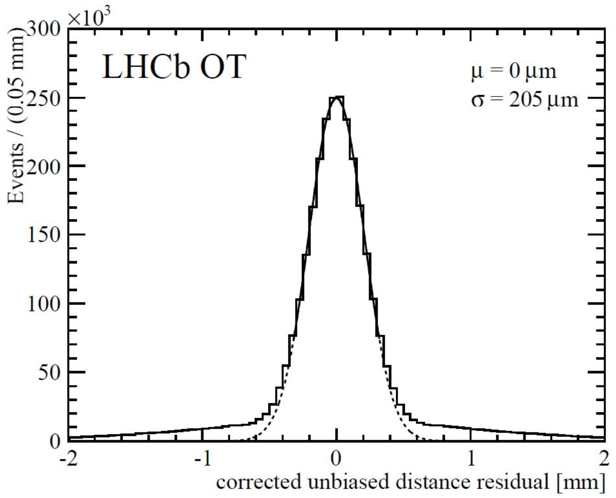


Figure 4.8: Events versus corrected unbiased distance residual [32]. The distance error and thereby the resolution of the detector is about 205 µm.

The red lines in Figure 4.9 indicate the maximum possible drift time with 35 ns. This equals the maximum of the simulated OT drift time spectrum shown in Figure 4.10a. However, the highest measured drift time was about 50 ns (see Figure 4.10b). This difference is due to the timing of the front end clock, which is not calibrated for each module separately, and the time of flight of the particle, starting at the interaction point, which again is assumed the same for each module.

Except for the occupancy, all[VI] of the mentioned aspects should not be affected by the LHCb upgrade with a higher luminosity and a higher multiplicity. Therefore the only part of the OT which has to be upgraded due to the new trigger design will be the electronics.

[VI] The distance to drift time calibration will still be done a few times per year, and therefore the values will change.

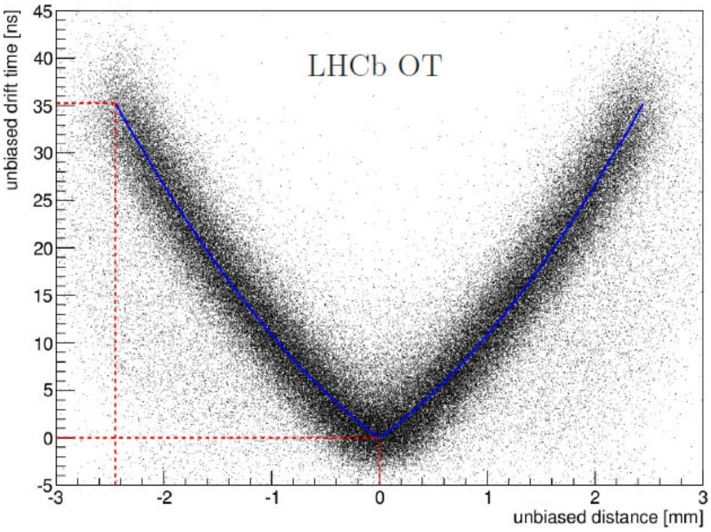
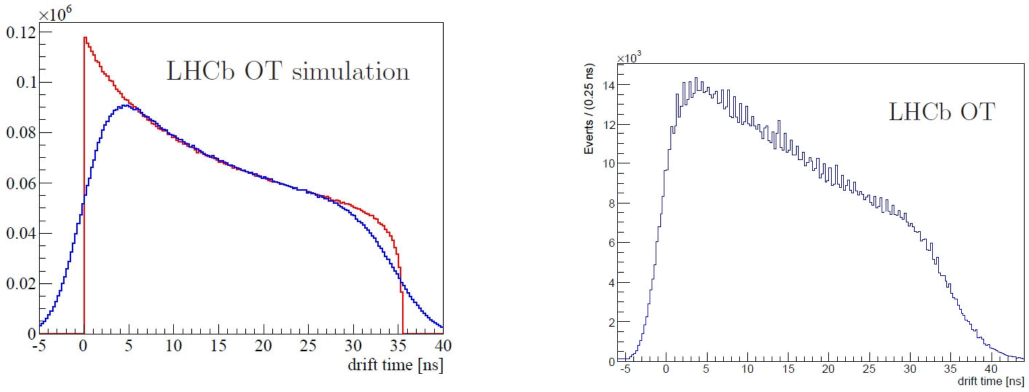


Figure 4.9: Unbiased drift time versus unbiased distance [32]. The distance to drift time relation is calculated with this information.



(a) A simple simulation of the OT drift time spectrum (red) and smeared with a time resolution of 3 ns (blue).

(b) A measured OT drift time spectrum.

Figure 4.10: The Outer Tracker drift time spectrum [32].

4.4 The Outer Tracker front end electronics upgrade

The tracker upgrade foreseen for the Outer Tracker would contain reusing the current straw tubes. Therefore, the main interest in development is the upgrade of the OT electronics, which consists of the front end electronics, handled in this section and the back end electronics, for which the firmware was developed in this thesis, explained in detail in Chapter 5. There are two options for the Outer Tracker Front End electronics upgrade, both using the current HV- and ASDBLR-boards and replacing the other parts. Anyway both options would zero suppress the data taken before sending it to the back end to reduce the bandwidth usage. This means, that only the data of hit channels is send to the back end, compressed at the beginning of the stream, while the rest of the data stream is filled with the next taken data.

4.4.1 The MicroSemi option

The MicroSemi[VII] option is under development by a group at NIKHEF[VIII]. It is based on an Actel Proasic3E flash based FPGA[IX] and uses a similar setting as the actual FE Box. The GOL board will be replaced by a Master GigaBit Transceiver (GBT) board, which will perform the same tasks the GOL board does at the moment. Besides forwarding the GBT information from the TDC boards to the off detector electronics, the Master GBT board hosts the power converters for the whole FE Box and a Slow Control Adapter (SCA). As Figure 4.11 illustrates an additional GBT for the fast control, called Timing and Fast Control (TFC), and the slow control is implemented on the Master GBT board to provide a connection to the overall controlling systems. These include the slow control for the configuration of the FE and the TFC, which provides the FE amongst others with the LHC-clock and Bunch Crossing Identification (BxID) information. The OTIS boards will be replaced by TDC boards. These will include the TDC FPGA, two widebus GBTs and an SCA. Each FPGA will contain a 32 channel TDC, which generates a five bit drift time with a resolution of about 780 ps. To decrease the bandwidth at the GBTs a zero suppression will be performed on the FPGA.

Additional tasks are monitoring and performing the TFC and slow control commands. The GBTs are used for sending the data out of the TDC board and the SCA performs as another control instance.

[VII] MicroSemi is an FPGA manufacturer formerly known as Actel.

[VIII] NIKHEF: National Institute for Subatatomic Physics, which is located in the Netherlands (Nationaal instituut voor subatomaire fysica).

[IX] The difference between flash based and SRAM FPGAs will be discussed in Chapter 6.

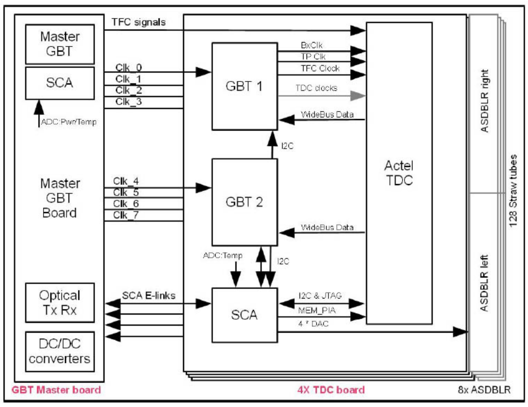


Figure 4.11: MicroSemi based upgrade plans as developed at NIKHEF [33]. This scheme with four TDC boards and a Master GBT board would replace the actual OTIS and GOL boards.

4.4.2 The Altera option

The Altera[X] option was developed at the Ruprecht-Karls-Universität Heidelberg [34]. The tested FPGA was an Static Random-Access Memory (SRAM) based Altera Arria I. This FPGA solution includes transceivers (transmitter and receiver) capable of sending the data as fast as the GBT chip in the MicroSemi option. Thus, no Master GBT board would be needed as every task would be performed not only on a single board, but also just on one FPGA for 32 straw tube channels. The TDC with a resolution of about 780 ps as well as the transceivers can be implemented on the FPGA along with the monitoring, the zero suppression and the interpretor for the TFC and slow control. Figure 4.12 shows a preliminary version of such a board, which has already been used for radiation tests and could be used in the OT upgrade with small adoptions in board design and functionality.

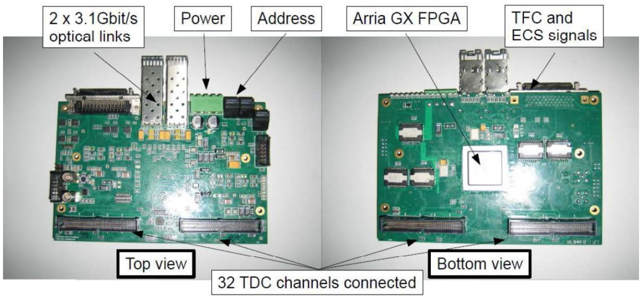


Figure 4.12: Altera based upgrade plans as developed at the Ruprecht-Karls-Universität Heidelberg [34]. A similar board could replace an OTIS board, whilst the GOL functions would be included.

[X] Altera is an FPGA manufacturer mainly known for their Stratix and Arria devices.

5 The LHCb back end electronics upgrade

The LHCb upgrade requires an upgrade of the whole back end electronics. The current LHCb electronics is designed to process incoming data with 1.1 MHz, whereas the requirement after the upgrade will correspond to an average of 40 MHz [25]. Due to this, a new back end electronics architecture is being developed. In the following, this electronics architecture upgrade will be discussed, whereat the information is taken from [35], [36] and [37], unless indicated otherwise.

5.1 The back end electronics upgrade structure

The back end electronics is located off detector in a dedicated room outside the radiative detector area, where no radiation hardness of the electronics is required. Therefore, the commercial Advanced Telecommunications Computing Architecture (ATCA) standard will be used[I] [35]. Positioned inside an ATCA crate, the back end electronics will consist of an ATCA carrier board with four Advanced Mezzanine Card (AMC) slots, as shown in Figure 5.1. A special ATCA carrier board, the ATCA40[II], is being developed to acquire the needs for the LHCb upgrade. With AMC40s for different functions, the ATCA boards get different purposes, determined by the AMC40 firmware. For example a TELL40 board would be an ATCA40 carrier board equipped with four AMC40s programmed for data acquisition. An example for a complete setup, which is able to take data, is shown in Figure 5.1. It is built of a TFC[III] crate including a TRIG40 and a S-ODIN board and a readout crate built up of TELL40 boards able to process data coming from the FEs and sending them to the farm. Further information about the boards is given in [36], whereas the data processing version of the AMC40s will be discussed in the course of this thesis.

[I] Commercial components reduce costs in research, development and production and are therefore favoured.

[II] The 40 in ATCA40 refers to the naming convention of the upgraded back end electronics to correlate to the 40 MHz readout scheme and differentiate it to the actual TELL1 back end electronics.

[III] The upgraded TFC is also called S-TFC to distinguish it from the current electronics, whereas the "S" stands for "Super".

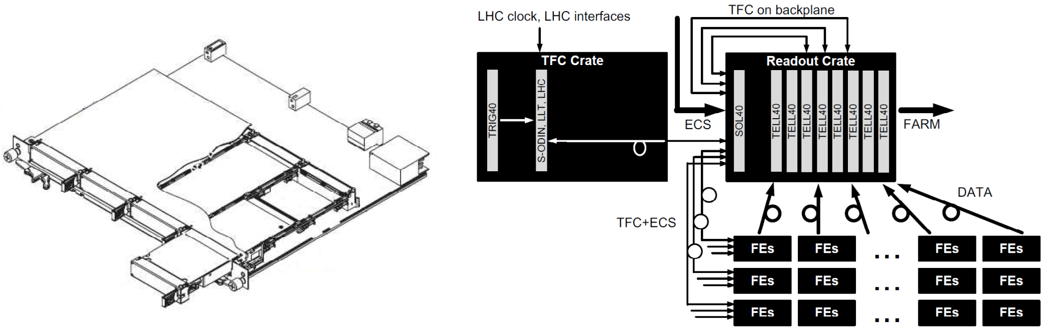


Figure 5.1: An outside view of an ATCA carrier board with four AMCs is shown on the left [35]. On the right a schematic view of a complete ATCA40 setup is illustrated [37]. A TFC crate with a connected readout crate, to which FEs are connected.

5.2 The ATCA40

The ATCA40 is a specialised ATCA based carrier board for the LHCb upgrade. As Figure 5.2 implies, four AMC40 boards can be equipped on an ATCA40 and connected via different electrical lines guided to different parts on the ATCA40. Besides basic voltage supply, the ATCA40 carries devices to interact between the AMC40s and the backplane and therefore other electronics inside the same crate.

The Credit Card Personal Computer (CCPC) is linked via four Peripheral Component Interconnect Express (PCIe) buses to the AMC40 boards and with a Gigabit Ethernet (GbE) link to the outside of the board. The CCPC reads and writes registers on the AMC40 FPGAs and connects to the Electronic Control System (ECS) through the Ethernet interface. The clock crossbar allows to route clock signals between the AMC40s and the backplane of the ATCA40 board, and from AMC40 to AMC40. It is used for transferring the LHC clock synchronously to all AMC40s. The serial crossbars purpose is to route high speed signals between the AMC40s, an additional connection to the backplane enables an exchange of signals between AMC40s on different ATCA40 carrier boards. A part of the transferred signals are not defined in advance, because every AMC40 configuration uses a different format to interact with other AMC40 boards of the same configuration[IV]. The only predefined signal is the TFC signal, which carries the commands for every AMC40 configuration in the same format. Additionally, a general purpose FPGA is positioned on the ATCA40 board, whose main task is to concatenate the throttle information and send it to the AMC40s. The last active part on the ATCA40 is the CIPMC. This controller allows changing the AMC40 boards without rebooting the ATCA40 and collects sensoring information from the AMC40 boards via the Intelligent Platform Management Bus (IPMB).

[IV] These signals are called X_FPGA in Figure 5.3.

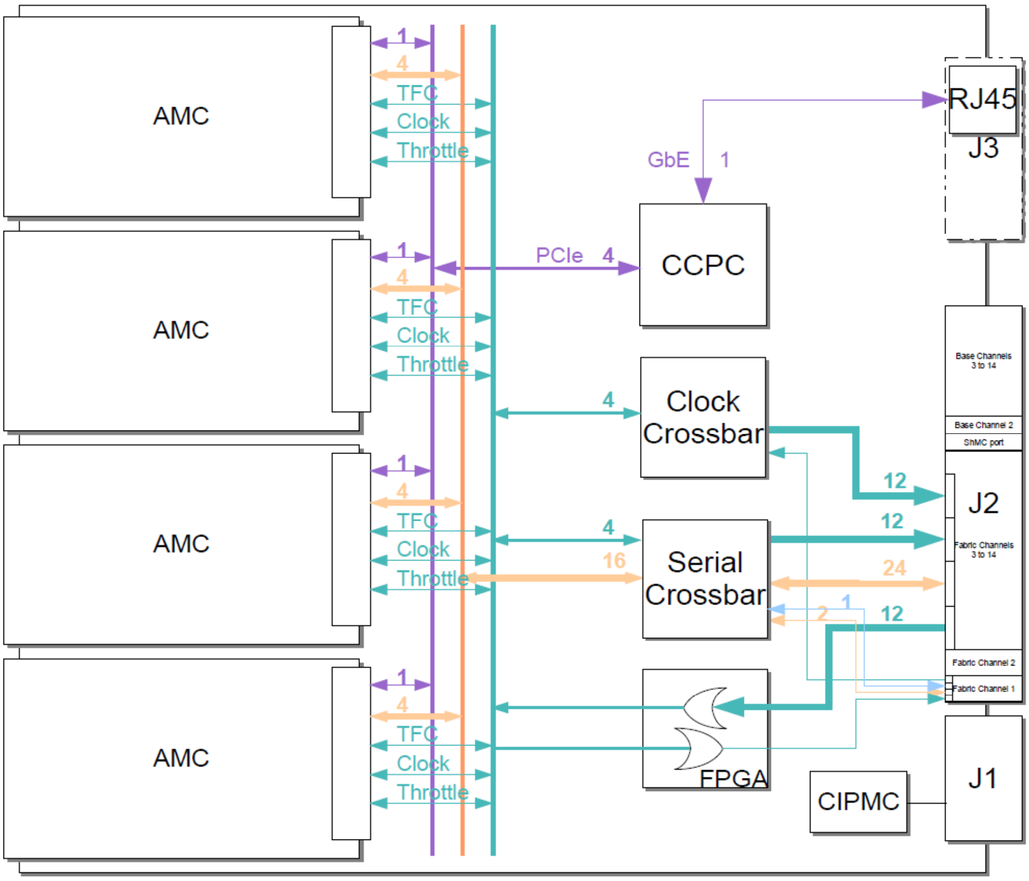


Figure 5.2: Schematic view of an ATCA40 board, with four AMCs connected [35]. All data carrying electrical lines are shown as well as the connectors to the backplane of the crate. Additionally, the active working parts are indicated. There are the CCPC, the Clock Crossbar, the Serial Crossbar, an FPGA and the CIPMC.

5.3 The AMC40

The AMC40 boards are connected via the AMC connector with different protocols to the ATCA40 carrier board as already described. Figure 5.3 shows a simplified architecture of the AMC40. The main component on the AMC40 is the "Stratix V GX"[V] FPGA. The FPGA performs nearly every active part on the AMC40, including the data processing in a TELL40 configuration, which will be explained in detail for the OT in Section 7.2. The FPGA is connected to every other part on the AMC40, which are the optical interface, the Random Access Memory (RAM), a Module Management Controller (MMC), the AMC connector, a Phase Locked Loop (PLL) and the jitter cleaner.

[V] The Stratix FPGA series is a product of Altera Corporation.

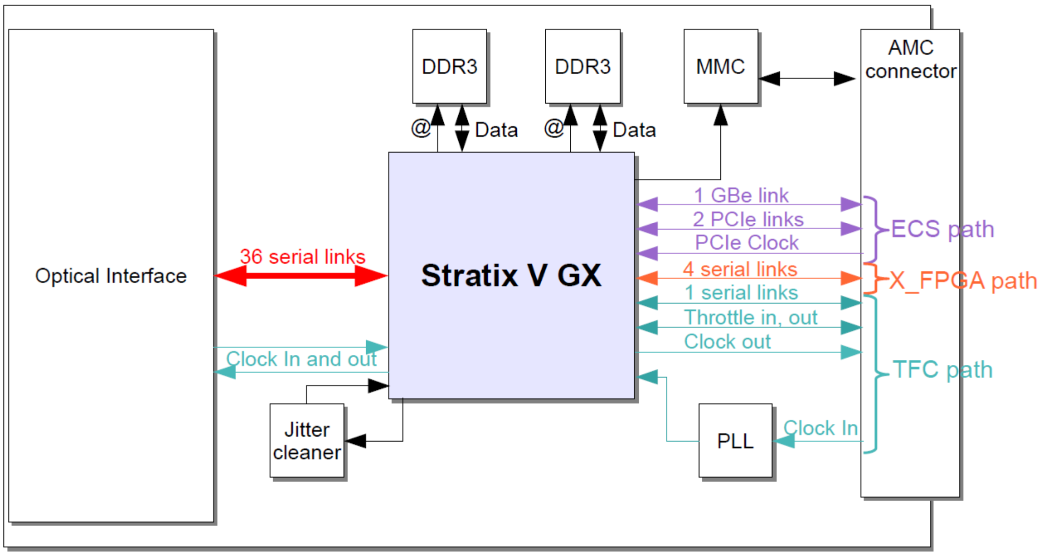


Figure 5.3: A schematic view of the AMC40 architecture [35]. The AMC40 is centred around the Stratix V GX FPGA. The optical interface ensures the connection to the FEs, whereas the AMC connector does the same to the ATCA40, via the described links. The other parts seen in the illustration are supporting parts and described in this chapter.

