Development and Data Analysis of the LAr Calorimeter Trigger Readout Prototype at ATLAS

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science

vorgelegt von

Philipp Horn
geboren am 01. November 1990 in Zittau

der

TECHNISCHEN UNIVERSITÄT DRESDEN

INSTITUT FÜR KERN- UND TEILCHENPHYSIK
FACHRICHTUNG PHYSIK
FAKULTÄT MATHEMATIK UND NATURWISSENSCHAFTEN

2016

Eingereicht am 29. März 2016

1. Gutachter:	Prof. Dr. Arno Straessner
2. Gutachter:	Prof. Dr. Kai Zuber

Abstract

Due to an increase in luminosity of the LHC, the trigger system of the ATLAS LAr calorimeter is improved. A higher granularity and a new readout chain is needed. First this master thesis examines the possible insertion of a histogram building device. It would be the only possibility to have access to untriggered data during normal operation. Secondly a system to control and change individual parts of the readout chain is tested. In the end actual calibration data is analyzed and compared to the simulation.

Kurzdarstellung

Durch eine Erhöhung der Luminosität muss das Trigger System des ATLAS LAr Kalorimeters verbessert werden. Eine feinere Granularität und ein neuer Teil der Auslese wird benötigt. Zuerst wird in dieser Masterarbeit die mögliche Einbringung eines "Histogrammers" untersucht. Dies wäre die einzige Möglichkeit, während des regulären Betriebs, Zugriff auf ungetriggerte Daten zu erhalten. Danach wird ein System zum Steuern der einzelnen Bestandteile der Auslese getestet. Zum Schluss werden Kalibrationsdaten analysiert und mit der Simulation verglichen.

Contents

1. Introduction 1

2. Physical Foundations 3
   2.1. The Standard Model . . . 3
   2.2. Proton Collider Physics . . . 4

3. The Experiment 7
   3.1. The Large Hadron Collider . . . 7
   3.2. The ATLAS Detector . . . 9
   3.3. The Liquid Argon Calorimeter . . . 11
   3.4. The Trigger System . . . 13

4. The Upgrade Plans 15
   4.1. The Timeline for the LHC . . . 15
   4.2. Changes at the ATLAS Detector . . . 16
   4.3. Readout Architecture . . . 18
   4.4. The LAr Digital Processing System . . . 20

5. Test Environment 23
   5.1. FPGA . . . 23
   5.2. Programming of the FPGA . . . 24
   5.3. SoCKit Development Kit . . . 26
   5.4. Avalon Interfaces . . . 28
   5.5. Ethernet connection . . . 30

6. Histogram Builder 33
   6.1. Motivation . . . 33
   6.2. DDR3 memory . . . 33
   6.3. Data handling process . . . 34
   6.4. Result and Discussion . . . 36

7. IPBus 39
7.1. Structure 39
7.2. Implementation into LATOME 40
7.3. uHal 42
7.4. Reading of the DDR3 via IPBus 45

8. Calibration 51
8.1. Introduction 51
8.2. Signal shape 52
8.3. Detector geometry 56
8.4. Linear Response 59
8.5. Saturation 61

9. Summary and Outlook 65

List of Figures 67

List of Tables 69

A. Compilation of the Produced Code 71

B. Calibration Result and Comparison 79

Bibliography 87

1. Introduction

What is everything made of? How did the universe look like shortly after the Big Bang? These are questions particle physics is trying to answer. In the recent years there were many tremendous achievements in this field. Several Nobel prices were awarded to physicists working in this area. Quantum field theories like the Standard Model have been developed and confirmed by many experiments [1].

Despite this success there are still many unanswered questions. For example dark matter and dark energy, which make up 95 percent of the universe, are still unexplained [2]. These and other problems like the asymmetry of matter and antimatter [3] or the role of gravitation need to be addressed and inspected [4]. To find answers special experiments are built.

One well established method is the acceleration and collision of particles and the analysis of the resulting collision products [5]. The aim is to have as much energy as possible available to generate heavier particles with a higher probability. The conditions during the experiment are in addition closer to the Big Bang. The particles are usually bent on a circular track, since then it passes the acceleration devices multiple times and reaches higher energies.

The next question is the choice which particle is picked for the collisions. Electrons are a good candidate because, being fundamental particles, their initial energy is completely transfered in the reaction, independently of the scattering parameters. The disadvantage is the low mass, which results in strong synchrotron radiation, when forced on a bent curve. Due to this effect, a lot of energy is lost and thus the maximum collision energy limited [6]. Protons on the other hand are much heavier and therefore less affected by synchrotron radiation. The limiting factor for these particles is the strength of the magnets, which are needed to keep them on a circular track.

Without changing the magnets a higher energy can be gained, if the radius of the circle is increased. Therefore experiments like the Large Hadron Collider (LHC) with a circumference of 27 km are built [7]. Along this machine multiple detectors analyze the reactions.

The biggest of these is called ATLAS, which is a general purpose detector searching for new physics. The whole master thesis is related to this detector and therefore an introduction is given in Chapter 3.

