A Control System for the Mu3e Data Acquisition

Masterarbeit in Physik
vorgelegt dem Fachbereich Physik, Mathematik und Informatik (FB 08)
der Johannes Gutenberg-Universität Mainz
am 23. Oktober 2019

Martin Müller

1. Gutachter: Prof. Dr. Niklaus Berger
2. Gutachter: Prof. Dr. Volker Büscher

Ich versichere, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Mainz, den 22. Oktober 2019

Martin Müller
Matrikelnummer 2714008
AG Berger
Institut für Kernphysik
Staudingerweg 9
Johannes Gutenberg-Universität D-55128 Mainz
mmarti04@students.uni-mainz.de

Abstract

The Mu3e experiment will search for the lepton flavour violating decay $\mu^+ \to e^+e^-e^+$ and is aiming for a sensitivity of one in 10^{16} muon decays. Since this decay is highly suppressed in the Standard Model to a branching ratio of below $\mathcal{O}\left(10^{-54}\right)$, an observation would be a clear sign for new physics.

In the Mu3e detector, six thin layers of silicon pixel sensors will be used to track electrons and positrons. Scintillating fibres and tiles will provide precise time information. The overall detector is expected to produce a data rate from 80 Gbit/s (Phase I) to 1 Tbit/s (Phase II), which will be processed in a three-layer, triggerless DAQ system using FPGAs and a GPU filter farm for online event selection.

In this thesis, parts of the DAQ system were developed and tested with a focus on the different control mechanisms and time synchronisation of the components. Integrating and synchronising all three detector types into a common data acquisition system is an important step on the path to the final Mu3e detector. The firmware structure written during this thesis is aimed upon making this possible.

Zusammenfassung

Das Mu3e Experiment wird mit einer Sensitivität von einem in 10^{16} Muon Zerfällen nach dem Leptonen-Familienzahl verletzenden Prozess $\mu^+ \rightarrow e^+e^-e^+$suchen. Da dieser Zerfall im Standardmodel zu einem Verzweigungsverhältnis von unter $\mathcal{O}\left(10^{-54}\right)$ unterdrückt ist, würde eine Detektion ein klares Signal für neue Physik darstellen.

Der Mu3e Detektor wird sechs dünne Lagen von Pixelsensoren für die Spurrekonstruktion von Elektronen und Positronen verwenden. Scintillationsdetektoren werden die nötige Zeitauflösung bereitstellen. Es wird erwartet, dass das gesamte System eine Datenrate von 80 Gbit/s in Phase I und 1 Tbit/s in Phase II des Experiments produzieren wird. Diese Daten werden in einem dreilagigem, triggerlosem Datennahmesystem verarbeitet, dass auf FPGAs und GPUs basiert.

In dieser Arbeit wurden Teile dieses Systems entwickelt und getestet, wobei ein Fokus auf die Kontrollmechanismen und Synchronisation der einzelnen Komponenten gelegt wurde. Die verschiedenen Detektortypen in ein gemeinsames System zu integrieren und in diesem zu Synchronisieren stellt einen wichtigen Schritt in Richtung des finalen Detektors dar. Die in dieser Arbeit entwickelte Firmware-Struktur zielt darauf ab dies zu ermöglichen.

Contents

1 Introduction **1**
- 1.1 The Standard Model of Particle Physics . . . 1
- 1.2 The Muon Decay . . . 2
- 1.3 Lepton Flavour Violation . . . 3
  - 1.3.1 Theories predicting $\mu^+ \to e^+e^-e^+$ . . . 4

2 The Mu3e Experiment **7**
- 2.1 The Muon Beam . . . 7
  - 2.1.1 The HIPA Accelerator . . . 7
  - 2.1.2 Muon production . . . 8
- 2.2 Background processes . . . 8
- 2.3 Detector Concept . . . 10
- 2.4 The MuPix Pixel Sensor . . . 11
- 2.5 Scintillating Fibres . . . 13
- 2.6 Scintillating Tiles . . . 14

3 Field Programmable Gate Arrays (FPGAs) **17**
- 3.1 Hardware . . . 17
  - 3.1.1 Transistors . . . 18
  - 3.1.2 Logic Gates . . . 18
  - 3.1.3 Flip-Flops . . . 20
  - 3.1.4 Synchronous Systems . . . 20
  - 3.1.5 Configuration of FPGAs . . . 21
  - 3.1.6 Logic Modules and LUTs . . . 22
  - 3.1.7 FPGAs used in this thesis . . . 23
- 3.2 Timing . . . 24
  - 3.2.1 Metastability . . . 25
  - 3.2.2 Clock Domain Transitions . . . 26
    - 3.2.2.1 Synchronisation of Parallel Signals . . . 27
- 3.3 IP-Cores . . . 27
  - 3.3.1 FIFOs . . . 27
  - 3.3.2 Phase Locked Loops (PLLs) . . . 28
  - 3.3.3 Soft Core Processors . . . 29
  - 3.3.4 Memory . . . 29
  - 3.3.5 Transceiver . . . 29

4 Data Transmission **31**
4.1 parallel and serial communication . . . 31
4.2 LVDS . . . 31
4.3 8b10b . . . 32
4.4 Transceivers . . . 33
4.4.1 PMA Block . . . 33
4.4.2 PCS Block . . . 34
4.5 PCIe & DMA . . . 35

5 The Mu3e Data Acquisition **37**
5.1 Overview . . . 37
5.2 Layer 1 – Frontend Boards . . . 38
5.3 Layer 2 – Switching Boards . . . 39
5.4 Layer 3 – Farm PCs . . . 40

6 Test Setup for the Mu3e Data Acquisition **41**

7 The Control System of the Mu3e DAQ **45**
7.1 MIDAS . . . 46
7.1.1 Midas Frontends . . . 46
7.1.1.1 Example: DHCP & DNS server control . . . 46
7.1.2 MIDAS Banks and Bank structure . . . 48
7.2 Reset and Clock System . . . 49
7.2.1 The Reset Protocol . . . 49
7.2.2 Generation and Distribution of the Clock and Reset Signal . . . 51
7.2.3 Timing measurements with the clock distribution board . . . 51
7.2.4 Receiving of Reset Commands . . . 52
7.2.5 Timing of Reset Commands . . . 53
7.2.5.1 The Problem with Timing in fast Transceivers . . . 53
7.2.5.2 About the use of Deterministic Latency Transceivers . . . 54
7.2.5.3 Implemented Solution for the Timing Problems . . . 55
7.2.5.4 Reset Connection to Sub-detectors . . . 58
7.2.5.5 Reset bypass from NIOS processor . . . 58
7.2.5.6 Reset Timing with LVDS Transceivers . . . 59
7.3 Slowcontrol . . . 61
7.3.1 Data Protocols . . . 61
7.3.1.1 MuPix Data . . . 61
7.3.1.2 MuTrig Data . . . 62
7.3.1.3 Slowcontrol Data . . . 62
7.3.1.4 Run Control Signals . . . 63
7.3.2 Upstream Control Data . . . 64
7.3.3 Downstream Control Data . . . 64
7.4 Clock Domains . . . 66

7.5 Midas Slow Control Bus (MSCB) 68
7.5.1 Data Transmission and Protocol 68
7.5.2 Implementation of an MSCB Node on the Frontend Board . . . 69
7.5.3 MIDAS frontend for memory access with MSCB 70

8 Conclusion and Outlook **73**

Appendices **75**

List of Figures **83**

List of Tables **85**

List of Abbreviations **87**

Bibliography **89**

1 Introduction

The standard model of particle physics summarises the human knowledge about the fundamental components of our universe. It describes a variety of processes using fundamental, indivisible particles, and it has helped us to understand, to describe and to predict observed phenomena and has proved to be very successful in doing so.

1.1 The Standard Model of Particle Physics

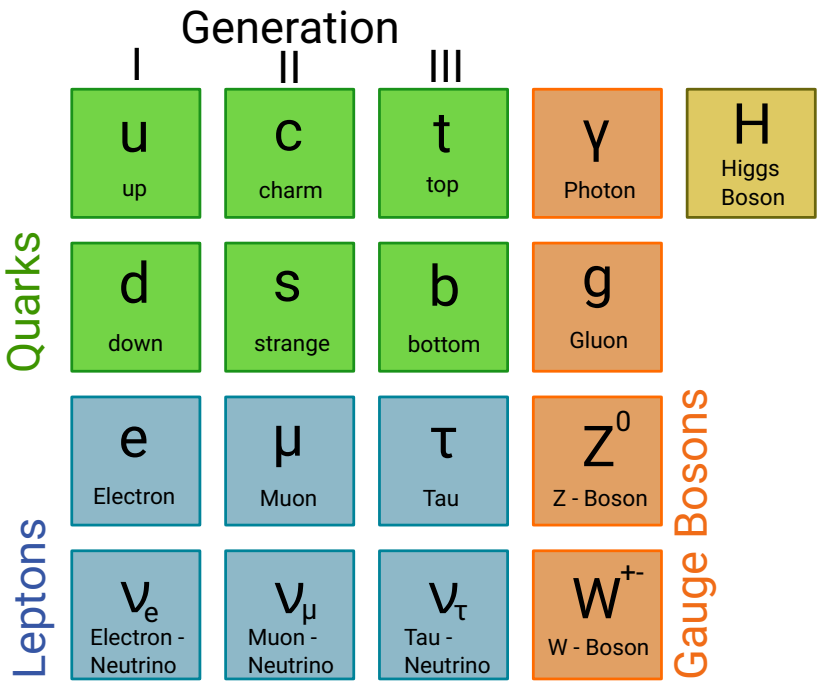


Figure 1: Overview of particles in the Standard Model of particle physics. The particles are grouped into the gauge bosons (orange), the Higgs-boson (yellow) and three generations of quarks (green) and leptons (blue).

The fundamental particles in the Standard Model of particle physics are shown in figure 1. The interactions between them are described with three fundamental forces, which are caused by the exchange of "force carrier" particles – the gauge bosons. The photon mediates the electromagnetic force, which attracts or repels charged particles from each other. Quarks are bond together by gluon exchange and form hadrons like protons and neutrons that make up, together with electrons, all visible matter. The

third fundamental force in the Standard Model (SM) is carried by the W- and Z-boson and is called the "weak" force, which is responsible for radioactive decays.

In addition to the quarks and gauge bosons, there are also the leptons. As well as the quarks, they come in three generations. The charged leptons (electron, muon and tau) interact via the electromagnetic and weak force, while their neutral counterparts (electron-, muon- and tau-neutrino) can only interact via the weak force. The last particle is the Higgs-boson, which was discovered in 2012 by the experiments ATLAS [1] and CMS [2] with the Large Hadron Collider (LHC) at CERN. The Higgs-field gives all other particles their mass. One could consider the prediction and discovery of the Higgs boson as the most important achievements of the Standard Model of particle physics.

Despite all the success of the SM, some phenomena remain unclear and cannot be explained with SM processes. These phenomena include the observation of additional, invisible, "dark" matter as well as gravitational interaction between particles and the matter-antimatter asymmetry. It is therefore clear that beyond standard model (BSM) research is necessary to improve our understanding and description of nature.

Some of the unexplained observations or tensions with the SM directly involve muons, like the proton radius puzzle [3] or the magnetic moment measured at Brookhaven National Laboratory, which deviates by 3.7 σ from the Standard Model prediction [4].

1.2 The Muon Decay

In the standard model, transitions between the three generations in figure 1 are not possible for leptons. This principle is called lepton flavour conservation. Therefore, the SM allows the decay modes shown in table 1 for the muon. All of them were experimentally found, and their branching ratios were measured. Almost all muons perform the Michel-decay shown in figure 2 with a mean lifetime of $(2.197034 \pm 0.000021) \cdot 10^{-6}$ seconds [5].

decay channel	BR [%]
$\mu^- \rightarrow e^- \overline{\nu}_e \nu_\mu$	≈ 100
$\mu^- \rightarrow e^- \overline{\nu}_e \nu_\mu \gamma$	1.4 ± 0.4
$\mu^- \rightarrow e^- \overline{\nu}_e \nu_\mu e^+ e^-$	3.4 ± 0.4

Table 1: Decay channels of the muon in the SM and their measured branching ratios (BRs). The decay modes for the positive muon are charge conjugates of the channels shown above. BR-values taken from [5]

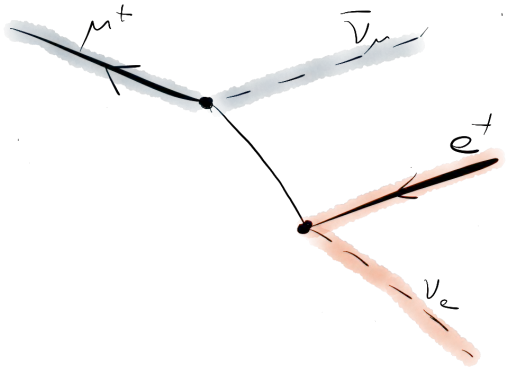


Figure 2: Main decay channel of the muon in the standard model, $\mu^+ \rightarrow \mathrm{e}^-\overline{\nu}_\mathrm{e}\nu_\mu$ (Michel-decay).

1.3 Lepton Flavour Violation

Lepton flavour violation is known to exist in the neutrino sector since the observation of neutrino oscillations in the Super-Kamiokande and other experiments [6]. Extending the SM with neutrino mixing leads to the possibility of charged lepton flavour violation via higher-order loop diagrams which include neutrino mixing. The muon decay channel $\mu^+ \rightarrow e^+e^-e^+$ is therefore allowed if non-zero neutrino masses are taken into account. The diagram of this type with the highest contribution is shown in figure 3 and has a predicted branching ratio smaller than 10^{-50} [7]. Direct decays

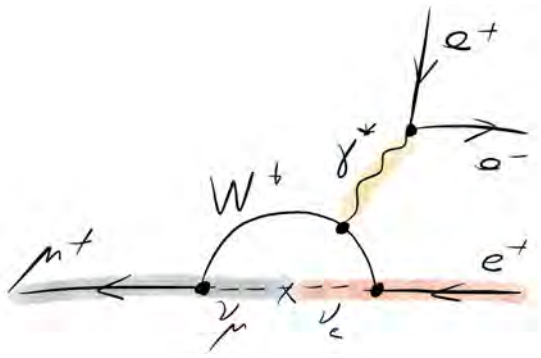


Figure 3: Feynman diagram of $\mu \rightarrow 3\mathrm{e}$ via neutrino mixing

$\mu^+ \rightarrow e^+e^-e^+$ at tree level or other loop diagrams that do not include a neutrino mixing are however forbidden in the standard model. Due to the unobservable small size of the branching ratios of decays with a neutrino mixing and the non-existence of other standard model contributions, the decay channel $\mu^+ \rightarrow e^+e^-e^+$ represents a very clean environment to search for physics processes beyond the SM. A single observation of such a muon decay would, therefore, be a clear sign for new physics.

Hence, the precision of any experiment searching for it will only be limited by the detection efficiency for this decay, the false-positive rate from background processes and the amount of muon decays observed.

Figure 4 shows a summary of searches and planned searches for lepton flavour violating processes and the upper limits obtained by or predicted for these experiments. The most recent experimental results show an upper limit in the area of 10^{-12} for the branching ratio of the lepton flavour violating muon decay channels $\mu \to e\gamma$, $\mu N \to eN$ and $\mu \to 3e$ at a 90% confidence level. For the tau channels $\tau \to \mu\gamma$ and $\tau \to 3\mu$ an upper limit around 10^{-8} was obtained [8].

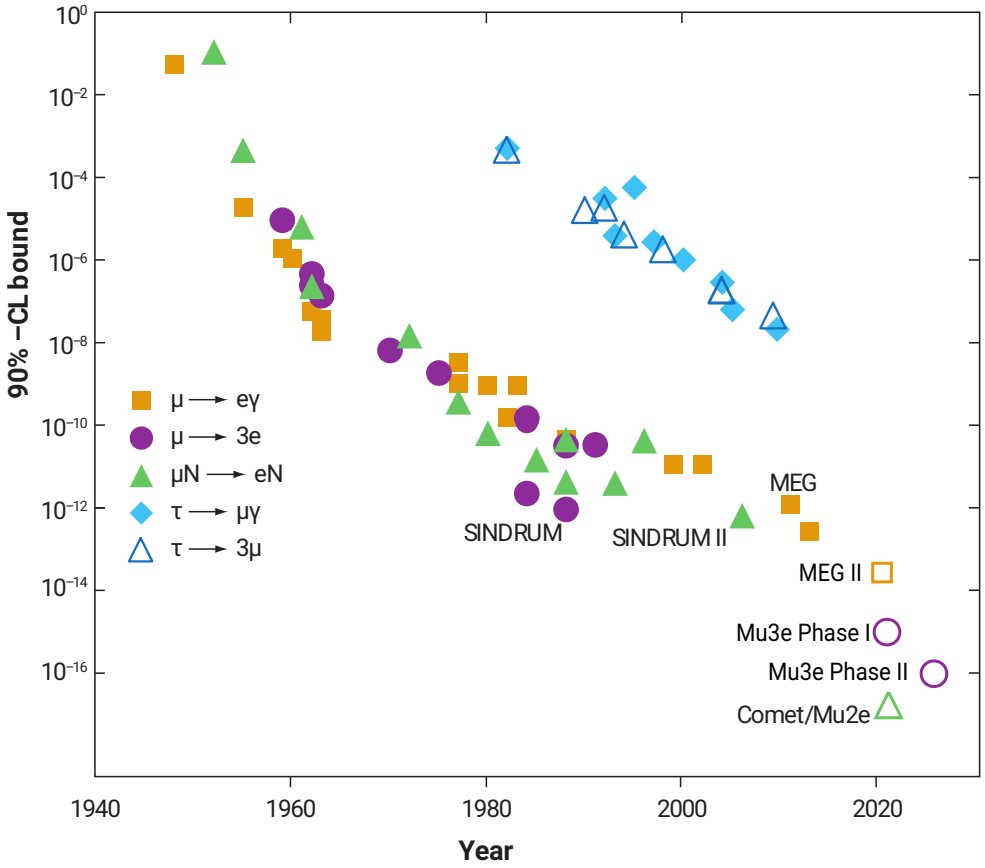


Figure 4: History of lepton flavour violation searches. Adapted from [8].

1.3.1 Theories predicting $\mu^+ \to e^+e^-e^+$

As discussed previously, the standard model does not include measurable possibilities for a $\mu^+ \to e^+e^-e^+$ decay. There are, however, beyond standard model (BSM) theories that predict measurable contributions to this decay channel, which makes an experimental search for it valuable in order to set exclusion limits on them or to launch further research in this field in case of an actual detection of $\mu^+ \to e^+e^-e^+$.

This section contains some of the theories that predict this decay. Some of the BSM theories predict $\mu^+ \to e^+e^-e^+$decays on tree level (figure 5), others expect it in BSM loop diagrams (figure 6).

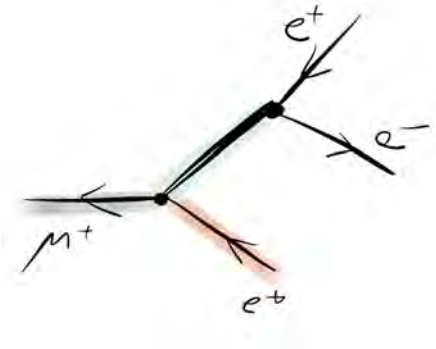


Figure 5: BSM Feynman diagram of a Mu3e decay at tree level.

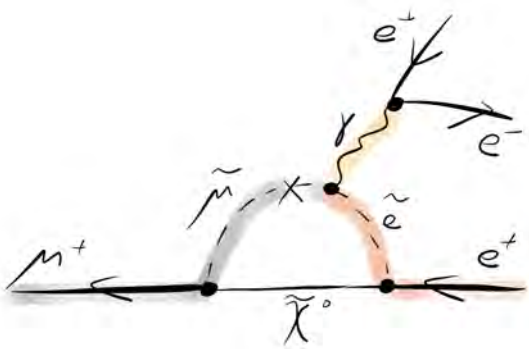


Figure 6: BSM Feynman diagram of a Mu3e decay in a loop containing SUSY particles.

Higgs-triplet

This kind of model proposes a Higgs-particle triplet that would naturally generate small neutrino masses. It also allows Mu3e decays on tree level mediated by the Higgs boson [9].

Z′

Z′- models introduce an additional heavy vector boson Z′. This boson is also expected to mediate Mu3e decays on tree level [10].

Supersymmetry

One of the possibilities for the loop diagram case is supersymmetry (SUSY). In supersymmetry, all fermions of the standard model have a supersymmetric bosonic partner and bosons have a supersymmetric fermionic partner. If these particles exist, a contribution to the decay $\mu^+ \to e^+e^-e^+$via a lepton flavour violation of the supersymmetric partners of the electron and the muon (selectron $\widetilde{e}$ and smuon $\widetilde{\mu}$) is expected in some models [11] (figure 6).

Other possibilities for Mu3e decays in loop diagrams arise in little-Higgs [12] or leptoquark [13] models. The set of examples given here is, however, not complete.

2 The Mu3e Experiment

The Mu3e experiment [14] is a particle physics experiment searching for the lepton flavor violating decay of a positive muon into two positrons and one electron ($\mu^+ \to e^+e^-e^+$). It will be constructed at the Paul Scherrer Institute in Switzerland and is expected to start data taking in 2021. The current upper limit for the branching ratio of $\mu^+ \to e^+e^-e^+$ is 10^{-12} at a 90% confidence level and was measured by the SINDRUM experiment in 1988 [15]. An overview of results from experiments searching for lepton flavour violating muon decays is shown in table 2. Mu3e is aiming to improve the sensitivity of SINDRUM by three orders of magnitude to 10^{-15} in Phase I and 10^{-16} in Phase II.

Decay channel	BR limit (90% CL)	Experiment	Year
$\mu^+ \to e^+e^-e^+$	$1 \cdot 10^{-12}$ [15]	SINDRUM	1988
$\mu^+ \to e^+\gamma$	$4.2 \cdot 10^{-13}$ [16]	MEG	2016
$\mu^-\mathrm{Pb} \to e^-\mathrm{Pb}$	$4.6 \cdot 10^{-11}$ [17]	SINDRUM II	1996
$\mu^-\mathrm{Ti} \to e^-\mathrm{Ti}$	$4.3 \cdot 10^{-12}$ [18]	SINDRUM II	1998
$\mu^-\mathrm{Ti} \to e^-\mathrm{Ca}^*$	$3.6 \cdot 10^{-11}$ [19]	SINDRUM II	1998
$\mu^-\mathrm{Au} \to e^-\mathrm{Au}$	$7 \cdot 10^{-13}$ [20]	SINDRUM II	2006

Table 2: List of experimental results for lepton flavour violating muon decays.

2.1 The Muon Beam

A muon beam of 10^8 muons per second will be provided in a secondary beamline (figure 7) in the πE5 area of the HIPA accelerator at the Paul Scherrer Institute (PSI), where the experiment will be located [14]. The rate of 10^8 muons per second is necessary to collect the statistics required to archive the desired sensitivity for the branching ratio of $\mu^+ \to e^+e^-e^+$ in one year of data taking. It is also necessary that the beam consists of positive muons, since a measurement of the channel $\mu^- \to e^-e^+e^-$ would involve the complications of producing muonic atoms [8]. For Phase II of the Mu3e experiment a new beamline which provides a muon rate of $2 \cdot 10^9$ muons per second is planned [14].

2.1.1 The HIPA Accelerator

The high intensity proton accelerator (HIPA) [21] consists of a chain of three accelerators, which deliver a proton beam with a current of up to 2.4 mA at 590 MeV. The

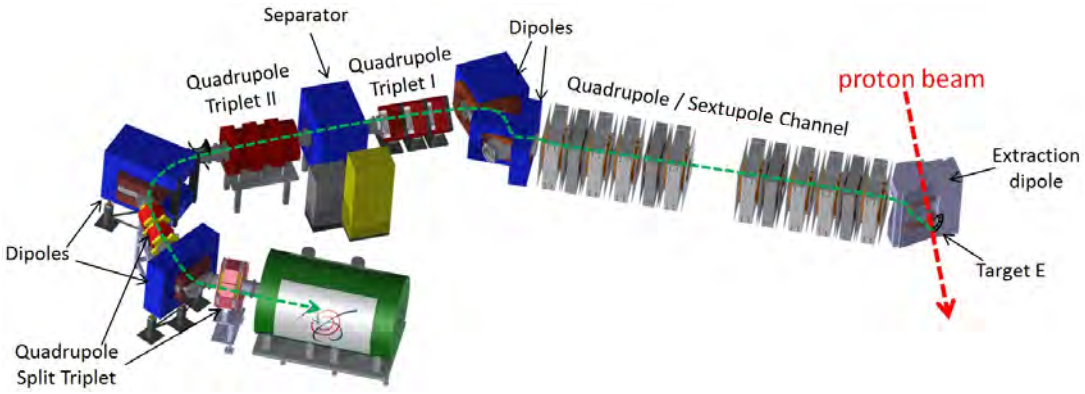


Figure 7: CAD model of the Mu3e beamline at PSI [22]

initial acceleration to an energy of 870 keV is provided by a Cockroft-Walton generator, followed by the acceleration to 72 MeV and 590 MeV in two ring cyclotrons.