Hereby, the RAM consists of two blocks of DDR3 RAM of a still undecided size with a maximum interface speed of 667 MHz. Its main purpose is to buffer data for the TELL40 configuration of the AMC40. The optical interface is made of three twelve-channel transmitters and three twelve-channel receivers from Avago[VI] able to handle data at up to about 10 Gbps. It is connected via 36 bidirectional serial links to the Stratix V so that each of the 36 optical links can be converted from optical signals outside the board to electrical signals on the board, the FPGA is able to interpret. The MMC assures the interface with the Intelligent Platform Management Interface (IPMI) bus, which allows exchanging AMC40 boards without rebooting the ATCA40 and communicates with the CIPMC on the ATCA40. The jitter cleaner is a support circuit which reduces the amount and strength of jitters on the connections of the FPGA to the other parts on the AMC40. Hereby, a jitter is to be understood as a fluctuation in the flank of a digital signal as shown in Figure 5.4.

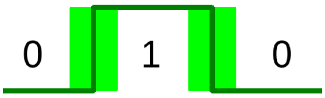


Figure 5.4: An example for a jitter in a flank of a digital signal. The flank in the dark green signal could vary in the green area, which would be called a jitter.

[VI] The used transceiver will be an Avago AFBR-77D1Z.

6 Field Programmable Gate Arrays

FPGAs are Integrated Circuits (IC) containing programmable blocks and connections, which in contrast to Application-Specific Integrated Circuits (ASICs) can be configured after manufacturing the chip. FPGAs are often used in R&D, as the ability to be reprogrammed allows an extensive testing and changing of its behaviour, for which an ASIC had to be realised as a new chip in hardware. So custom made electronics can be equipped with an FPGA and the logic on this FPGA can be tested and varied through the whole electronics lifetime. In electronics for large experiments FPGAs are preferred, due to their possibility of being enhanced and debugged during the experiment's lifetime.

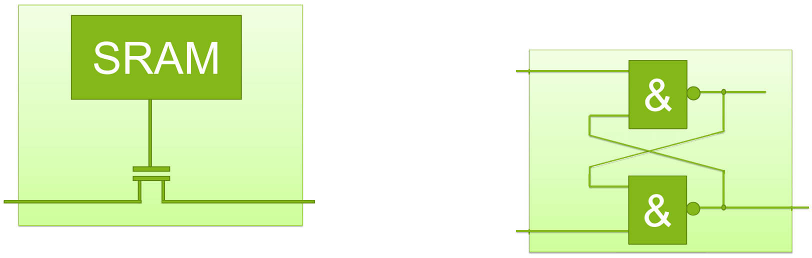
Due to the fact that the work of this thesis is done on FPGAs and the extensive use of FPGAs in particle detector electronics, a short introduction shall be given in this chapter. After a short introduction into FPGA technologies, internal structures of an FPGA are explained, followed by an explanation of the typical working procedure with FPGAs.

6.1 FPGA technologies

Different basic FPGA technologies are used in modern electronics. Most of them are only one-time programmable, like Fuse, Antifuse, Programmable Read-Only Memory (PROM) and Erasable Programmable Read-Only Memory (EPROM), and only used in specific cases, like low cost serial production or in extreme environments (e.g. under high radiation). Three reprogrammable FPGA technologies are SRAM, Electrically Erasable Programmable Read-Only Memory (EEPROM) and Flash, whereas Flash based memory is an advancement of EEPROM [38]. While EEPROM memory has to be erased completely before being rewritten, Flash memory is capable of being erased partially. Thus, the two most promising memory technologies, SRAM and Flash, will be discussed in the following, whereas the information in this chapter is taken from [38], [39] and [40].

FPGAs are basically built of transistors and working as Integrated Circuits (ICs). Since the connections between those transistor blocks are programmable, it is necessary to save the information for the FPGAs usage inside the transistor environment. So the main difference in technologies is the used memory.

SRAM is a volatile memory, whereas volatile means, that the stored information is lost at losing the FPGA voltage. An SRAM based programmable cell is built of an SRAM memory and a transistor, which is set through the information stored inside the SRAM memory, as Figure 6.1a illustrates. The SRAM memory, in its simplest way, works as an electronic flip-flop as it can be build of NAND elements in Figure 6.1b. There are more advanced versions of used flip-flops in SRAM memories, information about these can be found in [39] or [40].

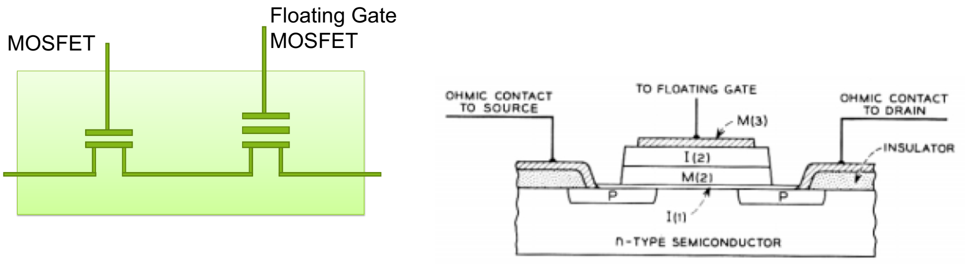


(a) An SRAM cell inside an FPGA. The SRAM memory defines the status of the transistor. Depending on the information stored inside the SRAM the transistor opens or closes and defines the connection as logical "1" or logical "0".

(b) A NAND flip-flop as possible memory in an SRAM. This flip-flop can only serve as a one bit memory, more complex ways to store information inside flip-flops can be found in [39] or [40].

Figure 6.1: SRAM memory structure.

As already mentioned Flash memory has the same working procedure as EEPROM, with the ability to delete smaller parts. Therefore, the raw functionality will be explained with an EEPROM. A schematic view of an EEPROM cell is shown in Figure 6.2a. It consists of two transistors, a normal Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET) and an EEPROM transistor (a floating gate MOSFET). A floating gate transistor is usable as a non-volatile memory unit. The floating gate can be programmed to keep the transistor open or closed by adding charge to the floating gate or removing it. A schematic view of a floating gate transistor is given in Figure 6.2b. Detailed information on using floating gate transistors can be found in [41], also more complex variations of floating gate usage can be found in [39] and [40]. Both technologies are used in current FPGAs, depending on the aim of their final use and cost. In Section 4.4 research on FPGAs with both technologies has already been mentioned. Especially the radiation hardness is of high interest and investigated for the OT FE electronics upgrade. Detailed information about the research in radiation hardness on the FPGAs intended for the OT FE upgrade can be found in [33] and [34].



(a) An EEPROM cell as it can be used inside an FPGA. The floating gate MOSFET is capable of staying open or closed, whereas the MOSFET in front of the floating gate is capable of erasing the floating gate MOSFET.

(b) Schematics of a floating gate transistor as invented by D. Khang [41]. Modern floating gate MOSFETs still use a similar setup.

Figure 6.2: EEPROM memory structure.

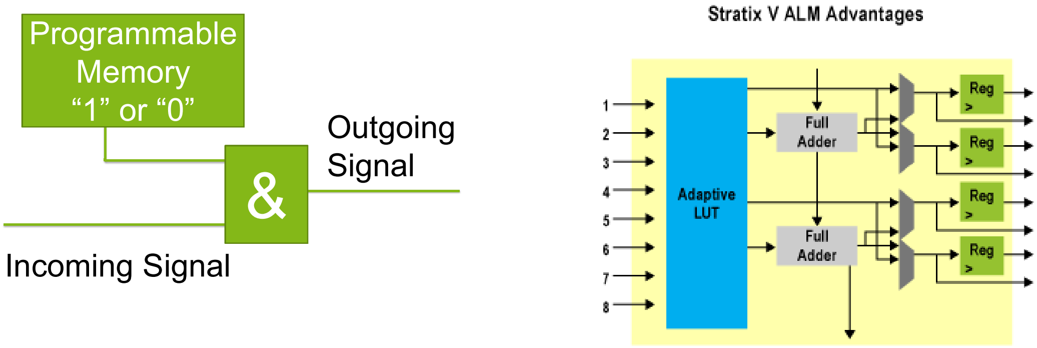
6.2 FPGA structures

Simple defined bits, respectively opened or closed transistors, will not provide any action since they will not change without reprogramming. Combining these fixed bits with more transistors, logic and connections inside and to the outside of the FPGA leads to combinatorics which can perform defined actions. An easy example is given in Figure 6.3a. Depending on the programming of the "Programmable Memory" the incoming signal can manipulate the outgoing signal. If the memory is programmed to be "0" the outgoing signal will remain "0", programmed as "1" the outgoing signal will be the same as the incoming signal. This is a simple example how a Look-Up Table (LUT) can be created, manipulated and used (in this case it is the simple LUT of an AND logic). This can be enhanced to bigger LUTs, with a definable number of input and output signals and various logic functions or combinations. The combination of LUTs, logic and transistors as a block is named and used differently among FPGA vendors. For example, Altera calls their combination Logic Element (LE) or Adaptive Logic Module (ALM) depending on the device family. A Stratix V ALM block diagram is shown in Figure 6.3b[I].

The other important configurable parts of an FPGA are the reprogrammable interconnections. With the help of this feature the programmed blocks can be connected in any desired combination. So each outcome of an LE can have an impact on any other used LE and therefore develop huge and complex logics. A simple generic architecture with four programmable FPGA LEs and interconnections is shown in

[I] Since the FPGAs used within this thesis are made by Altera their nomenclature will be used.

Figure 6.4. Modern FPGAs are built of thousands of these structures[II] and are able to perform a large amount of actions.



(a) A simple example for a programmable Logic Element. Depending on the programmed value inside the memory the outgoing signal behind the AND-logic is set to zero or the value of the incomming signal.

(b) A block diagram of a Stratix V ALM [42]. It consists of eight inputs with an adaptive fracturable LUT, two embedded adders and four dedicated registers.

Figure 6.3: Logic elements inside an FPGA.

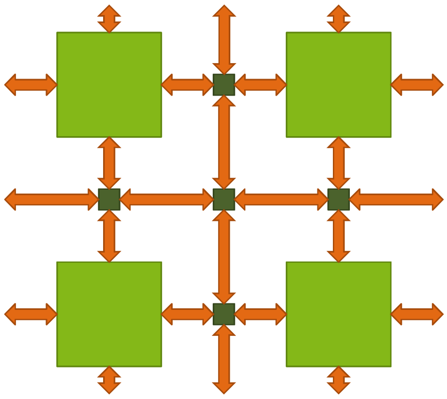


Figure 6.4: A top-down view of a simple generic FPGA architecture, with four LEs (green), five programmable interconnections (darkgreen) and the connecting lines between them (orange).

FPGAs are logic driven instances. Thus, a change on one signal changes the outcome. However, a clock driven logic is preferred. This means, that an external or internal quartz or another kind of clock provides the FPGA with a never-ending signal of alternating logic "1" and "0". The signal is changed with a specific frequency, which is measured from one rising flank to the next. So a distance of 25 ns between two

[II] The Stratix V GX 5SGXEA7, which will most likely be used on the AMC40 boards, consists of 938,880 registers and 234,720 ALMs, which would be an equivalent of 622,000 LEs [43].

rising edges would represent a clock frequency of 40 MHz. The clock frequency can be changed by logic inside the FPGA, but the principle of performing actions on a change of the clock remains. Since every logic is stored inside the LEs, the manipulated signals are sent from one block to another each clock cycle. Thus, every step programmed inside the FPGA is performed exactly at the same time each clock cycle, if not programmed otherwise.

6.3 Working with FPGAs

Developing a firmware for an FPGA is done in several steps: Coding, simulation, synthesis, programming and testing. These steps will be explained in the following.

The logic on an FPGA is described with a specific Hardware Description Language (HDL). The most widely used HDLs are Verilog and VHSIC Hardware Description Language (VHDL)[III] as they are standardised in the IEEE Std 1364-2001(Verilog) [44] and IEEE Std 1164-1993(VHDL) [45]. The language of choice in this thesis is the VHDL IEEE 1164 standard. This standard defines representative values for driving logic. The most significant ones are the two strong values "1" and "0", which are seen as logic "1", high or true, and "0", low or false. Based on this binary encoding and pairing multiple signals, every bit of data can be decoded to a higher level, like numbers (binary to decimal) or even alphabetical symbols using binary American Standard Code for Information Interchange (ASCII)[IV]. This variable interpretation of bits and bit vectors as boolean or values can drive decisions and be formed to logic serving for nearly every occasion. To keep track of all functions implemented inside the code, small functionable blocks are programmed. These blocks are called “entities” and consist of an entity declaration and an architecture. The declaration includes the input and output ports and their width and optional generics, in principle everything that connects to the outside of the entity. A simple example is given below.

```
ENTITY example IS
 GENERIC (
  example_value : std_logic := '0';
  example_counter : std_logic_vector(3 downto 0) := "0000");
 PORT (
  clk : in std_logic;
  reset : in std_logic;
  counter_input : in std_logic_vector(3 downto 0);
  counter_output: out std_logic_vector(3 downto 0);
  overflow : out std_logic);
END ENTITY;
```

[III] VHSIC stands for Very High Speed Integrated Circuit.

[IV] ASCII is a character encoding scheme for 128 specified characters including the alphabet.

The declaration in this example consists of two parts, a list of generics and a list of ports. First the generic list will be explained along with some syntax. This generic list consists of two "generics", named `example_value`, a "std_logic" type, and `example_counter`, a "std_logic_vector" type. The vector is generated with a width of four bit by `(3 downto 0)`, whereas the definition `downto` sets the counting of the single bits. A `downto` vector is initialised with the index of the highest number (in this case "3") on the left, so that the vector can be read as a binary combination[V], whereas a `to` vector is initialised and read vice versa. The `:=` interpreter sets a default value if no value is set from outside the block when it is initialised. The generics are then accessible from the outside of the block and a defined value can be assigned for each instance of this block.

The port list is the most important part of an entity description. It declares the incoming and outgoing signals and their width. The port list in the given example consists of three inputs (`clk`, `reset` and `counter_input`), symbolised by the declaration `in`, and two outputs (`counter_output` and `overflow`), symbolised by the declaration `out`. Hereby the `clk` signal will be used as the clock driving this block. With the declared ports and generics a simple adder will be programmed in the architecture (the behaviour) of the block, shown below.

```
ARCHITECTURE behaviour OF example IS
 signal temp : std_logic_vector(4 downto 0) := "00000";
BEGIN
 adding : process (clk)
 BEGIN
  IF rising_edge(clk) THEN
   IF reset = '1' THEN
    counter_output <= "0000";
    overflow <= '0';
    temp <= "00000";
   ELSE
    IF example_value = '1' THEN
     temp <= std_logic_vector (unsigned('0' & example_counter)
        + unsigned('0' & counter_input));
    ELSE
     temp <= '0' & counter_input;
    END IF;
    counter_output <= temp(3 downto 0);
    overflow <= temp(4);
   END IF;
  END IF;
 END PROCESS;
END ARCHITECTURE;
```

The first step inside the architecture, after naming and assigning it, is to declare the internal used signals, which in this case is the five bit wide vector `temp`. Afterwards,

[V] The binary combination is read as: 2^3 2^2 2^1 2^0.

BEGIN marks the beginning of the code to be processed. Next, a process is defined (adding), which is performed each time the signal in its sensitivity list experiences a change in its value (in this case the clk). An architecture may have any number of processes with the same, different or various sensitivity signals. The first if-statement checks whether the change on the signal clk is a rising edge from "0" to "1"(rising_edge{clk}). Not deciding on an edge, rising or falling, would perform the assigned logic at each change in signal, and thereby work with the double clock frequency. Then, the reset signal is checked and each outgoing and internal signal set to zero, by a positive decision.

Afterwards, the working logic description begins. If example_value is set to "1" the sum of example_counter and counter_input, including a leading "0" is written into the internal signal temp. One could also set this allocation to an outgoing signal. The redefining in unsigned and back to std_logic_vector are needed to instruct the compiler to handle the std_logic type as numbers. If example_value is set to "0" only the value of counter_input is set to temp with a leading "0". In the next clock cycle, simultaneously with the temp-setting logic, the stored value in temp is set to counter_output and the fifth bit is sent out in the overflow signal.

This code example will be used to explain the compilers behaviour. The code can be visualised as a block with inputs and outputs, as it is shown in Figure 6.5, and combined with other blocks by connecting their signals. A software like the used "HDL Designer"[VI] can interpret this code and visualise it as a flowchart as it can be seen in Figure 6.6. The code is then "synthesised" by a software of the manufacturer[VII]. This means it is translated into logic and adapted to the FPGA considering special needs of the preset FPGA. Along the way an optimisation is performed, which tries to minimise the used logic, by analysing the code. The generated logic equivalent to the example code is visualised in the Register Transfer Level (RTL) and shown in Figure 6.7. Afterwards, the produced file can be programmed on the FPGA and is ready to be used.

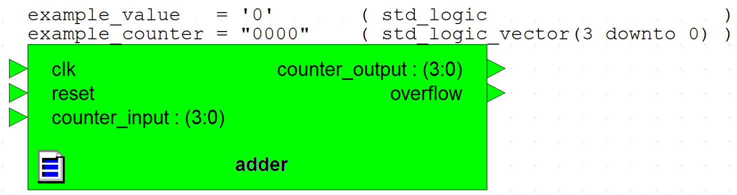

example_value = '0' (std_logic)
example_counter = "0000" (std_logic_vector(3 downto 0))
clk
reset
counter_input : (3:0)
counter_output : (3:0)
overflow
adder

Figure 6.5: The example code visualised as a block one level above the code.

To test the written code two possibilities are given, which both should be used. The first one is to simulate the code with a specific software[VIII]. This method allows to

[VI] HDL Designer is a software developed by Mentor Graphics. The used versions were 2009.a and 2012.b.

[VII] Altera provides their software QuartusII. Used Versions: 9.0 to 13.1.

[VIII] The used software was ModelSim developed by Altera.

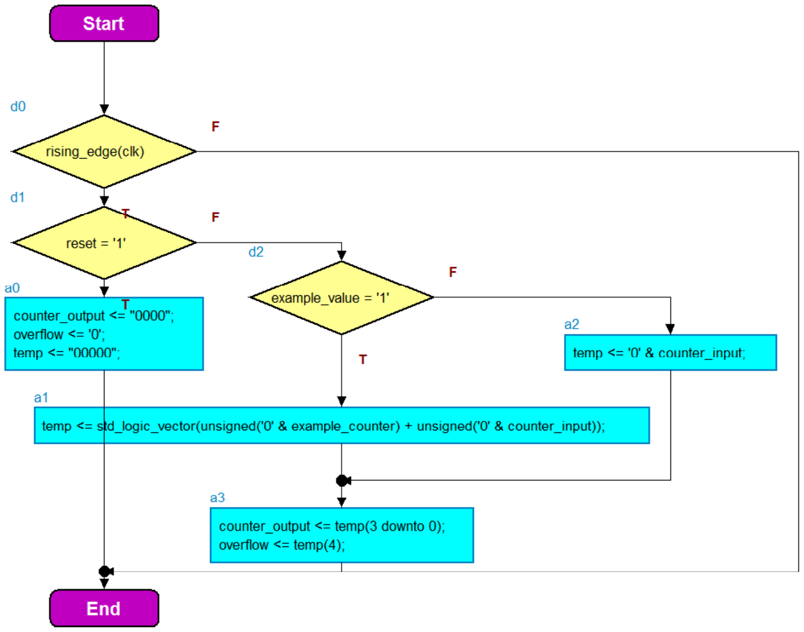

Start
d0
rising_edge(clk)
F
d1
T
reset = '1'
F
d2
a0
T
counter_output <= "0000";
overflow <= '0';
temp <= "00000";
example_value = '1'
F
T
a2
temp <= '0' & counter_input;
a1
temp <= std_logic_vector(unsigned('0' & example_counter) + unsigned('0' & counter_input));
a3
counter_output <= temp(3 downto 0);
overflow <= temp(4);
End

Figure 6.6: A flowchart of the example code generated by the HDL Designer.

show the value of each signal at every moment during processing. To simulate the built block correctly, stimuli are needed to perform all tasks which would be provided by the FPGA. Figure 6.8 shows a setup which can be simulated. All inputs to the block shown in Figure 6.5 are driven by a simulated stimulus, a clock, a pulse for the reset mechanism and a four bit counter for the input.