These unique machines already have made important discoveries, like the Higgs-Boson [8]. In the future it is however important to increase the number of collisions to detect very rare reactions, which are naturally the most interesting ones. Therefore the accelerator and detectors need to be upgraded. The motivation and plans for the future for these improvements of the experiment will be explained in Chapter 4. This section is also used to introduce the electronic readout of the LAr-Calorimeter. Most of the work during the master thesis is related to the improvement. It is divided into two parts, which are described in Chapter 6 and 7. The firmware used to program the detector signal processors had to be verified and therefore a test environment was set up in Dresden. The development is covered in Chapter 5.

Finally it was possible to gain access to the latest calibration data from ATLAS using a demonstrator system of the newly developed readout. In Chapter 8 this information is analyzed and interpreted. The result is compared to simulation to obtain more knowledge about its credibility.

2. Physical Foundations

2.1. The Standard Model

The Standard Model (SM) [9] [10] [11] is able to explain many experiments and summarizes most of our knowledge in particle physics. Therefore it is one of the most important and acknowledged theories and several experimental results could be successfully predicted. This chapter gives a short overview of the topic. For further information refer to [12].

All matter is made out of these kinds of elementary particles: leptons, quarks, force mediators and excitations of the Higgs field. The first two are fermions, which means that they obey the Pauli exclusion principle. Two identical fermions cannot occupy the same quantum state. In this way they also form stable bound states and construct matter.

The force mediators are gauge bosons. They do not obey the Pauli exclusion principle and all of them can be in the ground state. This makes them the interaction particles between the fermions and each force has their mediators. The photon for the electromagnetic, $W^{\pm}$ and the Z for the weak and the gluons for the strong force. The corresponding symmetries groups to the SM are $SU(3)_C \times SU(2)_L \times U(1)_Y$. $SU(3)_C$ requires the existence of gluons. $SU(2)_L \times U(1)_Y$ is the symmetry group of the electroweak unification with the weak isospin I and the weak hypercharge Y. Gravitation, which is generally considered to be too weak to affect the experiments, is not described in the SM. The Higgs boson plays an important role in the SM. It was just recently discovered and is the excitation of the Higgs field, which gives the particles their masses through weak interaction.

Three of the six leptons have an electrical charge. Therefore only these have the ability to interact with photons. The six quarks have not only an electric charge, but also a color charge and participate in the strong interaction. Table 2.1 summarizes the generations of the fermions. Each of these particles has a partner, the so called antimatter. They have the same mass, but opposite charge.

	Generation 1	 2	 3	Electrical charge	Color charge
Quarks	up down	charm strange	top bottom	+2/3 -1/3	red, blue, green
Leptons	electron (e) e-neutrino	muon (μ) μ-neutrino	tau (τ) τ-neutrino	-1 0	none

Table 2.1.: Overview of the different generations of fermions and their properties

Although the SM describes many observations in particle physics, it still leaves many aspects unexplained. Besides the missing gravitation [4] also dark matter and dark energy, which represents over 95 percent of the universe, are not included in the theory [2]. In addition some aspects of the SM have been proven wrong. The verification of neutrino oscillation [13] showed that neutrinos do have a mass, which is stated otherwise in the model [9] [10] [11]. This shows that there is still a lot more to explore and more precise measurements are needed.

An extension of the SM may explain many of these issues. A lot of popular models are based on the concept of Super Symmetry [14]. Therein it is implicated that every fermion has a bosonic partner and every boson has a fermionic partner. The exact symmetry has to be broken, since partners with the same mass were not found. There is no experimental evidence yet and new energy ranges must be explored. The lightest super symmetric particle would be a good candidate for dark matter [15].

2.2. Proton Collider Physics

The theory provides prediction for parameters, which can be verified by experiments. One of these parameters is the cross section σ, which is in our case the probability that a certain process occurs. A higher value means a greater probability that a certain reaction will happen.

A possibility to verify the prediction is the collision of particles and the analysis of the resulting collision products. The number of events N of a certain process are counted for a time t and set in relation with the cross section:

$$\frac{dN}{dt} = L\sigma$$

The proportionality factor is the luminosity L. This value is collider dependent and the aim is to increase it as much as possible, since there would be more events to occur.

Figure 2.1 displays the production cross section of several processes. The predictions of the theory are shown in gray and the results of experiments are displayed with different colors. The graph also shows, that many interesting physical processes have a small cross section, which makes it important to increase the luminosity to still be able to detect enough events to observe a significant signal [16].