2.1.2 Muon production

The protons from the HIPA are directed to a carbon target (Target E in figure 7), where they produce positive pions. These pions decay at rest on the surface of the target and produce muons via

$$\pi^+ \to \mu^+ \nu_\mu. \tag{1}$$

Due to momentum and energy conservation, these muons have a kinetic energy of 29.79 MeV:

$$E_\mu = m_\pi c^2 - p_\nu c = \sqrt{(m_\mu c^2)^2 + (p_\mu c)^2}, \quad p_\nu = p_\mu \tag{2}$$

$$\to p_\mu = \frac{(m_\pi^2 - m_\mu^2)c}{2m_\pi} = 29.79 \frac{\text{MeV}}{c} \tag{3}$$

A part of this energy is lost due to interactions in the target and thus the energy distribution of the muons that leave the target peaks around 28 MeV [14].

They are then collected by a set of focusing magnets with different orders and guided through a secondary beamline (figure 7). At the end of this arrangement the rate of 10^8 muons/s is provided in the πE5 area for the mu3e experiment.

2.2 Background processes

The suppression of background processes is the most important driver behind the design of the Mu3e detector, since other measurable SM contributions to the decay channel of interest do not exist. The most relevant background events for the Mu3e detector result from two different processes.

One of them is the radiative decay of a muon with an internal conversion of the photon into an electron positron pair with a branching ratio of $(3.4 \pm 0.4)\%$ [5] shown in figure 8.

$$\mu^+ \rightarrow \mathrm{e}^+ \nu_\mathrm{e} \overline{\nu}_\mu \mathrm{e}^+ \mathrm{e}^- \tag{4}$$

It can be distinguished from signal events with a measurement of the momentum of the electron and the two positrons, since momentum conservation requires them to add up to zero for muons decaying at rest.

$$\sum_{i=1}^{n=3} \vec{p_i} = 0 \tag{5}$$

This will not be the case for the radiative decay background, because some of the total momentum will be carried away by the two involved neutrinos, which will not be detected by the Mu3e detector.

The other relevant background is combinatorial background with a coincidence in space and time of two "normal" muon-Michel decays and a scattered electron from another source (e.g. bremsstrahlung) or a combination of a photon conversion or Bhabha-scattering with a Michel-decay. The probability for a detected interaction to contain this second background is, in contrast to the first example, rate depended, because it relies on the coincidence of independent processes. In order to suppress this kind of background, a precise timing information and vertex reconstruction is required, because the involved particles will not originate from the same interaction point.

Figure 8: $\mu^+ \rightarrow \mathrm{e}^+ \nu_\mathrm{e} \overline{\nu}_\mu \mathrm{e}^+ \mathrm{e}^-$ background process.

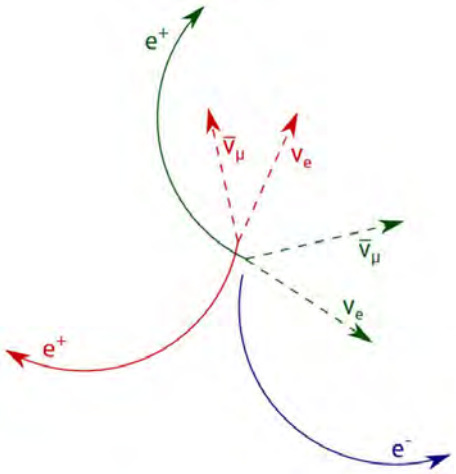


Figure 9: Combinatorial background process of the Mu3e experiment [23].

The desired sensitivity of 10^{-16} for the branching ratio of $\mu^+ \rightarrow \mathrm{e}^+ \mathrm{e}^- \mathrm{e}^+$requires a momentum resolution of 0.5 MeV/c [23] as well as timing information below the

nanosecond scale and good vertex resolution. These informations need to be acquired from the Mu3e detector to reach its sensitivity goal and are the reasons behind the design of the Mu3e detector described in the following sections.

2.3 Detector Concept

The phase I Mu3e detector consists of three cylindrical detector stations in a 1 T magnetic field. The central station hosts the hollow double cone target, a first double layer of pixel sensors, scintillating fibres and a second double-layer of pixel sensors. The pixel sensors are used for particle tracking to provide momentum resolution and to reconstruct the position of the interaction vertex. The scintillating fibres provide a precise timing information.

The muons are stopped inside the target and decay at rest. Some of their charged decay products will leave the central station through the pixel and fibre detectors and recurl, due to the magnetic field, into the central, upstream-recurl or downstream-recurl station, where their path and timing will be measured again with another double layer of pixel sensors and scintillating tiles. In an ideal case, a particle has therefore two precise timing measurements and six position measurements available to the reconstruction algorithms.

Particles with only one precise timing measurement might not be identified correctly as positron or electron, if both ends of their path intersect with the target. In this case, the direction of their recurl in the magnetic field is not measured and they cannot be identified.

The following sections will discuss the three sub-detectors (pixels, fibres and tiles) in more detail.

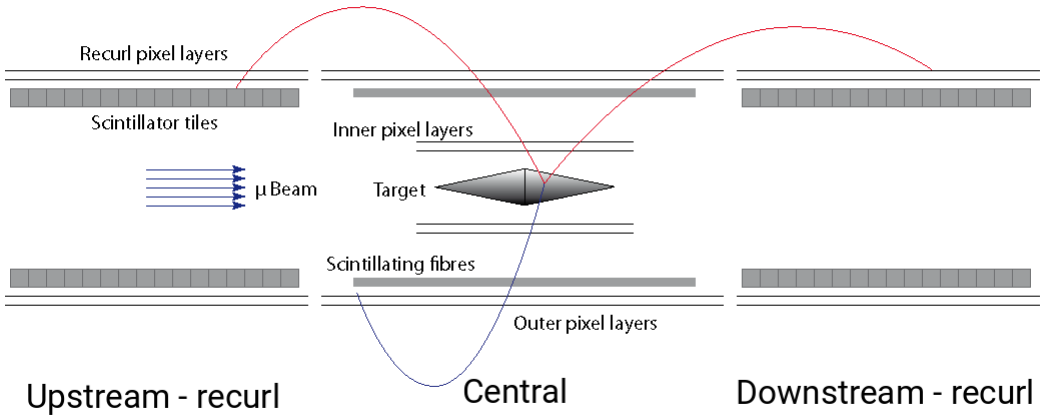


Figure 10: Schematic of the Mu3e detector showing the upstream-recurl, central and downstream-recurl detector station [22].

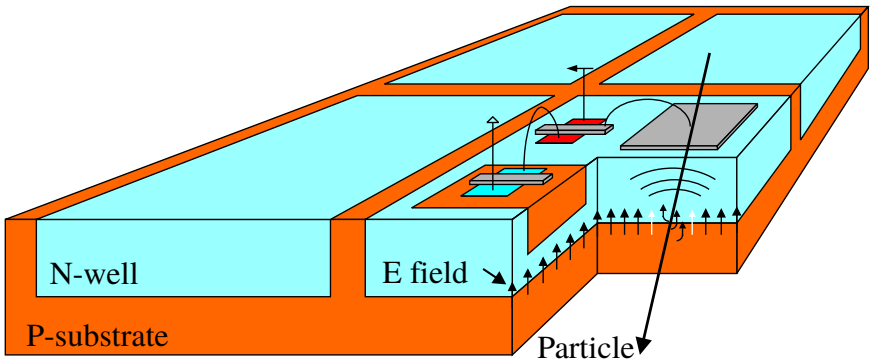


Figure 11: Schematic of a MuPix pixel sensor [24].

2.4 The MuPix Pixel Sensor

A pixel sensor consists of junctions of p-doped and n-doped semiconductor material. Conduction electrons from the n-doped material will combine with holes in the p-doped material and vice versa. The region between the n-doped and p-doped material is then free of charge carriers and a electric field from the p-doped to the n-doped material forms.

Any particles interacting in this depletion zone will form electron-hole pairs and lead to a measurable current through the p-n-junction, which can be measured by readout electronics of the pixel detector. A further increase of the size of the depletion zone can be gained, if a high reverse voltage is applied to the p-n junction. This increases the sensitivity due to the larger active area and also the time resolution because of a faster charge collection.

In order to meet the required momentum resolution for the Mu3e experiment, a new type of pixel sensor had to be developed. The momentum p of a charged particle can be measured with multiple layers of pixel sensors, by measuring the bending radius r of its path through a magnetic field B.

$$p = q \cdot r \cdot B \tag{6}$$

If one assumes a perfect knowledge about the magnetic field along the particle path, then the resolution of the momentum measurement depends on two factors. One of them is the physical pixel size of the used detector, the other one is disturbance of the path of the particle due to multiple coulomb scattering (figure 12).

In the Mu3e experiment, the energy of the particles is limited to the half of the muon mass, since the muons decay at rest. In this low energy regime, multiple scattering is the limiting factor for the momentum measurement and has to be minimised. This sets the constrain of a very low material budget upon the Mu3e pixel detector.

Such pixel detectors existed previously [25, 26, 27] and they achieved a low material budget by including readout electronics into the active pixel chip itself instead of bonding a second readout chip, but they did not allow to apply a reverse voltage, which is required to deal with the particle rates in Mu3e. A sensor was therefore developed

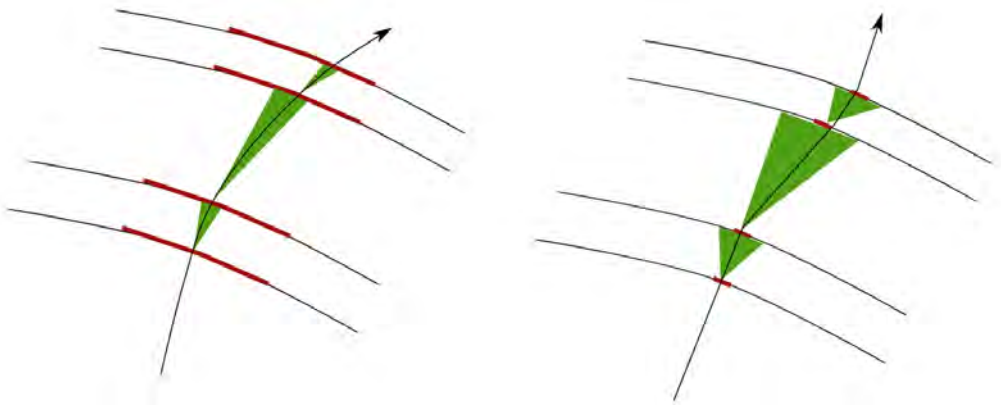


Figure 12: Schematic of a measurement with a pixel detector where the momentum resolution is dominated by the pixel size (left) and by scattering (right). The pixel size is shown in red, scattering effects are shown in green. Picture taken from [23].

in the Mu3e collaboration to combine the fast charge collection of a reverse-biased sensor with the low material budged of a sensor with included readout electronics.

The result of this development is the monolithic active pixel (MAP) MuPix sensor [24], which will have a size of 2x2 cm with a pixel size of 80x80 µm in its final version. It includes an amplification stage and line driver inside of each pixel and a digital periphery on chip, where the signal is digitalised and encoded.

The periphery also allows to set a per pixel tune value, that can be used to calibrate thresholds for all pixels of the sensor. This is relevant to have a handle on impurities in the semiconductor and other imperfections in the production process, which might change the response of single pixels relative to the rest of the chip. Furthermore, a reverse voltage of up to 100 V can be applied, the sensor can be thinned down to 50 µm and recent testbeam results have shown an efficiency above 99% and a time resolution of 11 ns [24, 28].

The sensor produces approximately 250 mW of heat per cm^2, which has to be actively cooled away with helium, since the MuPix needs to be operated in a atmosphere with low material budged. An increase in temperature would at some point lead to a dark count rate of the pixel sensors, since thermal fluctuations can produce electron hole pairs in the depleted region. It might also melt the glue that holds the sensor on the support structure.

In the final detector, the sensors will be glued to ladders on kapton strips, and mounted in cylindrical layers in each station. Another important topic related to the operation in the Mu3e experiment is the alignment of the single sensors and stations to each other, since the momentum measurement requires a precise knowledge of the position of each sensor [29]. Also, radiation damage has been observed and its effects on the lifetime and efficiency of the sensor are under investigation [30].

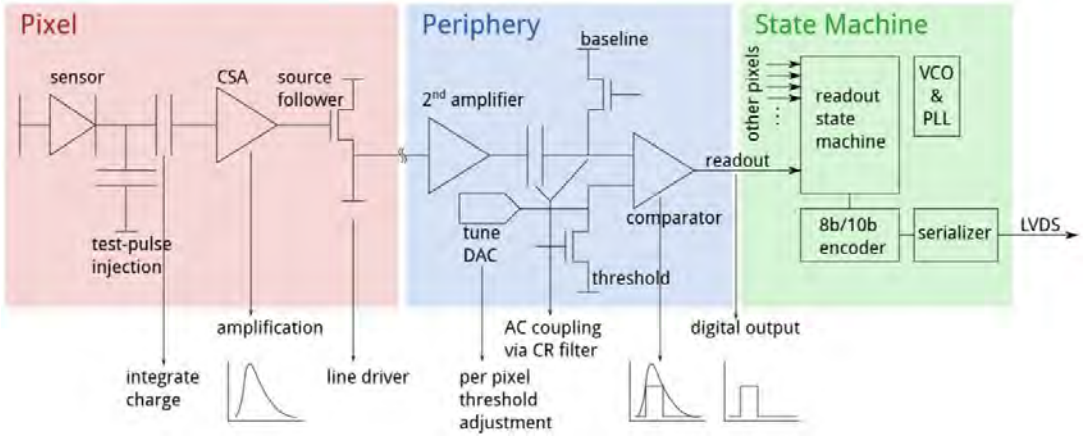


Figure 13: Schematic of the electronics that are implemented into the MuPix chip. The periphery and state machine have a dedicated, inactive area on the chip, which will overlap with the active area of the chips from a subsequent module. The components shown in red are implemented into each pixel of the chip [22].

2.5 Scintillating Fibres

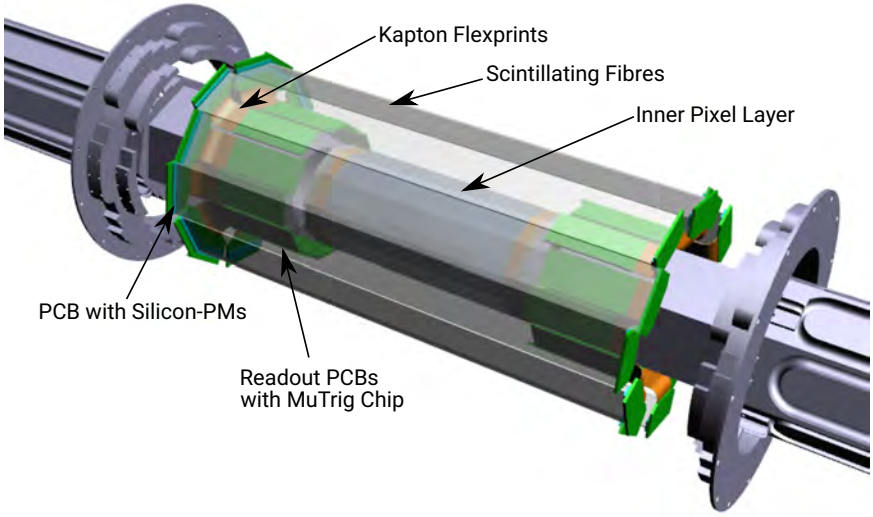


Figure 14: CAD drawing of the scintillating fibres in the central station of the Mu3e detector. The inner double layer of pixel sensors is visible through the fibres, which are transparent in this drawing. The outer double-layer of pixel sensors will be mounted around the fibre detector and is not displayed here. Adapted from [22].

The scintillating fibres are responsible for the timing measurement in the central station. They consist of a scintillator material, which gets excited into a higher energy level by the interaction of ionising radiation. In the de-excitation, the population of this excited energy state decays into the vibration substructure of the ground state and emits light with visible wavelengths in this process. The emitted light is then guided by internal reflection along the fibre and detected by silicon photomultipliers at the ends of the fibre. From there the signal is read out with the MuTrig ASIC, which is located on the mounting PCB.

This kind of detector is used in a single layer in the central station of Mu3e in a cylindrical shape with a radius of about 6 cm and a sensitive length of 29 cm [31].

The introduction of the fibre detector adds 0.3 % of the radiation length to the total material in the path of the particles. A single layer of pixel detectors corresponds to roughly 0.1 % of the radiation length [23].

Recent results have shown a time resolution for the fibre detector of <280 ps [32].

2.6 Scintillating Tiles

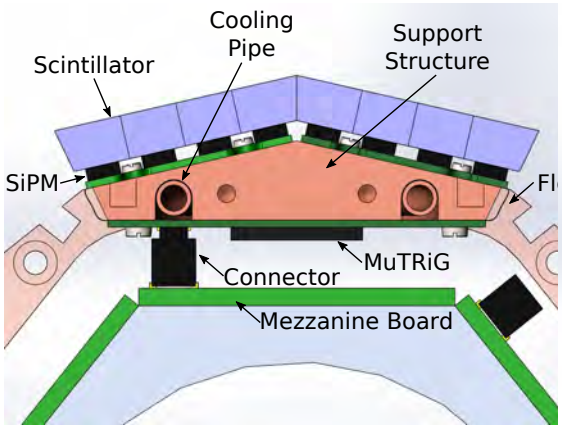


Figure 15: Schematic of a cross-section through a tile module. Scintillating tiles are mounted on a PCB with silicon photomultipliers.

Figure 16: CAD drawing of a Mu3e recurl station with tile detector modules

The tile detector is also built from scintillating material. It is located in the innermost layer of the upstream and downstream recurl stations and is the last detector that detects particles with a full track[1]. The radiation thickness of this part of the Mu3e detector is therefore not critical, which allows to segment this subdetector into single tiles with a size of 6.5x6x5 mm^3, each connected to a dedicated silicon photomultiplier.

[1] Six position measurements from pixel sensors and two time measurements from fibre and tile detector

The photomultipliers are then connected with a kapton flexprint to the MuTrig readout chip and a common readout PCB, which extends over the whole length of the recurl station. This part of the Mu3e detector is cooled with water, since the material budget is not relevant on the inside of the recurl stations.

Prototypes of this detector have shown a time resolution of 56 ps in testbeam campaigns, which would be sufficient to meet the requirements for the Mu3e experiment [33].

3 Field Programmable Gate Arrays (FPGAs)

The data from all parts of the Mu3e detector is collected and processed by a common data acquisition system, which will be discussed later in this thesis and is heavily based on field programmable gate arrays (FPGAs). This chapter will discuss the basic operating principles of these devices.

3.1 Hardware

Information processing systems can be classified into three different categories [34]:

- free-programmable systems
- hardwired systems
- reconfigurable systems

In a free-programmable system a software or "instruction flow" determines the sequence of operations performed by the system and the hardware structure is independent of the task. An example for this would be a central processing unit (CPU) used in a normal PC.

Hardwired systems are designed for a specific task. It is not possible to change the behaviour of such a system and the hardware structure is a result of the problem the device has to solve. The MuPix-Chip (or any other ASIC) is an example of a hardwired system.

The difference for the user between these two systems is the flexibility and performance[1]. A CPU or "general purpose processor" requires only a different software to change the problem solved by it. This flexibility advantage of the CPU comes with the penalty of a lower performance compared to an ASIC, since the device is not optimised for a specific task and has to read its instructions from memory.

In a reconfigurable system, like an FPGA, a change of the configuration of its hardware structure is possible. This allows to keep some of the performance advantages of hardwired systems, while still having a device which can be configured to solve arbitrary problems. In a direct comparison of the performance of an ASIC and a FPGA, the ASIC will still show better performance because the additional hardware needed for a reconfigurable system is not present in an ASIC.

[1] the exact meaning of "performance" in this context will become clear later in this chapter.

3.1.1 Transistors

All modern versions of the devices above are fabricated using lithographic methods to implement transistors on the surface of semiconductor material, usually silicon. A transistor is an element of an integrated circuit that can act as a switch. In the semiconductor material, this behaviour can be achieved with two n-doped wells in a p-doped substrate with an insulated gate above the two wells (figure 17).

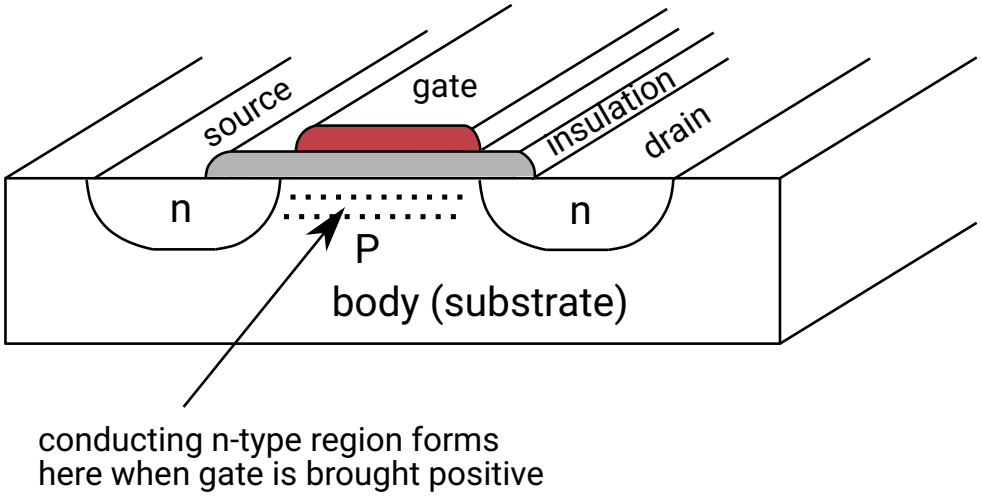


Figure 17: n-channel MOSFET Transistor. Adapted from [35].

A positive voltage applied to the gate increases the electron concentration beneath it and forms a conducting region in the substrate between the source and drain well. The gate acts as a switch between source and drain.

A similar behaviour is realised with the p-channel MOSFET with two p-doped wells in a n-doped substrate. In contrast to the n-channel MOSFET shown in figure 17, the p-channel MOSFET is conducting when 0 V is applied to the gate.

3.1.2 Logic Gates

With a combination of transistors (section 3.1.1), logic elements can be implemented into the silicon material. An example of such a combination is shown in figure 18. The output forms a boolean function which is only 0 if all inputs are 1, which is called a NAND gate.

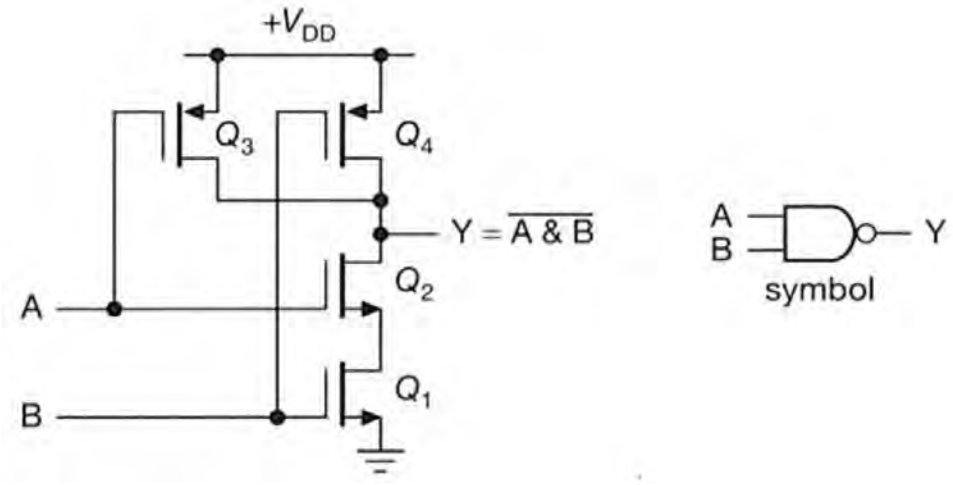
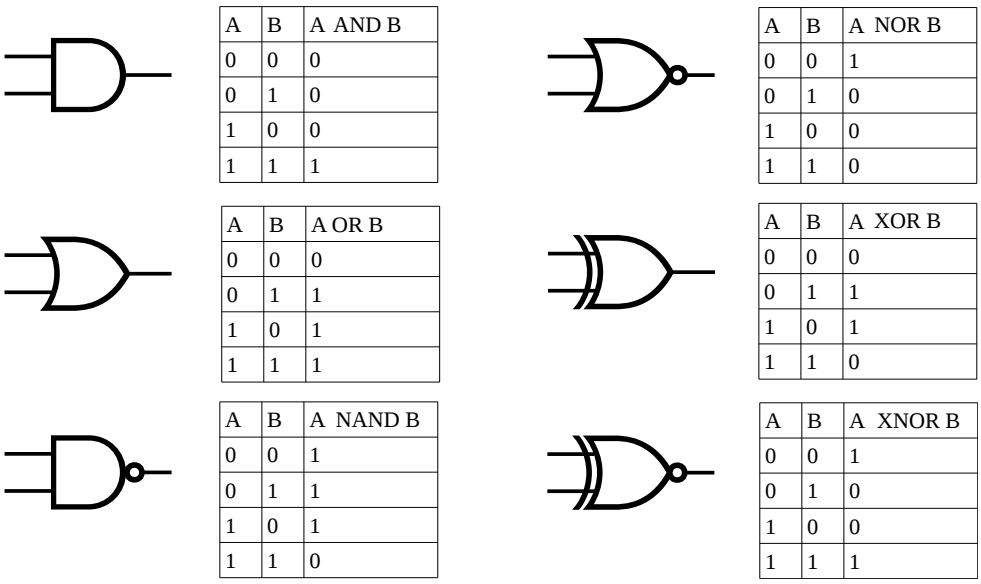


Figure 18: Implementation of a NAND gate. When both inputs are 1, the n-channel transistors Q1 and Q2 are conducting and pull the output to ground. If one of the inputs is 0, either Q3 or Q4 (p-channel transistors) are conducting and the output is 1 [35].