The simulated example code is visualised as signal value over time in Figure 6.9. On the left, the name of the simulated signal is shown, followed by the visualised signal variation over time including the value at that time. On every rising edge of the signal `clk` the process is executed. The `example_value` is added to the `counter_input` and stored in the five bit vector `temp`. On the following rising edge of the `clk` signal this value is split into the four bit vector `counter_output` and the std_logic value `overflow` when the fifth bit is needed to display the sum, while at the same time the `example_value` is added to the new `counter_input` and stored in the `temp` signal.

If the simulation shows the desired working procedure, the project, without the simulation stimuli, can be synthesised. Beforehand, the input and output ports of the project have to be allocated to the correct pins on the FPGA. During synthesis the code is ported and routed to the characteristics of the used FPGA. The code is also checked for further errors, which did not occur during simulation, and a timing analysis is done. This timing analysis shows if the code can be routed without any constraints to the working procedures in providing data at different blocks to the time expected. With a wrong timing, especially for an ongoing data processing,

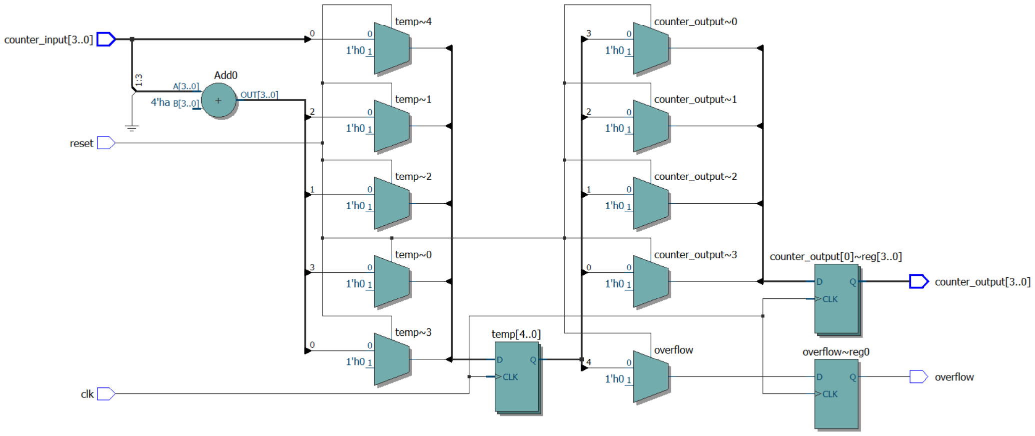


Figure 6.7: A view of the RTL generated from the example code by QuartusII.

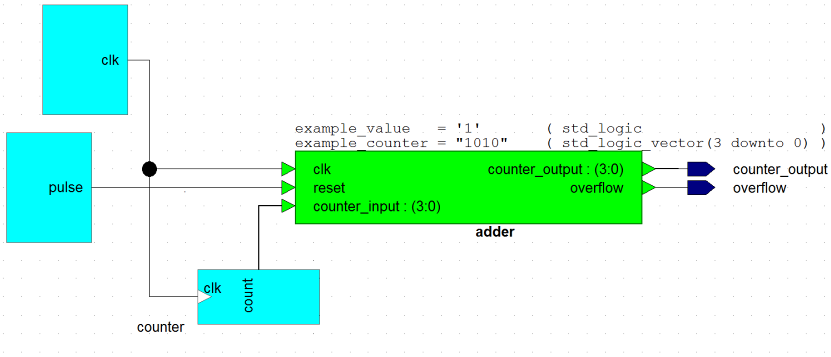


Figure 6.8: A block diagram of the example code and the stimuli (clock, pulse and counter) to be simulated with ModelSim.

parts of the data could be pulled out of their group and assigned to another.

The then produced file can be programmed onto the FPGA and tested. One way to program an FPGA is via a "joint test action group"[IX] connection, which can be established with any serial connection to the FPGA, e.g. USB or RS232. Another possibility is to store the program on a dedicated memory from which it is loaded on startup to the FPGA. There are also different ways to test a firmware on an FPGA. One is to simply put it into action and see if it works correctly. Another way is using a specified testboard or a development kit with indicators or a read out to determine the correct function of the firmware. For example a "Stratix IV GX Development Kit"[X], as shown in Figure 6.10, is used in this thesis. It is equipped

[IX] Joint test action group (JTAG) is the common name for the IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture.

[X] Used kit: Stratix IV GX Development Kit DK-DEV-4SGX230N.

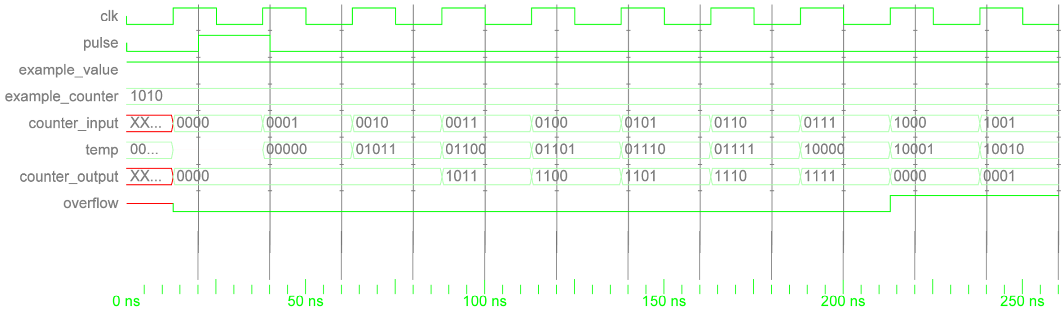


Figure 6.9: Simulation of the example code for the first 260 ns. The signals without the same name as in the example code are: `clk`=clock and `pulse`=reset. On every rising edge of the clock the example code is executed.

with an FPGA[XI], usable Light-Emitting Diodes (LEDs), a programmable Liquid-Crystal Display (LCD) and a few different Input/Output (I/O) buses, like PCIe. With this I/Os the firmware can be supplied with data to process and the outcome can be observed on a readable device. A few examples of test setups will be given in Chapter 8.

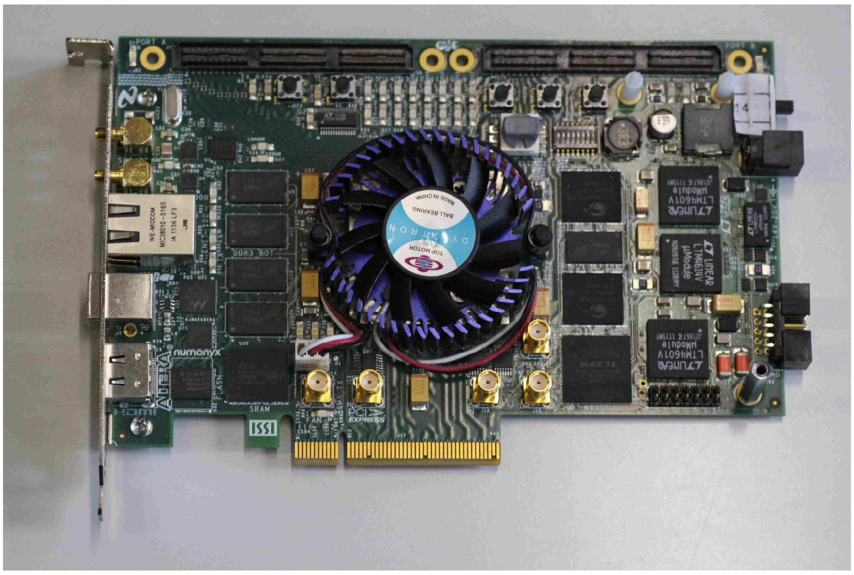


Figure 6.10: Picture of the used Stratix IV Development Kit DK-DEV-4SGX230N.

[XI] The DK-DEV-4SGX230N uses the Stratix IV EP4SGX230N-C2.

7 The AMC40 firmware for the OT upgrade

The firmware on the AMC40 is split into a common part and a subdetector specific part. The subdetector specific part is declared as "Data Processing" in the following. The common part will be provided by a group at the Laboratoire d'Annecy-le-Vieux de physique des particules (LAPP) in Annecy [46], while the data processing part for the OT is described in this thesis.

The task of the AMC40 and its firmware is to receive data sent from the FEs, process it and then send it out to the DAQ, as already explained in Section 5.3. In the following, the common part will be explained, followed by a detailed view of the subdetector specific data processing part for the Outer Tracker.

7.1 The AMC40 common part

The common part of the AMC40 firmware will be the same for each subdetector. Figure 7.1 gives an overview on the AMC40 design for the MiniDAQ[I]. Excluding the number of links and some additional functions, it resembles the layout of the final design. The purpose of each block, except the "Data Processing", will be explained shortly in the following. The relevant common parts for the AMC40 firmware design, as depicted in Figure 7.1, are "Decoding", "BCID Alignment", "LLT Decision" and "MEP building".

The "Decoding" block searches for the header in the incoming data stream and aligns the data. The BxID[II] and the header are separated and checked for inconsistencies. Afterwards the data is stored in a memory block.

The following "BCID Alignment" is a crucial part to the "Data Processing". Due to different cable's lengths and processing time inside the FEs, the data arrives asynchronous according to the timing on the AMC40 and the bunch crossings respectively. So, the data has to be aligned consistently with the BxID between the different optical link inputs to guarantee a faultless data processing. Furthermore, a timeout has to be performed; if data with the matched BxID from one or more

[I] The MiniDAQ is an AMC40 test system and will be described briefly in Chapter 8.

[II] Also called BCID as in Figure 7.1.

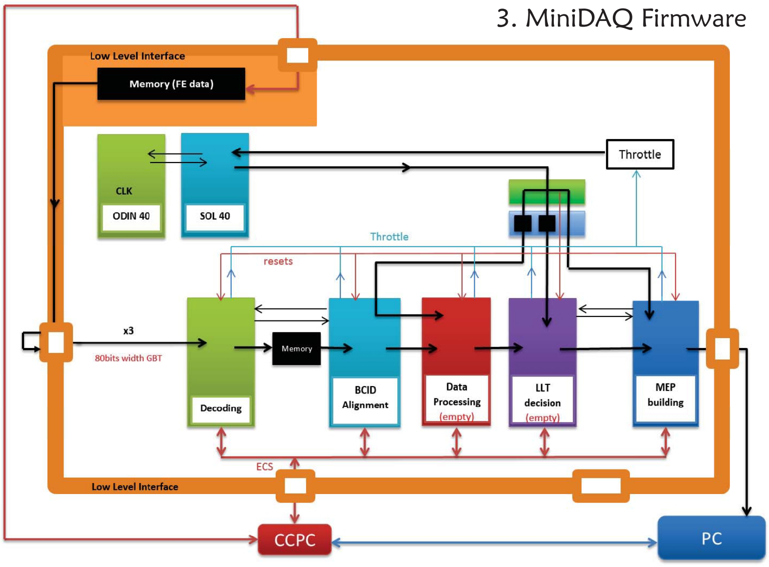


Figure 7.1: An overview of the firmware for the MiniDAQ, resembling the final firmware for the AMC40 except in some details [47].

optical links is not delivered within a previously defined timeslot, an error mechanism has to interrupt and either delete this BxID or use the data with a missing part. The maximum size of this timeslot will be defined by the amount of used memory inside the design, whereat a shorter time is a safer option to prevent a data overflow inside the memory.

After the data processing an “LLT Decision” is performed. With this option the Low-Level-Trigger can reduce the amount of data put through to the DAQ. Nevertheless, the goal to achieve is not to depend on this function and send all data to the HLT.

The last step of design relevant parts is the “MEP Building”, where a Multi-Event Package (MEP) is constructed out of several events. The previously by BxID sorted data will be put together to construct larger packages suitable for an Ethernet connection. This larger packages will include data of multiple events and can be sent to the DAQ regarding to the actual workload of the network. Thus, a capacity overload of the network can be prevented.

7.2 The AMC40 Outer Tracker data processing

The main task of the AMC40 firmware data processing part is handling the detector specific data. Since the firmware has to manage different detector occupancies, as already described in Section 4.3, and non zero suppressed data as required through [37], the OT data processing firmware is designed to process zero suppressed as well as non zero suppressed data with the blocks and connections. Therefore no additional logic is required on the FPGA, except for a small indicator if the data is zero suppressed or not. With this arrangement the OT data processing firmware is also capable of processing data with a detector occupancy of 100 % without efficiency loss.

Due to the relatively high drift time described in Chapter 4 the most important procedure for the Outer Tracker is to merge data from three consecutive bunch crossings. To make this possible, some preprocessing is necessary. After a short overview of the whole data processing for the OT, a detailed description of each part of the firmware is given.

7.2.1 Overview

Following the data flow through the data processing, the most important parts are receiving the data, sorting it, merging data of three consecutive bunch crossings and formatting it for the correct output. Figure 7.2 shows an illustrated data flow through the data processing. The data enters the processing part as a binary vector filled with a header, hit pattern and the drift times as already explained in Section 4.4 and visualised in Figure 7.3. In the receiving part the header is read and the information of the status bits and registers of the ECS part is used to perform the necessary actions in this and the following parts. The sorting mechanism was developed in two different ways, with a different philosophy standing behind each implementation. Both are explained in detail in Section 7.2.3. The main idea is still to react to wrong cable management on the hardware of the AMC40 board. Since it is not possible to send all the data from one FE over a single optical link[III], multiple links have to be aligned in the correct way. To react to a wrong cable alignment, the firmware has to check its correctness. Merging data in the OT format is the most crucial part. Due to the zero suppressed data, different merging algorithms were tested. The smallest one in logic usage was then preferred. At the end of the data processing chain the data is put into the correct order and established with a common header.

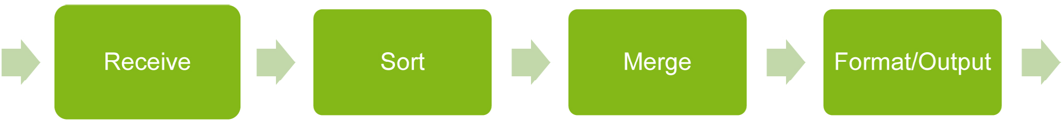


Figure 7.2: An overview of the main data processing working steps as it can be inserted into the data processing part of the AMC40, marked red in Figure 7.1.

[III] Each OT FE will use four optical links.

[IV] The final width of the BxID is not defined yet. A modification in the firmware can be easily obtained.

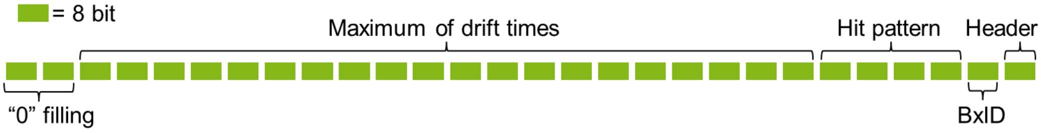


Figure 7.3: Content of an incoming 224 bit wide data bus. The bits 7 down to 0 are designated to the header, as the bits 15 down to 8 are for the BxID[IV]. The hit pattern is then placed in the bits from 47 down to 16, with the drift time following in 207 down to 48.

7.2.2 Receiving data

The data processing part of the AMC40 firmware for the OT will be given up to 24 buses with a maximum width of 224 bit each. This would include a maximum of two total GBT frames in wide bus mode for a single data stream. As already mentioned each incoming bus consists of a header, a BxID, a hit pattern and the corresponding drift times of one OT FE FPGA (see Figure 7.3). Since the data will be delivered from the common part arranged to the BxID, the main task of the receiving part is to interpret the header and ECS commands. The header can transport information to consider for the back end electronics (e.g. non-zero-suppressed data, sync-mode). The overall first bit to be checked is the ECS register which indicates if this bus is used or turned off due to hardware or software reasons. For each deactivated bus a signal is set to induce every following component using this bus to switch into an ongoing "reset-mode". Hereby the "reset-mode" sets every outgoing signal to zero, thus no action will be performed during this state. The logic will react as if a continuous reset signal is set.

The first bit of the header to be checked is the so called "sync" bit, which indicates the status of the transceivers on the AMC40. During sync-mode the AMC40 transceivers synchronise with the ones on the FEs. The OT plans to use part of the unused GBT frame during syncing to send the ID of the connected FE and FPGA. With the sync bit set, the receiving block sends the data stream to a component, which will check the connected IDs (see Section 7.2.3). Without the sync bit set, the data stream is sent to the first part of data processing inside the firmware. In addition, the receiving block checks the header for errors to report to the monitoring system. This is done by checking the error bits in the header and setting a signal indicated by the monitoring part of the firmware. Each receiving part is very small in logic occupation, which is crucial in developing the AMC40 firmware since the amount of logic on the FPGA is limited. The logic usage on the Stratix V GX 5SGXEA7[V] is shown in Table 7.1.

[V] The exact nomenclature for compiling is: 5SGXEA7N2F45C3.

Table 7.1: Logic usage on the Stratix V GX 5SGXEA7 for one receiving block. One block is needed for each connected optical transceiver input line.

Option	Logic utilisation (in ALMs)	Total registers
One block	119 of 234,720 <1% on FPGA	236
For one FE	476 of 234,720 <1% on FPGA	944

7.2.3 Sorting of incoming data streams

Two possibilities for sorting the incoming data streams were developed. The first one is user friendly and works with up to six connected FEs, but uses a lot of logic on the FPGA. The second one uses less logic, but needs the user to organise the optical fibre connection by himself. The sorting procedure will work during the sync bit is enabled, so no time is lost for finding the right connections during data taking. The IDs sent by the FEs consist of a twelve bit wide code, where ten bit identify the exact FE and two bit the FPGA on this FE. Both methods will be presented in the following.

Resorting incoming data streams

The first possibility of sorting data streams is to sort the incoming streams in logic. In this case the sorting mechanism works in two parts. The first part is an algorithm, which checks the incoming FE for their IDs and sorts it. Due to a lot of possibilities by connecting the FEs to the AMC40 the algorithm is limited to a maximum of six different FEs connected. If more than six different FEs are connected, an error flag is set and no sorting will be done.

The algorithm checks the FE IDs and assigns a three bit number to each of them. The scheme can be seen in Figure 7.4. A condition for this algorithm to work is a list of activated channels, which has to be provided via ECS as a simple std_logic vector. Thus, the possibility to deactivate channels and the use of any amount of them is included through this list. The FE connected to the first active channel will be given the three bit number "001". Now every other active channel will be checked for the same FE ID and given the number "001", since one FE will use multiple optical lines. Each channel with the same ID will then be deactivated for the next turn inside the algorithm. Now the first FE in an active channel will get the number "010" and the algorithm starts over, until all channels are marked as deactivated. In the second step the FPGA ID will be added to the FE ID number as a two bit addition. So the now five bit number of the first FPGA of the first FE would be

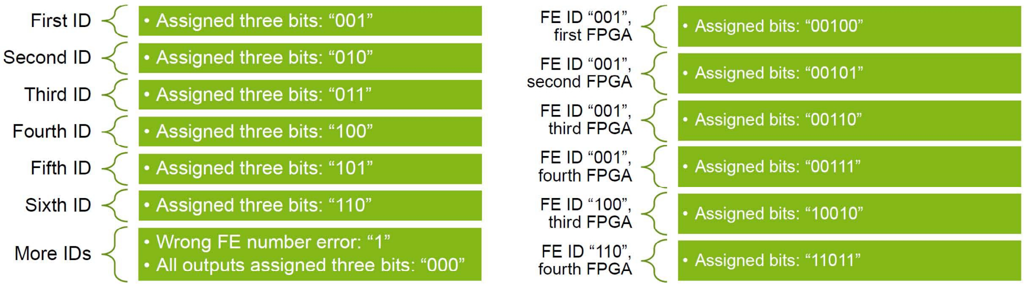


Figure 7.4: Scheme of assigned three bit numbers to different FEs on the left, and the combination with the added FPGA ID on the right.

identified by the bit vector "00100", while for example the fourth FPGA on FE five would be "10111". Note that the FPGA counting starts with "0" as the FE starts with "1". If more than six FEs are connected the algorithm sets an error bit for the ECS system and assigns each FE and FPGA the number "00000". With this setting every channel will be used as it is connected and further processing is not possible, except for a simple output.