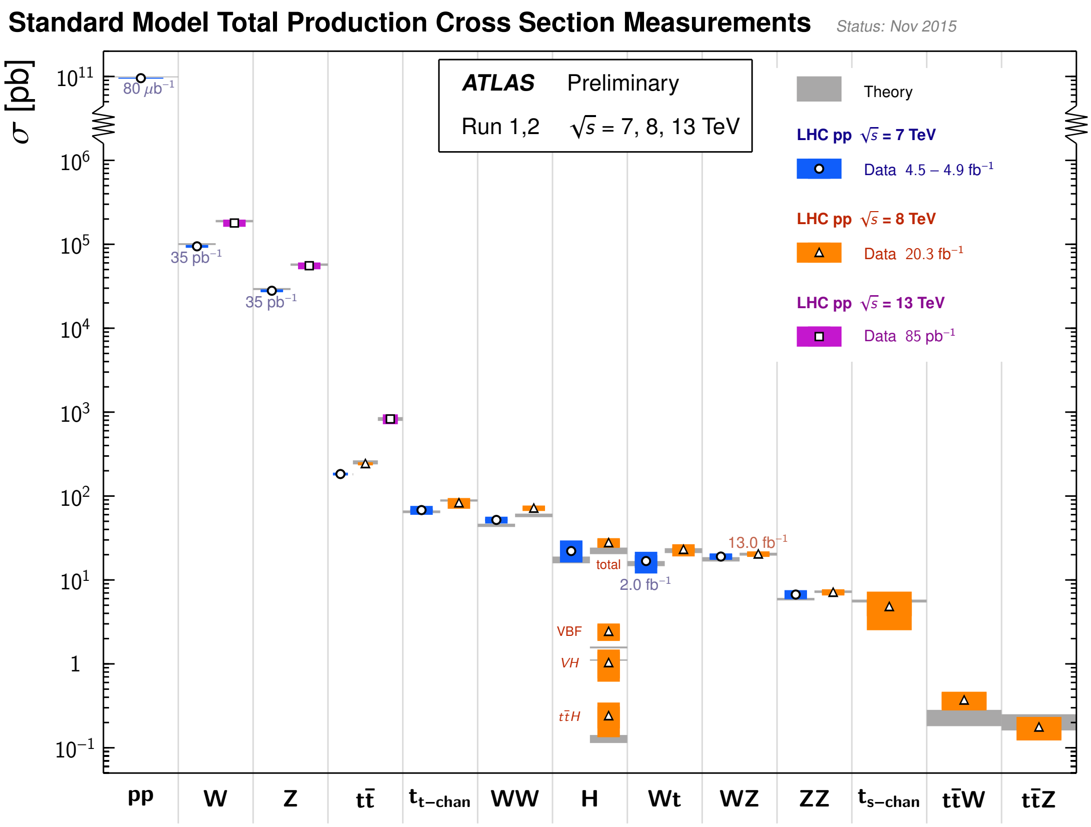


Figure 2.1.: The production cross section of several measurements are shown. The interesting reactions, like Higgs productions, have very small values. [17]

Another important collider parameter is the center-of-mass energy. It is the maximal possible energy, which is available in the collision. In the case of the LHC it is the sum of the energies of the colliding particles [5]. A higher center-of-mass value increases the probability to generate new heavy particles and is therefore desirable. At the LHC, protons are used for the collisions. They can be accelerated to high energies, but have the disadvantage of not being fundamental. The proton consists of quarks and gluons, which interact with each other during the collision. Therefore the initial energy which contributes to the hard scattering reaction is not known.

Unfortunately not every resulting particle can be detected. Neutrinos for example are very rarely interacting particles. The knowledge of the initial energy and the comparison to the sum of the energy of the detected particles would provide information about the neutrinos. A possibility to solve this would be to assume that the momentum of the protons transverse to the beamline is zero at the time of the collision. The conservation of momentum demands the transverse momentum of the sum of the resulting particles must be zero as well. If this value strongly deviates from zero, another undetected particle was generated [18].

In this master thesis, the transverse energy will be used:

$$E_\mathrm{T} = \sqrt{m^2 + p_\mathrm{T}^2}$$

The mass is m and p_T is the transverse (relative to the beam axis) momentum of the particle.

3. The Experiment

3.1. The Large Hadron Collider

The Large Hadron Collider (LHC) is the biggest particle collider in the world. It is located at the Swiss-French border near Geneva at CERN inside a tunnel with 27 km circumference in a depth between 45 m and 170 m [7]. The machine accelerates protons (or heavy ions) in two separate tracks and brings them to a collision at four intersection points. A sketch of the complex can be seen in Figure 3.1.

The acceleration is done by 16 radio-frequency cavities, which are located along the tracks. They use electromagnetic fields to build a resonance, in which the charged particle feels an accelerating force. This also leads to the existence of bunches of protons and not a continuous beam [20]. Important components are more than 1200 dipole magnets with a field of 8.4 T to keep the protons inside the collider. Both systems are based on the superconducting technology, which means they need to be cooled to around 2 K at all times. Since the particles travel 11,000 times around the circle every second, the tubes need to be highly evacuated to prevent the protons from being scattered with the residual gas [7].

At each intersection point a special detector has been built to examine the collisions, which are also referred to as Bunch Crossings (BC). Four main particle detectors are installed in the LHC.

The Large Hadron Collider beauty (LHCb) is specialized in heavy quark production and precise CP violation measurements. The current Standard Model cannot describe the asymmetric amount of matter and antimatter in the universe [3]. Therefore it is important to study rare decays of beauty and charm hadrons [21].

A Large Ion Collider Experiment (ALICE) is optimized for the study of heavy ion collisions. Its aim is to create a Quark-Gluon plasma, a state of matter where the quarks and gluons are freed. It is predicted that these only occur under high temperature and density [22].