Other combinations of transistors and also combinations of gates can be used to implement the complete set of logic gates (figure 19) and therefore arbitrary boolean logic into the silicon material.



A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

A	B	A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

Figure 19: Logic gate symbols and their truth tables.

3.1.3 Flip-Flops

A flip-flop is a logic circuit used to store the value of its input signal, and it can be built from logic gates, which were discussed in section 3.1.2. There are many types of flip-flops. An unclocked flip-flop is called latch and stores the value of its input if an enable or set signal is high.

A clocked flip-flop is called register and is used to store the value of a signal on the arrival of a clock edge. The input signal is sampled on the arrival of the edge and the output is then held on this level until the arrival of the next clock edge.

The clocked flip flop is of high importance for the design of digital logic and is shown in figure 20. The use of unclocked flip flops should be avoided in the context of FPGAs in most cases[2].

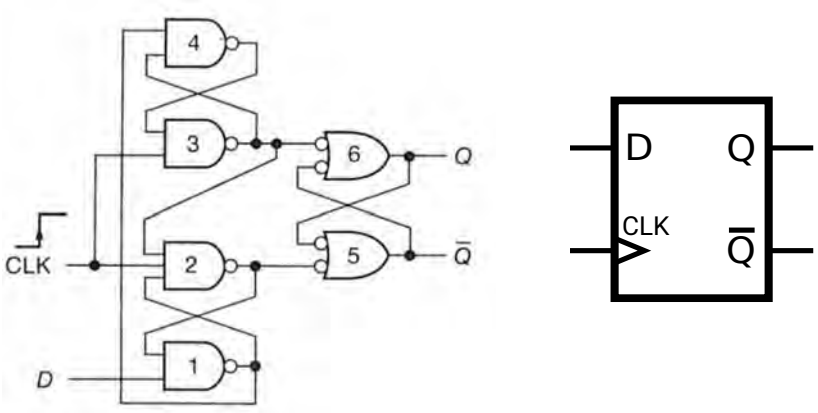


Figure 20: Clocked, positive edge triggered flip-flop build from logic gates [35] and circuit symbol.

3.1.4 Synchronous Systems

Combinations of flip-flops (Figure 20) and logic gates (3.1.2) can be used to build synchronous systems, where the same clock source drives all clock inputs of the used flip-flops[3].

A simple example of logic built as a combination of registers and logic gates is shown in figure 21. In an initial state where clk and signal_in are on a logic level 0, the output is also on a level of 0. A transition of signal_in from 0 to 1 will cause also the output to rise to a logic level of 1. However, the next clock edge will lower the output back to 0 when the register assigns D to Q. A rise of signal_in was therefore converted into a single pulse on the output and figure 21 represents an rising edge detector. If the edge of signal_in is synchronous with the clock edge (which is the case if it is also driven by other combinations of registers and logic gates), the output pulse will also

[2]The reason for this will be explained in the next section.
[3]I will call them registers from now on.

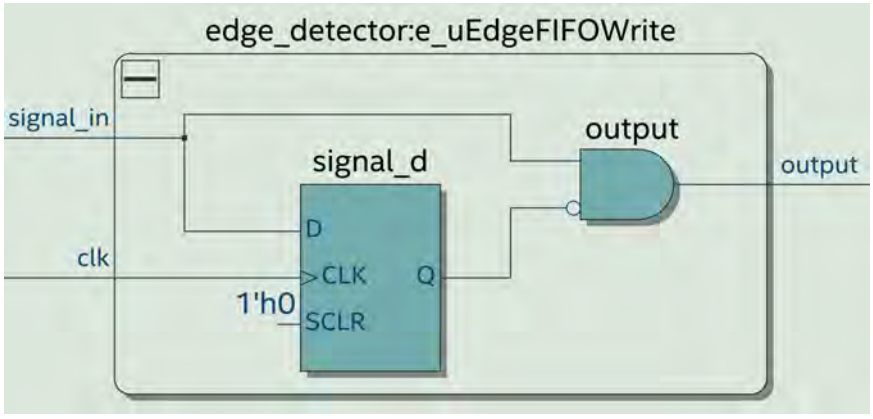


Figure 21: A simple example for a combination of a flip-flop with logic gates. This picture was taken from the Quartus Prime RTL Viewer.

be synchronised to the clock edge and its length will be identical to the length of one clock cycle. The output can then be used to drive further synchronous logic and build larger and more complicated synchronous systems.

This kind of system has very desirable properties. All the action takes place at the time of a clock pulse, and the system transits into a new stable state based on the state of the system right before the clock pulse [35]. This principle allows to construct logic sequences and is the foundation that CPUs, in general digital electronics and also FPGAs are built upon.

In this context, the term "performance" means the maximum clock frequency where a device is able to function. The data processing speed is a direct result of this frequency and it is limited by the time signals need to travel between different logic elements. If this time is larger than one clock cycle[4], synchronous operation is not possible. Minimising the travel time therefore increases the performance and can be achieved by an increased density of logic elements, which is after all just a denser packing of transistors in the silicon substrate.

3.1.5 Configuration of FPGAs

The difference between an FPGA and a hardwired system is that an FPGA allows to change the connections between different logic elements. The configuration for these connections is stored in static random access memory (SRAM) cells and has to be loaded into the device in order to implement the desired behaviour. This has to be done on each power up of the device, since the SRAM is volatile memory and the information is lost when the FPGA is powered down. An FPGA is usually reconfigured either via a serial connection from a PC or from a flash memory that contains the programming file.

[4] more restrictions in 3.2.

3.1.6 Logic Modules and LUTs

An FPGA is usually constructed from many identical copies of a base logic block or adaptive logic module (ALM) and the configuration process consists of the configuration of the interconnects between different ALMs and the configuration of the logic inside of each single ALM.

The design of an ALM can vary between manufacturers and also between different devices of the same manufacturer. A block diagram of the Altera ALM is shown in figure 22.

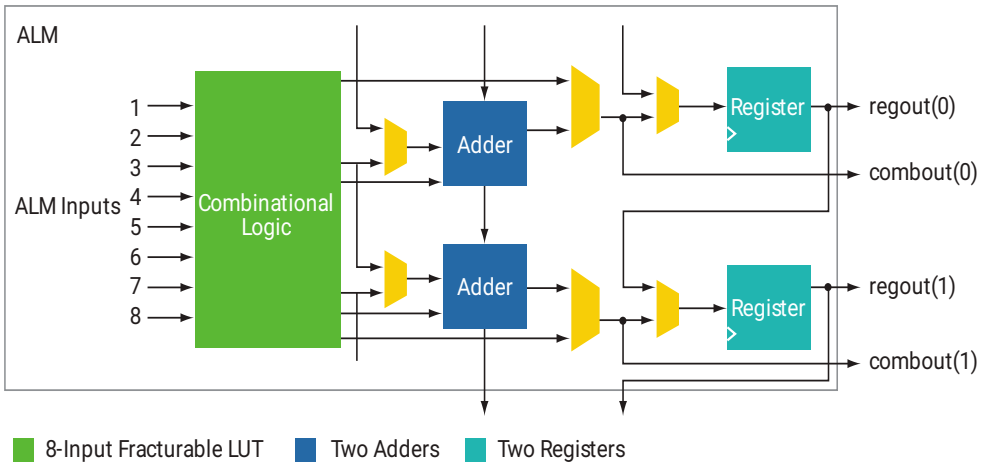


Figure 22: Block diagram of a Altera adaptive logic module (ALM) [36]. A detailed schematic of a ALM from a StratixIV FPGA can be found in figure 55 in the appendix.

Depending on the configuration, only a part of the elements shown in figure 22 are actually used. For logic paths with many levels of logic between one register and the next, only the combinational logic part of the ALM may be used and directly routed from combout(0) or combout(1) in figure 22 to the input of a logic block of another ALM.

The combinational logic in this block is implemented with a lookup table (LUT) which can be configured by the user by setting the LUT-Mask in SRAM memory (figure 23). A LUT with 2^n entries can implement any function of n inputs.

The LUT used in the Altera ALM is however only a full 6-input LUT, but it allows to implement two separate 4-input LUTs or two larger LUTs[5], if some inputs are shared between them. This is also the reason why the ALM contains two registers instead of one and is, according to Altera, more efficient in terms of space on the FPGA compared to single register ALMs [36].

[5] e.g. 5-LUT & 4-LUT, 5-LUT & 5-LUT

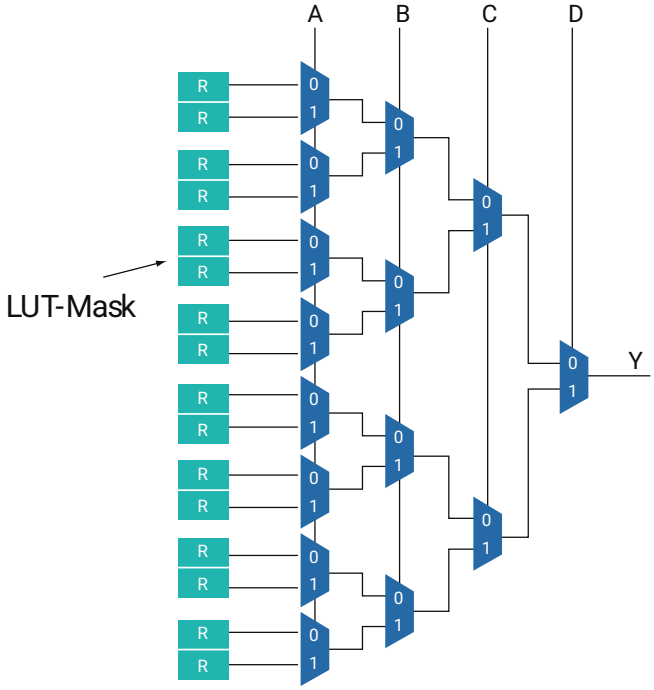


Figure 23: Block diagram of a lookup table. The LUT-Mask is stored in SRAM cells and configures the logic for the calculation of Y from the inputs A, B, C and D. The shown LUT can implement any boolean function of A, B, C and D [36].

3.1.7 FPGAs used in this thesis

There is only a handful of manufacturers that produce field programmable gate arrays. The two large ones are Altera[6] and Xilinx and then there are a few smaller companies with specialisation on devices with very low power consumption, radiation hardness or other special properties of their devices.

For this thesis FPGAs from the series ArriaV, Arria10, StratixV and Max10 from Altera and Kintex-7 from Xilinx are used on custom designed printed circuit boards (PCBs) or evaluation boards. The final readout system of Mu3e Phase I will contain the 241 FPGAs listed in table 3[7].

[6] Now owned by Intel
[7] Status September 2019.

Number of used FPGAs	Name	Version Number	Manufacturer
112	ArriaV	5AGXBA7D4F31C5N-ND	Altera
112	Max10		Altera
16	Arria10	10AX115N2F45E1SG	Altera
1	Kintex-7	XC7K325T-2FFG900C	Xilinx

Table 3: List of FPGAs used for the Mu3e Phase I readout system (status September 2019). ArriaV and Max10 FPGAs are used together on the same board.

3.2 Timing

All registers in an FPGA have requirements on the timing of an incoming signal to ensure proper operation. These requirements are given relative to the arrival time of the clock edge at the register. An arriving signal has to be stable for a setup-time t_{SU} before the clock edge and for a hold-time t_H after the clock edge (figure 24).

For a particular design, this is usually achieved with optimisation algorithms in the design software of the FPGA manufacturer, which tries to synthesise the VHDL or Verilog description written by the user into a design which meets timing requirements of the target device. However, timing optimisation is not guaranteed to succeed in this task since this software is also constrained by the available space and routing capabilities of the device. A design with a size closer to the entire available space of the target FPGA is therefore also more likely to violate setup or hold times because user logic cannot be placed in a way that a nearly empty FPGA would allow to do.

A failed timing closure of the design software does not necessarily mean that the synthesised design is not functional. When programmed into the FPGA, the timing performance depends on things like the temperature of the device, where the synthesis software assumes worst-case scenarios and includes security margins. In Quartus Prime[8] it is possible to analyse timing performance for user defined device temperatures, which allows to determine operating conditions where this design would be functional. If timing is close to succeeding, it is also possible to restart the process automatically with a set of different seeds for the optimisation. This can, in some cases, fulfil timing requirements without any change in the user logic because the optimisation might stop in a different local minimum.

Apart from a failed timing optimisation, there are also other reasons why the setup or hold time could be violated. The first one is an undefined arrival time of the incoming signal at the register. This is the case for all signals not synchronised to the register's clock. A button pressed by a human, for example, has the chance to violate timing requirements because its arrival time is undefined relative to the clock edge and is not necessarily stable for t_{SU} and t_H. Also problematic are all transitions between clock domains. A clock domain is a block of logic that is driven by the same clock. Many applications require having multiple clock domains and also signals between these clock domains. Such a signal also has a certain chance of violating setup or hold

[8]The design software from Altera used for most of the firmware written during this thesis.

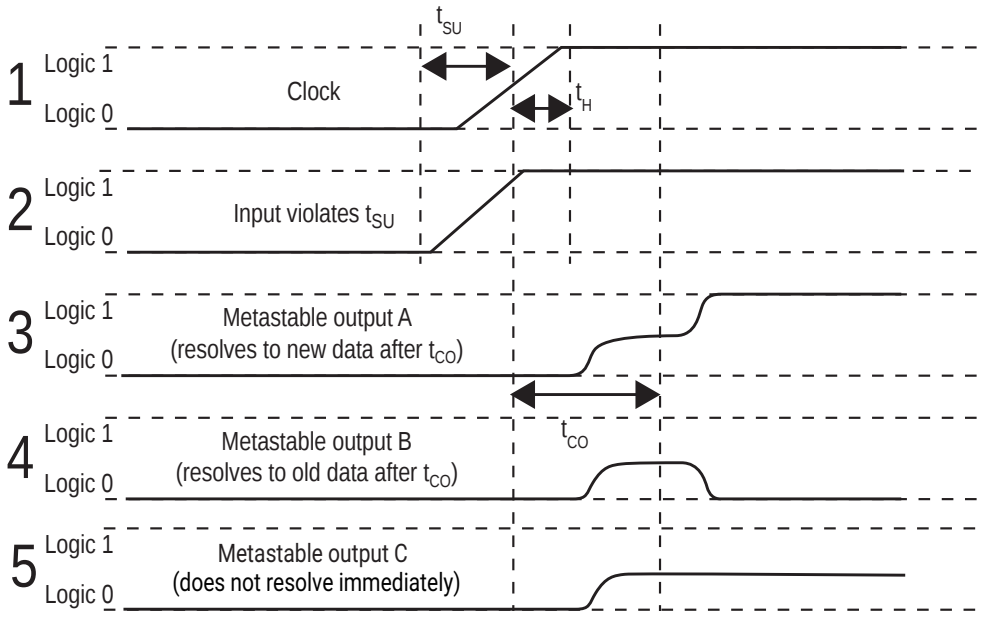


Figure 24: The Input in line 2 violates the setup time t_{SU} relative to the clock edge in line 1. Possible results are a metastable state that resolves to Logic 1 outside of the time window t_{CO} (line 3) or a metastable state that resolves to Logic 0 (line 4). In a very unlikely case, the output can also stay in a metastable state for a longer time period (line 5). Adapted from [37].

times since the arrival time at the register can shift due to a different frequency of the clock domain of its origin. Even if the clocks of two clock domains have the same nominal frequency, slight deviations of the frequency of the two oscillators that drive these two domains will still lead to timing violations. Methods for the transitions of signals into a different clock domain will be shown in section 3.2.2.

3.2.1 Metastability

If the hold (t_H) or setup time (t_{SU}) of a register is violated due to one of the reasons described above, this register might go into a metastable state. In regular operation, the output voltage of a register is on a defined logic level of 0 or 1 after the clock-to-output delay t_{CO}. With a timing violation (line 2 in figure 24), the output voltage can also be somewhere between the two logic levels, which is called a metastable state. This metastable state will usually resolve quickly into one of the defined logic levels. For small violations of t_{SU}, the output is more likely to resolve into the intended logic level (line 3), and for larger violations the output will most likely resolve to the previous logic level (line 4). The exact timing of the input signal determines the resulting logic level and also the time required for the output voltage to resolve to this level. In a very unlikely case, the register can also stay in a metastable state for a longer period of time. However, this state will also resolve into a defined logic level

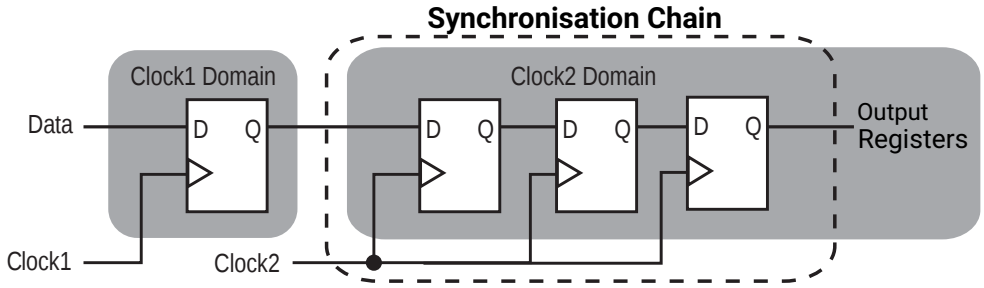


Figure 25: 3-level synchroniser chain. Adapted from [37]

due to small voltage fluctuations.

All results of a timing violation are not guaranteed to meet the specified clock-to-output delay and the output of the register is therefore also at risk to violate the timing requirements of any following logic driven by it. Timing violations have the ability to propagate through a system and can cause failures much larger than the undefined output of a single register [37].

3.2.2 Clock Domain Transitions

In order to avoid any failures due to timing violations, one has to divide the design into clock domains. Timing requirements inside of each domain are checked and optimised by the design software. Failures require a change in the user logic or a restart of the optimisation with a different seed.

Transitions between two clock domains require synchronisers to synchronise all signals to the new clock safely. A synchroniser is a chain of registers driven by the clock of the destination clock domain. The output of each register is only connected to the input of the next register in the chain. In a configuration like this, a single register that succeeds in delivering a valid signal to its successor causes all following registers to meet timing requirements. The input is therefore synchronised to the destination clock by manually inserting registers and increasing the available time for a metastable state to settle on a logic level. As discussed in section 3.2.1, the time that a metastable state needs to resolve depends highly on the exact timing of the input and is therefore subject to statistical fluctuations. Even with a synchroniser chain, system failures are still possible. The probability for such a failure is calculated for all identified synchroniser chains by the design software and expressed in a value called MTBF (mean time between failures).

The overall failure rate of the design can then be calculated by using:

$$failure_rate_{design} = \frac{1}{MTBF_{design}} = \sum_{i=1}^{number\ of\ chains} \frac{1}{MTBF_i} \tag{7}$$

Where the individual mean time between failures of a single chain is:

$$MTBF_i = \frac{e^{\frac{t_{MET}}{C_2}}}{C_1 \cdot f_{CLK} \cdot f_{DATA}} \tag{8}$$

C_1 and C_2 are constants and depend on the device and operating conditions, f_{CLK} and f_{DATA} are the frequencies of the destination clock and input data and t_{MET} is the sum of all output timing slacks[9] of the registers in the chain [37]. A longer synchronisation chain therefore increases the MTBF exponentially.

3.2.2.1 Synchronisation of Parallel Signals

Synchroniser chains are not directly usable for parallel signals. Applying a synchroniser chain to each bit of the signal will synchronise each bit to the new clock, but will not necessarily provide the full parallel signal in the same destination clock cycle, which makes it unusable. This problem can be avoided by using a dual-clock FIFO (first in - first out) data buffer. More details about FIFOs and how they can be used to do clock domain transitions will follow in section 3.3.1. In summary, the parallel data is written to buffer memory, some control signals are transmitted to the target clock domain using synchroniser chains, and the memory is read from the target domain once the control signals have arrived.

3.3 IP-Cores

An intellectual property core or IP-core is a block of logic (for example a FIFO) that can be included into FPGA firmware.

In this section a few IP-cores will be discussed, since they represent important building blocks of the firmware developed during this thesis and are necessary to understand some of the topics in the following chapters. We will start with the first in/first out buffer (FIFO) in order to clarify their use for synchronisation of parallel signals mentioned in section 3.2.2.1.

A IP-core is either provided as a hard-IP-core, integrated as a unchangeable copy into the FPGA, or as a soft-core, where the logic can be implemented into the user part of the FPGA if necessary.

Hard-IP-cores can be seen as an ASIC inside of an FPGA. They also come with the performance improvements of an ASIC and require less space than soft-IP-cores, but cannot be removed in designs which do not need this specific IP-core.

Soft IP-cores on the other hand are provided as source code for open source cores or as netlist to protect the source code of a licensed core.

3.3.1 FIFOs

FIFOs are used to store or buffer data, or to synchronise signals between different clock domains. They have a read and a write side and signals written to it will exit

[9] Slack is the available time until the setup or hold time of the next register would not be fulfilled.

the FIFO on a command on the read side in the same order that they where written to it. The data is stored either in on-chip random access memory or in logic elements, depending on the configuration of the IP-core. The read and write side of the FIFO can run on the same or on different clock domains. In the case of different clock domains, the internal control signals on the write side of the FIFO are synchronised with synchroniser chains to the read side and provide the output control signals. A read command to the memory is processed synchronous to the clock of the read side of the FIFO.

However, since the control signals are synchronised with a synchroniser chain, every double sided FIFO has a limited mean time between failures (MTBF), which scales with the inverse clock frequency squared[10] according to equation 8.

The size of the synchroniser chains is usually a parameter in the configuration of the IP and determines not only the MTBF, but also the delay between a write to the FIFO and the availability of the data on the read side.

3.3.2 Phase Locked Loops (PLLs)

Phase locked loops are used to derive different clock frequencies from a reference clock. They are implemented as hard-IPs and the maximum number of PLLs in a design and also their position is limited by the target device.

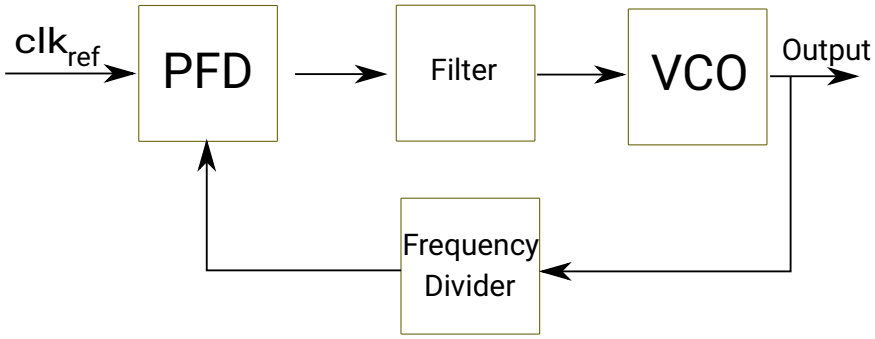


Figure 26: Simplified schematic of a phase locked loop.

The phase and frequency of the reference clock of a phase locked loop (PLL) is compared to a clock generated by a voltage controlled oscillator (VCO) in the phase-frequency detector (PFD). This component generates an output that determines if the VCO frequency should run faster or slower, which is then converted in a filter or charge pump to an input voltage to the VCO. The output of the VCO is the output clock of the PLL.

The frequency of the PLL can be adjusted by inserting a clock divider between the VCO output and the PFD. Also non-integer relations of the frequency of clk_{ref} to the output frequency are possible, if the setting of the frequency divider is dynamically changed between two or more values. This is then called a "fractional" PLL (fPLL).

[10] $f_{CLK} \cdot f_{DATA}$

The frequency divider divides, for example, a specific fraction of clock cycles to frequency x, the others to frequency y and the mean of the resulting input voltage to the VCO will cause it to run with a frequency z, which has a non-integer relation to the frequency of clk_{ref} [38].