In the second part the data is then sorted as shown in the scheme in Figure 7.4. Due to rearranging all the data streams a lot of logic is used. Each channel is sorted depending on the number assigned in the algorithm. So the channel with the assigned code "00100" is put on the first internal channel, while the channel with the assigned code "11011" is processed on the 24th channel. This is a simple re-routing sorted by the assigned code to get the incoming connections arranged to the correct internal processing channels. Since a lot of combinations are possible the usage of logic on the FPGA is very high. The usage for the code assigning process on the Stratix V is shown in Table 7.2 and for re-routing in Table 7.3.

Table 7.2: Usage of logic on the Stratix V 5SGXEA7 for assigning an internal ID. One block is needed for the whole firmware.

Option	Logic utilisation (in ALMs)	Total registers
One block	269 of 234,720 <1% on FPGA	424

Table 7.3: Usage of logic on the Stratix V 5SGXEA7 for re-routing the incoming streams. One block is needed for the whole firmware.

Option	Logic utilisation (in ALMs)	Total registers
One block	23,130 of 234,720 10% on FPGA	5664

Error on wrong placement

The second possibility of sorting data is just warning the user instead of sorting. With no sorting on the FPGA the logic usage shown in Tables 7.2 and 7.3 is reduced by about 10% of logic on the FPGA. Instead of the previously shown sorting algorithm, a simple check is done to guarantee the correct placement of the optical fibres on the AMC40. The first four buses on the AMC40 are then to be used by one FE, the second four by the next FE and so forth. Hence the correct cabling falls to the user.

The cabling is checked on the FPGA by an algorithm, which allows to deactivate single buses. For every four channels the algorithm searches for the first active channel and remembers the FE ID. This ID is then compared in a second step with each of the other FE IDs in this group of four active channels. In addition the correct combination of cable input due to the FPGA ID is also verified. This procedure is visualised with different examples in Figure 7.5. To minimise the use of logic, a limitation is built into this concept. The FEs may only be connected in consecutive optical connections in groups of four. Thus the links "0" to "3" as well as "16" to "19" would be ok to use, but a combination breaking this group of four would not be allowed, e.g. "11" to "14". This limitation is acceptable due to the maximum amount of processable FEs on one AMC40 as it will be discussed in Section 7.2.8. This multiplexer variation is implemented for the use of two and three connected FEs, working with a channel list, which has to be distributed to the block by ECS. Their logic usage is shown in Table 7.4.

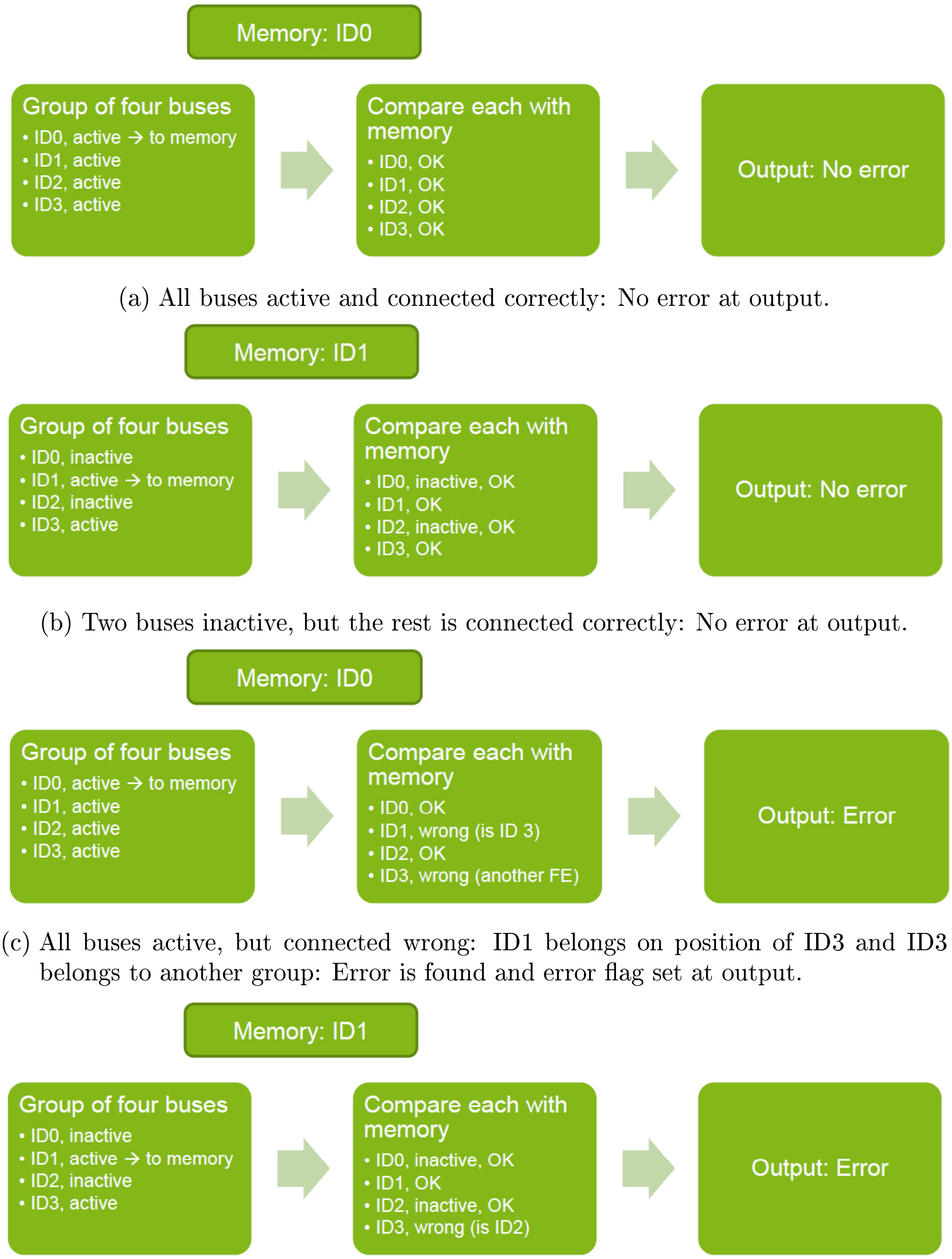
Table 7.4: Usage of logic on the Stratix V 5SGXEA7 for multiplexing paired input buses for two and three FEs. One block is needed for the whole firmware.

Option	Logic utilisation (in ALMs)	Total registers
Two FEs	1,919 of 234,720 <1% on FPGA	1,793
Three FEs	2,701 of 234,720 1% on FPGA	2,689

Table 7.5: Usage of logic on the Stratix V 5SGXEA7 for checking the correct assignment of fibre plugging. One block is needed for each processable FE.

Option	Logic utilisation (in ALMs)	Total registers
One block	310 of 234,720 <1% on FPGA	306

As already mentioned and shown in Table 7.5 this procedure uses less logic than sorting the streams inside the FPGA, although it has the disadvantage of charging the user with the responsibility to wire the optical connections in the correct order.



(a) All buses active and connected correctly: No error at output.

(b) Two buses inactive, but the rest is connected correctly: No error at output.

(c) All buses active, but connected wrong: ID1 belongs on position of ID3 and ID3 belongs to another group: Error is found and error flag set at output.

(d) Two buses inactive and connected wrong: ID3 should be on position of ID2: Error is found and error flag set at output.

Figure 7.5: Decision making procedure of the algorithm warning the user.

7.2.4 Merging hit pattern of consecutive bunch crossings

The most crucial part in data processing of the Outer Tracker to correctly perform tracking, is taking data from three consecutive Bunch Crossings (Bx), due to the relatively high Drift Time (DT) in the straw tubes. With a drift time under 50 ns [32] a hit during Bx0 can be measured in Bx0, Bx1 or Bx2 as Figure 7.6 illustrates.

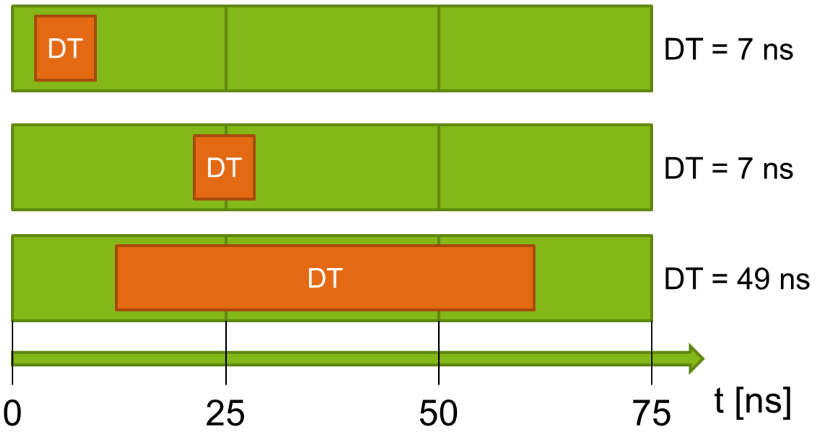


Figure 7.6: Three possibilities of drift time measuring. From top to bottom: A DT of 7 ns measured as 8 ns in Bx0. A 7 ns DT started in Bx0 and measured as 3 ns in Bx1. A 49 ns DT started in Bx0 and measured as 12 ns in Bx2.

Since the FE sends data for every Bx, the data has to be merged on the AMC40 to reduce the network bandwidth usage and CPU time on the DAQ. The data does not only consist of drift times, but also of a hit pattern that has to be merged. Figure 7.7 shows the scheme in which the hit pattern is merged, with Table 7.6 giving some examples. This is done for each bus with a 32 bit wide hit pattern. The incoming hit pattern is coded in a 32 bit wide vector in which every bit indicates a channel on the FE belonging to the FPGA taking the data. Hereby a "0" indicates a channel without a hit, while a "1" indicates a hit in this channel.

Table 7.6: Examples for hit pattern merging results regarding to the algorithm illustrated in Figure 7.7. The merged hit pattern is encoded to the first hit Bx.

Merged	Bx0	Bx1	Bx2
00	0	0	0
01	1	0	1
01	1	1	0
10	0	1	1
11	0	0	1

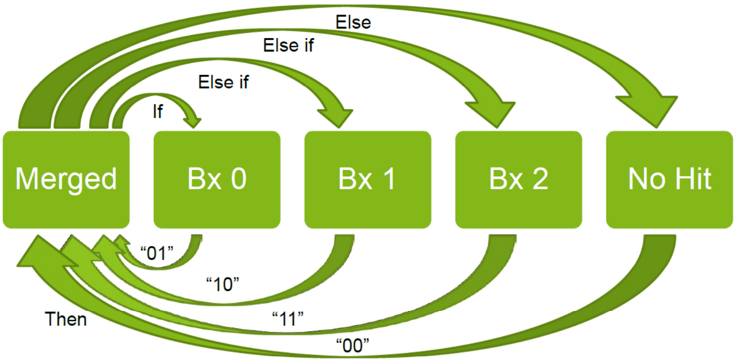


Figure 7.7: Illustration of the hit pattern merging algorithm. For each undefined channel in the merged hit pattern the algorithm checks "if" the channel is hit in the first Bx, if not it checks (`else if`) the second Bx and afterwards the third. If none is hit (`else`) this channel in the merged hit pattern is set to "no hit" ("00").

To get detailed information about the bunch crossing the first indicated hit appeared, the hit pattern is enlarged to a 64 bit wide vector and coded differently. In this 64 bit wide vector two consecutive bits belong to each channel[VI] and are coded to the first Bx a hit was indicated, with "00" as no hit, "01" as a hit in the first Bx, "10" as a hit in the second Bx and "11" as a hit in the third Bx. Since the hit pattern is always 32 bit wide, each channel has its designated bit and merging three of these vectors does not use a lot of logic on the FPGA. An example of this simple if-else statement is given in Figure 7.7 whereas the logic usage is shown in combination with the drift time merging in Table 7.9 on page 56. This is due to the fact that the hit pattern and the drift times are merged inside the same block, so only one query is used.

7.2.5 Merging drift times of consecutive bunch crossings

Due to the zero suppressed data, merging the drift times is not as easy as merging the hit pattern. Their positioning inside the vector with a maximum width of 160 bit is not allocated to the hit channel. Therefore, a connection or sorting has to be established between the hit channel and the zero suppressed DT. Different approaches have been tested to find the most suitable method. Three different methods, including the chosen one, are explained in detail in the following.

[VI] Channel 0: `1 downto 0`. Channel 1: `3 downto 2`. ... Channel 31: `63 downto 62`.

The elegant approach

The most elegant, but also most logic consuming approach would be to do the whole merging algorithm in one single step. The method itself seems very simple, if one would not work with hard routed connections on an FPGA.

For each channel the FPGA had to check the merged hit pattern if the channel was hit or not. If a hit is found, the corresponding drift time would be put to its correct destination from the zero suppressed DT vector of the corresponding Bx to a merged DT vector. The destination should be known to the FPGA by checking the number of hits in the previous channels. The already zero suppressed vector would be built out of the DT in the correct order corresponding to the merged hit pattern.

As simple the approach might look, as difficult it is to be implemented. The variability of possible combinations for each possible hit pattern and counter as well as a lot of multiplying, including the not available dynamic cache on an FPGA made it nearly impossible to use this method. There had to be a solution in logic for every possible combination hard coded on the FPGA. Different hit pattern combined with zero suppressed data create a high amount of possibilities the correct DT had to be picked from.

The attempt was made for a whole FE module, but it was not possible to synthesise this due to the reason of the high logic requirement on the Stratix IV FPGA[VII], therefore this method was dismissed for further tests.

The step-by-step arranging

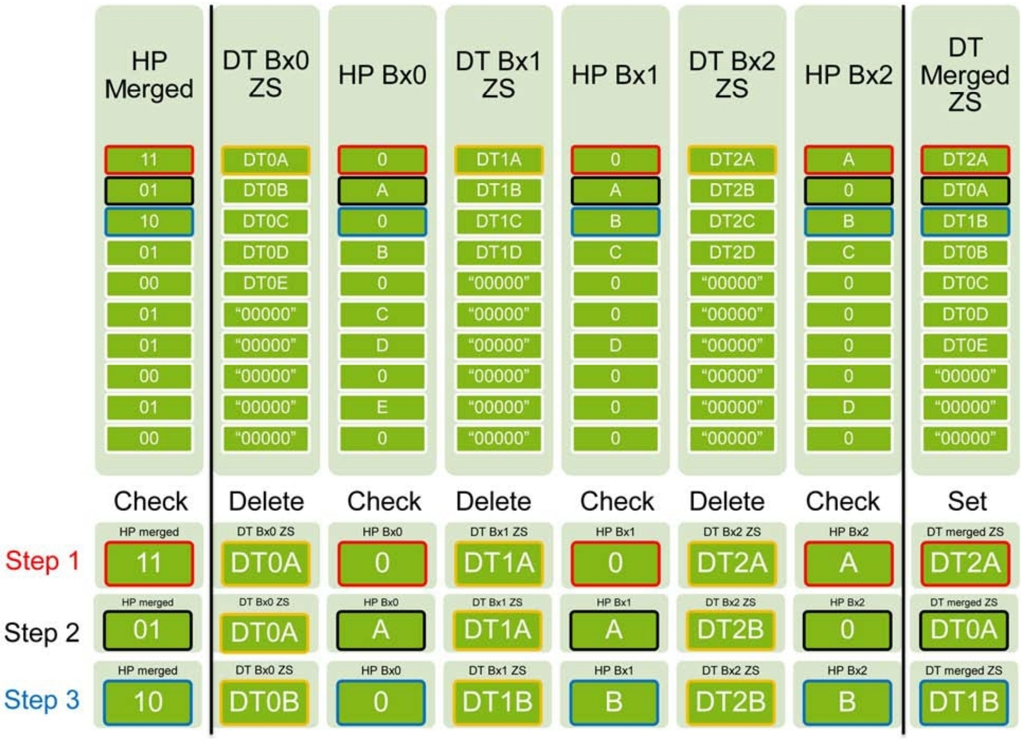
The second promising method is a modification of the "elegant approach". Hereby the arranging method is stretched to 32 steps. In each step a single channel of the merged hit pattern is checked for a hit. If a hit is found, the corresponding drift time is put to the correct designation of the new merged vector. Afterwards the same action is performed for the next channel. Some adaptations have to be made to optimise this method.

To find the correct space for the new drift time to put, a counter has to be carried along, so that each step knows where to put the drift time into the merged vector. The counter starts at "0" in the first step and if a hit was detected it is increased by one. Each following step has to calculate the correct five bit wide position to put the next DT. For example in the first step the DT would be set to the vector positions `4 downto 0`, whereas each other step has to calculate their positions and put the DT to counter $\times 5 + 4$ `downto` counter $\times 4$.

Additionally all three drift time vectors have to be carried along, including another

[VII] The attempt was made on a Stratix IV (EP4SGX230KF40C2).

optimisation to get rid of more counters. Each carried unmerged drift time vector is shifted by five bits. Respectively, the lowest five bits, which represent a single drift time, are deleted, if a hit in the executed channel is found and the following data is shifted to the front of the vector. With this shifting, the drift time for the executed channel is always located in the first five bits of the vector and the executing logic has just three options where to search for the correct drift time. Unfortunately, each carried unmerged vector has to be checked for a hit, so not only the merged hit pattern has to be checked, but each unmerged hit pattern has to be checked to prevent data loss. Figure 7.8 illustrates this approach, which is also discarded. Another simpler approach was developed and found to be even less logic consuming as the step-by-step arranging, whose logic usage is shown in Table 7.7.



HP Merged	DT Bx0 ZS	HP Bx0	DT Bx1 ZS	HP Bx1	DT Bx2 ZS	HP Bx2	DT Merged ZS
11	DT0A	0	DT1A	0	DT2A	A	DT2A
01	DT0B	A	DT1B	A	DT2B	0	DT0A
10	DT0C	0	DT1C	B	DT2C	B	DT1B
01	DT0D	B	DT1D	C	DT2D	C	DT0B
00	DT0E	0	“00000”	0	“00000”	0	DT0C
01	“00000”	C	“00000”	0	“00000”	0	DT0D
01	“00000”	D	“00000”	D	“00000”	0	DT0E
00	“00000”	0	“00000”	0	“00000”	0	“00000”
01	“00000”	E	“00000”	0	“00000”	D	“00000”
00	“00000”	0	“00000”	0	“00000”	0	“00000”

	Check	Delete	Check	Delete	Check	Delete	Check	Set
	HP merged	DT Bx0 ZS	HP Bx0	DT Bx1 ZS	HP Bx1	DT Bx2 ZS	HP Bx2	DT merged ZS
Step 1	11	DT0A	0	DT1A	0	DT2A	A	DT2A
Step 2	01	DT0A	A	DT1A	A	DT2B	0	DT0A
Step 3	10	DT0B	0	DT1B	B	DT2B	B	DT1B

Figure 7.8: Illustration of an example for the step-by-step arranging method. In each step as marked in terms of colour (Step 1: red; Step 2: black; Step 3: blue) the hit pattern (HP) column is checked, indicated by the word “Check”. In the drift time (DT) columns only the yellow marked part is checked and deleted if hit and thereby set to the merged DT column. Hereby ZS stands for zero suppressed.

Table 7.7: Usage of logic on the Stratix V 5SGXEA7 for one step of the step-by-step arranging method. 32 blocks are needed for each FE FPGA, thus 128 are needed for each processable FE.

Option	Logic utilisation (in ALMs)	Total registers
One block	396 of 234,720 <1% on FPGA	713
For one FE	50,688 of 234,720 22% on FPGA	91,296

Working with zero unsuppressed data

The method found to be the least logic consuming is the simplest one in function. The zero suppressed drift time vector is unsuppressed, then merged and afterwards zero suppressed again. In the following each of these steps is explained in detail.