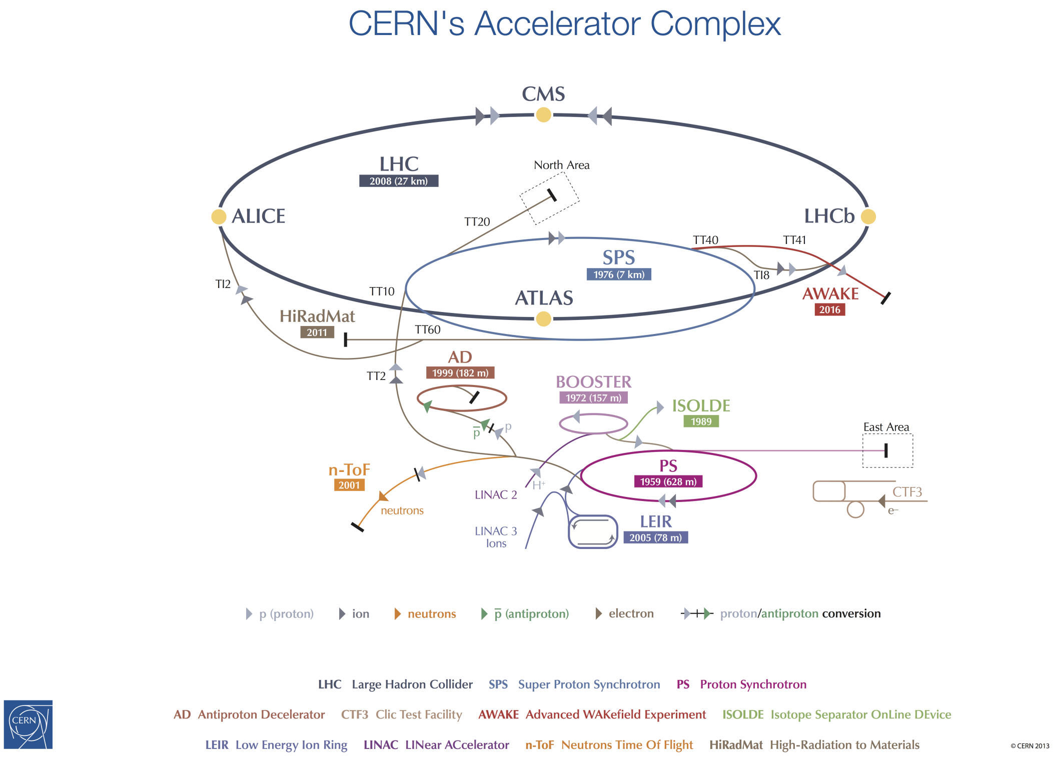


Figure 3.1.: A sketch of the different accelerators and detectors at CERN are shown. [19]

The Compact Muon Solenoid (CMS) experiment is one of the two general purpose detectors. It has the task to explore the new high energy range. The first benchmark to discover the Higgs particle has already been accomplished. An important aspect is the measurement of the momentum of muons, which led to the use of a very strong magnetic field [23].

This thesis is focusing on the ATLAS (standing for "A Toroidal LHC ApparatuS") experiment, described in the more detail in the following.

3.2. The ATLAS Detector

The ATLAS detector is with 25 m x 25 m x 44 m size the largest of the four experiments [24]. The discovery of the Higgs was a first benchmark similar to CMS. Now additional studies of the properties of this particle will be made. Furthermore the search for physics beyond the Standard Model, like additional Higgs bosons, is very important. Super Symmetry and extra dimensions are interesting models, which may be confirmed with additional data [25].

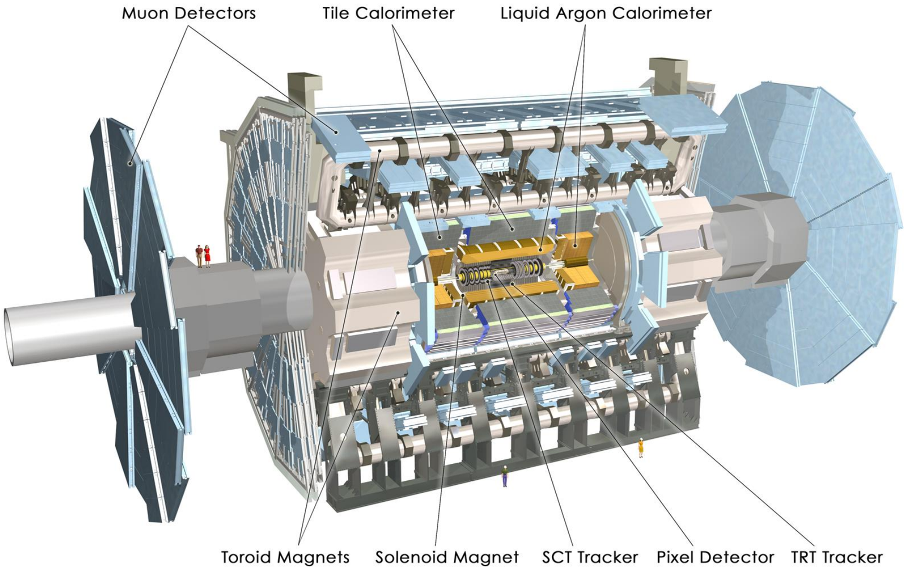


Figure 3.2.: The structure of the ATLAS detector with the different sections is displayed. [26]

The structure of ATLAS is shown in Figure 3.2. The high energy particles from the LHC enter the detector from the left and the right side and collide in the interaction region in the center. Closest to this point is the Inner Detector (ID), which consists of semiconductor pixel and silicon micro-strip (SCT) trackers and the Transition Radiation Tracker (TRT). The first two are precise tracking detectors with high granularity. The TRT also offers good electron identification capabilities [24]. The ID is surrounded by a superconducting solenoid, which provides a 2 T magnetic field. Charged particles are therefore forced on a bent track and its curvature gives information about the momentum of the particle.