3.3.3 Soft Core Processors

Soft core processors are a implementation of a CPU in the configurable logic of a field programmable gate array. It is possible to add different peripherals, like memory, io-ports or interrupt inputs to the IP. The software for this type of processor is compiled outside of the FPGA and uploaded as binary file together with the configuration of the FPGA. Multicore systems can be implemented if the available space on the FPGA is sufficient.

The processor used in this thesis is called NIOS II and was developed by Altera for the use in their FPGAs. It is based on a 32-bit RISC architecture [39].

Data-transfers between a soft core processor and user logic in the FPGA usually require some kind of buffer in form of dual port memory or a FIFO, since the latencies of a processor are unreliable and depend on the software that is currently executed.

3.3.4 Memory

An FPGA contains also memory blocks in addition to the logic modules (ALMs) shown in figure 22. These memory blocks are used for example to implement a FIFO or memory for a soft-core processor.

One can also configure them in a standalone IP core to act as a random access memory (RAM) or read only memory (ROM). The memory content can be accessed after a few clock cycles, if the correct address is provided to the input of the IP.

3.3.5 Transceiver

Another very important type of IP cores are transceivers. Their function and specific types used in this thesis will be discussed in 4.4.

4 Data Transmission

A data acquisition system is used to transport, process and finally store data. Some of the protocols and interfaces used in the Mu3e DAQ are briefly explained in this chapter. Not further explained here, but also relevant for Mu3e are SPI [40], I2C [41], graycode [42] and JTAG [43].

4.1 parallel and serial communication

There are in principle two ways to transmit data on the bit level: parallel or serial. In a serial protocol, the data bits are transmitted one after the other over a single line. In parallel communication, multiple lines are used and the data is transmitted in n-bit wide words.

Parallel communication offers a higher bandwidth compared to serial communication if identical frequencies are used. This comes with the complication, that each bit of a n-bit parallel word has to be clearly assigned to the same parallel cycle in order to receive the correct data word. A difference in latency between the single lines in a parallel interface can, therefore, make it impossible to receive the data in the correct order. Other disadvantages of a parallel interface are the increased size due to the larger number of wires and the possibility of crosstalk between these wires. Crosstalk is the distortion of signals that especially densely packet digital lines can cause in their neighbouring cables.

4.2 LVDS

LVDS (low-voltage differential signaling) is a widely used signal standard that is also used at many points in the Mu3e DAQ. An LVDS signal uses a differential pair of cables to transmit data. The ends of these cables are connected at the receiver with a specific resistance and the driver sinks or drives a specified current into both connections. Data is then received by measuring the voltage drop over the resistance of the receiver, which is a fixed positive or negative value due to the defined driver current and resistance of the receiver [35].

LVDS signals are less affected by electromagnetic noise, since both wires are routed together and will have similar noise pick-up, which does only shift the overall voltage level but not the voltage drop over the receiver resistance.

4.3 8b10b

8b10b [44] is an encoding scheme, where all possible 8 bit words are encoded into 10 bit words. The additional two bits are used for bit error control and to achieve that an equal number of ones and zeros are transmitted. Any number of transmitted bits which is larger than 20 is guaranteed to have a difference between transmitted ones and zeros of 2 or smaller [35]. This leads to the result of a effective current flow of 0 on the transmitting cable.

In addition to this, also a transmission of 6 or more sequential 1s or 0s is not possible with the 8b10b protocol and the therefore guaranteed regular logic level transitions allow to recover the transmitter clock from the serial data stream on the receiver side.

When the restrictions above are implemented, the 12 symbols shown in table 4 are left from the additional possibilities gained from the two bits.

Name	Hex-Value	10b code with positive disparity
K.28.0	1C	1100001011
K.28.1	3C	1100000110
K.28.2	5C	1100001010
K.28.3	7C	1100001100
K.28.4	9C	1100001101
K.28.5	BC	1100000101
K.28.6	DC	1100001001
K.28.7	FC	1100000111
K.23.7	F7	0001010111
K.27.7	FB	0010010111
K.29.7	FD	0100010111
K.30.7	FE	1000010111

Table 4: The 12 8b10b control symbols [44].

These 12 symbols are called control characters and can be used to transmit link control commands, since they cannot appear in normal data if the serial link is correctly transformed into a parallel signal. The sequence 1100000, which is present in the symbols K.28.1, K.28.5 and K.28.7, is a unique sequence of bytes that cannot appear at any point in the serial data. This sequence can therefore also not appear in the parallel signal, regardless of a correct alignment. This is the reason why the control symbols that contain this sequence are used to align the parallelisation of the serial link.

8b10b encoding is used for many modern data links, such as Ethernet, HDMI, SATA, USB3.0, DVI and DisplayPort [45] and is also used for all fast data links in the Mu3e data acquisition.

4.4 Transceivers

Transceivers are used to receive and transmit data. Because the Mu3e DAQ is based on Altera FPGAs, their structure on these devices will be explained here, but similar principles also apply to other implementations.

On Altera FPGAs transceivers are implemented as hard-IP-cores, which are hard-wired to specific pins of the FPGA. It is, therefore, not possible to route the serial output and input signals freely through the FPGA user logic. The transceivers are divided into a physical medium attachment (PMA) and physical coding sublayer (PCS) block and the following explanations of these blocks are based on [46].

4.4.1 PMA Block

The PMA block of the transceiver is directly connected to the physical output or input pin of the FPGA. Received data is feed into the channel PLL, which is located in the transceiver PMA. This PLL is able to recover a synchronised clock from the data stream with the serial bit frequency, because the data has a guaranteed amount of level transitions due to the 8b10b encoding.

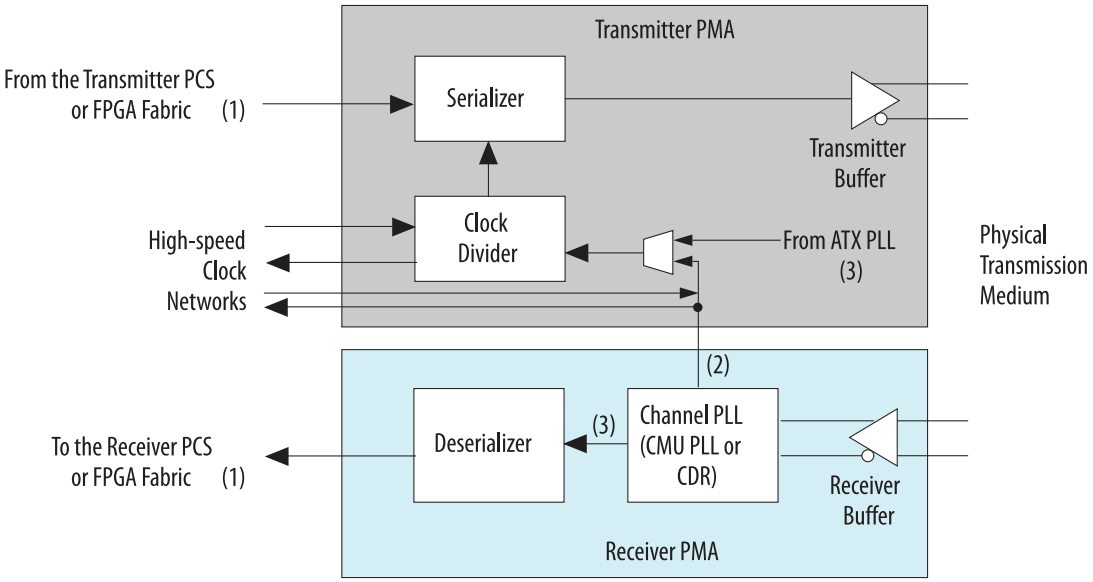


Figure 27: Diagram of a PMA block of an ArriaV transceiver [46]

The clock is then used in the deserialiser to convert the serial data into a parallel data signal, which is then forwarded to the PCS block. In the PMA, the word alignment of the parallel data is chosen randomly[1]. The recovered clock is also sent to the transmitter PMA, where it is divided into a clock with the frequency of the parallel data and in some configurations[2] also to drive the serialiser in the transmitter PMA.

[1] At least for standard configurations of the transceiver. More about this will follow later

[2] More about the effects of different configurations will follow during this thesis

4.4.2 PCS Block

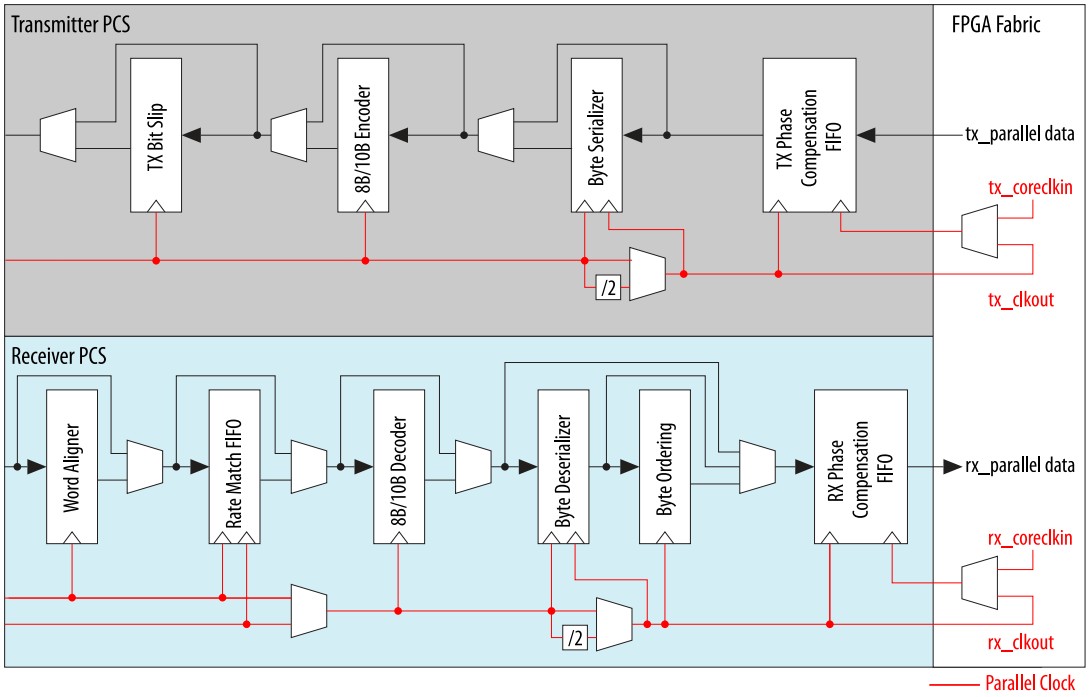


Figure 28: Diagram of a PCS block of an ArriaV transceiver. Adapted from [46]. The left side is connected to the PMA block of the previous section.

The PCS block contains digital processing logic for the received data before it is given to the user in the FPGA fabric. Data from the PMA block is word aligned using the control characters from section 4.3 and then 8b10b decoded. For some applications where a very wide interface is wanted, the rate match FIFO and byte ordering blocks are needed and convert the output of the word aligner into a wider parallel signal. In order to connect the received data to the FPGA fabric, it is written to a phase compensation FIFO to safely synchronise the data into user clock domains.

However, if the clock frequency of this clock domain is slightly smaller than the clock of the device which has transmitted the data[3], then this is a point where data could potentially be lost because of the limited size of the phase compensation FIFO.

The transmitter PCS is much simpler compared to the receiver. Data from the FPGA fabric is written to a phase compensation FIFO, which is read with the divided clock from the channel PLL in the PMA. The bytes are serialised in cases with very wide interfaces and then 8b10b encoded and sent to the PMA block.

[3]For example due to slight variations of the frequencies of the oscillators

4.5 PCIe & DMA

The use-case of peripheral component interconnect express (PCI express) interfaces in the Mu3e data acquisition will be explained in the next chapter. PCIe is used as connection to different peripherals on the motherboard of a PC (graphic cards for example) and some FPGA boards are able to act as such a peripheral of the PC motherboard. PCIe is, in contrast to its older version PCI, not a bus. Each component is connected with a set of differential wires to the PCIe network. This network can be compared to an ethernet network with switches (north- and southbridge for the PCIe case) which connect different devices (periphery components). In this network packets are transmitted with the format shown in table 5 [47].

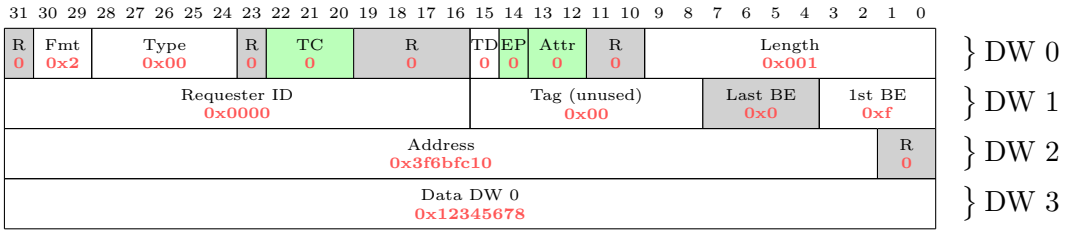


Table 5: Byte field of an example PCIe write request. Table from [47] adapted by [48].

Not only the CPU, but all components in the network can sent these packets. This means, that also an FPGA-board can transmit packets in order to write to memory, for example. Such a write process to memory is called a direct memory access or DMA. It does not involve the CPU and does, therefore, also not need any processor time.

To access the data written to memory from the CPU, it needs to know the address regions to which the FPGA board is writing. This is done with a PCIe driver on the PC, which allocates physical memory into the virtual memory of the CPU, keeps this allocation fixed and informs the FPGA board about the address of the memory that it is allowed to write to. This information (the page addresses and lengths of the DMA memory) is programmed into the base address registers (BARs) of the FPGA [23].

A PCIe driver on the PC, and a DMA engine and PCIe control block on the FPGA were implemented by [23] and other members of the Mu3e collaboration for the use in the Mu3e experiment. This implementation uses four BARs on the FPGA, which are called write/read registers and write/read memory during this thesis.

5 The Mu3e Data Acquisition

This section discusses the data acquisition (DAQ) of the Mu3e experiment. The readout ASICs of the fibre and tile detector (the MuTrig chip) and also the MuPix sensors are considered as part of the detector and not treated as part of the DAQ in this section.

5.1 Overview

The Mu3e data acquisition is a three layer system. The lowest layer consists of 112 custom designed "frontend boards" (FEBs), which directly connect to the Mupix chips or to the readout ASICS of the timing detectors via an 1.25 Gbit/s LVDS link for each MuPix or ASIC. Each of these frontend boards connects to one of four switching boards outside of the magnet via an optical fibre running on a data rate of 6.25 Gbit/s.

One Switching board receives all the data from the upstream- and another one all the data from the downstream-recurl station. The data from the central station is divided into pixel and fibre data with a dedicated switching board for each detector type in this station.

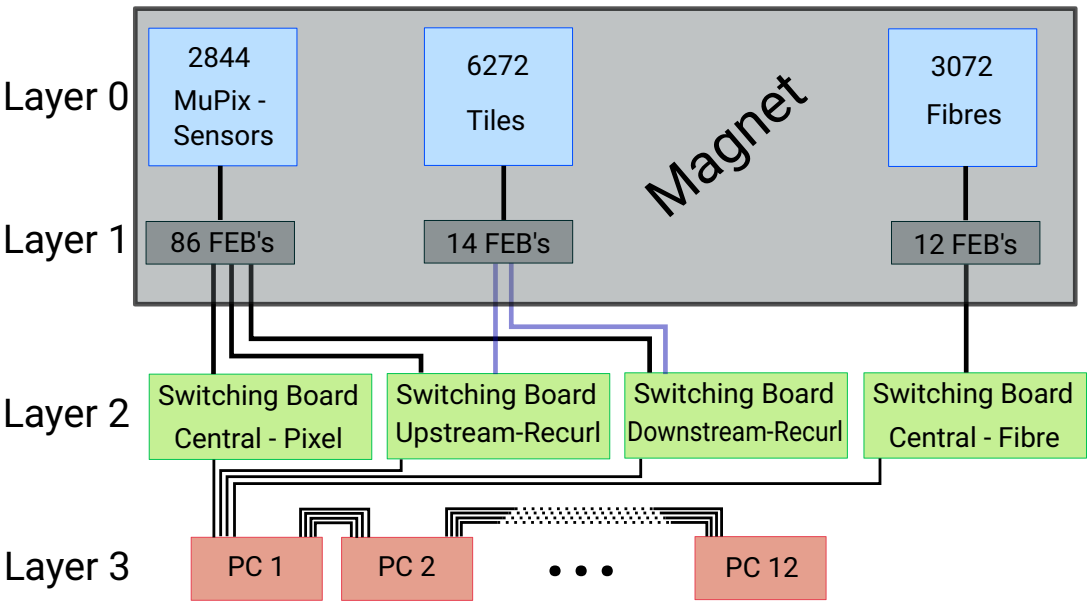


Figure 29: Overview of the layers of the Mu3e data acquisition.

The last layer consists of a daisy-chain of PCs, which host one receiver board and

one graphics card each. The first PC in the chain connects to each switching board via optical fibres. All following PCs in the chain have a connection with the same bandwidth in both directions.

The overall data rate from all subdetectors is expected to be at 80 Gbit/s in phase I [23] and 1 Tbit/s in phase II of the Mu3e experiment. This data has to flow through all layers of the data acquisition and a physics based processing begins in the last layer. The system does not contain a trigger signal and therefore all the data needs to be streamed to the highest layer.

5.2 Layer 1 – Frontend Boards

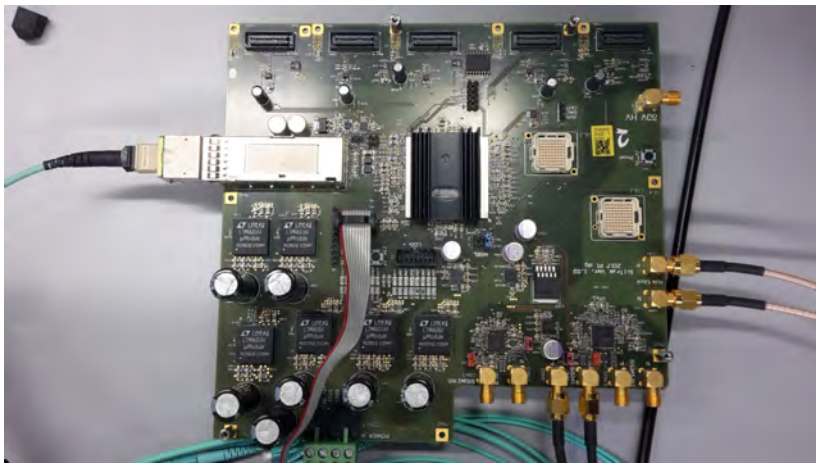


Figure 30: Picture of the prototype for the frontend board (FEB). The final version of the board will contain an ArriaV FPGA instead of the StratixIV included here and needs to be significantly smaller.

Layer 1 of the data acquisition is built from 112 frontend boards (FEBs). They are located inside of the magnet in a helium atmosphere and were developed in the Mu3e collaboration in order to combine all the required functionalities with the tight available space at their intended position. They include an optical transceiver to connect to the switching board, an ArriaV and a Max10 FPGA, and the necessary connectors to connect and operate the MuPix or MuTrig chip.

A single FEB will only connect to either only MuPix chips or only MuTrig chips. The data received from them is not sorted in time. The MuPix reads detected hits in his pixel matrix row by row and the assigned timestamps for each hit do not correspond directly to the order in which they are send off. This is corrected by a hitsorter implemented in firmware on the FPGA of the frontend board in order to simplify further processing of the data.

Online analysis of the MuTrig data will also require a time sorting of hits at some point in the readout system. The final decision where this is going to take place is

not made at the moment, but one of the possibilities is to implement a hitsorter on the FEB similar to the one for the MuPix, if the resources of the FPGA are sufficient for this solution.

Apart from data readout, the FEB has also the responsibility to provide the configuration procedures for MuPix and MuTrig as well as time information in order to assign timestamps to each hit.[1]

5.3 Layer 2 – Switching Boards

Four switching boards form the second layer of the data acquisition system. They are located in a PC outside of the magnet and are connected to the first layer with optical fibres. In the final experiment it is planned to use the PCIe40 board developed by the LHCb collaboration [49] shown in figure 31. However, this board was not available during this thesis and had to be replaced by a different board for system tests.

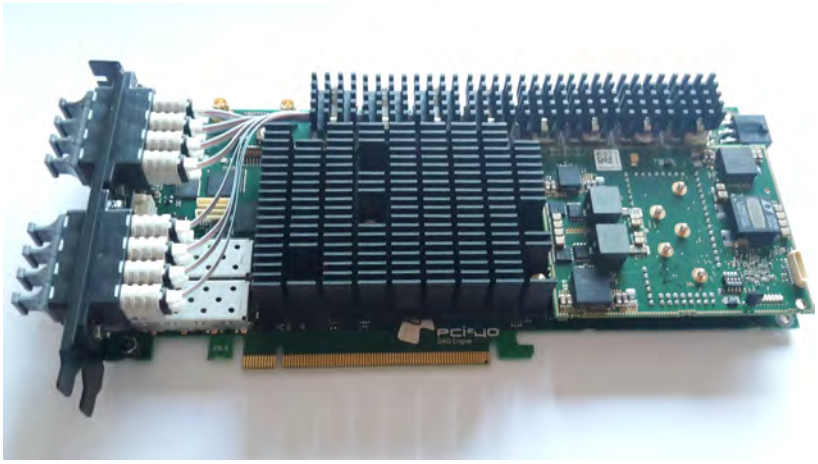


Figure 31: Picture of the PCIe40 board developed by the LHCb collaboration.

The PCIe40 board contains an Arria10 FPGA, a PCI express interface and eight Avago Minipods [50], giving it the possibility to connect to 8x12 optical fibres.

The purpose of the switching board and of the second layer of the data acquisition is to align and merge the data streams of the connected frontend boards into a single data stream. The exact width, amount and assignment to subdetectors of the optical outputs of the switching board to the next layer is still subject to discussions, but it will be necessary to time align data streams from different frontend boards which are connected to the same detector type.

A mixing of tile and pixel data in the two boards which are connected to the up- and downstream recurl station is not foreseen. These two boards might have an individual fibre for their pixel and tile data, respectively. However, some time alignment between these two individual fibres could also be useful.

[1] Timestamps and synchronisation will be discussed in detail in section 7.2.

Because these operations in the switching board and also the transmission through the whole system take some time, the initial timestamp[2] will become ambiguous. To avoid this, more timestamp bits have to be added as the data flows through the system. Also added in this process is a larger chip address and the data of a single hit will require more bits in order to clearly identify it. The interface width to layer 3 of the system might therefore be increased by combining n fibres into a n x 32 bit interface.

5.4 Layer 3 – Farm PCs

Figure 32: Picture of the DE5a-Net development board, which is used as optical receiver inside of each farm PC in the Mu3e DAQ.

When the data arrives at the last layer of the data acquisition, it is received by a DE5a-Net development board inside of a PC. This board sends a time slice of the received data from all channels to the software running on the PC using a direct memory access (DMA). There, the data is evaluated by online tracking algorithms running on a graphics processing unit (GPU). If the reconstructed track is of interest, the data is sent to a data collection server and stored for offline analysis.

During the processing of one time slice, the receiver board forwards all received data to its optical outputs and only takes another time slice if the previous one was processed by the software.

Identical copies of this setup are then dasiy-chained together until no data is forwarded from the last PC in the chain. It was shown that 12 PCs equipped with a Nvidia GTX1080Ti graphics card are able to achieve this for the expected data rates in the first phase of the experiment [23]. With newer graphic cards available at the start of the experiment, this number can potentially be reduced.