The zero unsuppressing is done in 32 consecutive steps, where in each the actual channel is checked for a hit. If a hit is found, the first drift time in the incoming vector (`5 downto 0`) is put into the designated five bit spot of the unsuppressed vector. In addition the incoming drift time vector is shifted by five bits, so that the used DT is deleted and the following shifted to the front. If no hit is found, this spot is set to "00000" and the incoming drift time vector remains unchanged. Figure 7.9 illustrates the procedure of zero unsuppressing data with a few example steps. This procedure minimises the amount of possibilities the FPGA has to consider and no calculation is needed. Each of these steps works with a simple query, but 32 steps are needed for a whole zero unsuppressing of one FE FPGA. Table 7.8 shows the logic usage for the whole unsuppressing mechanism on the Stratix V.

Table 7.8: Usage of logic on the Stratix V 5SGXEA7 for the zero unsuppressing algorithm. One block is needed for each FE FPGA, thus four are needed for each processable FE.

Option	Logic utilisation (in ALMs)	Total registers
One block	3588 of 234,720 2% on FPGA	7168
For one FE	14,352 of 234,720 6% on FPGA	28,672

Merging the unsuppressed data from three consecutive bunch crossings is then made simultaneous with merging the hit pattern. As it is shown in Figure 7.10 the algorithm checks in which bunch crossing the first hit happened for every channel, builds the hit pattern as already explained, and puts the drift time on the same place in the merged vector. Due to less combinatorial possibilities concerning where to search for the correct drift time, less logic on the FPGA is used in comparison to the other methods. At this point a non zero suppressed vector is built with the merged data of three consecutive bunch crossings. The combinatorial amount of used logic for merging the drift times and the hit pattern for a 32 bit wide vector is shown in Table 7.9.

Table 7.9: Usage of logic on the Stratix V 5SGXEA7 for merging the hit pattern and the unsuppressed drift times. Four blocks are needed for each processable FE.

Option	Logic utilisation (in ALMs)	Total registers
One block	161 of 234,720 <1% on FPGA	256
For one FE	644 of 234,720 <1% on FPGA	1,024

The last step is zero suppression of the data. In this case the zero suppression is performed in four steps taking only four clock cycles to finish, instead of 32 clock cycles as it is done on the FE. The vectors are split into eight equal parts, with the length of four channels (8 bit wide hit pattern) and their corresponding drift times (40 bit wide drift time vector). In the first step each of the parts is checked for one of the 16 possible hit constellations as indicated in Table 7.10 and zero suppressed by picking and sorting the data from the drift time vector as Figure 7.11 illustrates. Afterwards two split vectors are merged to one by carrying a hit counter, which indicates the position the vectors have to be merged to remain zero suppressed. Figure 7.12 shows an example for this repeating procedure working with increasing vector widths. At the end of this chain a zero suppressed vector containing all DTs and a counter representing the number of hits in this vector are available for further processing.

The logic usage on the Stratix V FPGA is shown in Table 7.11 for the whole zero suppressing algorithm. The “working with zero unsuppressed data” method is preferred, since it is the least logic consuming of the three presented ones.

Table 7.10: Hit possibilities with four channels. Due to the merged coding ("01", "10", "11" for a hit; "00" for no hit) the "0", indicating no hit, in this table is coded as `="00"`, whereas a hit is retrieved as `/="00"` (unequal "00").

No. of Hits	Possibilities
0	0000
1	1000, 0100, 0010, 0001
2	1100, 1010, 1001, 0110, 0101, 0011
3	1110, 1101, 1011, 0111
4	1111

Table 7.11: Usage of logic on the Stratix V 5SGXEA7 for the zero suppression, including merging the drift time vectors. Four blocks are needed for each processable FE.

Option	Logic utilisation (in ALMs)	Total registers
One block	1,861 of 234,720 $<$1% on FPGA	1,772
For one FE	7,444 of 234,720 3% on FPGA	7,088

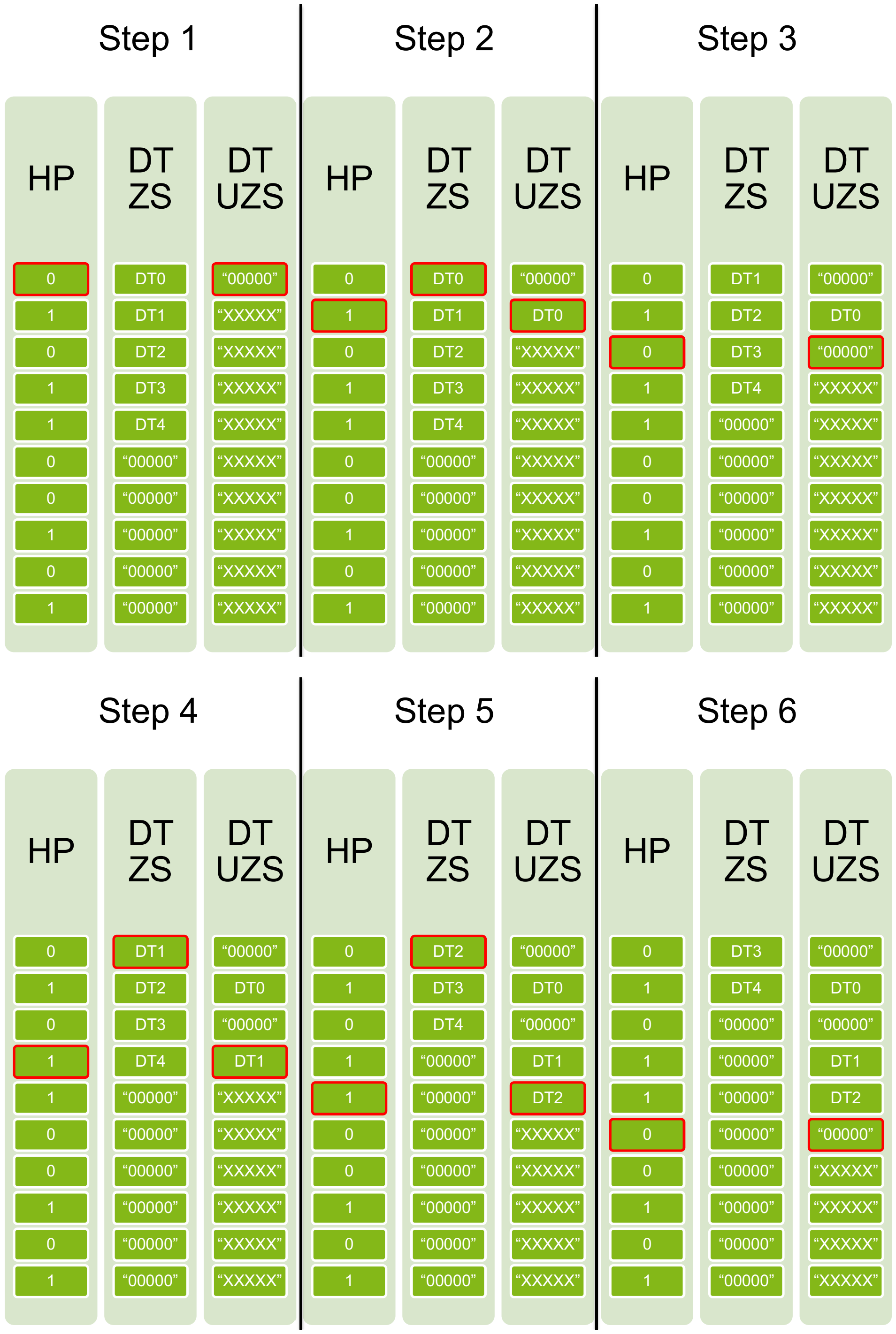
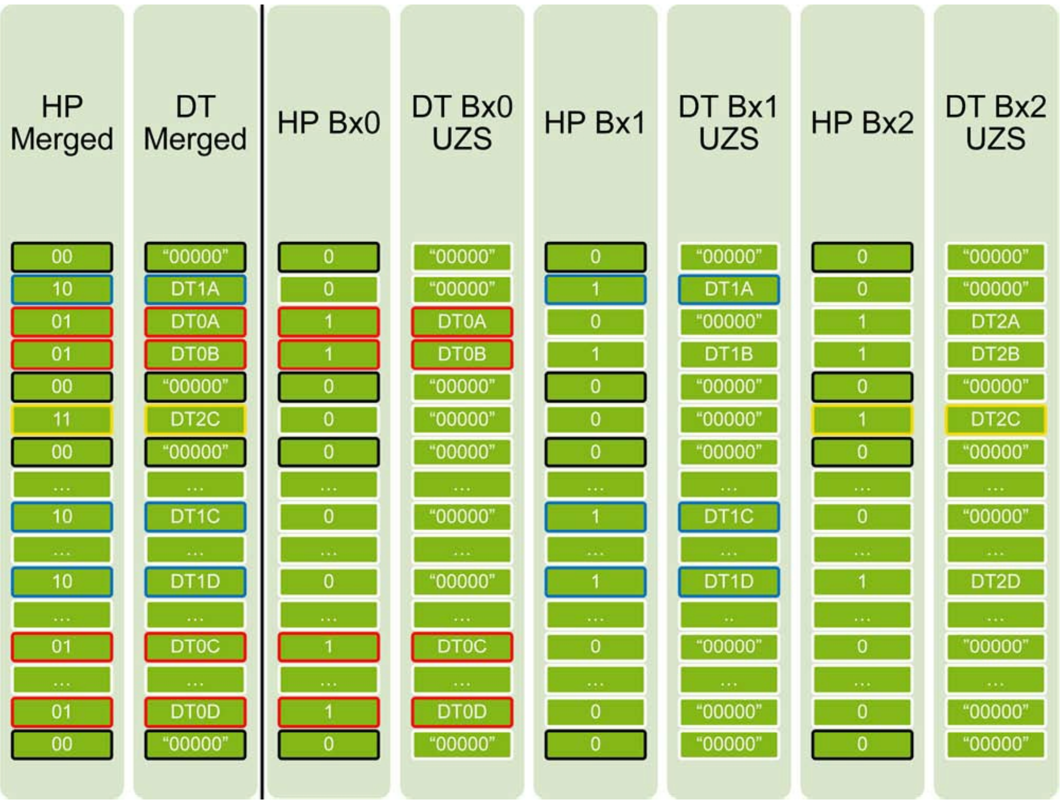

Step 1
HP
0 1 0 1 1 0 0 1 0 1
DT ZS
DT0 DT1 DT2 DT3 DT4 “00000” “00000” “00000” “00000” “00000”
DT UZS
“00000” “XXXXX” “XXXXX” “XXXXX” “XXXXX” “XXXXX” “XXXXX” “XXXXX” “XXXXX” “XXXXX”
Step 2
HP
0 1 0 1 1 0 0 1 0 1
DT ZS
DT0 DT1 DT2 DT3 DT4 “00000” “00000” “00000” “00000” “00000”
DT UZS
“00000” DT0 “XXXXX” “XXXXX” “XXXXX” “XXXXX” “XXXXX” “XXXXX” “XXXXX” “XXXXX”
Step 3
HP
0 1 0 1 1 0 0 1 0 1
DT ZS
DT1 DT2 DT3 DT4 “00000” “00000” “00000” “00000” “00000” “00000”
DT UZS
“00000” DT0 “00000” “XXXXX” “XXXXX” “XXXXX” “XXXXX” “XXXXX” “XXXXX” “XXXXX”
Step 4
HP
0 1 0 1 1 0 0 1 0 1
DT ZS
DT1 DT2 DT3 DT4 “00000” “00000” “00000” “00000” “00000” “00000”
DT UZS
“00000” DT0 “00000” DT1 “XXXXX” “XXXXX” “XXXXX” “XXXXX” “XXXXX” “XXXXX”
Step 5
HP
0 1 0 1 1 0 0 1 0 1
DT ZS
DT2 DT3 DT4 “00000” “00000” “00000” “00000” “00000” “00000” “00000”
DT UZS
“00000” DT0 “00000” DT1 DT2 “XXXXX” “XXXXX” “XXXXX” “XXXXX” “XXXXX”
Step 6
HP
0 1 0 1 1 0 0 1 0 1
DT ZS
DT3 DT4 “00000” “00000” “00000” “00000” “00000” “00000” “00000” “00000”
DT UZS
“00000” DT0 “00000” DT1 DT2 “00000” “XXXXX” “XXXXX” “XXXXX” “XXXXX”

Figure 7.9: Six steps of the zero unsuppressing mechanism. Each step checks the assigned position inside the hit pattern (HP), marked red. If no hit is found the same five bit wide position in unsuppressed drift time vector (DT UZS) is set to `00000`. If a hit is found, always the first five bits of the zero suppressed drift time vector (DT ZS) are set to the five bit wide position the step works at and the DT ZS is then shifted by five bits.



HP Merged	DT Merged	HP Bx0	DT Bx0 UZS	HP Bx1	DT Bx1 UZS	HP Bx2	DT Bx2 UZS
00	“00000”	0	“00000”	0	“00000”	0	“00000”
10	DT1A	0	“00000”	1	DT1A	0	“00000”
01	DT0A	1	DT0A	0	“00000”	1	DT2A
01	DT0B	1	DT0B	1	DT1B	1	DT2B
00	“00000”	0	“00000”	0	“00000”	0	“00000”
11	DT2C	0	“00000”	0	“00000”	1	DT2C
00	“00000”	0	“00000”	0	“00000”	0	“00000”
...	...	...	...	...	...	...	...
10	DT1C	0	“00000”	1	DT1C	0	“00000”
...	...	...	...	...	...	...	...
10	DT1D	0	“00000”	1	DT1D	1	DT2D
...	...	...	...	...	..	...	...
01	DT0C	1	DT0C	0	“00000”	0	“00000”
...	...	...	...	...	...	...	...
01	DT0D	1	DT0D	0	“00000”	0	“00000”
00	“00000”	0	“00000”	0	“00000”	0	“00000”

Figure 7.10: The merging mechanism for zero unsuppressed drift times. If a hit is indicated in one of the hit patterns (HP Bx0/1/2) the drift time inside the drift time vector (DT Bx0/1/2 UZS) is set to the merged drift time vector (DT merged) and the hit pattern is merged as explained in Section 7.2.4 inside the merged hit pattern (HP Merged), visualised in red ("01"), blue ("10") and yellow ("11"). If no hit is indicated the five bit wide space is set to 00000, visualised in black ("00").

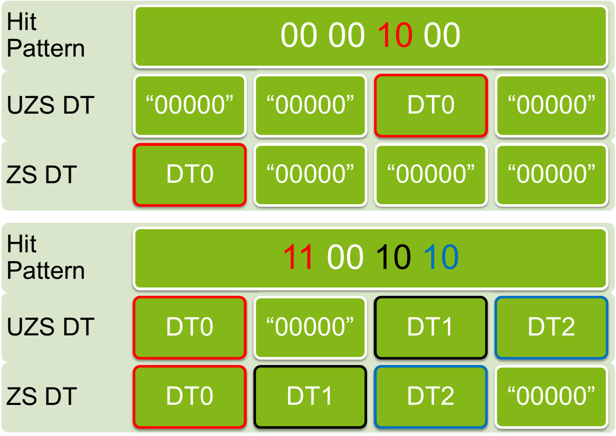


Figure 7.11: Two examples for the first zero suppressing step. Depending on the hit pattern checked as indicated in Table 7.10 the drift times are shifted to the front of the vector, whereas the rest is filled with zeroes.

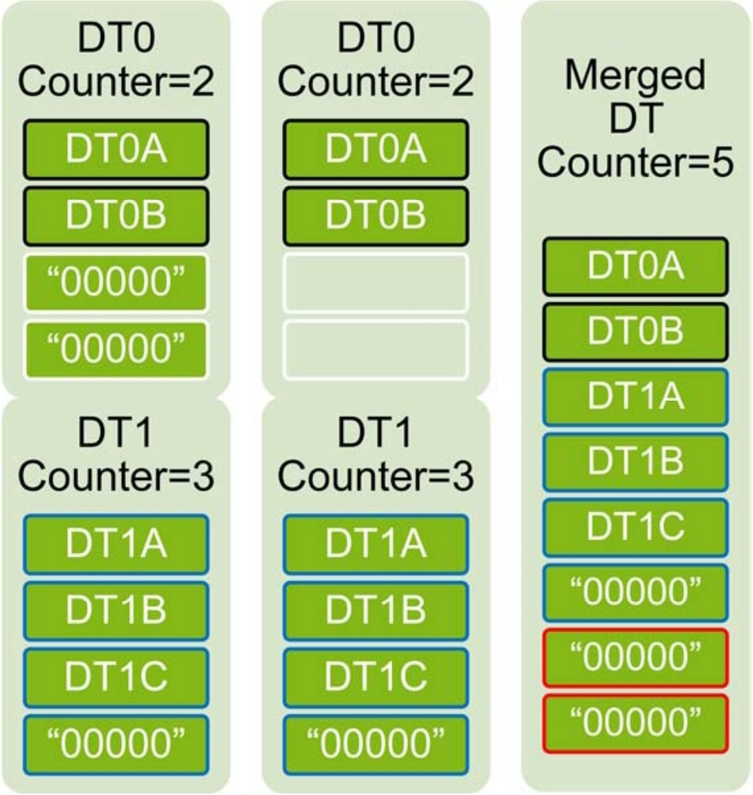
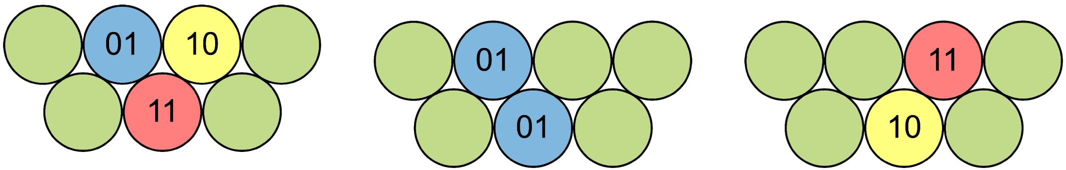


Figure 7.12: Example for merging zero suppressed drift time vectors. The vectors `DT0` and `DT1` are merged. Therefore the hit counter of `DT0` is checked and any content from position "2" onward (white) replaced by the vector `DT1` (blue). The rest of the vector is filled with zeroes (red).

7.2.6 Clustering

An additional function in the AMC40 firmware for the Outer Tracker is finding clusters of hits over three bunch crossings. The clustering algorithm searches for hits in neighbouring channels representing possible particle tracks.

Since an OT module is built with two layers of straw tubes and positioned on the LHCb x-y axis, a particle crosses the module by hitting both layers as already explained in Chapter 4. A cluster is defined as a hit in one front layer straw tube and a hit in at least one adjacent straw tube of the rear layer. Figure 7.13 illustrates the possibilities of cluster building across three consecutive bunch crossings, where every labeled colour indicates one bunch crossing.



(a) One hit in Bx2 (red) in the front layer and two hits in adjacent straw tubes in the back layer in Bx0 (blue) and Bx1 (yellow).

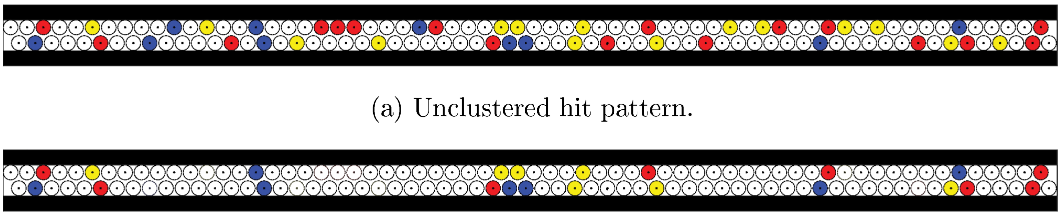
(b) One hit in Bx0 (blue) in the front layer and one hit in an adjacent straw tube in the back layer in Bx0 (blue).

(c) One hit in Bx1 (yellow) in the front layer and one hit in an adjacent straw tube in the back layer in Bx2 (red).

Figure 7.13: Three possibilities for clusters. Each of the three shown pattern is accepted as a cluster. The Bx are interchangeable.