The calorimeter system measures the energies of the particles. It consists of an electromagnetic and a hadronic section. The first one is the Liquid Argon (LAr) calorimeter. A detailed insight is given in the next section. The second one is the Tile calorimeter. It provides energy and position information of absorbed hadrons.

The calorimeter is surrounded by the muon spectrometer. As a particle with a long lifetime and little energy loss in matter, most muons pass the hadronic calorimeter. The muon chamber has the task to measure their momentum as precisely as possible. Strong toroidal magnets provide a field, which forces the muons on a bent curve, similar to the tracking in the ID.

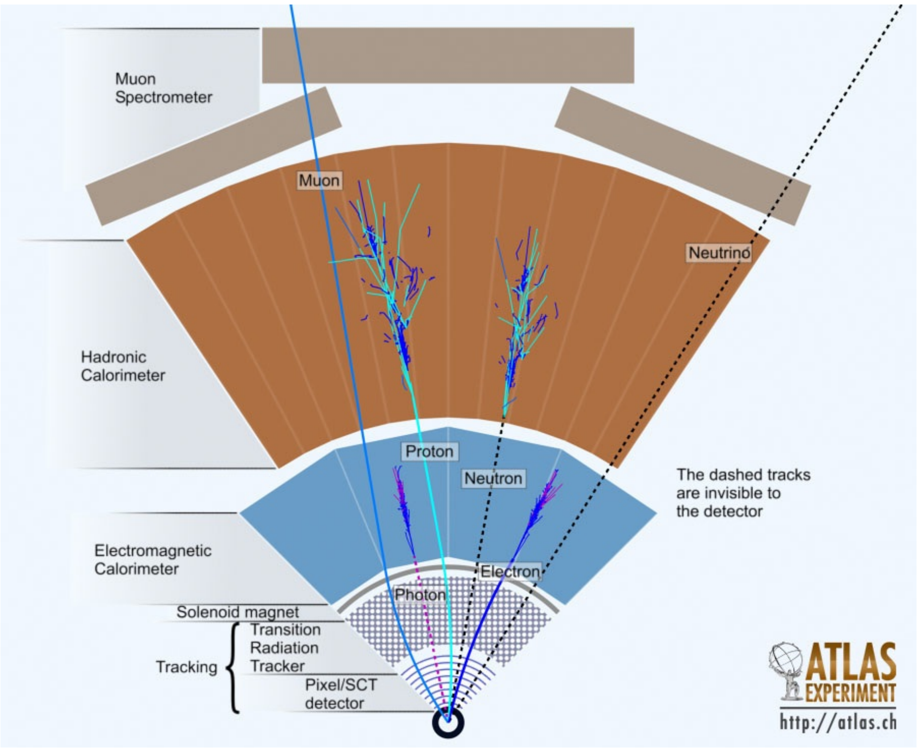


Figure 3.3.: The interaction of different particles with the detector sections is shown. Only charged particles are visible in the Tracker. Most particles develop a shower in the calorimeter, which will be covered in Section 3.3. [27]

Figure 3.3 summarizes the interactions between the particles and different parts of the detector.

ATLAS uses a coordinate system with the interaction point as the origin. The z-axis is parallel to the beam, the positive x-axis is pointing from the origin to the center of the LHC ring and the positive y-axis is pointing upwards. The azimuthal angle ϕ is measured around the beam axis. Instead of the polar angle θ, which is the angle from the beam axis, the pseudorapidity $\eta = -\ln(\tan(\frac{\theta}{2}))$ is used [24]. It has the advantage to be Lorentz invariant under boosts along the z-axis.

3.3. The Liquid Argon Calorimeter

The idea of a calorimeter is the complete absorption of the energy of the particle to transform it into a measurable quantity. The physical process inside an electromagnetic calorimeter is very well understood. Electrons and photons lose energy by interaction with the surrounding material via different processes. At high energies electrons mostly emit photons, which is called Bremsstrahlung, while photons produce electron-positron pairs. These alternating interactions develop a shower in the calorimeter until the energy is too low to produce additional particles. At these energies electrons scatter mainly with atoms and molecules, which causes ionization or thermal excitation. Photons lose energy through Compton scattering and the photoelectric effect [28].

An important parameter is the radiation length X_0. It depends on the characteristics of the detector material and represents an average distance the electrons needs to travel to reduce its energy to $1/e$ of its original value. This parameter can also be used for a photon beam. After a distance of about $\frac{9}{7}X_0$ the intensity is reduced to $1/e$ of the original value. The parameter is used as a scale for the length of an electromagnetic shower and important for the construction of calorimeters [29]. The length of the LAr calorimeter in ATLAS is in the order of $30X_0$ [24].