[2] A 10 bit timestamp is send by the pixel sensor. More information about data protocols will follow in section 7.3

6 Test Setup for the Mu3e Data Acquisition

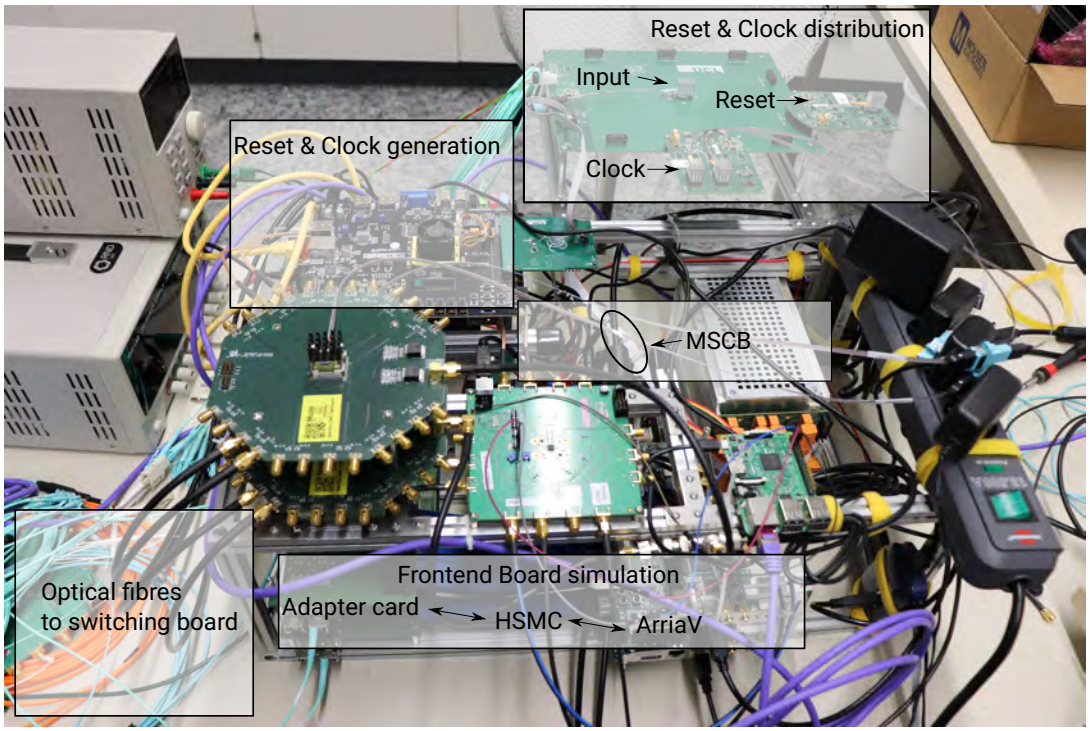


Figure 33: Test setup for the Mu3e data acquisition system

A test setup of the system described in chapter 5 was built during this thesis. It does not contain the full number of boards and also not the final version of each component, but it covers all three layers and also a horizontal expansion in layer 1.[1]

The purpose of this setup is to build a slice of the final system, which is also as close as possible in terms of the used components, firmware and software to the system used in the final experiment. This should allow to test and develop firmware and software for the data acquisition and to build the different control systems around it.

It contains two boards with an ArriaV FPGA, which have an HSMC connection to an adapter card with optical transceivers. This is visible on the bottom of the aluminium frame in figure 33. The two ArriaV boards simulate the behaviour of eight

[1] This means it has more than one frontend board.

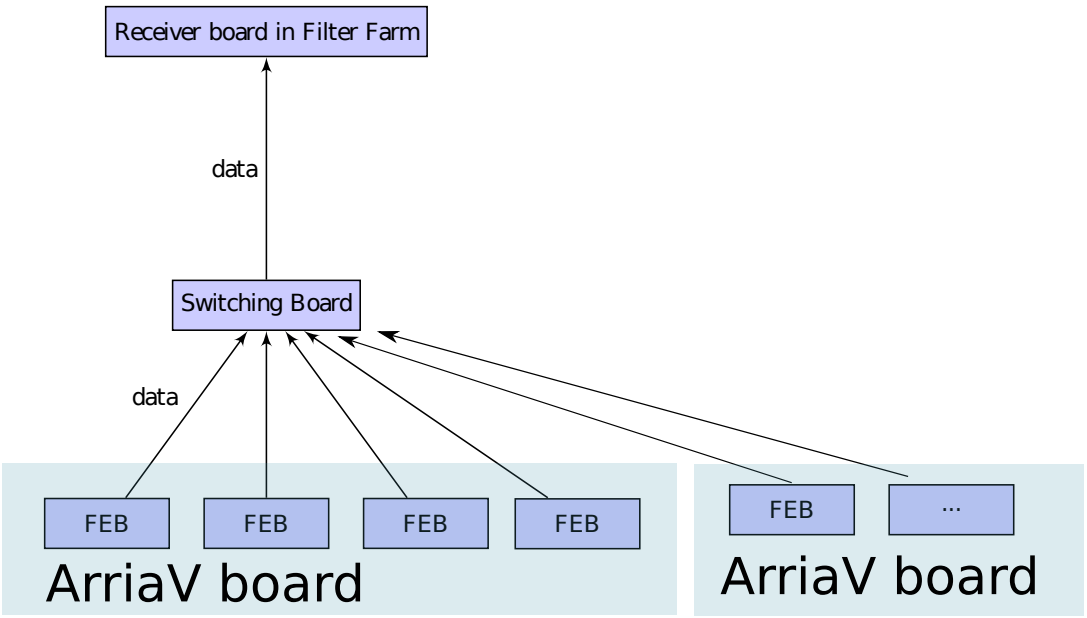


Figure 34: Diagram of the test setup for the Mu3e data acquisition system

individual frontend boards and are connected with optical fibres to the PC with the switching board and from there to a single farm PC.

The reason why only four frontend boards can be simulated per ArriaV board is that the fast transceivers on a ArriaV FPGA are arranged in banks of six channels, each containing two transceiver triplets. Each transceiver block contains a single PLL and only the mid channel PLL of each triplet is capable to drive other channels. In CDR operation, transmitters are driven by the recovered clock of each channel. In CMU operation, the PLL that drives the transmitters replaces the CDR PLL of the one channel in each triplet that is capable to drive other channels. This channel has therefore lost its receiver PLL and can only be used as transmitter, since no other PLL is available to lock on the input data. This means that only 4 bidirectional channels per transceiver bank can be used, if their transceivers are not operated in CDR mode[2] [46]. Therefore, only four individual frontend boards can be simulated.

Over these four channels per board, pseudo-random data can be sent in order to test the alignment procedure in the switching boards or the PCI express readout and file writing in the farm PC. The data is sent in the correct data format, which will follow in section 7.3, and bits corresponding to row or column address in the pixel matrix e.g. are filled with pseudo random data from a linear-feedback shift register.

The components on the top of the picture in figure 33 are part of the control system and are used for the distribution and generation of the global clock and reset signals. These signals and especially their timing is subject of section 7.2. Section 7.5 will

[2] An exception to this are channels with a data rate below 3.125 GBit/s. In this case the transmitter can be clocked from a PLL outside of the transceiver block.

then discuss the MSCB system, which is also shown in this picture.

The setup shown here was expanded later with a frontend board prototype to connect to actual detectors and an additional ArriaV development board for timing tests.

These tests also required some possibilities for the steering of different clock phases, which is done by a raspberry-pi minicomputer with a inter-integrated circuit (I2C) connection to a SI5338-EVB clock chip [51]. The final version of the frontend board will include two of these chips and they will be configurable from the FPGA.

7 The Control System of the Mu3e DAQ

The control system of the Mu3e data acquisition will contain three different control channels to communicate with the frontend boards inside of the magnet. The default solution is to send the information to the switching board over the PCIe interface and from there over optical fibres to the frontend boards. Control data coming from the frontend boards has to be merged with detector data into the same interface and then demerged again on the switching board into the PCIe interface.

The software side of the control system will be built within a software framework called MIDAS, which will be explained in section 7.1.

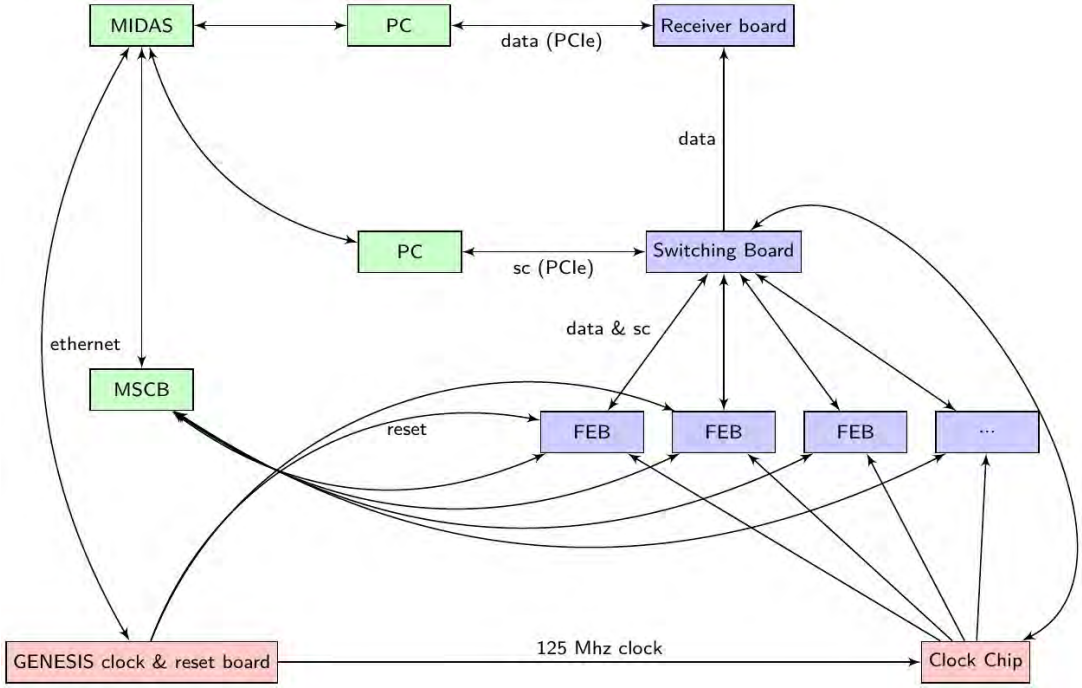


Figure 35: Overview diagram of the Mu3e DAQ control system

Apart from the default path of control data, there is also a backup solution with the name MSCB (section 7.5). This system has a very small bandwidth compared to the data transmission with optical connections and is intended for situations where the optical connection fails. In such a situation, it is still necessary to get critical information, such as temperatures or pressures, out of the experiment.

Another important use-case is the configuration of the FPGA on the FEB. In normal operation, the optical connections will be used for this task, but in case that this fails,

or that the uploaded firmware was not functional, the reconfiguration of the FPGA might not be possible since the optical transceivers require a correct firmware to function. Without a backup solution, this would mean that a very time consuming disassembly of the experiment would be necessary to gain access to the components in the magnet. Such a situation has to be avoided.

The steering of the switching and receiver boards will be done via the PCIe connection. Backup solutions are not necessary in this case since physical access to these boards is given at any time and they can be reprogrammed with a USB cable from their host PC.

The third control channel to the FEBs is the clock and reset system, which is responsible for all signals that need a precise and synchronised timing. They are transmitted to each frontend board with two individual fibres. The reasons why a precise timing is necessary and why this needs a dedicated optical connection will be discussed in the according section (7.2).

7.1 MIDAS

MIDAS, or Maximum Integrated Data Acquisition System, is a software framework for data acquisition systems. It is continuously developed at the Paul Scherrer Institute since 1993 and includes data logging, slowcontrol, alarm systems, user specific software and many other parts of a DAQ into a single system, which can be accessed and controlled from any web browser.

7.1.1 Midas Frontends

The user or device specific software in MIDAS is written in a format which is called MIDAS frontends[1]. A MIDAS frontend runs on a PC and communicates with its device or system using software written by the user. The information exchanged with the rest of the DAQ is then divided into detector data, which is intended for permanent mass storage and analysis, and into control information. Control information can be sent to or requested from a online database in shared memory.

This shared memory is then accessed by the mhttpd webserver, which hosts the user interface web pages. Custom web pages can be written for specific devices or systems using HTML and javascript and user interactions on these pages can be hot-linked into functions in the frontend user code. This allows to have a distributed, remote-controllable system, where one PC hosts the webserver and many other PCs with an ethernet connection to the host can control the devices connected to them and run their own frontend user code for this locally.

7.1.1.1 Example: DHCP & DNS server control

An example for a MIDAS frontend is the DHCP and DNS server control that was implemented during this thesis. A PC acts as a gateway server to the local Mu3e

[1] This has nothing to do with the frontend board.

network in order to avoid problems with the university network and to include devices with a fixed IP address. It hosts a DHCP and DNS server and runs a MIDAS frontend to control the two servers. Devices in the local network will send DHPC requests, the MIDAS frontend will receive these requests, copy the assigned IP address and hostname to the online data base and a list of all ethernet devices in the local network will be visible with their IP address and hostname on a webpage.

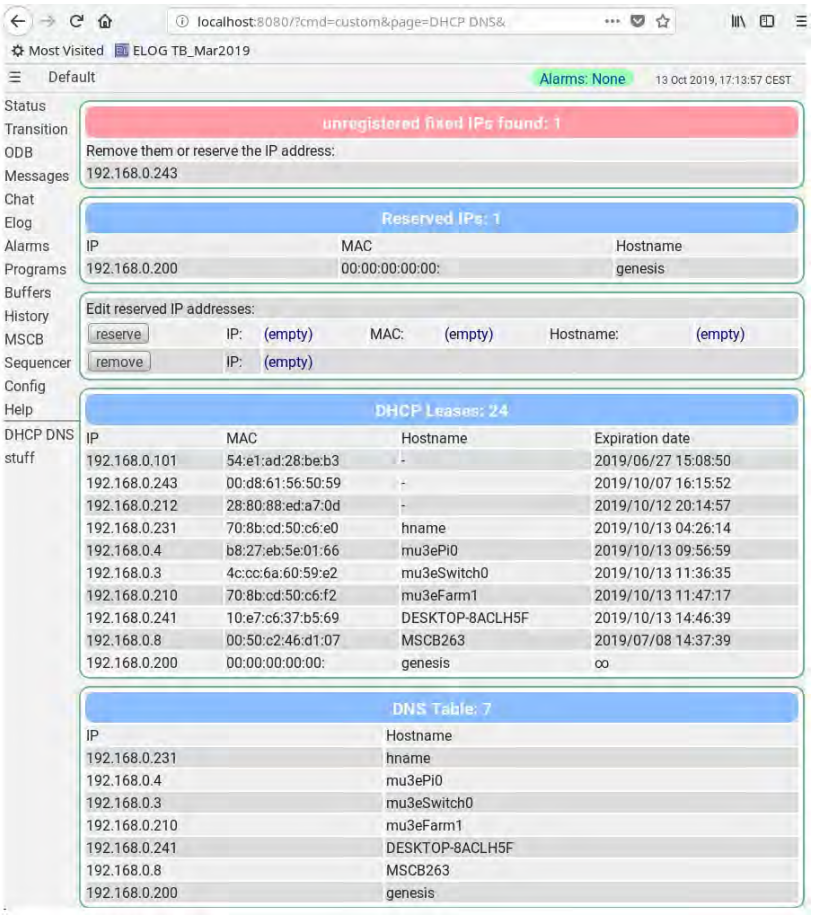


Figure 36: MIDAS custom page for DHCP and DNS server

On this webpage, the user can edit the hostnames that the DNS server provides. A periodic ping to all addresses in the subnet was implemented in order to find devices with fixed IPs, which have not gained their IP from a proper DHCP request. These devices are then also shown on the webpage and the user has the possibility to reserve IP addresses for these specific devices using their MAC address, which is useful for components which do not have the possibility to send a DHCP request or have a hardcoded IP address[2].

[2]FPGAs for example.

7.1.2 MIDAS Banks and Bank structure

The data from MIDAS frontends which is intended for mass storage is sent to a data collection server within the network and stored in so called MIDAS-banks. An event in MIDAS consists of the event header, which contains general information of the event and then a number of banks in this event. These banks could, for example, contain the information of the pixel detectors for one time slice. Another bank within the same event could contain the tile detector data and these two banks could originate from two different MIDAS frontends running on two different PCs.

The speed at which the MIDAS logger can write these events to a file depends on the write speed of the mass storage and on the speed of the software of each involved frontend.

A speedtest was performed with a MIDAS frontend that reads a specific number of bytes from the DMA buffer in a loop.

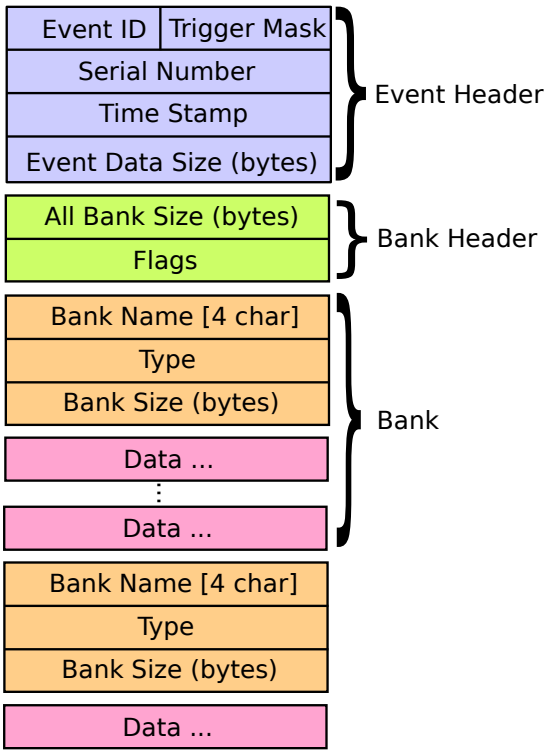


Figure 37: MIDAS bank structure. Picture adapted from [52].

It could be observed, that the used DMA readout frontend hits the limitations of the mass storage, when more than 200 bytes are written from the DMA buffer into mass storage in a single event (figure 38). This depends of course on the used PC hardware and on the operations that the frontend software performs, however, the important information here is, that it is possible to reach the mass storage speed limitations, if the event size is large enough.

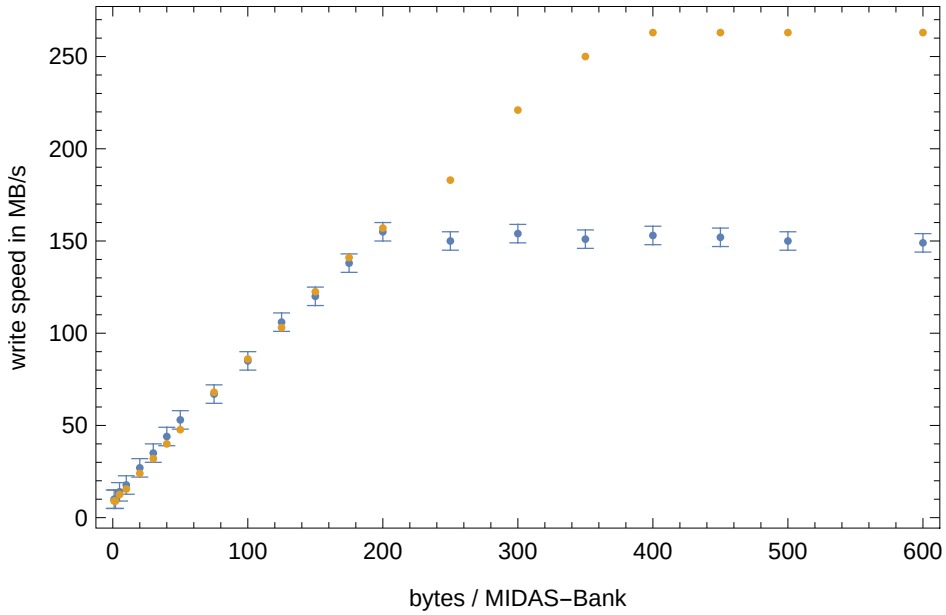


Figure 38: Write speed of the MIDAS logger for a dummy frontend writing banks with different sizes to a hard drive (blue) and an SSD.

7.2 Reset and Clock System

The operation of a detector system like the Mu3e detector requires the distribution of precise time information to all components directly involved in particle detection. In order to perform the reconstruction of tracks, a significant reduction of the parameter space has to be performed. This has to be achieved by applying precise timing information to each individual particle detection. It is therefore absolutely necessary, that all involved detectors can attribute a precise timing information to each detection of a particle. The clock and reset system is responsible to provide this time information to the components in the Mu3e detector. Since the detector with the lowest time resolution reaches a resolution of 70 ps, the synchronisation across the whole detector and the overall precision of the clock and reset system has to be much smaller than 70 ps, if the usable precision of an individual detection should be preserved.

The Mu3e clock and reset system will provide the required time information by transmitting a clock and a reset signal. These have to arrive at the lowest level of the data acquisition chain with a precision of below 70 ps and will be used to run and reset a counter in the ASICs and in the frontend FPGA boards, which will be used as a global timestamp of the Mu3e detector.

7.2.1 The Reset Protocol

The reset line will not transmit a single true or false signal. It is an optical connection with an interface width of 8 bit and 8 bit commands will be used to cycle through

a set of possible states of the frontend board with defined and synchronised timing. The possible states and the commands to change between them are shown in tables 6 and 7.

State	Comment
Normal DAQ	
`Idle`	
`Run Prepare`	
`Sync`	Resets active
`Running`	
`Terminating`	
Tests and Calibration	
`Link Test`	
`Sync Test`	
Hard Resets	
`Reset`	
Others	
`Out of DAQ`	

Table 6: FEB System States [53]

Command	Code	Payload
Run Prepare	0x10	32 bit run
Sync	0x11	-
Start Run	0x12	-
End Run	0x13	-
Abort Run	0x14	-
Start Link Test	0x20	T.b.s.
Stop Link Test	0x21	-
Start Sync Test	0x24	T.b.s.
Stop Sync Test	0x25	-
Test Sync	0x26	T.b.s.
Reset	0x30	16 bit mask
Stop Reset	0x31	16 bit mask
Enable	0x32	
Disable	0x33	
Address	0x40	16 bit addr

Table 7: Reset link protocol [53]

In a normal run sequence the FEB cycles through the states `idle`, `run prepare`, `sync`, `running` and `terminating` before the `idle` state is reached again. During the `run prepare` state, the run number is transmitted and all FIFOs in the FPGA are cleared. When the run prepare signal is acknowledged by all FEBs, the sync signal is send out and resets the timestamp counting in all FEBs until the sync state is exited with a start run signal. With the arrival of this signal all time counters across the detector are supposed to start synchronised counting.

When the run is stopped, the frontend board will continue to send data in the `terminating` state until its buffers are empty and transit back into the idle state by itself.

The four additional states, which are not present in a normal run cycle have their own, special purpose. The `link test` and `sync test` states can be entered only from the `idle` state and are intended for bit error rate tests (BERTs) and synchronisation test, respectively. `Out of DAQ` can also only be entered from the idle state and allows to take a FEB out of the data acquisition system. A FEB in this state will not respond to any other commands except for an enable signal and will not send detector data to the switching board. The last state is the `reset` state, where 16 individual reset bits in the FEB firmware can be set to 0 or 1 in order to hard-reset parts of the system. The reset state can be entered from any test or normal DAQ state and will always

exit into the idle state.

All commands in table 7 can also be transmitted with a target address. This requires the address command and the address of the frontend board to be transmitted right in front of the intended command.

7.2.2 Generation and Distribution of the Clock and Reset Signal

The commands above and the global system clock are generated in a GENESYS2 board with a Kintex-7 Xilinx FPGA, which is connected to a MIDAS frontend with an ethernet connection. The run start sequence described in the last section will have to be implemented into the MIDAS internal run start procedure for the final system. For testing purposes the user can send each signal in table 7 individually from the user interface, where also the power consumption and temperatures of the system are shown (Appendix figure 56).

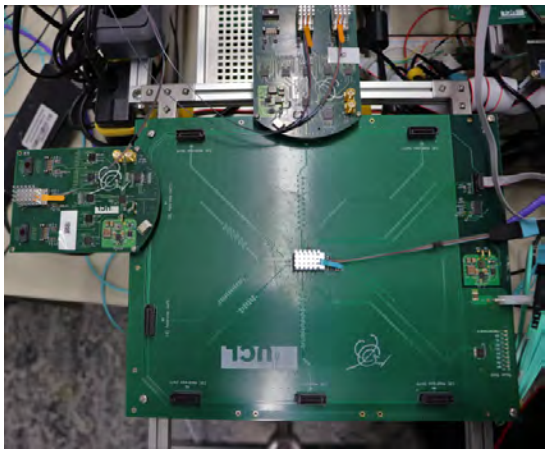


Figure 39: Picture of the clock distribution board.

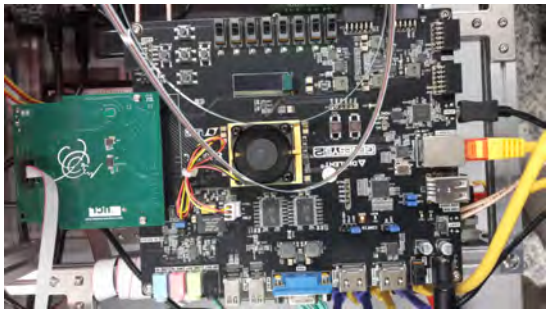


Figure 40: Picture of the Genesys2 board

The GENESYS2 board is then connected to the clock distribution board with an optical connection. This board generates 144 active optical copies of the clock signal and 144 active optical copies of the reset line and is connected to the FEBs.

7.2.3 Timing measurements with the clock distribution board

Measurements were performed to test the timing of some of the 288 output channels of the clock distribution board shown in figure 39. The trace-lengths on the board should in principle provide identical delays to all output channels. The optical outputs were converted into LVDS signals by an Avago-MiniPOD [50] receiver and then measured with an oscilloscope.

The mean of the jitter of the clock signals was found to be at (1.44 ± 0.36) ps and the delays between different channels varied over a range of 47 ps with a standard

deviation of 14 ps. Relevant for the timing in the Mu3e detector is the measured jitter, since constant delays can be removed with the clock chips that will be included on the frontend boards. A correlation between the measured jitter and the connected daughter-board slot on the clock board was not found.