This simple causal connection is checked with the merged hit pattern during a single clock cycle. Thereby each channel in the front layer is checked for a coincidence with the two adjacent in the back layer, e.g. channel 5 and channels 121 and 122. An example for a merged unclustered hit pattern and a clustered one is given in Figure 7.14. The clustering mechanism is positioned directly after drift time merging and an optional feature. Two parameters are embedded to distinguish between different working options, to be able to configure the clustering. These parameters and their impact are explained in Table 7.12.

Such a clustering algorithm is not necessary for standard data processing, but a strong relief to the DAQ where this algorithm would be done instead. The changeless channel comparing with new data every clock cycle is predestined to be done on an FPGA. Additionally, the logic usage on the Stratix V is very low, as shown in Table 7.13. Therefore, the clustering is a valuable addition to the Outer Tracker data processing firmware.



(a) Unclustered hit pattern.

(b) Clustered hit pattern.

Figure 7.14: A merged hit pattern before (unclustered) and after clustering (clustered). The different colours indicate different bunch crossings: Bx0 ("01", blue), Bx1 ("10", yellow), Bx2 ("11", red). In the clustered hit pattern every hit outside a cluster is deleted.

Table 7.12: The parameters and their impact as used for clustering.

Parameter	Status	Impact
cluster_onoff	On = '1'	Clustered pattern is used for zero suppression.
cluster_onoff	Off = '0'	Unclustered pattern is used for zero suppression.
add_ pattern	On = '1'	The pattern not used for zero suppression is added to the output vector.
add_ pattern	Off = '0'	Only the pattern used for zero suppression is put into the output vector.

Table 7.13: Usage of logic on the Stratix V 5SGXEA7 for clustering. One block is needed for each processable FE.

Option	Logic utilisation (in ALMs)	Total registers
One block	723 of 234,720 $<$1% on FPGA	1253

7.2.7 Output formatting

The last step in the data processing block is to format the output vector. The data inside the data processing block is split into different vectors and additional information is produced. An output vector is generated for each FE FPGA, to avoid unnecessary long vectors. The maximum length for each vector is 315 bit (`314 downto 0`). It consists of a header, the BxID, hit patterns and drift times, aligned as shown in Table 7.14.

Table 7.14: The format for an output vector of one FE FPGA with a maximum length of 315 bit.

Content	Position	Length in bit
Header	`8 downto 0`	9
BxID	`16 downto 9`	8
Position	`26 downto 17`	10
First hit pattern	`90 downto 27`	64
Second hit pattern	`154 downto 91`	64
Drift times	`314 downto 155`	160

The header is the same as the one sent by the FE extended by one bit. This bit can be used as a general error bit and set with each possible error flag inside the data processing and is not yet finally defined. Another option is to use this bit as an indicator for the amount of hit patterns inside the output vector. The final use of this bit or even using more than one can easily be adapted.

Although the output is filled with the merged data, only one BxID is assigned to each vector. This BxID refers to the first bunch crossing inside the merged data and the data includes the drift times of the following two bunch crossings.

Following the BxID a ten bit wide code is integrated, defining the position of the FE and FPGA the data was taken. This position was sent by the FE during the sync-mode and is sent out with every event data.

Two hit patterns are then arranged after the position, the first one is the hit pattern used for zero suppressing. The second one is an optional pattern and can be set via the clustering options. Thus, the first hit pattern could be the one delivered by the FE and the second one the clustered in addition or vice versa. Another option is to only send the first hit pattern and shift the following drift time positions by 64 bit to the front.

The last part of the output vector is filled with zero suppressed drift times. The maximum length of this part is 160 bit, referring to 32 possible hits times five bit for each hit. Since the data is zero suppressed all drift times are positioned on the front of this vector part and the rest is filled with zeroes.

In addition to this output vector two pieces of information are provided as detached vectors. The first one is a simple copy of the eight bit coded BxID to spare the user and following blocks to search for it inside the output vector. The second one is a nine bit coded counter which indicates the occupancy of the output vector. The value of this vector indicates the last bit of the output vector filled with data. After this bit the output is filled with zeroes. This information can be used in MEP-building to compress the data and save bandwidth to the DAQ.

The presented output vector is not fixed in this status. Due to a not yet defined data format needed after the data processing, the format of the vector can be changed or varied easily in code. Since the merging mechanism is very simple, the logic usage on the FPGA is very small (see Table 7.15).

Table 7.15: Usage of logic on the Stratix V 5SGXEA7 for formatting the output vector. One block is needed for each optical transceiver input line.

Option	Logic utilisation (in ALMs)	Total registers
One block	179 of 234,720 <1% on FPGA	334
For one FE	716 of 234,720 <1% on FPGA	1336

7.2.8 Complete design

The complete Outer Tracker data processing firmware design includes every presented part and a few additional error handling blocks. Since the usage in logic is a crucial point on the AMC40, due to the high amount of common logic in the whole design, only a small not yet defined part is available for the data processing firmware. The first estimate on the available logic for data processing is about 20 %–30 % [48]. Due to that, different structure options for the firmware are considered and will be explained in the following.

Reasoned variations are a different number of FEs being connected to one AMC40 as well as sorting the data streams by software or by reporting an alignment error, as already mentioned in Section 7.2.3, and using the possibility to send data always not zero suppressed, which would get rid of the 32 clock cycle long unsuppressing mechanism. With this constraints in mind seven combinations were reviewed:

- Six FEs connected with software sorting
- Six FEs connected without software sorting
- Five FEs connected without software sorting

- Four FEs connected without software sorting
- Three FEs connected without software sorting
- Two FEs connected without software sorting
- Three FEs connected without software sorting and with receiving zero unsuppressed data

The first variation made is to replace the software sorting algorithm with mentioning a wrong connection to the user. Both designs use all 24 input lines which would correspond to a maximum of six connected FEs. Table 7.16 shows the comparison between those two options. Replacing the software sorting reduces the logic utilisation by about 12%. Since the fibre connection is done only once, if no error occurs, the error mentioning mechanism is preferred. If it is possible to spare logic by bearing the user with more responsibility this option should be chosen and will be used in further variations.

Table 7.16: Usage of logic on the Stratix V 5SGXEA7 for the options with six FEs and a variation of the sorting algorithm. Replacing the software sorting algorithm with just mentioning a connection error uses 12% less logic on the FPGA.

Option	Logic utilisation (in ALMs)	Total registers
Six FEs, sorting	178,976 of 234,720 76% on FPGA	273,598
Six FEs, no sorting	149,701 of 234,720 64% on FPGA	267,930

To decrease the usage of logic on the FPGA even further, the design will be tested with less processable FEs. Table 7.17 shows the tested variation from five down to two FEs. The design has not been built containing one processable FE only. This decision was made by observing a linear function in logic usage during the synthesis for the higher amount of used FEs and the fact that requiring one AMC40 per FE would boost the cost factor for the then needed amount of TELL40 setups. The observed linear increase is shown in Figure 7.15, where a linear fit results in the following usage:

$$\begin{aligned}\text{ALMs}(FE) &= 24322.9\,\text{ALMs/FE} + 3394.6\,\text{ALMs}\\ \text{Registers}(FE) &= 44033.7\,\text{Registers/FE} + 3369.2\,\text{Registers}\end{aligned}$$

With this result the logic usage for one FE would be about 12% of the FPGAs logic.

Table 7.17: Usage of logic on the Stratix V 5SGXEA7 using less FEs than in Table 7.16 for the option without software sorting or rather reporting an error on wrong placement. The design with only two used FEs needs about 22% of the FPGAs logic, which would correlate to the assumed range.

Option	Logic utilisation (in ALMs)	Total registers
Five FEs	124,752 of 234,720 53% on FPGA	223,282
Four FEs	99,804 of 234,720 43% on FPGA	178,634
Three FEs	77,423 of 234,720 33% on FPGA	136,543
Two FEs	51,751 of 234,720 22% on FPGA	91,131

Since 22% logic usage for two processable FEs could still be at the edge of the available space, another option was tested. Since one of the most logic requiring functions is unsuppressing the data before merging it, the option of receiving zero unsuppressed data is tested. The design is built to be able to process zero unsuppressed data for debugging and other options, as it is requested in [37]. If the data would be sent zero unsuppressed by the FEs it would double the bandwidth and therefore require twice the amount of optical fibres between the FE and the AMC40, thus a maximum of three FEs could be connected to one AMC40. This option could in principle be the same as the design options already mentioned, but get rid of the unsuppressing mechanism. The synthesis for this method has been performed with three connected FEs and the result is shown in Table 7.18. Only using about 16% of the logic on the Stratix V would definitly allow each AMC40 to process three connected FEs using all 24 optical inputs. So a double of the amount of optical fibres would halve the amount of used logic on the FPGA.

Table 7.18: Usage of logic on the Stratix V 5SGXEA7 for the whole firmware design processing three FEs receiving zero unsuppressed data.

Option	Logic utilisation (in ALMs)	Total registers
Three FEs	37,290 of 234,720 16% on FPGA	56,287

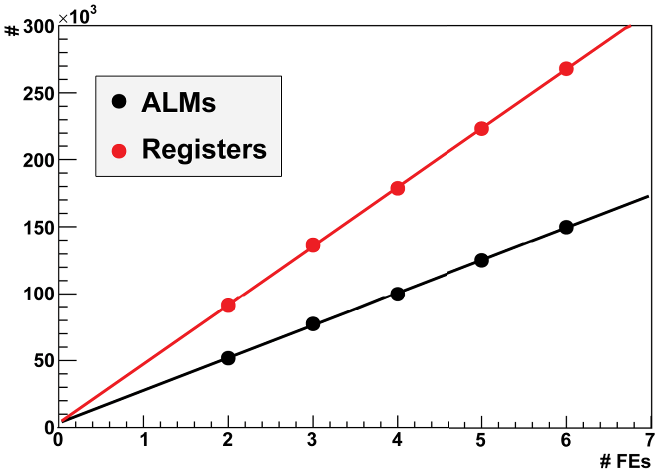


Figure 7.15: Linear fit for the usage of logic on the Stratix V 5SGXEA7 for the designs without sorting.

7.2.9 Simulation

The simulation of the described parts was done with ModelSim[VIII] 6.6g on logic level. So just the raw logic function was simulated and checked. For each block a designated fixed pattern was built and inserted into the logic. The results have been checked and no error was found.

The whole firmware for two FEs without the sorting algorithm and was also simulated with random generated data. This simulation was done for every possible running condition. Figure 7.16 shows a simulation with zero suppressed data input ("nzs_mode" low) for one FE. The data is clustered ("cluster_onoff" high) and the clustered hit pattern is used for zero suppressing of the output drift times ("cluster_hitmap" high). The not clustered hit pattern is added to the output vector ("add_pattern" high).

As an example for the functionality of the data processing firmware, the data from the processed bunch crossing shown in Figure 7.16 at 3240.000 ps will be discussed in detail. Each FE is separated into four quarters, which will be processed altogether and put out in four quarters again. To be able to merge the data from three consecutive Bx, the data of the three processed Bx is shown in the simulation. Each consists of a header, a BxID, a hit pattern and zero suppressed drift times as shown in Table 7.19, whereas the output is formatted as already explained in Section 7.2.7.

The input data for the simulated step is visualised in Figure 7.17, whereas the drift times can be found in the Appendix. Figure 7.17 shows the correct functioning of

[VIII] ModelSim is a simulation software provided by MentorGraphics.

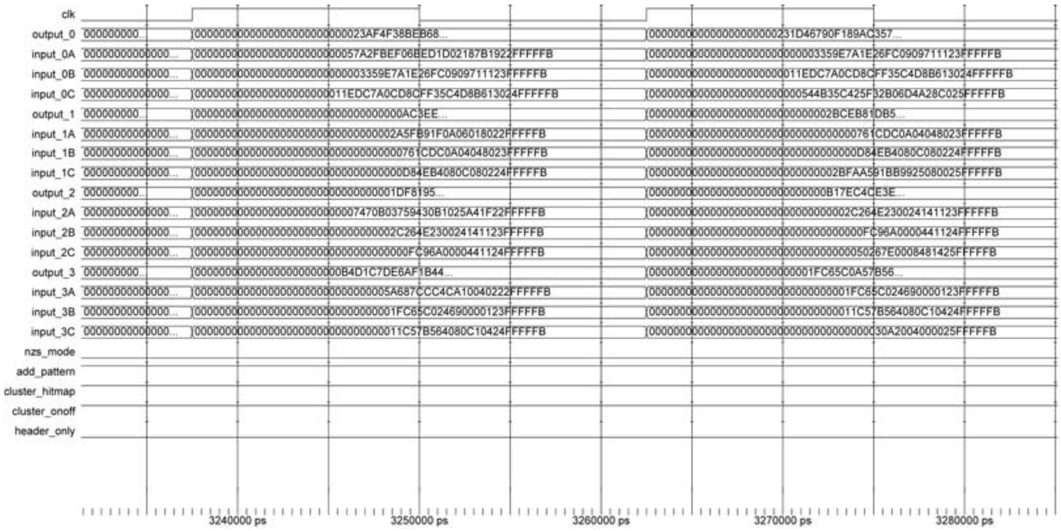

clk
output_0
000000000...
000000000000000000000000000023AF4F38BEB68...
00000000000000000000000231D46790F189AC357...
input_0A
0000000000000000...
00000000000000000000000000057A2FBEF06BED1D02187B1922FFFFFB
0000000000000000000000000003359E7A1E26FC0909711123FFFFFB
input_0B
0000000000000000...
0000000000000000000000000003359E7A1E26FC0909711123FFFFFB
000000000000000000000000011EDC7A0CD8CFF35C4D8B613024FFFFFB
input_0C
0000000000000000...
000000000000000000000000011EDC7A0CD8CFF35C4D8B613024FFFFFB
0000000000000000000000000544B35C425F32B06D4A28C025FFFFFB
output_1
000000000...
000000000000000000000000000000000000AC3EE...
0000000000000000000000000000002BCEB81DB5...
input_1A
0000000000000000...
00000000000000000000000000000000002A5FB91F0A06018022FFFFFB
0000000000000000000000000000000000761CDC0A04048023FFFFFB
input_1B
0000000000000000...
0000000000000000000000000000000000761CDC0A04048023FFFFFB
00000000000000000000000000000000000D84EB4080C080224FFFFFB
input_1C
0000000000000000...
00000000000000000000000000000000000D84EB4080C080224FFFFFB
000000000000000000000000000000002BFAA591BB9925080025FFFFFB
output_2
000000000...
0000000000000000000000000000000001DF8195...
00000000000000000000000000000B17EC4CE3E...
input_2A
0000000000000000...
000000000000000000000000000007470B03759430B1025A41F22FFFFFB
000000000000000000000000000000002C264E230024141123FFFFFB
input_2B
0000000000000000...
000000000000000000000000000000002C264E230024141123FFFFFB
00000000000000000000000000000000000FC96A0000441124FFFFFB
input_2C
0000000000000000...
00000000000000000000000000000000000FC96A0000441124FFFFFB
000000000000000000000000000000000000050267E0008481425FFFFFB
output_3
000000000...
0000000000000000000000000000B4D1C7DE6AF1B44...
000000000000000000000000001FC65C0A57B56...
input_3A
0000000000000000...
0000000000000000000000000000000005A687CCC4CA10040222FFFFFB
00000000000000000000000000000000001FC65C024690000123FFFFFB
input_3B
0000000000000000...
00000000000000000000000000000000001FC65C024690000123FFFFFB
000000000000000000000000000000000011C57B564080C10424FFFFFB
input_3C
0000000000000000...
000000000000000000000000000000000011C57B564080C10424FFFFFB
000000000000000000000000000000000000000030A2004000025FFFFFB
nzs_mode
add_pattern
cluster_hitmap
cluster_onoff
header_only
3240000 ps
3250000 ps
3260000 ps
3270000 ps
3280000 ps

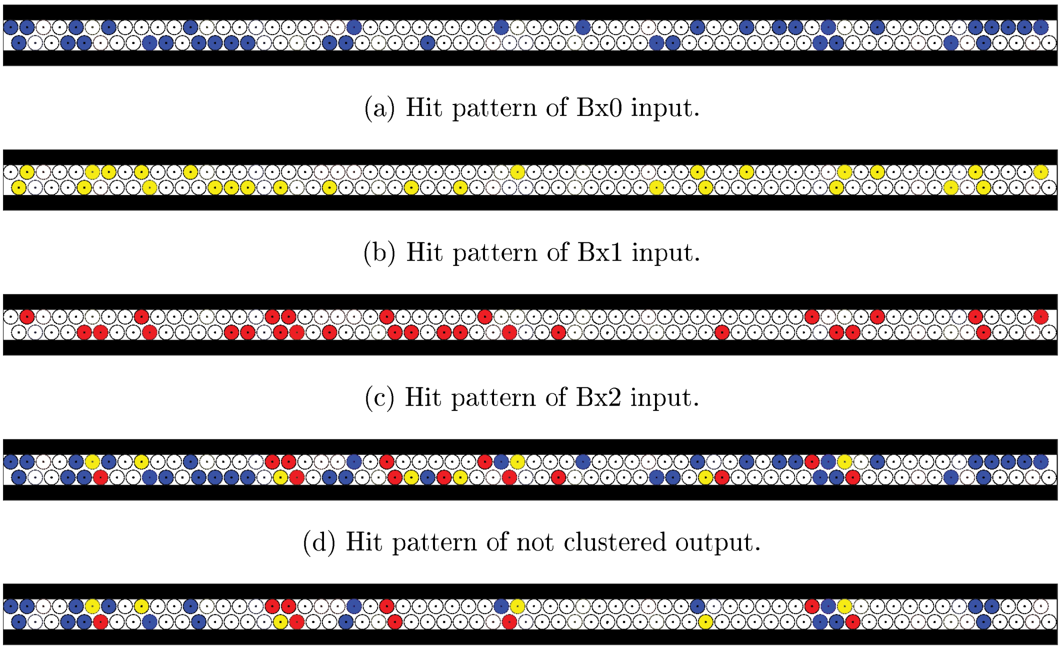
Figure 7.16: An extract of the ModelSim data processing simulation with random generated data. the outputs “0” to “3” indicate the processed data for a quarter of one FE. The inputs belong to the same FE quarter and the letters “A”, “B” and “C” indicate consecutive bunch crossings. The signal “clk” represents the clock driving the logic, whereas the other signals indicate the adjustable running conditions.

Content	Position	Length
Header	23 downto 0	24 bit
BxId	31 downto 24	8 bit
Hit pattern	63 downto 32	32 bit
Drift times	223 downto 64	160 bit

Table 7.19: Content of the input vector in the discussed simulation.

the data processing firmware for every function besides the sorting and drift time merging. The hit pattern is merged and clustered and the firmware works as it is programmed for the running condition simulated. Every other possible combination was also tested and no errors were found, including the “header_only” mode, which only processes the header of each Bx and ignores anything else.

The properness of the drift time merging was also checked and no errors were found. This can be rechecked in the Appendix. The sorting mechanism was also simulated, with direct pattern input, and working without failure as explained in Section 7.2.3.



(a) Hit pattern of Bx0 input.

(b) Hit pattern of Bx1 input.

(c) Hit pattern of Bx2 input.

(d) Hit pattern of not clustered output.

(e) Hit pattern of clustered output.

Figure 7.17: Visualised hit patterns as indicated in Figure 7.16. The three input BxIDs are correctly merged into the merged output hit pattern in Figure 7.17d, whereas the clustering mechanism generated a correct clustered hit pattern output in Figure 7.17e.

7.2.10 Bandwidth consideration

With the whole firmware working and able to process data with a detector occupancy of up to 100 %, the capability of receiving and transmitting the data on and off the AMC40 has to be considered.

As already mentioned an upgraded front end will send data via four optical links to the TELL40. Depending on the occupancy in the detector the bandwidth usage varies as shown in Figure 7.18. The theoretical input rate of the AMC40 equals the theoretical output rate of about 120 Gb/s[IX]. But since the data is processed inside the AMC40 OT data processing firmware the output is higher than the input. As Figure 7.18 illustrates, a simple output with only one hit pattern and zero suppressed data exceeds the input value in about 6 Gb/s, while the addition of a second hit pattern exceeds this value by 10 Gb/s.

To take the output limit into account, the used output bandwidth for each amount

[IX] The input rate splits into 24 5 Gb/s optical links, while the output rate splits into twelve 10 Gb/s ethernet connections.