The LAr calorimeter is shown in Figure 3.4. It is divided into the Electromagnetic Barrel (EMB) and two Electromagnetic End-Cap (EMEC) components. The LAr Hadronic End-Cap (HEC) and the Forward Calorimeters (FCal) are also displayed in the figure, but they are in fact hadronic calorimeters. All of them are so called sampling calorimeter, consisting of alternating layers of an absorber and active medium. The first is a dense material to absorb the energy of the particle, which in the case of the EMB is lead. Liquid argon is used as an active medium, which provides the detectable signal for the energy measurement [24]. That is achieved by installing electrodes at the end of the LAr and applying a high voltage, which creates an electromagnetic field. The passing particles ionize the Argon and the generated electrons and ions drift to the electrodes due to the field. This results in a triangular shaped current pulse.

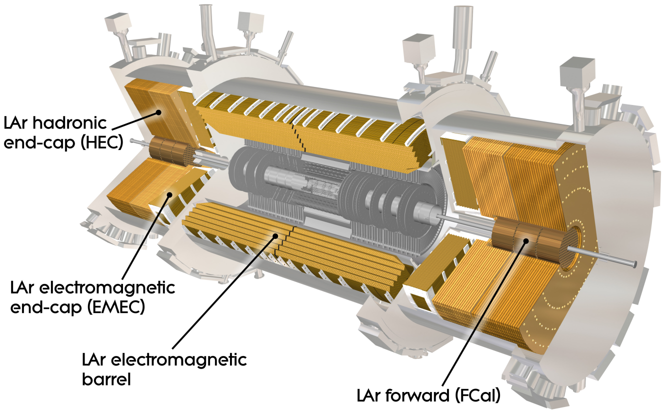

LAr hadronic end-cap (HEC)
LAr electromagnetic end-cap (EMEC)
LAr electromagnetic barrel
LAr forward (FCal)

Figure 3.4.: The structure of the LAr Calorimeter and its components are displayed. It is located outside the Inner Detector of ATLAS. [30]

To achieve full coverage in ϕ an accordion geometry is used, which is visible in Figure 3.5. In addition the image shows the segmentation of the calorimeter in different layers. The first in the front is finer segmented in η. This offers the discrimination between a single $e^{\pm}/\gamma$ and a π^0, which decays into two photons with a small opening angle. The largest amount of energy is deposited in the second layer and the last one collects the tail of the electromagnetic shower. The barrel has an additional single layer of liquid argon, the Presampler, which is passed first by the particles. The figure also gives information about the size of each elementary cell. The signal is extracted from each cell separately, which results in a total of 182,468 readout channels [24].

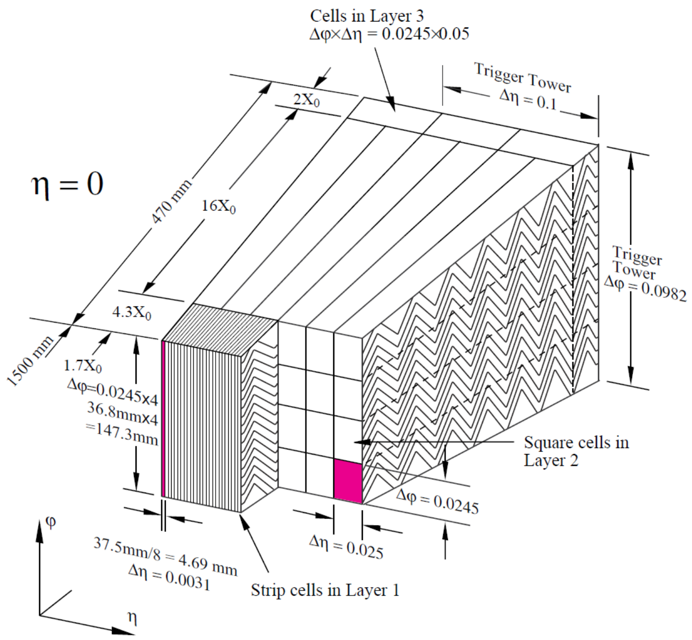


Figure 3.5.: The accordion geometry of the LAr Calorimeter is shown. The detector consists of the finely segmented front layer, the widest middle layer and the back layer. A single elementary cell is highlighted. The figure uses the ATLAS coordinate system (refer to Section 3.2) [24]

3.4. The Trigger System

There are 40 million collisions per second at the interaction point and each event produces about several Mb of information in the ATLAS detector. It is not possible to store all of these for later analysis. In addition most of these events are well-known inelastic pp scattering events. Interesting physics events typically produce high p_T signature and are usually rare. These need to be selected for storage. The solution is a trigger system, which decides about the importance of an event. It is divided in three levels: Level-1 (L1), Level-2 (L2) and event filter [24].

The L1 trigger gets information from the muon spectrometer and the calorimeter system. It looks for high p_T muons, electrons/photons, jets (cones of particles including mostly hadrons), τ-leptons decaying into hadrons and events with large missing transverse energy ($E_\mathrm{T}^\mathrm{miss}$). The decision must be made in 2.5μs and the maximum acceptance rate is 75 kHz. This means that 75,000 events per second are sent to the L2 trigger. The calorimeter signatures alone produces a trigger rate of 20 kHz.