7.2.4 Receiving of Reset Commands

A "state controller" state machine (figure 41) was written to decode the reset commands on the frontend board and it was tested with the setup from chapter 6. The parallel reset input signal is interpreted and the state of the FEB is changed after the complete payload is received. The amount of clock cycles between the arrival of the command and state change depends, therefore, on the type of command, since different commands have different payload lengths.

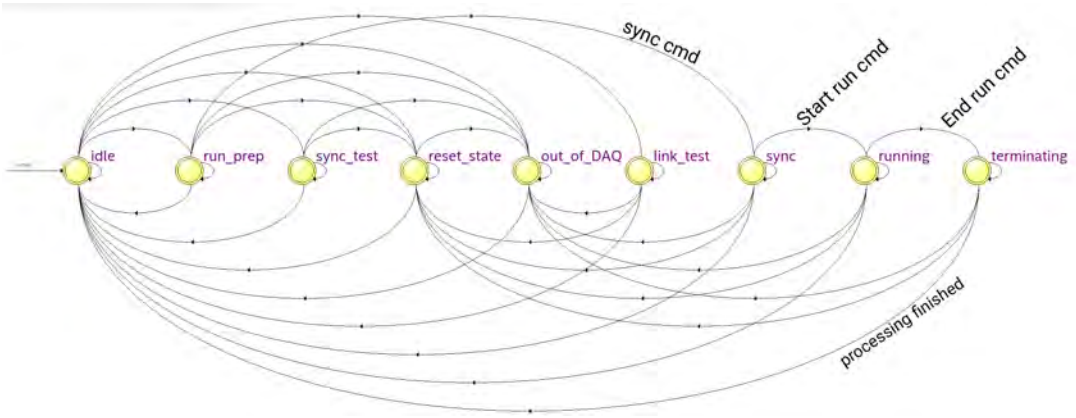


Figure 41: States and transitions between the different FEB states.

When an address command is received, all not addressed boards still follow the commands and payloads on the reset line in order to determine the exact clock cycle where they are supposed to react to commands again. All boards are addressed by default and an address command causes all not addressed boards to ignore the following command. The address system could also be easily extended to implement group addressing with an appropriate address scheme, for example to address only the FEBs of a single detector station or sub-detector type.

Another possibility is to intentionally delay each state change by n clock cycles for time alignment. This should in principle not be necessary, since the cables to all FEBs will have the same length and therefore also the variation of the transmission time of the signal to each FEB should be below the length of a clock cycle (8 ns).

However, this assumption requires that the state controllers on all FEBs receive the reset command in the same clock cycle, which can be a problematic assumption as the next section will show.

7.2.5 Timing of Reset Commands

As described in the introduction, the timing of the reset commands is critical for the operation of the Mu3e detector. A delay, or at least an unknown or not constant delay, between the reset of timestamp counting on different frontend boards would directly cut into the time resolution of the detector. An uncertainty of one clock cycle on the state change of a reset command, for example, would completely remove the purpose of the fibre and tile detectors, since their time resolutions are in the order of 100 ps.

7.2.5.1 The Problem with Timing in fast Transceivers

The issue here is that exactly this can happen with standard configurations of fast transceivers on the used FPGAs. The transceiver recovers a clock from the serial input data and divides this clock down to the appropriate frequency of the parallelised output signal of the transceiver. The reset line is operated at 1.25 GHz and with an 8b/10b decoded 8 bit wide interface. This means, that the clock division in the transceiver divides a 1.25 GHz clock into a 125 MHz clock. The transceiver has ten possibilities to do so, since there are ten rising edges of the fast clock available, to which the phase of the slow clock could be aligned to.

On an asynchronous reset, the recovered slow clock from the transceiver will end up in one random phase out of these ten possibilities and its parallel output (our reset line) will be aligned to this random phase.

If also the placement restrictions of non-CDR transceivers (see chapter 6) are considered, then one could think that these transceivers were designed for a situation where no extra clock line is present and the receiver relies on the recovered clock without a fixed phase relation to the transmitter. This timing behaviour is presumably not a problem for many applications of FPGAs, but, however, unacceptable for a particle physics experiment and has to be resolved, which is the topic of this section.

In the situation described above, the phase of the output data will randomly choose one of ten values. The real situation in our system is slightly different since the reset of the transceiver is not asynchronous. The transceiver reset is driven by the global system clock, which has a fixed phase relation to the serial input data of the transceiver. Therefore, the release of the reset of the clock division will take place in a specific cycle of the fast recovered clock. Slight phase variations of the PLL can still introduce a phase uncertainty, but only between two neighbouring fast clock cycles.

The arrival of reset commands can, however, still jump between two clock cycles when it is re-synchronised to the global clock. This could be seen in measurements and is shown in figure 42.

Another potential problem for the timing is that the transceiver, even if it ends up in a somehow constrained state after a synchronous reset, is still able to function at all other phases of the fast clock. Under normal circumstances it would have no reason to enter into one of these states, but it has the possibility to do so. When the measurement for figure 42 was performed, it was necessary to move clock phases into the critical region using the clock chip and raspberry-pi in the test-setup. During this

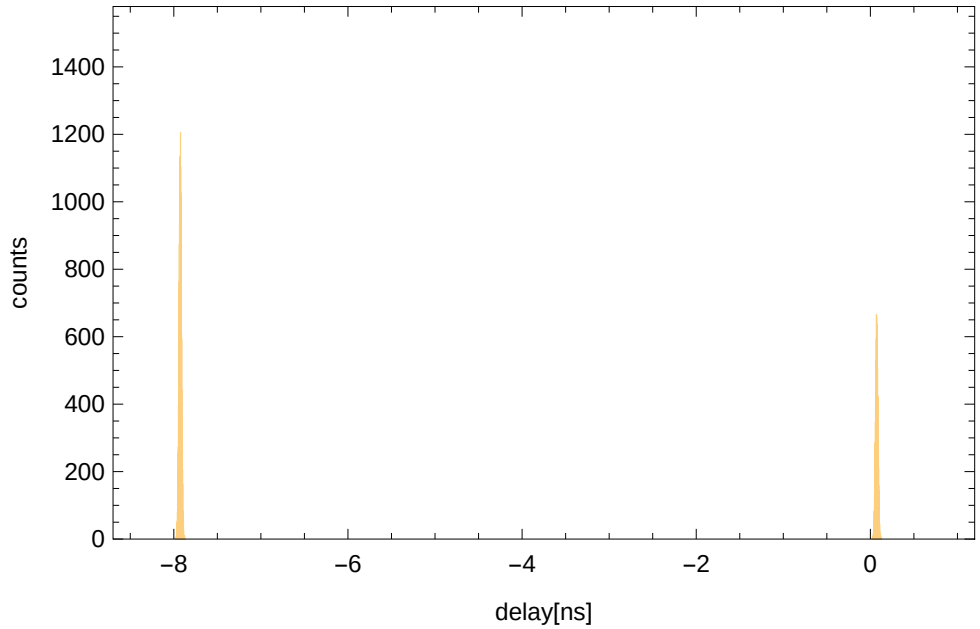


Figure 42: Delay of the arrival of a reset signal in a FEB, measured relative to a reference signal. Each count in the histogram was measured after a reset of the transceiver and the phase of the reset line to the system clock was intentionally moved into the phase-region where this behaviour is visible. This region is about 80 ps wide.

process it was observed that the PLL of the transceiver was sometimes able to adjust to the new phase situation without loosing lock and without causing a synchronous reset.

The internal PLL of the transceiver has to be able to follow slight phase variations between its output clock and the reference clock in order to function (see section 3.3.2). If something, like a reconfiguration of the clock chip, happens to one of the input phases it might be able to follow the change and cause a different output clock phase. This situation was very unlikely, but could be reproduced multiple times with the test-setup. However, a synchronous reset always forced the phase back into its normal position.

This behaviour is also something that needs to be avoided for the Mu3e DAQ. The transceiver still functions, but with a different latency relative to the input commands.

7.2.5.2 About the use of Deterministic Latency Transceivers

One might be tempted to use transceivers in deterministic latency mode to solve the problems from section 7.2.5.1. In the deterministic latency mode, the input data is aligned with clock slips in the deserialiser instead of word alignment in the PCS [46]. This procedure possesses a deterministic latency in the sense that the parallel output

data will have a fixed phase relation to the word boundaries of the serial input, which is in our case also a fixed phase relation to the global system clock.

The problem is, that it is not ensured that all FEBs align to the same clock cycle of the global system clock. They will all have the same phase relative to a rising edge of the global clock, but possibly to two different edges.

It should in principle be possible to avoid this, because the cable lengths of all reset and clock lines to the FEBs are intended to be equal. The deterministic latency transceiver could, therefore, be a feasible solution, when the timing of each FEB is measured before the system enters the magnet. Once it has entered the magnet, there is no direct way to test time alignment of the FEBs[3].

The approach which was taken for the Mu3e DAQ should allow "in-magnet" timing tests and is explained in the next section.

7.2.5.3 Implemented Solution for the Timing Problems

The first necessary step is to re-synchronise the reset line to the global Mu3e system clock. The state controller is still operated in the clock domain of the recovered clock and only the decoded state output is forwarded to the global 125 MHz clock domain. This has the advantage that it is not possible to introduce unintended state changes due to metastability. If the state controller would be located in the global clock domain, a metastable parallel input from the transceiver clock could be interpreted as a different command. The phase relation in such a case is not optimal and has to be changed to prevent this. In order to do that, the clock chip needs to be reconfigured, which requires a SPI connection to an active NIOS2 processor. The problem here is, that the NIOS2 might have an active reset when he is needed to resolve the wrong phase relation, which cannot be removed, since the system that could move the FEB out of the reset state is not functional due to a false interpretation of commands because of the wrong phase relation. This does not seem like a very likely or unsolvable situation, but can be avoided with the placement of the state controller on the transceiver clock, since there is no strong reason to have it in the global clock domain.

The chosen approach to ensure the time alignment of the reset commands is to constantly measure the phase between the recovered clock of the transceiver and the global clock of the FEB. If the global clocks on all FEBs are synchronised, then the measured phases of the FEBs will be distributed around some value depending on their slight differences in cable length and the one or two 1.25 GHz cycles to which they align on a reset. The mean phase relation to the reset lines of the FEBs can then be moved (by using all clock chips) to a value where two neighbouring fast clock cycles cannot cause a different cycle of the reset in the slow clock domain. The state change is then transmitted into the global clock domain without the use of a synchroniser[4]

[3] Although, it should be possible to observe time misalignment of a FEB once the tracking starts to fail for its specific regions in the detector. But this could also have other reasons and is not a direct sign for time misalignment.

[4] A synchroniser chain is not useful here, since it introduces non-deterministic latency for phase relations where it is necessary.

and the phase of the two clocks is inserted into the timing optimisation of the design software.

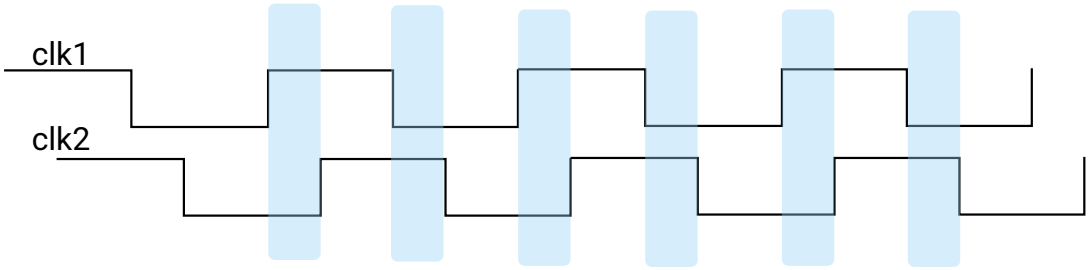


Figure 43: Working principle of the phase measurement between clk1 and clk2.

To do this, a phase measurement entity was implemented and tested in hardware. The entity makes use of a free running clock with arbitrary frequency to measure the phase between the related clocks clk1 and clk2 in figure 43. Rising edges of the free running clock will be randomly distributed relative to clk1 and clk2, since they are driven by a different oscillator. On a rising edge of the free running clock, the two other clocks will be sampled and compared. If their current value is not equal to the value of the other clock, a counter is increased (blue regions in figure 43). This is done for a measurement time T and then the value of the counter is compared to the total amount of rising edges of the free clock in the time T. The phase difference between clk1 and clk2 can then be calculated using:

$$\text{phase difference} = \frac{counts}{T \cdot f} \cdot \pi \tag{9}$$

where f is the frequency of the free running clock.

The entity was tested with an ArriaV development board in the test setup described in chapter 6. The measurement result is shown in figure 44. With the description given above, the entity is only able to measure the absolute value of the delay, therefore the y-axis contains only positive values. A phase shift of +1 ns should produce the same value in the counter as a phase shift of -1 ns. It should be possible to solve this ambiguity by sampling the value of clk2 with the rising edge of clk1. If clk2 is low at this position, then it is shifted in one direction relative to clk1 and if it is high it is shifted in the other direction. However, this was not implemented for this test.

For the purpose of the reset system, a resolution of the phase measurement in the order of 1 ns would be sufficient. It can clearly be seen in figure 44 that differences on this time scale can be identified by the phase measurement entity. In principle the resolution should only be limited by the measurement time, but in reality jitter, rise times and other things can have an influence on the resolution.

The measurements in figure 44 form a plateau near a 4 ns delay on the x-axis for some reason. A part of the explanation for this could be the 55 % duty cycle which was observed for the recovered clock (appendix figure 57).

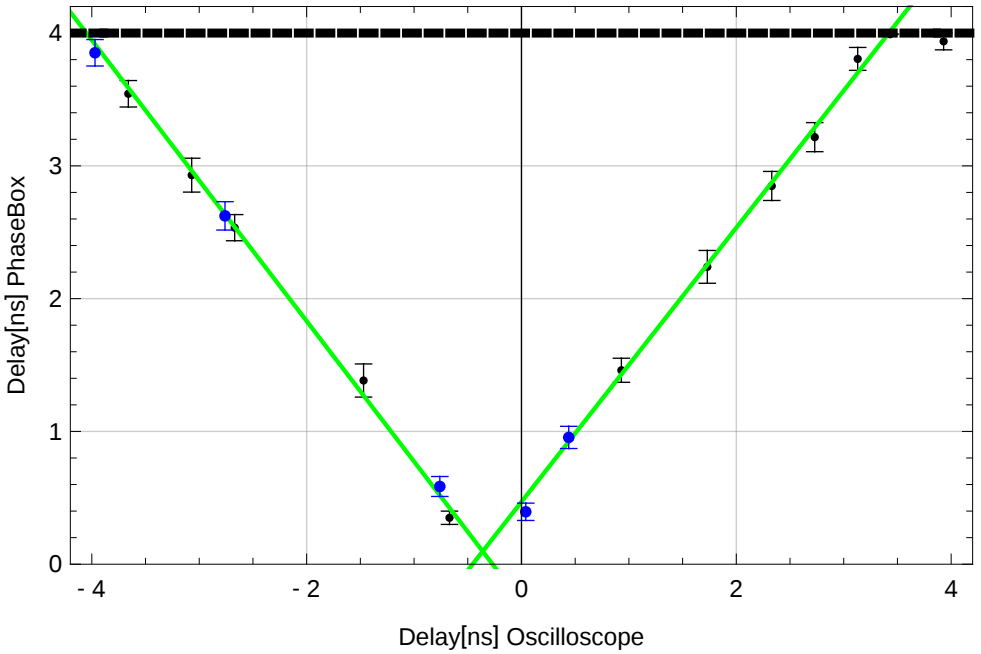


Figure 44: Measurement results from the phase entity. On the x-axis the delay between a reference clock and the global clock measured by an oscilloscope is shown. The y-axis shows the mean of 15 delay measurements with the phase entity for a 50 MHz free running clock and a measurement time of 2^{26} counts. The point of intersect with the x-axis is of no meaning, since this was moved to 0 ns by cable lengths of the reference signal.

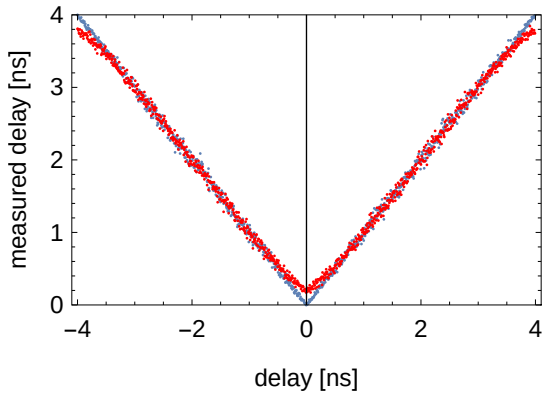


Figure 45: Simulation of the phase measurement entity without jitter (blue) and with a 50 ps jitter on one of the input clocks(red)

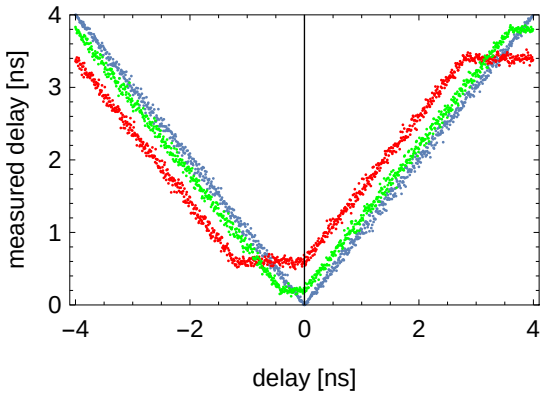


Figure 46: Simulation of the phase measurement entity with different duty cycles of one input clock. 50 % (blue), 55 %(green), 65 %(red). The second input clock was on a fixed duty cycle of 50 % for all simulations.

A simulation of the entity was performed to simulate the effect of different duty cycles on the measurement (figure 46) and the variation with 55 % duty cycle shows indeed a behaviour similar to the measurement. In addition to the duty cycle simulation also a simulation with jitter was written, which is shown in figure 45.

7.2.5.4 Reset Connection to Sub-detectors

The reset signal to the sub-detectors is raised to active when the sync state is entered by the frontend board. In this connection a single bit is transmitted instead of a command. When the reset bit arrives at the MuPix or MuTrig, it has to be again resynchronised to the global system clock. This is necessary because the jitter of signals and clocks generated by an ArriaV FPGA can be in the order of 100 ps, which is large compared to a jitter of a clock from a dedicated clock chip and can have a significant influence on the time resolution of the fibre and tile detectors.

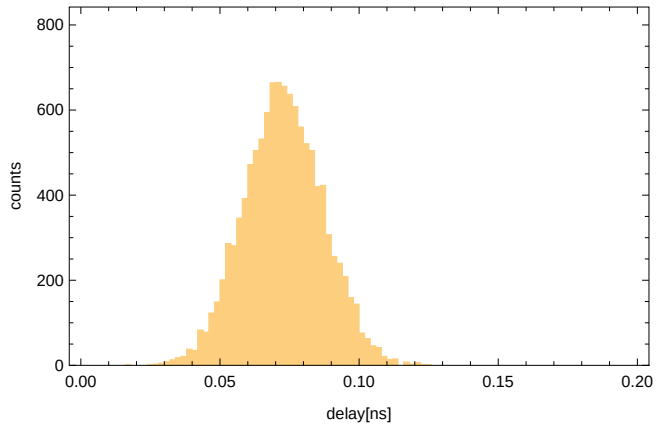


Figure 47: Example of jitter on a FPGA output signal. The delay was measured against a reference clock.

The resynchronised reset is then used in the readout ASICs to start timestamp counting which is synchronised across the whole detector system when the frontend boards go from the sync state into the running state.

7.2.5.5 Reset bypass from NIOS processor

The system needs to be operated and tested in testbeam campaigns and lab tests before the Mu3e experiment is assembled. Only two copies of the hardware for the reset system (clock distribution board with adapter cards) are available in the collaboration, which is the reason why it was required to find a solution to operate the setup without the reset hardware. This was done with the possibility to bypass the reset line from an output of the NIOS2 processor. The bypass replaces the parallel connection between the transceiver and the state controller and can be activated, deactivated and controlled from the NIOS2 command line without a change in the firmware. The

signals are still processed in the state controller in the same way as the actual reset line would be processed.

When the bypass is used, time synchronisation across multiple frontend boards is not possible.

7.2.5.6 Reset Timing with LVDS Transceivers

The final version of the frontend board probably will not have the available space to host two fast optical transceivers. The reset line will therefore be converted into an LVDS signal and will be received by an LVDS transceiver instead of a fast transceiver in the frontend FPGA. The considerations of the previous sections still hold and the phase measurement entity is used for the same purpose.

An LVDS version of the reset firmware was implemented on an additional ArriaV development board in the test setup described in chapter 6 and the tests performed with the fast optical transceivers were repeated with the new version to proof the practicability for the reset system on the FEB.

Altera's LVDS SERDES transmitter/receiver does not include an automatic alignment of the word boundaries. It provides an input port, where the alignment can be slipped by one bit. Therefore, an entity was written to control and align the LVDS receiver. The procedure in the entity follows the instructions in 1.5.3.4 of [54].

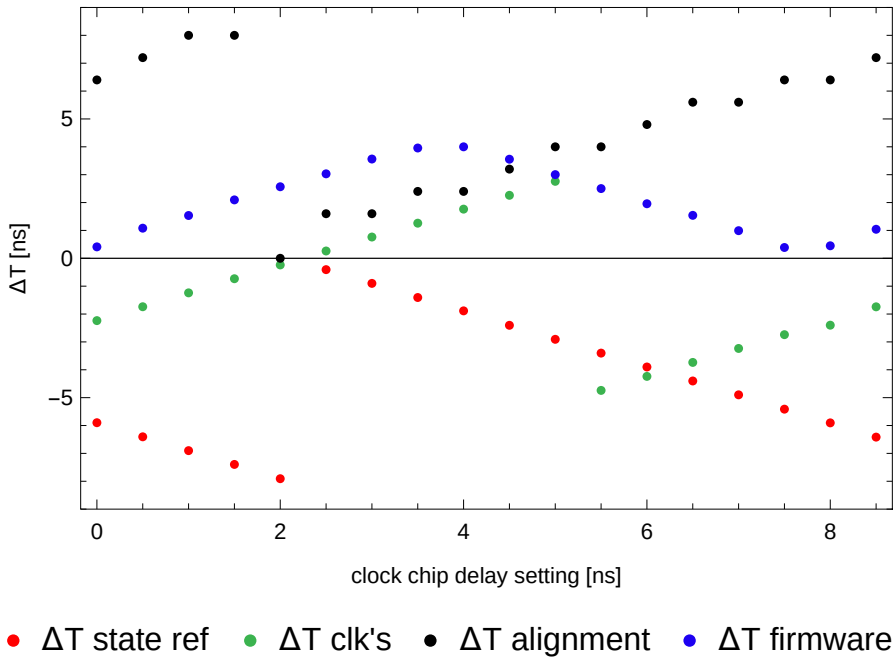


Figure 48: Phase measurements with the LVDS receiver. All measurements are relative to a reference clock. The x-position of the green measurement points relative to the other measurements is not relevant since it was measured at a different point in the system.

Figure 48 shows the results of the reset timing measurements with the LVDS receiver. The phase of the clock (green) was varied relative to the reset word boundaries using the phase setting in the clock chip (x-axis). At some point the arrival of the reset command had to happen in a different clock cycle when the delay settings were scanned. This happened at a setting around 2 ns (red).

As for the fast transceivers, this critical region can be avoided using the results from the phase measurement entity (blue). In addition to this information, also the amount of alignment clicks from the alignment entity (black) seems to correspond directly to the phase of the recovered clock and can, therefore, also be used to avoid the region where the reset command could end up in a different clock cycle.

7.3 Slowcontrol

The second possibility to communicate with the frontend board is the optical connection to the switching boards. The control data which is transmitted over these connections is referred to as slowcontrol in the following sections, even if the optical connection operates at 6.25 GBit/s and this term might not be perfectly appropriate.

7.3.1 Data Protocols

The connection to the switching boards transmits four types of data packets : MuPix-data, MuTrig-data, slowcontrol-data and run control signals. The first three mentioned types of data packets start with a preamble, where the comma word K28.5 is located in the 8 least significant bytes in order to identify the start of a packet. The other bytes of the 32-bit wide preamble are occupied by an FPGA ID and 6 type identifier bytes to identify the type of packet and also the origin of the transmission. The end of a packet is indicated with K28.4 in the least significant bytes in the interface.

7.3.1.1 MuPix Data

The structure of MuPix data packets is shown in table 8. After the preamble a data header with a 48 bit timestamp is transmitted, followed by a sequence of sub-headers and MuPix hit data.