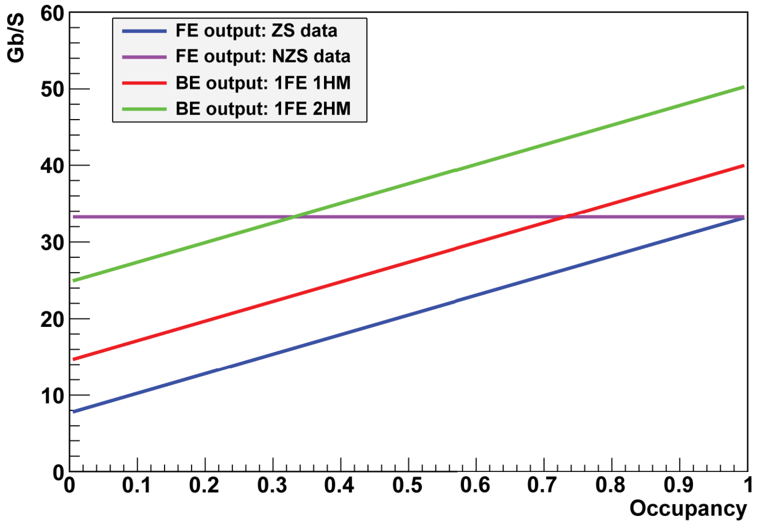


Figure 7.18: A comparison between the detector occupancy and the bandwidth usage on front end (FE) and back end (BE). The FE data output with an occupancy of 100 % equals the bandwidth usage of not zero suppressed data (NZS). The addition of a second hit pattern (2HM) adds an offset of about 10 Gb/s.

of front ends, with the highest offset, dependent on the detector occupancy is shown in Figure 7.19.

Up to two processed front ends it is possible to send processed data up to an occupancy of 100 % out of the AMC40. Any higher amount of processed front ends would have to truncate the data above an average occupancy, which would be about 60 % with three processed front ends and about 20 % with four processed front ends, respectively. With five or six processed front ends the offset itself would exceed the available output bandwidth.

At this point the LLT has to be considered, which could decrease the output as already mentioned in Section 7.1. With an LLT decision rate of 80 % an occupancy of up to 100 % can be send out for three processed front ends, whereas an LLT decision rate of 60 % is needed to send out data for six processed front ends and a detector occupancy of up to about 34 %, as shown in Figure 7.20. Though this average occupancy would be higher than the OT detector occupancy during the first LHC data taking period [32].

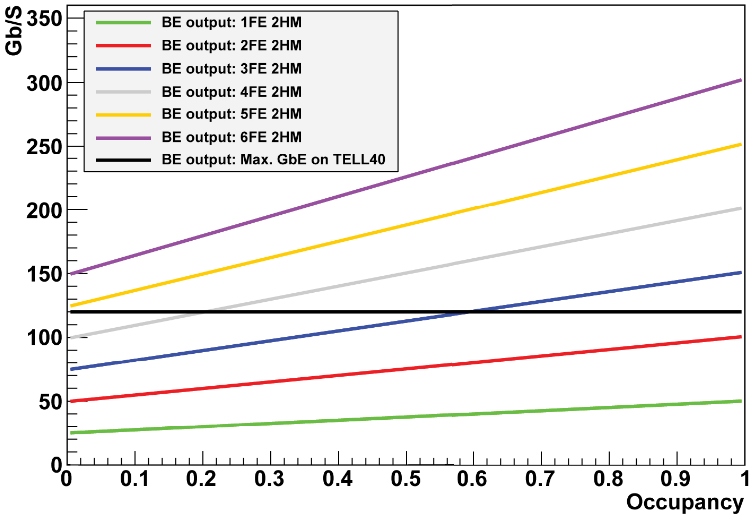


Figure 7.19: The output bandwidth usage of an AMC40 with a different amount of front ends processed. The black line (Max. GbE) marks the maximum of usable output bandwidth on an AMC40. The other lines represent an amount of front ends (1FE-6FE) with a two hit pattern data scheme (2HM).

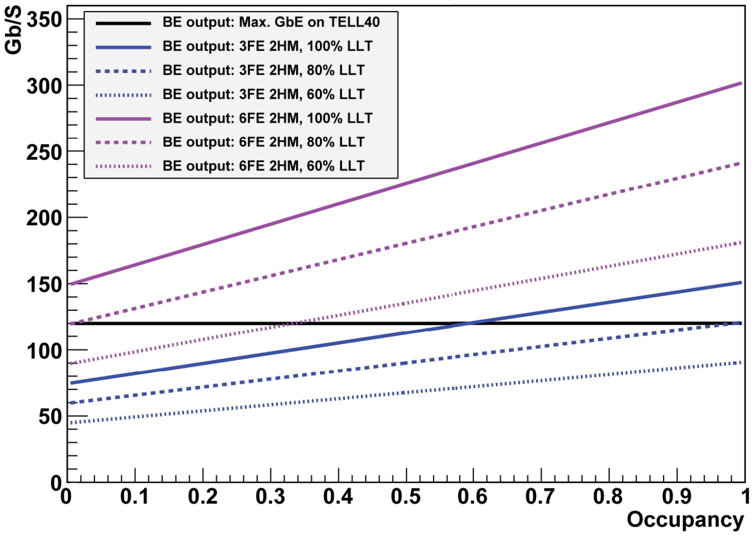


Figure 7.20: The output bandwidth usage for three (3FE) and six (6FE) processed front ends, with an LLT decision rate of 100 % (100 % LLT), 80 % (80 % LLT) and 60 % (60 % LLT) of output data. In addition the maximum of available output bandwidth on an AMC40 is shown (Max. GbE).

8 Test systems

A finished firmware has to be tested on the FPGA to detect errors not discovered during simulation. Since the final hardware for the back end electronics is not yet available, test systems have been developed to work with. Two test systems will be presented in the following: The MiniDAQ and a combination of a commercial FPGA development kit and the SantaLuz board developed during this thesis.

8.1 The MiniDAQ

The MiniDAQ, as shown in Figure 8.1, is a development of a group at the Centre for Particle Physics of Marseilles (CPPM) [46]. It consists of an AMC40 prototype connected to a carrier board and is able to use six of the 24 optical connections. The low level interface and the common firmware for the MiniDAQ are still under development. Additionally, the MiniDAQ itself is only available in small numbers and with a not yet finished prototype AMC40 board. Therefore, this test system is not used to verify the functionality of the Outer Tracker data processing firmware presented in Chapter 7.

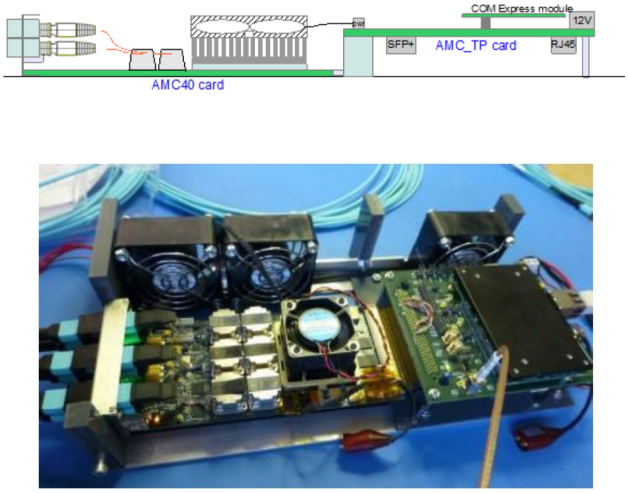


Figure 8.1: A picture of the MiniDAQ, consisting of an AMC40 prototype and a carrier board [49].

8.2 The SantaLuz board

Another already available and relatively simple possibility to test firmware is to use an already existing commercial FPGA development kit and add some extensions. In this case the Stratix IV GX development kit DK-DEV-4SGX230N is extended by the SantaLuz board developed during this thesis.

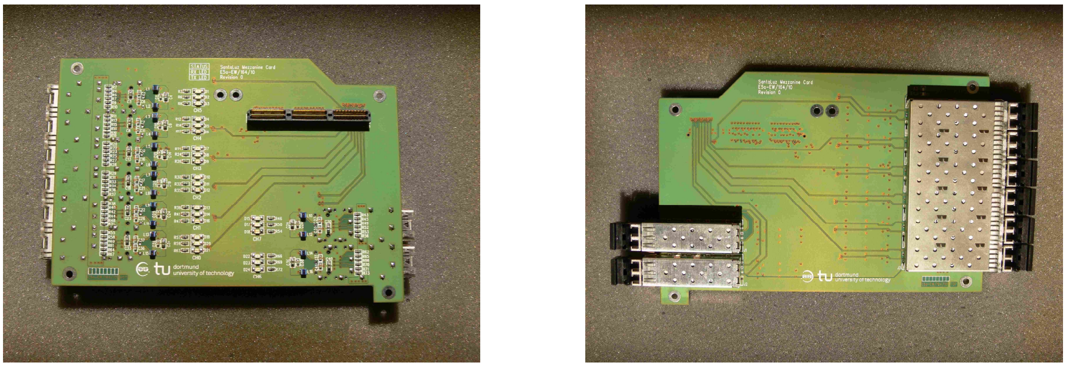


Figure 8.2: The SantaLuz board. On the left the boards front can be seen, with its HSMC connector and the status LEDs. On the right the backside is shown with the cages equipped with optical transceivers.

The SantaLuz board, shown in Figure 8.2, is a HSMC. Its main purpose is to provide the commercial FPGA board with eight high speed optical transceivers via an HSMC connector cable. The SantaLuz board is routed to guarantee faultless data transmission up to 5 Gb/s, but higher rates are also possible. Besides cages for optical transceivers it is equipped with status LEDs accessible via the HSMC connection.

The reliability of the board has been tested with a Bit Error Rate Test (BERT). The BERT has been performed with each transceiver at 6.25 Gb/s and an error rate below 3×10^{-15} could be established. To avoid errors due to cabling issues, only 1 m long optical cables were used, so that the error rate was not inflicted by light loss inside the cables. The second test was a static load test with all transceivers. Again an error rate below 3×10^{-15} was achieved.

To value the quality of the data transmission through the SantaLuz board an eye diagram can be used. Figure 8.3 shows an eye diagram taken at the Mu3e experiment, where the SantaLuz board is used in combination with Stratix V GS development boards[I] for data acquisition [50, 51, 52]. The eye diagram was taken with a data rate of 8 Gb/s and shows a well opened eye. A wide jitter can be seen at the transition points, but no transition is visible inside the eye opening which indicates a good signal transmission. The height of the eye opening is measured to be about 120 mV

[I] Used Stratix V: 5SGSMD5K2F40C2N.

and thereby provides a difference between the high and low state the Stratix V GS receiver is able to understand[II].

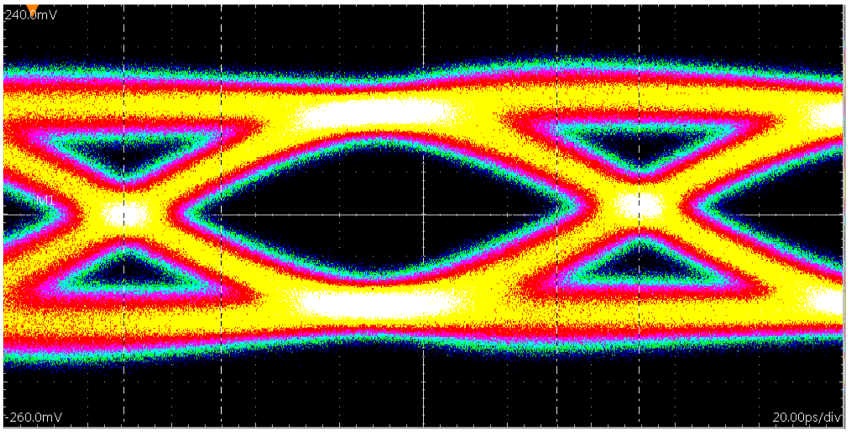


Figure 8.3: Eye diagram of the SantaLuz board, taken with a transfer rate of 8 Gb/s on the receiver site at the Mu3e experiment [52], observable through the 125 ps distance between the two crossing points. The eye is well opened and, with a height of about 120 mV, the difference between the high and low states is good enough for the Stratix V GS receiver to distinguish.

A test setup with a SantaLuz board equipped with eight optical transceivers[III] connected to a Stratix IV development board is shown in Figure 8.4.

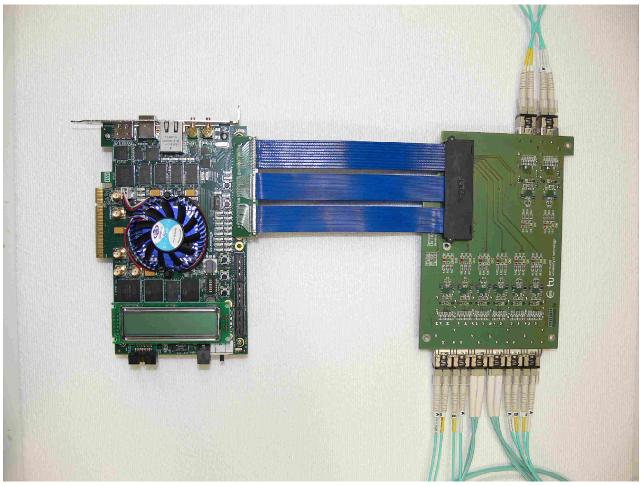


Figure 8.4: A picture of a SantaLuz board fully equipped with eight optical transceivers connected to a Stratix IV development kit via an HSMC cable.

[II] The minimum differential eye opening at the receiver serial input pins for a Stratix V GS receiver is 85 mV [53].

[III] The used optical transceivers are Avago AFBR-57D7APZ.

The Stratix IV development kit comes with an PCIe connector and can be attached to any PCIe host. Using the development kit inside a PC, a setup can be obtained which enables the user to receive and send data from and to any other device using optical connections via a PC.

Since no OT front end is yet available to provide the test setup with data, the random data generator mentioned in Section 7.2.9 was programmed with the firmware option for two connected FEs to the Stratix IV board and a data output via PCIe was established. The output was checked for firmware errors and none was found. So the OT data processing firmware is working successfully on a Stratix IV FPGA.

9 Conclusions

In this thesis a complete data processing firmware for the Outer Tracker has been presented. The firmware is capable of performing all data processing tasks needed to allow the OT to take data after the LHCb upgrade 2018/19, with a detector occupancy of up to 100 %. The main tasks involve receiving, processing and formatting the data for a correct output.

The receiving mechanism reacts to set error bits and is able to react to disabled input lines. One important reaction is to check the "sync" bit, which indicates whether event data or the FE ID is sent, and lead the information to the correct processing blocks.

Two possibilities have been developed to react to a wrong cable management on the hardware: A software sorting on the FPGA using the FE IDs and a safety procedure which, instead of sorting the incoming input lines, warns the user of a wrong cable management. Since the available logic on the to be used Stratix V FPGA is a crucial point the warning mechanism is preferred, due to the fact that it consumes about 10% less logic of the FPGA.

The most crucial part was finding the merging algorithm, which consumes the least amount of logic. Three variations have been presented from FPGA overwhelming to acceptable logic usage. The unsuppressing mechanism has become the method of choice, due to its simplicity and relatively low amount of logic usage. The unsuppressing method extends the zero suppressed data to a vector with defined positions for each drift time, merges it afterwards and zero suppresses it again.

In between the merging and the zero suppressing step, the OT specific clustering mechanism is situated. It generates an additional hit pattern, which removes hits outside OT defined clusters. This additional hit pattern can be used as an option for the zero suppressing mechanism to reduce the outgoing bandwidth even further. If not used for the zero suppressing algorithm, the clustered hit pattern can be added to the data string. Since the clustering is a simple cross check in data and performed in only one step on an FPGA, the AMC40 is the perfect place to perform this task and reduce CPU usage on the DAQ.

The complete OT data processing firmware has been synthesised in different configurations. Variations in reacting to wrong cable management, in the number of connected FEs and in the incoming data format have been analysed. Since the available space for the data processing part is not yet defined, a final statement can not be made. The first estimate of 20 %–30 % of the whole FPGA reduces the possibilities to a few. With this estimated value, the reaction to a wrong cable management has to be warning the user. The number of connected and parallel processed FEs then depends on the used data format. With the planned zero suppressed data format only two or less FEs are able to be processed simultaneously with a logic usage of about 12% for each FE. With the use of a data format without zero suppression and the associated use of eight optical links per FE, the processing of three connected FEs is possible and uses about 16% of the logic on the FPGA for all of them.

With the option of only processing up to three front ends per AMC40 and the average occupancy in the Outer Tracker during the data taking period in 2011/2012, the available output bandwidth on an AMC40 would not be surpassed. With a higher amount of available logic elements on the FPGA for data processing the average detector occupancy could become a limiting factor.

A final simulation of the designs worked without errors and every step inside the processing chain could be followed in logic. The simulation on logic level showed no errors neither with a fixed pattern input nor with random generated data.

To test the firmware on hardware an extension card was developed during this thesis. The SantaLuz board can be attached to an FPGA board via HSMC and provides cages for up to eight optical transceivers. An FPGA development kit can then be read out via PCIe on a PC. Due to the lack of available upgraded OT FEs, the firmware was tested with random generated data on the FPGA and read out via PCIe. As the simulation implied, no errors were found during these tests. However, the integration of the data processing firmware into the common code of the AMC40 could lead to new problems and therefore would need further tests in the finished design. The SantaLuz board itself was tested due to its liability with a BERT and an error rate below 3×10^{-15} could be established.

In summary, the developed firmware is ready to be used for data taking in the upgraded OT back end electronics after including it into the common code. With the extension of the monitoring mechanisms, which were developed in [54] under supervision of this thesis, the data processing is complete for data taking and monitoring as it is actually done with the TELL1. The sorting algorithm as well as the zero suppressing mechanism can also be used in various other detectors with only small changes in the data format, whereas the rest of the firmware is very detector specific. Raw algorithms are still possible to be adapted for other detector electronics.

The SantaLuz board is a working extension for every FPGA based board including an HSMC connector and already used in the data acquisition for the Mu3e experiment [50, 51].

Acronyms

ALICE	A Large Ion Collider Experiment
ALM	Adaptive Logic Module
AMC	Advanced Mezzanine Card
ASCII	American Standard Code for Information Interchange
ASDBLR	Amplifier, Shaper, Discriminator and BaseLine Restorer
ASIC	Application-Specific Integrated Circuit
ATCA	Advanced Telecommunications Computing Architecture
ATLAS	A Toroidal LHC ApparatuS
BERT	Bit Error Rate Test
Bx	Bunch Crossings
BxID	Bunch Crossing Identification
CCPC	Credit Card Personal Computer
CIPMC	Central Intelligent Platform Management Controller
CKM	Cabibbo-Kobayashi-Maskawa
CMS	Compact Muon Solenoid
CPPM	Centre for Particle Physics of Marseilles
CP	Charge-Parity
DAQ	Data Acquisition
DDR	Double Data Rate
DT	Drift Time
ECAL	Electromagnetic Calorimeter
ECS	Electronic Control System

EEPROM	Electrically Erasable Programmable Read-Only Memory
EPROM	Erasable Programmable Read-Only Memory
FE	Front End
FPGA	Field Programmable Gate Array
GbE	Gigabit Ethernet
GBT	GigaBit Transceiver
GEM	Gas Electron Multiplier
GOL	Gigabit Optical Link
HCAL	Hadronic Calorimeter
HDL	Hardware Description Language
HLT	High-Level-Trigger
HSMC	High Speed Mezzanine Card
HV	High Voltage
I/O	Input/Output
IC	Integrated Circuit
IPMB	Intelligent Platform Management Bus
IPMI	Intelligent Platform Management Interface
IT	Inner Tracker
L0	Level-0
LAPP	Laboratoire d'Annecy-le-Vieux de physique des particules
LCD	Liquid-Crystal Display
LED	Light-Emitting Diode
LE	Logic Element
LHCb	Large Hadron Collider beauty
LHC	Large Hadron Collider
LLT	Low-Level-Trigger

LUT	Look-Up Table
LVDS	Low Voltage Differential Signaling
MEP	Multi-Event Package
MMC	Module Management Controller
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
MWPC	Multi-Wire Proportional Chamber
OTIS	Outer Tracker Time Information System
OT	Outer Tracker
PCIe	Peripheral Component Interconnect Express
PID	Particle Identification
PLL	Phase Locked Loop
PROM	Programmable Read-Only Memory
QCD	Quantum Chromodynamics
RAM	Random Access Memory
RICH	Ring Imaging Cherenkov
RTL	Register Transfer Level
SCA	Slow Control Adapter
SM	Standard Model of particle physics
SPD/PS	Silicon Pad Detector/PreShower
SRAM	Static Random-Access Memory
TDC	Time to Digital Converter
TELL1	Trigger Electronics Level 1
TFC	Timing and Fast Control
TT	Tracker Turicensis
VELO	Vertex Locator
VHDL	VHSIC Hardware Description Language

Appendix

Files of used blocks in presented designs

Table 1: Explanation of the abbreviations used in Table 2.