The L1 trigger also selects Region of Interest (ROI). These are the regions in the detector, which have possible trigger objects. The L2 trigger uses the information of position and energy to further select events. The acceptance rate is 3.5 kHz with a processing time of 40 ms. The event filter reduces the rate to 200 Hz by using offline analysis procedures. The average processing time is about four seconds. The last two levels use the full granularity of the detector.

4. The Upgrade Plans

4.1. The Timeline for the LHC

The accelerator and the detectors have operated very well since the beginning of the LHC collisions in 2009. The discovery of a SM-like Higgs boson was announced in July 2012 and many new limits on new physics have been set. The maximum collision energy was 8 TeV and a luminosity of $7.7 \times 10^{33}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ has been reached [31] during Run 1. To further analyze the new particle and continue the search for physics beyond the SM both values have to be increased. To accomplish this, CERN has developed an upgrade plan for the upcoming years [32].

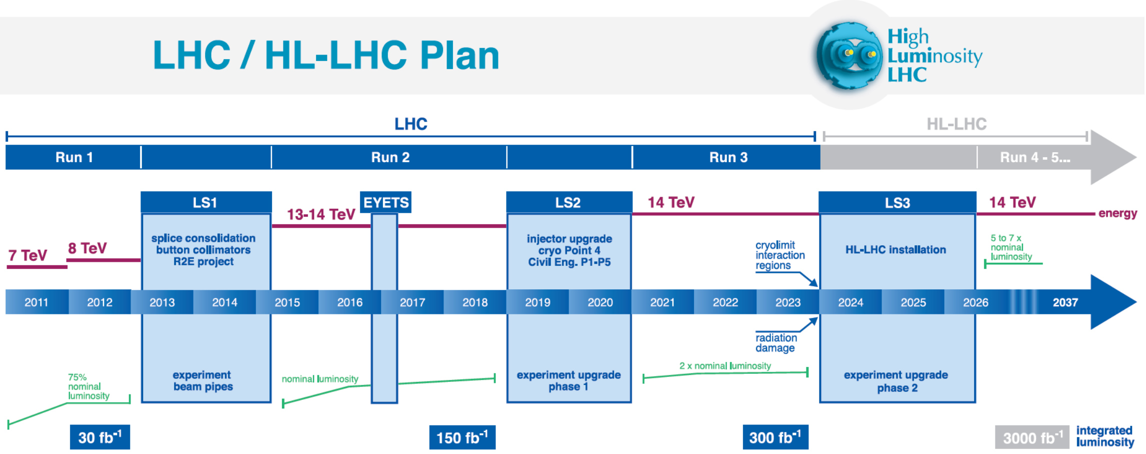


Figure 4.1.: The project plan for LHC is displayed. The data collection takes place during the runs and the long shutdowns are used to improve the accelerator and detectors. [32]

As seen in Figure 4.1 the upgrade will be done in three phases. The improvements are made during the Long Shutdowns between the runs, where new data is collected. After the very successful Run 1 the Long Shutdown 1 (LS1) lasted from 2013 to 2015 and the Phase-0 upgrade was implemented during this time. The aim was to achieve design beam energy and luminosity by for example repairing the magnet splices. The maintenance is necessary to reduce the risk of a recurring of an accident as in September 2008. During powering tests one of these splices faulted and invoked mechanical damage and delayed the start of the experiment by six months [33].

Because of these changes the LHC is now able to achieve 13 TeV collision energy and a luminosity of $1 \times 10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ during Run 2 [34]. This will give the opportunity to explore a complete new energy range and collect more data for physics analyses. This procedure will be finished in 2018 for the Long Shutdown 2 (LS2). This Phase-1 upgrade includes changes like a new injector complex, which will double the luminosity compared to the value from Run 2.

After three years of data taking in Run 3 the Long Shutdown 3 (LS3) and its Phase-2 upgrade will give way for the high luminosity LHC (HL-LHC). A Mayor upgrade of multiple components will make a value of five times the design luminosity possible [34]. These changes will ensure to continue the operation of this successful experiment for many years, beyond 2035.

4.2. Changes at the ATLAS Detector

During the long shutdowns the individual detectors need to be upgraded for various reasons. For example, during LS1 a new layer of the pixel detector was installed [35]. In the future, problems like aging and the radiation damage in the ID will have to be addressed as well, which will mainly be done in the Phase-2 upgrade during the LS3 [36]. This thesis will concentrate on the Phase-1 upgrade of the LAr calorimeter [34], which will be installed during the LS2.

The motivation for this upgrade derivates from the enhanced luminosity of the LHC. The increase of the average number of interactions per BC will go from 20 to 60 compared to Run 1. Therefore more events with high p_T signatures will be recorded and the acceptance rate of the already discussed trigger system would go up. Unfortunately the L2 trigger cannot process a bandwidth higher than 100 kHz [34]. There are two options to keep the trigger acceptance rate from L1 at this level. The first would be to increase the p_T lepton threshold, which is at the moment at 25 GeV. A lot of interesting physics starts at 40 – 50 GeV, because this is about half the mass of the Z/W$^{\pm}$ bosons and therefore the electroweak scale. An increase of the threshold is therefore tried to be avoided.