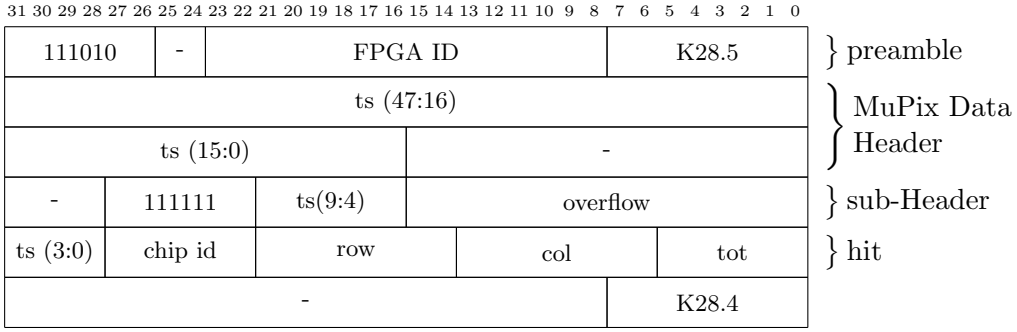


Table 8: Structure of MuPix Data

The sub-headers contain bits 9 to 4 of the timestamp and 16 overflow bits to indicate loss of data due to bandwidth limitations. The next full header with a 48 bit timestamp is always sent when bit 10 of the timestamp flips. The header can therefore contain multiple sub-headers and their hit data, which contains the column and row of the hit in the pixel matrix, time-over-threshold (tot), chip ID and the lowest four bits of the timestamp.

This data packet, and also two data packets that are about to follow, is not necessarily sent without interruptions. Any gaps between packets and also positions between preamble and trailer where no data is available will be filled with comma words.

7.3.1.2 MuTrig Data

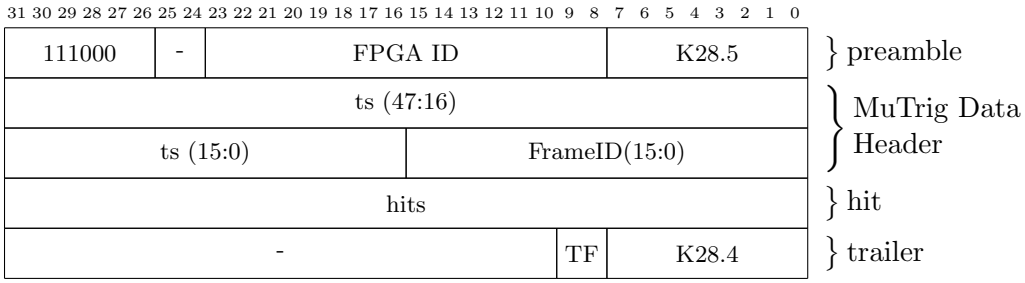


Table 9: Structure of a MuTrig Data Packet

The structure of a MuTrig packet is very similar to the structure of a MuPix packet. The preamble contains a different type identifier to distinguish it from MuPix packets and the header contains a 16 bit frame ID in addition to the 48-bit timestamp. This is then followed by hit data, whose format will not be further discussed here, until the end of the packet is reached. A sub-header is not present in a MuTrig data packet.

The idea is to use a common firmware for MuPix and MuTrig data as soon as possible in the DAQ chain, which is the reason why these two packet types have a very similar structure. It is intended that the firmware of a frontend contains a part which is identical for MuTrig and MuPix FEBs, and a part that is MuPix, fibre or tile specific. This allows to develop control systems and communication to the FEB for all FEBs together, while treating the connected sub-detector type as a kind of "user" of this system. The communication between the common and detector-specific firmware will be discussed in section 7.3.3.

7.3.1.3 Slowcontrol Data

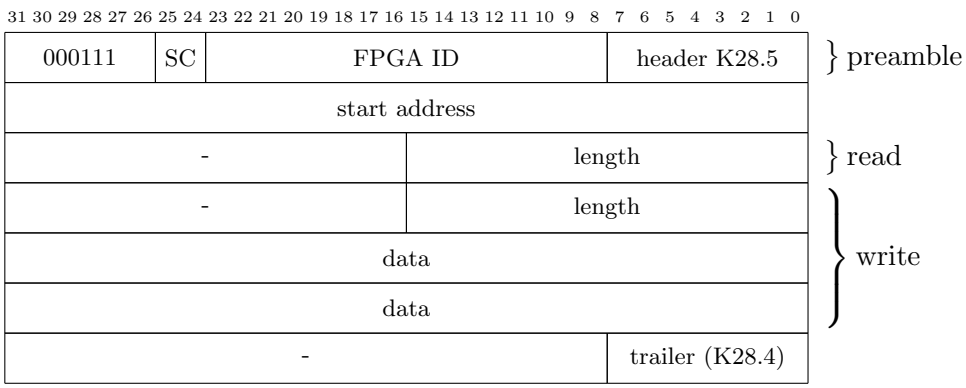


Table 10: Structure of Slow Control

A slowcontrol packet always interacts with memory on the FPGA and can have four purposes. It can either act as a read or write request to the FEB or as a response or acknowledgement to a read or write request, respectively. Whether or not the FEB should also be able to send slowcontrol packets without a request from the switching board is not decided yet. The firmware that was written in this thesis to handle and merge these packets with data packets does allow this situation.

Slowcontrol packets are the only type of packet which can be transmitted in both directions on the optical link between the switching board and the frontend board. The first 32-bit word after the preamble contains the memory address from which or to which data should be read, should be written, is read or is written. The next word contains the length of the data.

In case of a read command the packet ends here with a trailer. For a write command it is followed by the data that should be written to the specified address and its successor addresses.

A reply to a read command has the identical structure as a write command and the data after the length contains the contents that were read from FPGA memory in this case. A reply to a write command contains no data words and an additional 1 in bit 17 of the word in which the length is sent out to acknowledge a successful write to FPGA memory.

7.3.1.4 Run Control Signals

Run Control signals are single 32-bit words, which can be transmitted at any time between or within data packets. For other packets a mixing of different packet types has to be strictly avoided and they need to be transmitted one after the other. Run control signals can be identified with the K30.7 comma word in bit 7-0, which is unique for all communication on this link.

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
run control signal	K30.7

Table 11: Structure of a run control signal

They are currently used for the acknowledge signals for state changes of the state controller. One of the signals is the run prepare acknowledge signal, which indicates the readiness of the FPGA to enter the sync state. The MIDAS frontend that controls the reset link has to wait for this signal to be received from all FEBs before the sync and start run signals can be transmitted over the reset link. The second run control signal that is currently in use is the run end signal, which is transmitted when the processing of the data of the connected detectors was finished in the terminating FEB state. When the run end signal is send of, the FEB is allowed to fall back into the idle state by itself and the switching board is given the permission to end data taking for this FEB.

Other use-cases of run control signals might follow as the development of the Mu3e DAQ proceeds.

7.3.2 Upstream Control Data

The handling of upstream[5] control data packets was developed in [48] and is briefly discussed here.

The read or write command is first written into the PCIe write memory (see section 4.5) of the switching board, from where it is transferred to the FEB via the optical connection. On the FEB this command is decoded and the data is written or read from the corresponding addresses in the frontend memory.

Replies or acknowledges are fed through the downstream chain (next section) and end up in the PCIe read memory of the switching board, where they can be accessed by software of its host-PC again.

The memory to which this read and write commands are directed is a dual-port-RAM. The other port of the memory is connected to a NIOS2 soft-core-processor on the FEB, which can then act based upon the memory content. The instructions for the NIOS2 processor can be written with the same method to reserved addresses in the memory.

An alternative way to access this memory using the MSCB system will be shown in section 7.5.3.

7.3.3 Downstream Control Data

Replies to upstream control data packets or also messages sent from the FEB firmware without a previous request have to be transmitted together with the detector data and run control on the same optical fibre. An entity to merge these different sources together into a single link was written in this thesis and is explained in this section.

The detector data and also the slowcontrol data is written into one buffer FIFO for each data type. The data is not in the format described in the previous section when it is written to the FIFOs. Start and end of a packet are indicated with 4 additional bits in the 36-bit wide FIFOs[6]. The entity which merges the data reads it from the read side from the FIFOs and is also responsible to convert packets into the format described above.

The data that is sent from this entity to the optical link depends on the state of the FEB from the state controller (section 7.2). In the idle state, no packets from the data FIFO are accepted and only slowcontrol packets with the correct format are built and sent over the link. On a state change to run prepare, the run prepare acknowledge signal is sent by this entity. If other parts of the firmware require time to perform run preparations, an option to wait for the AND of ready bits can be added here[7].

[5] Against the direction of detector data-flow.

[6] Less than 4 additional bits would also be sufficient, but FIFOs with such a configuration are not available.

[7] Currently nothing is using this, but the firmware is still in development.

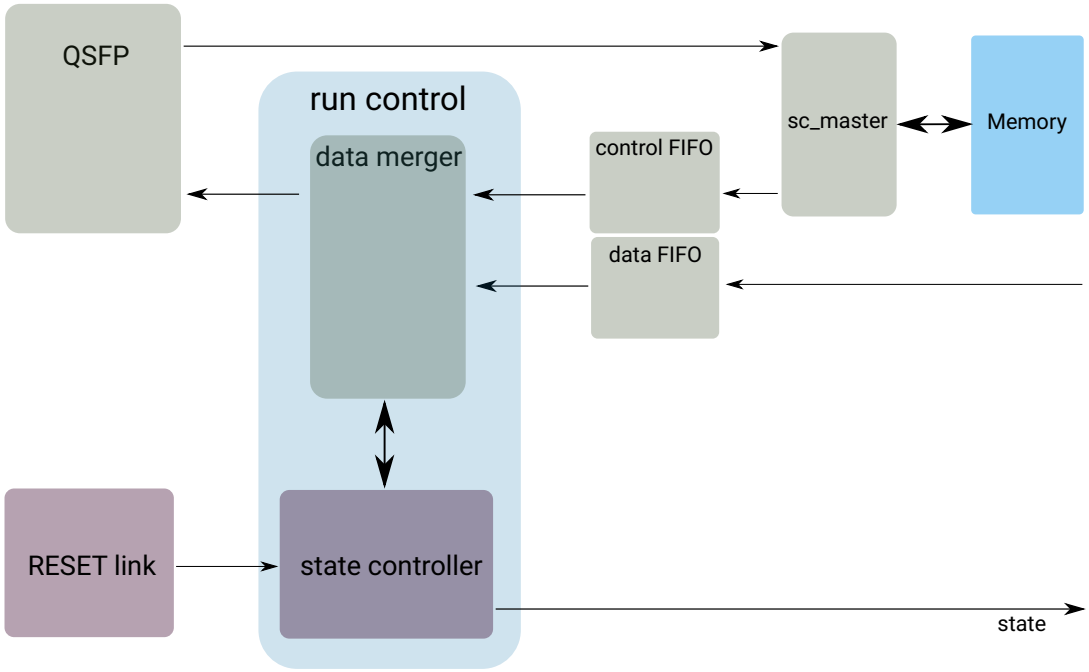


Figure 49: Block diagram of the merging of control and detector data.

If the sync state is entered, no new packets from the FIFOs are accepted. Slow-control packets, which are in transmission during the state change, are transmitted until they are finished and then the link is silent (apart from comma words). In the running state one has two options to configure the data merger: data priority or slow control priority. The priority setting determines from which FIFO the data merger should accept new packets, if packets are available in both FIFOs. Since both sides of the FIFOs are operated with the same clock, it is possible that the FIFO which is not on priority overflows, if the other FIFO occupies the link due to large data rates.

An occupation of the link will also happen if a started packed is not closed. In this case the merger will fill the link with word alignment until the end of packet marker arrives in the FIFO and causes a trailer signal on the link.

The problems above should in principle not occur if the other parts of the firmware are functioning correctly. However, there are a few options how these situations can be avoided. One of them is to implement a timeout for the end of packet after which the merger would end the packet by itself. Another one is to send a single-word run control signal indicating a FIFO overflow. Also possible is a clear of the FIFO on an overflow to avoid packet corruption. None of these options was implemented, since they fix a problem which should not be produced in the first place. Sending a single-word runcontrol to indicate overflows is, however, probably a viable option and will be implemented in the future.

When the run should be stopped and the state controller goes into the terminating state, the data merger will proceed as before until the detector-specific firmware runs

out of data. Then the merger will send the run end run-control signal to the link and give the state controller the permission to fall back to the idle state.

In the link test state a complete bypass of the data merger is used. When the state is entered, the merger finishes the current packet and opens a new packet by sending a header with an ID that corresponds to link tests. Once this is done, it grants exclusive forwarding to the transceiver from an input port, where bit error rate test (BERT) entities can be connected. When the link test state is exited, the data merger stops to forward the data and closes the packet, which was open since the begin of the link test, with a trailer signal. The sync test state is not in use at the moment and eventually obsolete due to the permanent phase measurements in the reset system.

The combination of state controller, data merger and FIFO was simulated and debugged with ModelSim[8] and the whole system was tested with the test setup in the lab. First operations of the system in a testbeam were ongoing while this thesis was written.

7.4 Clock Domains

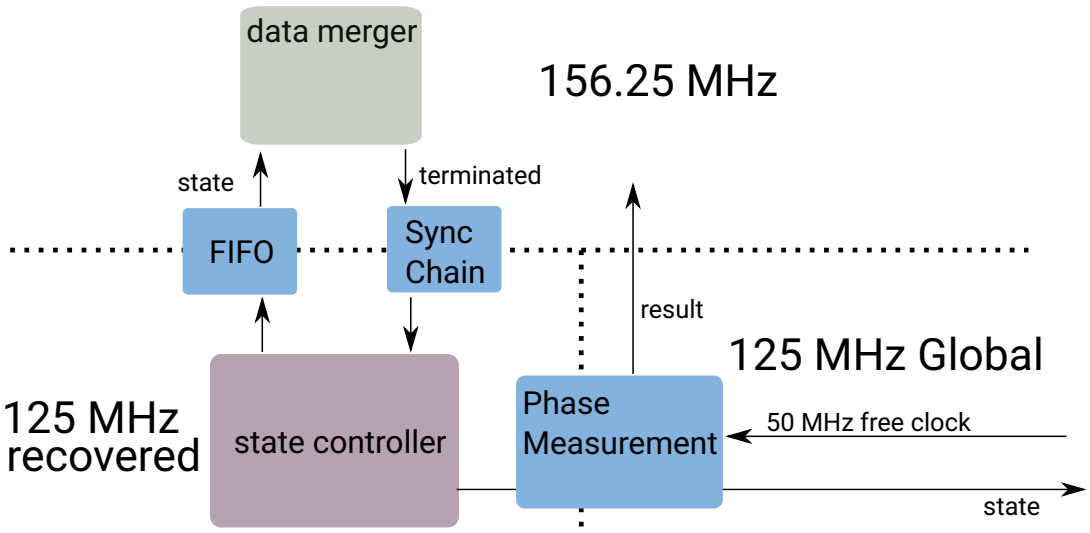


Figure 50: Clock domain block diagram

The system is distributed across three clock domains: the 125 MHz global clock, the 125 MHz clock which is recovered from the reset link and the 156.25 MHz clock that is operating the fast transceiver. Signals between these clock domains can cause problems, as outlined in section 3.2.1.

The FEB state from the state controller has to be transmitted into all clock domains. For the clock domain transition into the 156.25 MHz region a FIFO can be used, since no defined timing of the state change needs to be maintained. The transition into the

[8] A hardware description language (HDL) simulation software.

global clock is done with phase measurements and without a synchroniser as discussed in previous sections. The phase measurement entity is using an asynchronous 50 MHz clock to perform the measurements. The result is transmitted into the 156.25 MHz region, where the NIOS and slowcontrol entities are located. Currently this is done without a synchroniser, since the result changes roughly every second synchronised to an independent clock and can be sampled from the NIOS on a similar timescale. Therefore, an already very large MTBF is expected for this transition.

The permission to enter the idle state is synchronised into the recovered clock domain using a 3-level synchroniser chain.

In order to guarantee timing stability of the reset signals and also for the timing closure of common firmware, it might be a good idea to fix the physical position on the FPGA for parts of the firmware, especially everything in the recovered clock domain and as much as possible in the global clock domain. The Quartus software provides tools to do this, but they were not investigated further at the moment.

This could be particularly helpful once different versions of the firmware for the MuPix, tile and fibre detector are compiled for different FEBs and they want to operate together. A different detector-specific firmware will also have an influence on the placement and timing optimisation of the common firmware parts by default, which can be avoided if these parts are fixed at a specific location on the FPGA.

7.5 Midas Slow Control Bus (MSCB)

The third and also the last way to communicate with the fronted board is the MIDAS slow control bus or MSCB. As described previously, this is intended as a backup solution for the optical connection. If the optical connection fails, it is still necessary to get some critical information out of the detector and a reprogramming of the FPGA is also only possible over MSCB in this case.

MSCB is a serial bus which is fully integrated into the MIDAS slow control system. It consists of an MSCB sub-master and MSCB nodes. The sub-master is connected to the nodes with a serial bus, which is operated at 115 kBaud in the default setting. The connection to MIDAS is then made from the sub-master via ethernet.

Apart from MIDAS, no additional software is needed to operate an MSCB sub-master and the connected nodes. A device which includes the MSCB protocol in hardware or runs an MSCB node in software will self-document itself into the MIDAS online data base. This means that all available variables of this node, their names, current values, length and also their unit are transmitted to MIDAS when the node is initially connected to MSCB and these informations can be shown and edited in the MIDAS web interface without any device specific driver software on the PC.

7.5.1 Data Transmission and Protocol

Some of the important MSCB commands are shown in table 12. The complete set of commands and a complete description of the MSCB system can be found in [55].

Command	Name	Comment
0x0A	CMD_ADDR_NODE16	16-bit address command
0x10	CMD_ADDR_BC	Broadcast addressing
0x1A	CMD_PING16	16-bit ping command
0x28	CMD_GET_INFO	Request for initialisation information
0xA1	CMD_READ	read variable
0x80	CMD_WRITE_NA	write variable
0xBF	CMD_READ_MEM	read memory
0xB7	CMD_WRITE_MEM	write memory

Table 12: Table of some important MSCB commands [55]

These commands are sent with a payload of variable length and a CCITT-8 cyclic redundancy check (CRC) to identify faulty transmissions. An additional 9th bit is added to distinguish addressing commands from data bytes.

No clock is transmitted between the sub-master and the nodes and the communication is, therefore, asynchronous. MSCB relies upon start and stop bits to synchronise master and node for a short time period. When the default logic level (high) is left, nine bits are transmitted and the level goes back to its default. Because transmitter and receiver both know the nominal baud rate, their clocks do not need to be exactly synchronised, since the receiver only needs to sample nine bits. The next nine

bits follow after some time period and start again with a start bit that synchronises both ends of the transmission for these nine bits only. A oscilloscope screenshot of a example MSCB transmission can be found in figure 58 in the appendix.

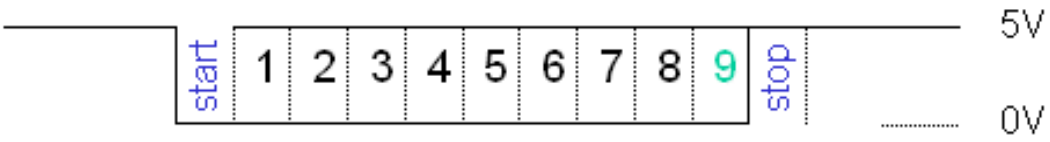


Figure 51: Diagram of an MSCB transmission

7.5.2 Implementation of an MSCB Node on the Frontend Board

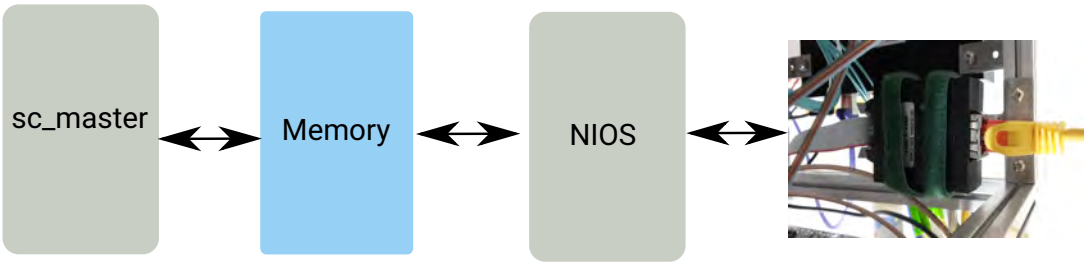


Figure 52: Block diagram of the two different ways to access the memory on the frontend board. The slow control master can access it and communicate with MIDAS through the data merger, optical connection to the switching board and a PCIe interface to the PC. The NIOS can access memory and communicate with MIDAS through MSCB and ethernet.

An MSCB node was implemented on the frontend board to communicate with the NIOS processor and to access the same memory that can also be accessed from the slow control master and the optical control channel from section 7.3.

In the current implementation, the MSCB commands are interpreted by software running on the NIOS processor. The processor then also executes the MSCB commands to write to or read from memory.

A command that arrives at the frontend board is parallelised and written to a showahead[9] FIFO, where it is available for the NIOS. Replies to MSCB are also written to a FIFO, serialised and then sent to the MSCB sub-master.

Later on during this thesis, the input to the NIOS was changed to use interrupts when MSCB commands are available in order to avoid a periodical check of the FIFO empty signal from NIOS software. This would also mean that each byte transmitted on the bus causes an interrupt for the processor of all frontend boards connected to this sub-master, which is a quite significant and unnecessary amount of processing

[9] A FIFO that provides the next data word at its output before it was requested. A read command to this FIFO acts as a read acknowledge.

time for the NIOS processors. This is the reason why also an MSCB addressing entity was implemented in firmware to sort out all commands that are not intended for the node that the entity is operated on. Also implemented into this firmware part is a limited buffering of the MSCB transmission which leads to only a single interrupt per command to the NIOS, instead of one interrupt for each transmitted byte.

The newest version of MSCB uses 2.5 V signals and in the final experiment these signals will be converted into an optical signal and back to guarantee a complete electrical decoupling of all DAQ components inside of the magnet. However, the MSCB hardware available during this thesis was only capable to operate with a 5 V signal standard, which had to be converted to a lower voltage in order to connect the MSCB bus to the used Altera FPGAs. This was done with a converter chip [56] and the converted bus was then connected to the test boards over their LCD headers.

All connected boards use the same line to transmit their data. Therefore, only one board at a time is allowed to drive a logic level into this cable. Multiple driving boards would effectively be a shortage and could potentially damage the involved components. Some output pins of FPGAs support three possible states: 0, 1 and high-impedance. In the last state, no voltage is driven to the pin, it is set to a high-impedance against ground and can be used as input instead of an output. This third state was used as default status of the implemented MSCB nodes.

7.5.3 MIDAS frontend for memory access with MSCB

A MIDAS frontend and custom web page was written to access the NIOS memory. The goal here was to create the possibility to use the MSCB system in the same way that the optical connection can be used for control information, in principle resulting in a completely redundant system with a very reduced bandwidth.

The MIDAS web page used for NIOS memory access with the optical connection is shown together with the MSCB version in figures 53 and 54. The MSCB version does not require the panels for the PCIe memory, but provides the same functionality otherwise.

The user interfaces shown here are, of course, only intended for testing purposes. A final system will rather have user interfaces for sub-detectors or detector modules instead of a simple memory access. The low level functionalities will be buried beyond more complicated functions. However, also for these functions a complete redundancy with two control paths to execute any commands on the frontend board is possible.

At the moment, the function to access the FEB memory from the optical connection is located in the PCIe driver. For the future it might be useful to move this function out of the PCIe driver and include the choice of the control channel (optical or MSCB) as a parameter of this function.

Apart from the MSCB nodes on the ArriaV development board and the FEB prototype, also some tests with a pure hardware implementation of a MSCB node on a MAX10 FPGA were performed by students in a lab course [57]. In this project, a memory access command was used to request an ADC value from the MAX10 (without an actual memory access). The firmware was built upon the existing firmware for

the ArriaV and FEB prototype boards.

Eventually the full functionality and the full protocol of an MSCB node is not required for the FEB operation in the Mu3e experiment. If all required functionality could be boiled down to memory access on the frontend board, then it could be reasonable and useful to consider a full hardware implementation of this functionality using the other side of the dual-port memory in figure 52. The decoding of MSCB memory access commands could be in principle implemented into the sc_master hardware, as the student-project has shown. The NIOS would not need to run any MSCB decoding software in this case and the MSCB system would also not rely upon a running processor. Whether this is a viable option or not remains to be seen.