Description	Toplevel filename	Abbr.
Six FEs with sorting	TOP_struct.vhd	6(S)
Six FEs with error	TOP_without_sort_struct.vhd	6(E)
Five FEs with error	TOP_without_sort_5FE_struct.vhd	5(E)
Four FEs with error	TOP_without_sort_4FE_struct.vhd	4(E)
Three FEs with error	TOP_without_sort_3FE_struct.vhd	3(E)
Two FEs with error	TOP_without_sort_2FE_struct.vhd	2(E)
Three FEs with error, zero unsuppressed data	TOP_without_sort_3FE_NZS_struct.vhd	3(EN)

Table 2: Filenames are to be extended with “ehavior.vhd” for each _b. . . and with “truct.vhd” for each _s. . . .

Filename	6(S)	6(E)	E(E)	4(E)	3(E)	2(E)	3(EN)
FE_select_3_b. . .	0	0	0	0	1	0	1
FE_select_b. . .	0	0	0	0	0	1	0
incomming_stream_b. . .	24	24	20	16	12	8	12
sorting_2_s. . .	1	0	0	0	0	0	0
address_cluster_merger_b. . .	1	0	0	0	0	0	0
FE_arrangement_b. . .	24	0	0	0	0	0	0
fe_part_arrangement_s. . .	1	0	0	0	0	0	0
position_value_output_b. . .	1	0	0	0	0	0	0
fe_part_arranger_small_b. . .	24	0	0	0	0	0	0
connection_check_s. . .	0	1	0	0	0	0	0
connection_check_5FE_s. . .	0	0	1	0	0	0	0
connection_check_4FE_s. . .	0	0	0	1	0	0	0
connection_check_3FE_s. . .	0	0	0	0	1	0	1
connection_check_2FE_s. . .	0	0	0	0	0	1	0
FE_arrange_check_channels4_b. . .	0	6	5	4	3	2	3
FE_arrange_check4_b. . .	0	6	5	4	3	2	3
splitting_s. . .	1	1	0	0	0	0	0

splitting_5FE_s...	0	0	1	0	0	0	0
splitting_4FE_s...	0	0	0	1	0	0	0
splitting_3FE_s...	0	0	0	0	1	0	1
splitting_2FE_s...	0	0	0	0	0	1	0
gbt_splitter_b...	24	24	20	16	12	8	12
PROCESSING_s...	1	1	0	0	0	0	0
PROCESSING_5FE_s...	0	0	1	0	0	0	0
PROCESSING_4FE_s...	0	0	0	1	0	0	0
PROCESSING_3FE_s...	0	0	0	0	1	0	0
PROCESSING_2FE_s...	0	0	0	0	0	1	0
PROCESSING_3FE_NZS_s...	0	0	0	0	0	0	1
PROCESSING_FE_s...	6	6	5	4	3	2	0
PROCESSING_FE_NZS_s...	0	0	0	0	0	0	3
Un_0_supp_Allinone_b...	24	24	20	16	12	8	0
hitmap_buffer_b...	24	24	20	16	12	8	12
BX_merger_32bit_NZS_b...	24	24	20	16	12	8	12
pre_cluster_data_check_b...	6	6	5	4	3	2	3
clustering_merged_b...	6	6	5	4	3	2	3
cluster_selector_b...	6	6	5	4	3	2	3
UN_0_Supp_quarter_b...	24	24	20	16	12	8	12
ZS_splitter_b...	24	24	20	16	12	8	12
header_buffer_zs_b...	24	24	20	16	12	8	12
zs_partial_4hits_b...	192	192	160	128	96	64	96
zs_merger_4_to_8_hits_b...	96	96	80	64	48	32	48
zs_merger_8_to_16_hits_b...	48	48	40	32	24	16	24
zs_merge_16_to_32_hits_b...	24	24	20	16	12	8	12
OUTPUT_s...	1	1	0	0	0	0	0
OUTPUT_5FE_s...	0	0	1	0	0	0	0
OUTPUT_4FE_s...	0	0	0	1	0	0	0
OUTPUT_3FE_s...	0	0	0	0	1	0	1
OUTPUT_2FE_s...	0	0	0	0	0	1	0
Output_merger_HMHM_b...	24	24	20	16	12	8	12

Files

The files are available as download at:

http://www.physik.tu-dortmund.de/downloads/OT_TELL40_dp_v1.0.rar

Data illustrated in Section 7.2.9

Table 3: Drift times belonging to the illustrated data in Section 7.2.9. Zero suppressed with indication of the channel (ch), the drift time in binary (bin) and decimal (dec).

	Bx0			Bx1			Bx2			Output	
ch	bin	dec	ch	bin	dec	ch	bin	dec	ch	bin	dec
0	11101	29	0	11100	28	4	11100	28	0	11101	29
3	01000	8	4	10111	23	5	11010	26	3	01000	8
4	11011	27	8	01001	9	8	11100	28	4	11011	27
8	10111	23	12	11100	28	13	11111	31	5	11010	26
9	00110	6	13	00001	1	14	01100	12	8	10111	23
11	11000	24	14	11101	29	16	01100	12	11	11000	24
12	11011	27	16	11001	25	17	10011	19	16	11001	25
13	10111	23	19	10011	19	19	00001	1	17	10011	19
14	01111	15	24	10101	21	23	11010	26	20	11110	30
19	10001	17	27	11001	25	24	00011	3	23	11010	26
20	11110	30	39	11100	28	26	10111	23	30	10001	17
25	01010	10	42	00110	6	27	11101	29	42	00110	6
39	11111	31	50	00111	7	30	10001	17	49	01110	14
40	01000	8	57	01100	12	33	10100	20	50	11111	31
49	01110	14	59	00111	7	43	10101	21	51	10000	16
50	11111	31	64	00011	3	50	10011	19	59	10101	21
57	00101	5	68	10001	17	51	10000	16	67	10010	18
59	10101	21	74	10011	19	59	01101	13	68	10101	21
64	01011	11	76	01100	12	64	01010	10	76	01100	12
65	11000	24	82	00010	2	68	01011	11	77	00000	0
66	10000	16	85	10110	22	74	10010	18	78	11111	31
67	10010	18	96	00010	2	78	11111	31	85	11101	29
68	10101	21	116	00000	0	98	10110	22	96	00010	2
74	11011	27	119	10111	23	104	11010	26	97	00100	4
77	00000	0	121	01100	12	110	11110	30	104	11010	26
79	10110	22	122	11100	28	111	01010	10	106	00110	6
80	10000	16	126	01111	15	119	11100	28	110	11110	30
82	00011	3				126	01000	8	111	01010	10
85	11101	29							116	10011	19
92	00000	0							119	10111	23
97	00100	4							121	01111	15
106	00110	6							122	11100	28
116	10011	19							123	01000	8
121	01111	15							126	10011	19
123	01000	8							127	10110	22
126	10011	19									
127	10110	22									

Bibliography

[1] LHCb Collaboration, *The LHCb Detector at the LHC*, JINST 3 (2008) S08005, DOI: 10.1088/1748-0221/3/08/S08005

[2] CERN, *LHC Performance and Statistics*, (07.10.2014), http://lhc-statistics.web.cern.ch/ LHC-Statistics/index.php

[3] CERN Bulletin, *Some LHC milestones*, Issue Nr. 38/2008, (04.11.2014), http://cds.cern.ch/journal/CERNBulletin/2008/38/News%20Articles/1125888

[4] I.J.R. Aitchison, A.J.G. Hey, *Gauge theories in particle physics: Non-Abelian gauge theories: QCD and the electroweak theory*, Bristol, Institute of Physics (Great Britain) (2004), ISBN: 0-7503-0950-4

[5] ATLAS Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider*, JINST 3 (2008) S08003, DOI: 10.1088/1748-0221/3/08/S08003

[6] CMS Collaboration, *The CMS Experiment at the CERN LHC*, JINST 3 (2008) S08004, DOI: 10.1088/1748-0221/3/08/S08004

[7] ALICE Collaboration, *ALICE: Physics Performance Report, Volume I*, J. Phys. G: Nucl. Part. Phys. 30, 1517 (2004), DOI: 10.1088/0954-3899/30/11/001

[8] T. Muta, *Foundations of quantum chromodynamics*, New Jersey, World Scientific (2010) ISBN: 978-981-279-354-6

[9] W.Greiner and J. Reinhardt, *Quantenelektrodynamik*, Thun, Deutsch (1995), ISBN: 3-8171-1426-5

[10] C. Weinheimer and K. Zuber, *Neutrino Masses*, Annalen der Physik, 525 (2013) 565-575, DOI: 10.1002/andp.201300063, arXiv:1307.3518 [hep-ex]

[11] F. Englert and R. Brout, *Broken Symmetry and the Mass of Gauge Vector Mesons*, Phys. Rev. Lett. 13, 321 (1964), DOI: 10.1103/PhysRevLett.13.321

[12] P.W. Higgs, *Broken Symmetries and the Masses of Gauge Bosons*, Phys. Rev. Lett. 13, 508 (1964), DOI: 10.1103/PhysRevLett.13.508

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

[14] CMS Collaboration, *Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC*, Phys.Lett. B 716, 30-61 (2012), DOI: 10.1016/j.physletb.2012.08.021, arXiv:1207.7235 [hep-ex]

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

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

[17] J.T. Wishahi, *Measurement of CP violation in $B^0 \rightarrow J/\psi K_S^0$ Decays with the LHCb Experiment*, Dissertation TU Dortmund (2013), http://hdl.handle.net/2003/32999

[18] CKMfitter Group (J. Charles et al.), Eur. Phys. J. C41, 1-131 (2005) [hep-ph/0406184], updated results and plots available at: http://ckmfitter.in2p3.fr

[19] LHCb Speakers Bureau, "$\bar{b}b$ production angle plots", (07.10.2014), http://lhcb.web.cern.ch/lhcb/speakersbureau/html/bb_ProductionAngles.html

[20] G. Lutz, *Semiconductor Radiation Detectors*, Berlin, Springer (1999), ISBN: 3-540-64859-3

[21] H. Büker, *Theorie und Praxis der Halbleiterdtektoren für Kernstrahlung*, Berlin, Springer (1971), ISBN: 3-540-05443-X

[22] P.A. Cherenkov, *Visible Radiation Produced by Electrons Moving in a Medium with Velocities Exceeding that of Light*, Phys. Rev. 52, 378 (1937), DOI: 10.1103/PhysRev.52.378

[23] LHCb RICH Group, *Performance of the LHCb RICH detector at the LHC*, Eur. Phys. J. C 73 (2013), 2431, DOI: 10.1140/epjc/s10052-013-2431-9, arXiv:1211.6759 [physics.ins-det]

[24] I. Machikhiliyan for the LHCb calorimeter group, *The LHCb electromagnetic calorimeter*, J. Phys.: Conf. Ser. 160 012047, DOI: 10.1088/1742-6596/160/1/012047

[25] LHCb Collaboration, *Framework TDR for the LHCb Upgrade : Technical Design Report*, CERN-LHCC-2012-007 ; LHCb-TDR-12 (2012)

[26] LHCb Collaboration, *Strong constraints on the rare decays $B_s^0 \to \mu^+\mu^-$ and $B^0 \to \mu^+\mu^-$*, Phys. Rev. Lett. 108, 231801 (2012), arXiv:1203.4493 [hep-ex]

[27] LHCb Collaboration, *Differential branching fraction and angular analysis of the decay $B^0 \to K^{*0}\mu^+\mu^-$*, Phys. Rev. Lett. 108 (2012) 181806, DOI: 10.1103/PhysRevLett.108.181806, arXiv:1112.3515 [hep-ex]

[28] LHCb Collaboration, *Measurements of the CP-violating phase ϕ_s in the decay $B_s^0 \to J/\psi\phi$*, Phys. Rev. Lett. 108, 101803 (2012), DOI: 10.1103/PhysRevLett.108.101803, arXiv:1112.3183 [hep-ex]

[29] R. Aaij, et al., *The LHCb Trigger and its Performance in 2011*, JINST 8 P04022 (2013), DOI: 10.1088/1748-0221/8/04/P04022, arXiv:1211.3055 [hep-ex] (2012)

[30] S. Akar, et al., *Review Document: Low Level Trigger (LLT)*, CERN-LHCb-PUB-2014-037 (2014)

[31] J. Albrecht, et al., *Review Document: Full Software Trigger*, CERN-LHCb-PUB-2014-036 (2014)

[32] LHCb Outer Tracker Group, *Performance of the LHCb Outer Tracker*, JINST 9 (2014) P01002, DOI: 10.1088/1748-0221/9/01/P01002, arXiv:1311.3893 [physics.ins-det]

[33] A. Pellegrino and W. Vink, *FPGA-based, radiation-tolerant on-detector electronics for the upgrade of the LHCb Outer Tracker Detector*, Poster at TWEPP 2013 (2013), https:// indico.cern.ch/event/228972/session/9/material/poster/0?contribId=151

[34] C. Färber, *Feasibility study to use an SRAM-based FPGA in the readout electronics of the upgraded LHCb outer tracker detector*, Dissertation (2014), urn:nbn:de:bsz:16-heidok-162973

[35] J.-P. Cachemiche, et al., *Readout board specifications for the LHCb upgrade*, LHCb Technical Note (2012), EDMS Id: 1251709 v.1

[36] K. Wyllie, et al., *Electronics Architecture of the LHCb Upgrade*, LHCb Technical Note (2013), LHCb-PUB-2011-011

[37] F. Alessio and R. Jacobsson, *System-level Specifications of the Timing and Fast Control system for the LHCb Upgrade*, LHCb Technical Note v1.4 (2014), LHCb-PUB-2012-001

[38] C. Maxfield, *The Design Warrior's Guide to FPGAs*, Burlington, MA: Newnew/Elsevier (2004), ISBN: 978-0-7506-7604-5

[39] B. Prince, *Semiconductor memories*, Chichester, Wiley (1995) ISBN: 0-471-94295-2

[40] J. Schulze, *Konzepte siliziumbasierter MOS-Bauelemente*, Berlin, Springer (2005) ISBN: 978-3-540-27547-3

[41] A. Kolodny, et al., *Analysis and modeling of floating-gate EEPROM cells*, IEEE Transactions on electron devices, VOL. ED-33, NO. 6, JUNE 1986, 835-844, DOI: 10.1109/T-ED.1986.22576

[42] ALTERA Corporation, *High-Performance FPGA Architecture*, ALTERA website (29.09.2014), http://www.altera.com/devices/fpga/stratix-fpgas/about/fpga-architecture/stx-architecture.html

[43] ALTERA Corporation, *Stratix V FPGA Family Overview*, ALTERA website (29.09.2014), http://www.altera.com/devices/fpga/stratix-fpgas/stratix-v/overview/stxv-overview.html

[44] IEEE Computer Society, *1364-2005 - IEEE Standard for Verilog Hardware Description Language*, IEEE Standard (2006), DOI: 10.1109/IEEESTD.2006.99495

[45] IEEE Computer Society, *1164-1993 - IEEE Standard Multivalue Logic System for VHDL Model Interoperability (Stdlogic1164)*, IEEE Standard (1993), DOI: 10.1109/IEEESTD.1993.115571

[46] LHCb Collaboration, *LHCb Trigger and Online Upgrade: Technical Design Report*, CERN/LHCC 2014-016 ; LHCb TDR 16 (2014)

[47] G. Vouters, et al., *Global Status*, AMC40 firmware Meeting (9.04.2014), https://indico.cern.ch/event/254741/session/0/contribution/1/material/slides/0.pdf

[48] Private communication with Guillaume Vouters (20.02.2013)

[49] G. Vouters, et al., *Mini DAQ Simulation*, LHCb Upgrade AMC40 firmware Meeting (12.06.2013), https://indico.cern.ch/event/249774/session/1/contribution/4/material/slides/0.pdf

[50] A. Blondel, et al., *Research Proposal for an Experiment to Search for the Decay $\mu \rightarrow eee$*, Research Proposal (2013), arXiv:1301.6113 [physics.ins-det]

[51] S. Bachmann, et al., *The proposed trigger-less TBit/s readout for the Mu3e experiment*, JINST 9 C01011 (2014), DOI: 10.1088/1748-0221/9/01/C01011

[52] S. Corrodi, *Fast Optical Readout of the Mu3e Pixel Detector*, Master thesis (2014), http://www.psi.ch/mu3e/ThesesEN/MasterCorrodi.pdf

[53] ALTERA Corporation, *Stratix V Device Datasheet* SV53001-3.3 (11.2014)

[54] D. Buchmann, *Monitoring auf einem Testsystem der Backend- Elektronik für das Upgrade des Outer Tracker am LHCb*, Master thesis TU Dortmund (2014), http://www.e5.physik.uni-dortmund.de/media/download/research/tp/lhcb/diplomarbeiten/buchmann-master.pdf

Danksagung

Als erstes möchte ich Herrn Prof. Dr. Bernhard Spaan für die Betreuung und die Möglichkeit danken diese Promotion durchzuführen. Herrn Dr. Reiner Klingenberg möchte ich danken, dass er sich als Zweitgutachter dieser Dissertation zur Verfügung gestellt hat.

Dem gesamten Lehrstuhl für Experimentelle Physik 5 möchte ich für die mal mehr, mal weniger fruchtbare Zusammenarbeit der letzten Jahre danken. Nicht nur bei der Arbeit, sondern auch bei verschiedensten Veranstaltungen wie „Exkursionen“, beim Grillen, oder einfach nur während der mittwöchlichen „Halo Night“. Hervorzuheben sind hierbei meine zwei Dauerbürokollegen Robert und Mirco, mit denen immer sehr interessante Diskussionen um absolut uninteressante Themen in einem gewissen Bürojargon möglich waren. Zudem gilt ein besonderer Dank Kai und Matthias, die mir immer mit Rat und Tat zur Seite standen und die nötige Infrastruktur in Betrieb hielten, oder auch mal als „Notizbuch“ dienten. Zu erwähnen sind natürlich auch noch einige Ehemalige des Lehrstuhls, namentlich sei vor allem Mirco dafür gedankt mich in die Arbeit mit FPGAs eingeführt zu haben.

Mein Dank gilt natürlich auch den Kollegen vom CERN, aus Heidelberg und NIKHEF, ohne die Teile dieser Arbeit nicht möglich gewesen wären.

Abseits des Büroalltags gab es natürlich auch Menschen die mich in den letzten Jahren begleitet haben. Zunächst seien hier Thorsten, Birgit, Alex und Dominik genannt, die mir bei der ein oder anderen Gelegenheit (u.a. Montags, oder am Anfang noch Dienstags) halfen den Blick auf Dinge abseits der Uni zu richten. An dieser Stelle geht auch ein besonderer Dank an die Jungs und Mädels vom Physikerfußball, die es bis heute nicht geschafft haben mir bleibende Schäden zuzufügen.

Meiner Familie danke ich für die Unterstützung jedweder Art während der gesamten Promotion. Hilfe war immer da wenn ich sie benötigte.

Zu guter Letzt möchte ich mich besonders bei meiner Verlobten Natalie bedanken, die mich immer unterstützt und motiviert hat. Sie war immer für mich da und half mir vor allem beim Zusammenschreiben der Dissertation nicht die Nerven zu verlieren.

Danke!