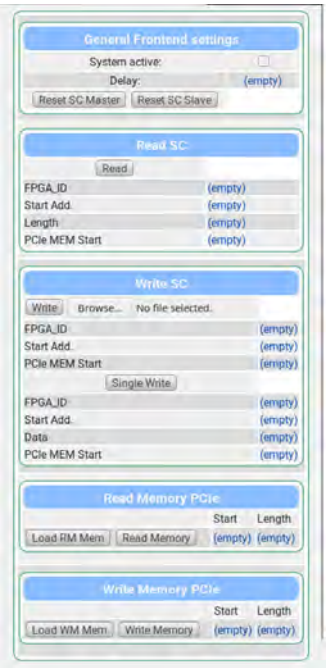


Figure 53: MIDAS control GUI for NIOS memory access using the optical connection, developed in [48]

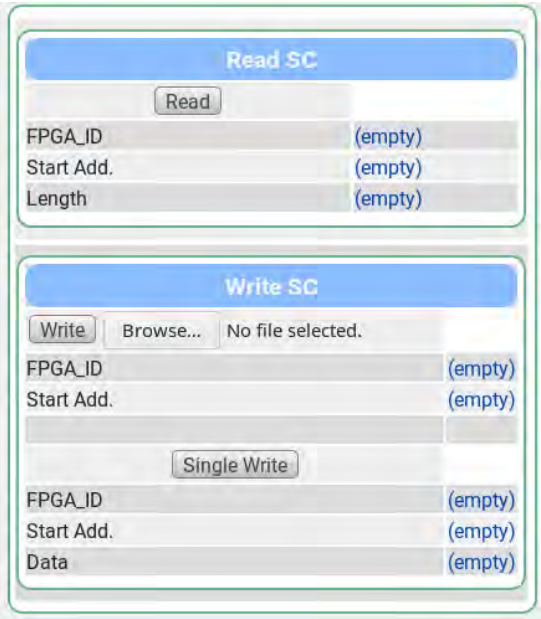


Figure 54: MIDAS control GUI for NIOS memory access using MSCB

8 Conclusion and Outlook

Mu3e is a planned high intensity particle physics experiment searching for the lepton flavour violating decay $\mu^+ \rightarrow e^+e^-e^+$with a sensitivity goal of one in 10^{16} muon decays. In order to achieve this, a data rate of up to 80 GBit/s has to be processed in a triggerless data acquisition system for the first phase of the experiment.

This data is produced by high-voltage monolithic active pixel sensors, scintillating tiles and scintillating fibres. A concept for the synchronisation of these three different detectors was presented and implemented on field programmable gate arrays. The concept uses permanent phase measurements in firmware to ensure an alignment with a global 125 MHz clock system. An LVDS version and also an optical version of the synchronisation system was implemented and the performed tests have shown their suitability to synchronise the components in the Mu3e detector. The precision of this synchronisation is limited by the jitter of the clock distribution, which was found to be (1.44 ± 0.36) ps at the output of the distribution board. This would be sufficient to avoid significant influences on the overall time resolution of the Mu3e detector.

Apart from synchronisation of the three sub-detectors, also an integration of their control systems into a common solution is necessary. It is intended to use the MIDAS system for this purpose. Two redundant control channels were implemented and have been tested during this thesis. One of these channels transmits the control data together with the detector data through optical fibres to the outside of the magnet, where it is accessed from a PCIe interface. The other channel uses a implementation of a MSCB node, which is partially located in software and partially in hardware. Whether a full hardware implementation of this system is feasible remains to be examined once the exact requirements are clear.

A detailed integration of functionalities of the sub-detectors and also their control software into the new data acquisition has to follow in the future and will be based upon the procedures that each sub-detector was using in his preliminary DAQ used during his development phase.

The setup built during this thesis can act as a blueprint and testing environment for the final Mu3e data acquisition. The components used here will be gradually replaced by the final components as they become available.

Appendices

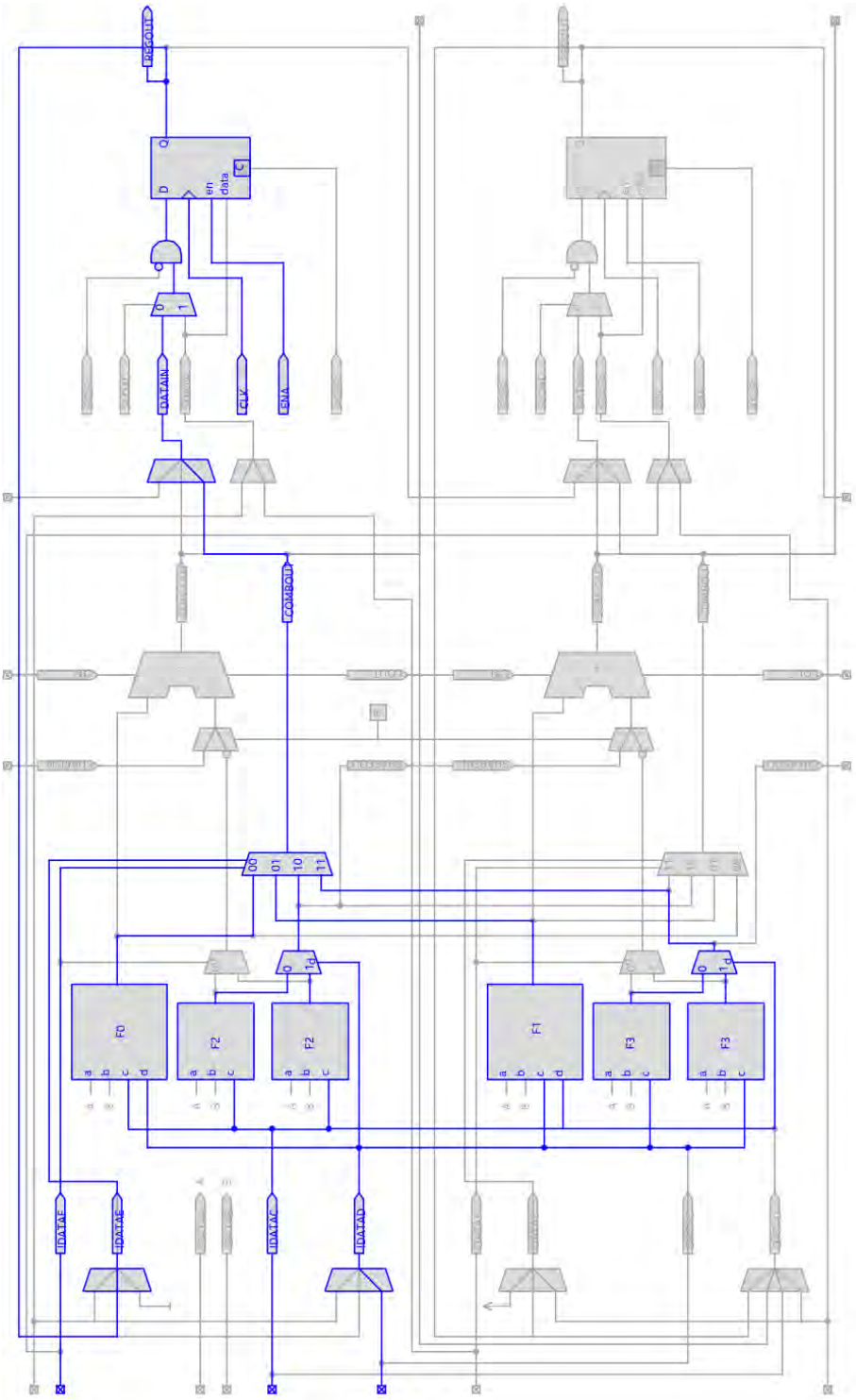


Figure 55: Schematic of an adaptive logic module (ALM) from a Altera StratixIV FPGA. Taken from Quartus Prime Chip Planner.

Figure 56: Custom web page for the reset system

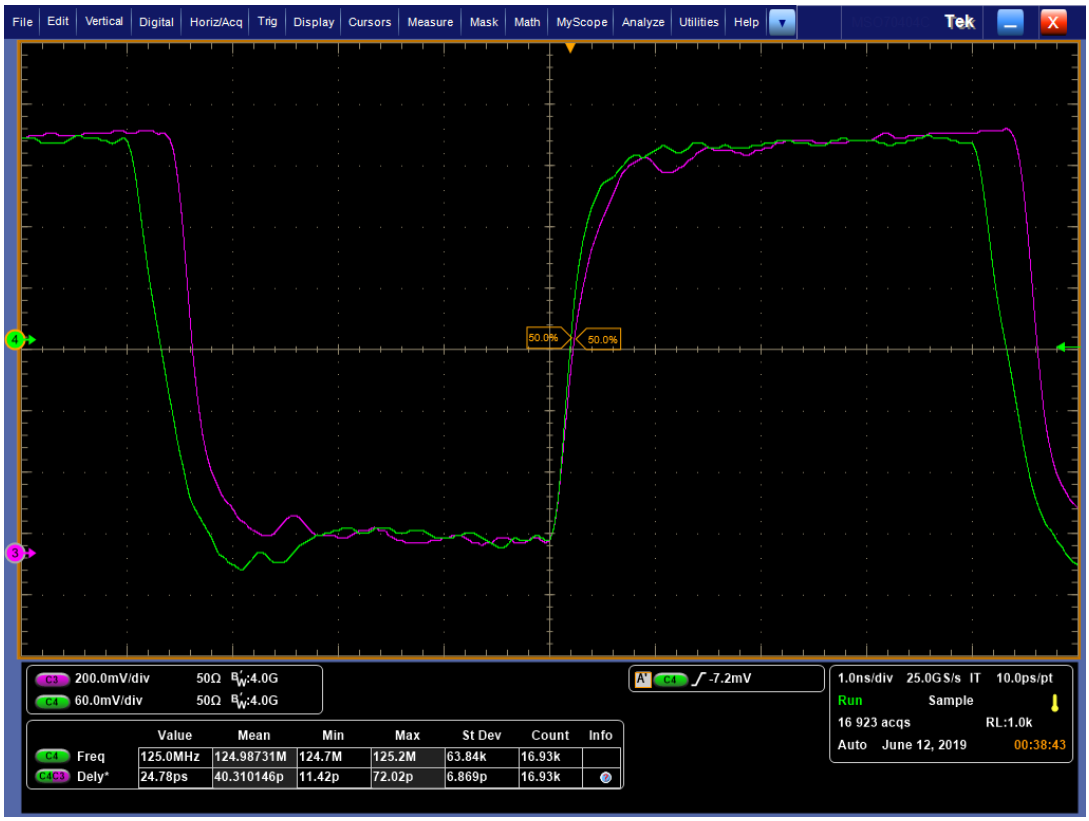


Figure 57: Duty cycle measurement of the recovered clock (purple)

Figure 58: Example of a MSCB transmission measured with a oscilloscope. The blue line is the MSCB bus, the green line is an output enable signal for the used converter chip.

List of Figures

1 Particles in the standard model 1
2 Michel decay of the muon 3
3 $\mu \to 3e$ via neutrino mixing 3
4 History of Lepton Flavour Violation searches 4
5 Mu3e decay at tree level 5
6 Mu3e decay in a SUSY loop 5

7 CAD model of the Mu3e beamline 8
8 $\mu^+ \to e^+\nu_e\overline{\nu}_\mu e^+e^-$ background process 9
9 Combinatorial background 9
10 Schematic of the Mu3e detector 10
11 MuPix schematic 11
12 Momentum resolution schematic 12
13 Schematic of the electronics in the MuPix 13
14 CAD drawing of the scintillating fibres 13
15 Schematic of a tile module 14
16 CAD drawing of a recurl station 14

17 n-channel MOSFET 18
18 Implementation of a NAND gate 19
19 Table of logic gates 19
20 Clocked flip flop 20
21 Example logic 21
22 ALM Block diagram 22
23 LUT Block diagram 23
24 Timing violations and Metastability 25
25 Synchronsiation Chain 26
26 Schematic of a phase locked loop(PLL) 28

27 Diagram of a PMA block of an ArriaV transceiver 33
28 Diagram of a PCS block of an ArriaV transceiver 34

29 Overview of the Mu3e DAQ 37
30 Picture of the frontend board prototype 38
31 Picture of the PCIe40 Board 39
32 Picture of the DE5a-Net development board 40

33 Test setup for the Mu3e DAQ 41

34 Diagram of the test setup for the Mu3e DAQ 42

35 Mu3e DAQn control diagram 45
36 MIDAS custom page for DHCP and DNS server 47
37 MIDAS bank structure 48
38 Midas logger write speed 49
39 Picture of the clock distribution board 51
40 Picture of the GENESYS2 board 51
41 Reset state machine 52
42 Jumps of the arrival of reset commands 54
43 Working principle of the phase measurement 56
44 Measurement results from the phase entity 57
45 Phase measurement simulation with jitter 57
46 Phase measurement simulation with a different duty cylce 57
47 Example for jitter of an FPGA output signal 58
48 Phase measurements with the LVDS receiver 59
49 Diagram of data and control merging 65
50 Clock domain block diagram 66
51 Diagram of a MSCB transmission 69
52 caption 69
53 NIOS memory GUI with optical connection 71
54 NIOS memory GUI using MSCB 71

55 Schematic of an StratixIV ALM 78
56 Custom web page for reset system 79
57 Duty cycle measurement 80
58 Example of a MSCB transmission 81

List of Tables

1 Decay channels of the muon in the SM 2

2 Experimental results for LFV muon decays 7

3 List of FPGAs used for the Mu3e PhaseI readout system 24

4 8b10b control symbols 32
5 Byte field of a PCIe write request. 35

6 FEB System States [53] 50
7 Reset link protocol [53] 50
8 Structure of MuPix Data 61
9 Structure of a MuTrig Data Packet 62
10 Structure of Slow Control 62
11 Structure of a run control signal 63
12 Important MSCB commands 68

List of Abbreviations

ADC analog-to-digital converter

ALM adaptive logic module

ASIC application specific integrated circuit

BAR base address register

BERT bit error rate test

BSM beyond standard model

CDR clock data recovery

CPU central processing unit

CMOS complementary metal-oxide semiconductor

daq digital-to-analog converter

DAQ data aquisition

DMA direct memory access

DPA dynamic phase alignment

FIFO first in/first out buffer

FPGA field programmable gate array

GPU graphics processing unit

GUI graphical user interface

HDL hardware description language

HIPA high intensity proton accelerator

HSMC high speed mezzanine card

HV-MAPS high voltage monolithic active pixel sensors

I2C inter-integrated circuit

I/O input/output

IP interlectual property

JTAG joint test action group

LSB least significant bit

LUT lookup table

LVDS low voltage differential signaling

MAP monolithic active pixel

MSB most significant bit

MTBF mean time between failures

PCB printed circuit board

PCI express peripheral component interconnect express

PCS physical coding sublayer

PLL phase locked loop

PMA physical medium attachment

PSI Paul Scherrer Institute

QSFP quad small formfactor pluggable

RAM random access memory

ROM read only memory

SFP small formfactor pluggable

SM Standard Model

SPI serial periphal interface

SRAM static random access memory

TCL tool command language

UART universal asynchronous receiver-transmitter

VCO voltage controlled oscillator

Bibliography

[1] Georges Aad et al.
Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC.
Phys. Lett., B716:1–29, 2012.

[2] Serguei Chatrchyan et al.
Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC.
Phys. Lett., B716:30–61, 2012.

[3] Randolf Pohl, Ronald Gilman, Gerald A. Miller, and Krzysztof Pachucki.
Muonic hydrogen and the proton radius puzzle.
Annual Review of Nuclear and Particle Science, 63(1):175–204, 2013.

[4] G. W. Bennett et al.
Final Report of the Muon E821 Anomalous Magnetic Moment Measurement at BNL.
Phys. Rev., D73:072003, 2006.

[5] K. Nakamura et al.
Particle physics booklet.
Journal of Physics, G37:075021, Jul 2010.

[6] Y. Fukuda et al.
Evidence for oscillation of atmospheric neutrinos.
Phys. Rev. Lett., 81:1562–1567, 1998.

[7] A. Blondel et al.
Research Proposal for an Experiment to Search for the Decay $\mu \to eee$.
ArXiv e-prints, January 2013.

[8] William J. Marciano, Toshinori Mori, and J. Michael Roney.
Charged lepton flavor violation experiments.
Annual Review of Nuclear and Particle Science, 58(1):315–341, 2008.

[9] Mitsuru Kakizaki, Yoshiteru Ogura, and Fumitaka Shima.
Lepton flavor violation in the triplet Higgs model.
Phys. Lett., B566:210–216, 2003.

[10] J. Bernabéu, E. Nardi, and D. Tommasini.
μ-e conversion in nuclei and z' physics.
Nuclear Physics B, 409(1):69 – 86, 1993.

[11] Yoshitaka Kuno and Yasuhiro Okada.
Muon decay and physics beyond the standard model.
Rev. Mod. Phys., 73:151–202, Jan 2001.

[12] Monika Blanke, Andrzej J Buras, Björn Duling, Anton Poschenrieder, and Cecilia Tarantino.
Charged lepton flavour violation and(g-2)mu in the littlest higgs model with t-parity: a clear distinction from supersymmetry.
Journal of High Energy Physics, 2007(05):013–013, may 2007.

[13] K. S. Babu and J. Julio.
Two-Loop Neutrino Mass Generation through Leptoquarks.
Nucl. Phys., B841:130–156, 2010.

[14] A. Blondel et al.
Research Proposal for an Experiment to Search for the Decay $\mu \to eee$.
2013.

[15] U. Bellgardt et al.
Search for the Decay mu+ $\to$ e+ e+ e-.
Nucl. Phys., B299:1–6, 1988.

[16] A. M. Baldini et al.
Search for the lepton flavour violating decay $\mu^+ \to e^+\gamma$ with the full dataset of the MEG experiment.
Eur. Phys. J., C76(8):434, 2016.

[17] W. Honecker et al.
Improved limit on the branching ratio of $\mu \to$ e conversion on lead.
Phys. Rev. Lett., 76:200–203, Jan 1996.

[18] C. Dohmen et al.
Test of lepton-flavour conservation in $\mu \to$ e conversion on titanium.
Physics Letters B, 317(4):631 – 636, 1993.

[19] J. Kaulard et al.
Improved limit on the branching ratio of$\mu \to$ e conversion on titanium.
Physics Letters B, 422(1):334 – 338, 1998.

[20] W. Bertl et al.
A search for μ-e conversion in muonic gold.
The European Physical Journal C - Particles and Fields, 47(2):337–346, Aug 2006.

[21] Mike Seidel et al.
Production of a 1.3 MW Proton Beam at PSI.
Conf. Proc., C100523:TUYRA03, 2010.

[22] The Mu3e collaboration.
Mu3e internal Wiki.

[23] D. vom Bruch.

Pixel Sensor Evaluation and Online Event Selection for the Mu3e Experiment.
PhD. Thesis, Heidelberg University, 2017.
https://www.psi.ch/sites/default/files/import/mu3e/ThesesEN/DissertationVomBruch.pdf.

[24] Ivan Perić.
A novel monolithic pixelated particle detector implemented in high-voltage CMOS technology.
Nucl. Instrum. Meth., A582:876–885, 2007.

[25] Jerome Baudot et al.
First test results of mimosa-26, a fast cmos sensor with integrated zero suppression and digitized output.
pages 1169 – 1173, 12 2009.

[26] I Valin et al.
A reticle size CMOS pixel sensor dedicated to the STAR HFT.
Journal of Instrumentation, 7(01):C01102–C01102, jan 2012.

[27] C. Cavicchioli and others.
Design and characterization of novel monolithic pixel sensors for the alice its upgrade.
Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 765:177 – 182, 2014.

[28] Heiko Augustin et al.
The MuPix System-on-Chip for the Mu3e Experiment.
Nucl. Instrum. Meth., A845:194–198, 2017.

[29] U. Hartenstein.
Track Based Alignment for the Mu3e Pixel Detector.
PhD. Thesis, JGU Mainz, 2019.
https://www.psi.ch/sites/default/files/2019-05/DissertationHartenstein.pdf.

[30] H. Augustin et al.
Irradiation study of a fully monolithic HV-CMOS pixel sensor design in AMS 180 nm.
Nucl. Instrum. Meth., A905:53–60, 2018.

[31] S. Corrodi.
A Timing Detector based on Scintillating Fibres for the Mu3e Experiment.
PhD. Thesis, ETH Zurich, 2018.
https://www.psi.ch/sites/default/files/import/mu3e/ThesesEN/DissertationCorrodi.pdf.

[32] Antoaneta Damyanova.
Development of the Scintillating Fiber Detector for Timing Measurements in the Mu3e Experiment.
PhD. Thesis, University of Geneva, 2019.
https://www.psi.ch/sites/default/files/2019-10/DissertationDamyanova.pdf.

[33] H.P. Eckert.
The Mu3e Tile Detector.

PhD. Thesis, Heidelberg University, 2015.
https://www.psi.ch/sites/default/files/import/mu3e/ThesesEN/DissertationEckert.pdf.

[34] Rainer [Hrsg.] Waser, editor.
Nanoelectronics and information technology : advanced electronic materials and novel devices.
Wiley-VCH, Weinheim, 2., corr. ed. edition, 2005.

[35] Winfield Horowitz, Paul ; Hill, editor.
The art of electronics, volume [Hauptbd.].
Cambridge Univ. Press, Cambridge [u.a.], 2. ed., repr. edition, 1997.

[36] Fpga architecture.
Altera Corp., San Jose, CA, USA, 2006.
White Paper WP-01003-1.0.

[37] Jennifer Stephenson et al.
Understanding metastability in fpgas.
Altera Corp., San Jose, CA, USA, 2009.
White Paper WP-01082-1.2.

[38] Texas Instruments.
Fractional/Integer-N PLL Basics.
Technical Brief SWRA029, 1999.
http://www.ti.com/lit/an/swra029/swra029.pdf.

[39] Nios ii processor reference guide.
Altera Corp., San Jose, CA, USA, 2019.

[40] Susan C. Hill et al.
Queued serial peripheral interface for use in a data processing system.
US patent US4816996 , 1989.

[41] NPX Semiconductors.
I2C-bus specification and user manual.
2014.
https://www.nxp.com/docs/en/user-guide/UM10204.pdf.

[42] F. Gray.
Pulse code communication, March 17 1953.
US Patent 2,632,058.

[43] Randy Johnson et al.
Jtag 101.
Intel Corp. White Paper, 2009.
www.intel.com/content/dam/www/public/us/en/documents/white-papers/jtag-101-ieee-1149x-paper.pdf.

[44] Albert X. Widmer Peter A. Franaszek.
Byte oriented DC balanced (0,4) 8B/10B partitioned block transmission code.
US patent US4486739A, 1982.

[45] 8b/10b encoding — Wikipedia, the free encyclopedia, 2019.
[Online; accessed 19-October-2019].

[46] Arria V Device Handbook Volume 2: Transceivers.
Altera Corp., San Jose, CA, USA, 2019.
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/arria-v/av_5v3.pdf.

[47] Xillybus Ltd.
Down to the TLP: How PCI express devices talk (Part I).
http://xillybus.com/tutorials/pci-express-tlp-pcie-primer-tutorial-guide-1.

[48] Marius Köppel.
Data Flow in the Mu3e Filter Farm.
Master Thesis, JGU Mainz, 2019.
in preparation, will be available here: https://www.psi.ch/en/mu3e/theses.

[49] P. Durante, N. Neufeld, R. Schwemmer, G. Balbi, and U. Marconi.
100 gbps PCI-express readout for the LHCb upgrade.
Journal of Instrumentation, 10(04):C04018–C04018, apr 2015.

[50] Avago Technologies.
MiniPOD^{TM} AFBR-812VxyZ, AFBR-822VxyZ Product Brief.
AV02-4038EN, 2013.

[51] Si5330/34/35/38 EVALUATION BOARD USER'S GUIDE.
Silicon Labs, Austin, TX 78701, USA, 2011.
https://www.silabs.com/documents/public/user-guides/Si5338-EVB.pdf.

[52] MIDAS Wiki.
TRIUMF.
https://midas.triumf.ca/MidasWiki, Accessed: 05.10.2019.

[53] Niklaus Berger.
Run Start and Reset Protocol.
Mu3e Internal Note 0046, 2018.

[54] LVDS SERDES Transmitter / Receiver IP Cores User Guide.
Altera Corp., San Jose, CA, USA, 2017.
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_altlvds.

[55] Stefan Ritt.
Midas Slow Control Bus (MSCB).
PSI, 2001.
https://elog.psi.ch/mscb.

[56] Analog Devices.
ADM2682E/ADM2687E (Rev. C) Data sheet.
2015.
https://download.mikroe.com/documents/datasheets/ADM2682E_2687E.pdf.

[57] M. Heppener and Y. Witzky.

Analysis of Max 10 FPGA Board.
2019.
student lab project, JGU Mainz.

Acknowledgements

I would like to thank everyone who has supported me during this thesis.

First of all, I would like to thank Prof. Niklaus Berger for the opportunity to write my master thesis on this interesting topic in his group and for the advice and discussions during this thesis. I would also like to thank Prof. Büscher, who agreed to take over the role of my second referee.

The design choices in the Mu3e DAQ caused regular discussions with Alexandr Kozlinskiy, Marius Köppel and Niklaus Berger. I very much enjoyed these.

Thanks also to all members of the AG Berger and the Mu3e collaboration for the relaxed atmosphere and interesting meetings.

Also many thanks to everyone who volunteered for the proof reading of my thesis: Carsten Grzesik, Alexey Kivel, Julien Laux and Marius Köppel.

Last but not least I would like to thank my family for the support during my studies.