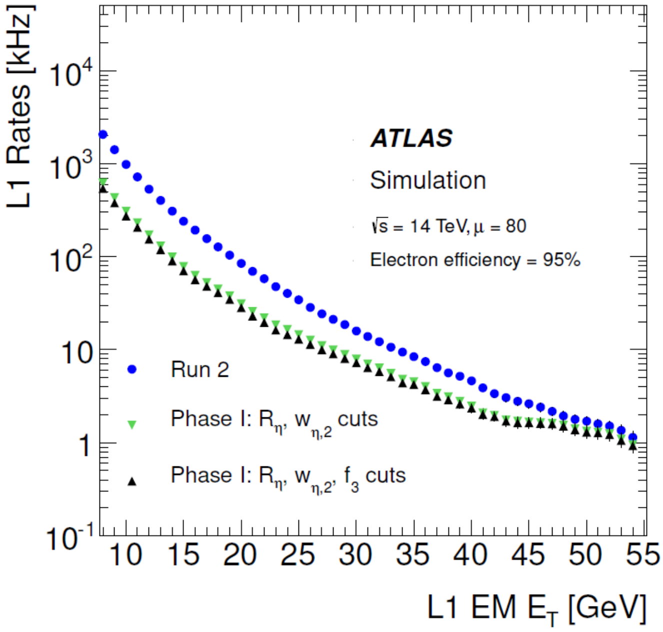


Figure 4.2.: The acceptance rate of the L1 trigger is simulated in respect to the energy threshold. Naturally the bandwidth can be decreases by increasing the threshold. Another possibility is the introduction of shower parameters (e.g. R_η, $\omega_{\eta,2}$), which also lowers the trigger rate. [34]

The better solution is to introduce new shower recognition algorithms for electrons and photons and an improved background rejection. The results are shower parameters, which are calculated by grouping different cells. A more precise insight is given in [34]. The shower profile of physical interesting electrons is for example more collimated than background jets, which leads to other shower parameter values. The result of a simulation regarding the L1 trigger upgrade is presented in Figure 4.2. To apply these recognition algorithms more efficient, a higher granularity of the calorimeter is needed.

In Figure 3.5 of the calorimeter chapter the elementary cells of the different layers are shown. At the moment the L1 sums the cells of a $\Delta\eta \times \Delta\phi = 0.1 \times 0.1$ area of all layers to a so called Trigger Tower (TT). The new concept are Super Cells (SC). One TT consists of 60 elementary cells in the EMB section and will be divided in 10 SC. In the Presampler 4 elementary cells are combined into one SC. The Back-layer part also forms one SC, while the other two layers both have four SCs. A summary of the old and new system, including the number of cells, is shown in Figure 4.3. This improvement leads to the necessity to exchange parts of the current electronic system [34].

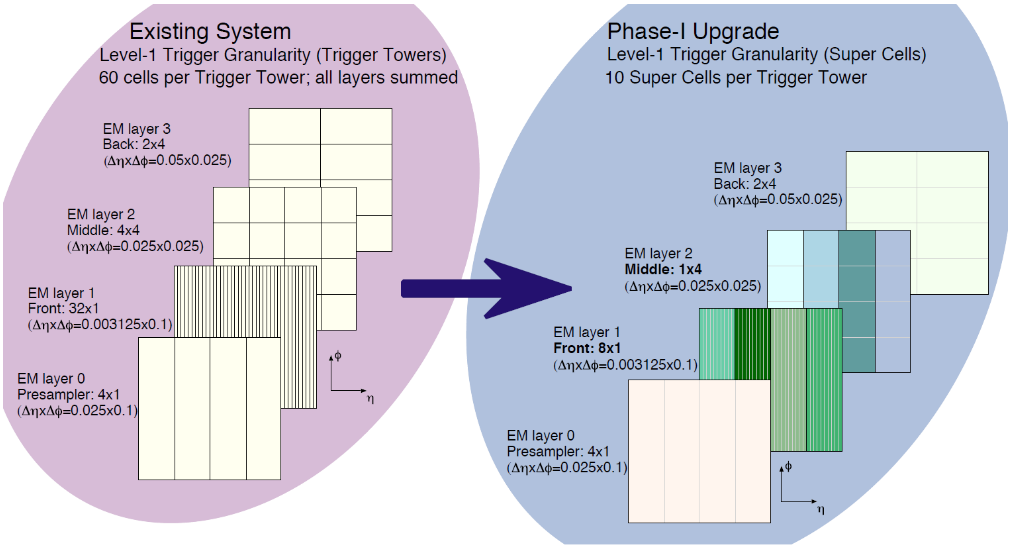


Figure 4.3.: The elementary cells in a $\Delta\eta \times \Delta\phi = 0.1 \times 0.1$ area are shown in all layers. On the left side all of them get summed to one TT. On the right side 10 SC are formed with the same cells. [34]

4.3. Readout Architecture

In Figure 4.4 the Phase-1 readout electronics are shown, including the new parts marked with a red rectangle. The left half is the front end electronics, which is installed directly on the detector and has to be radiation resistant. The location of the right half is in a separate room near ATLAS and called "back end electronics". The analog signal arrives from the detector in the upper left corner and passes a preamplifier. Then the shaper reforms the pulse. The signals of the regular readout get stored in switched capacitor arrays (SCA), where they wait for the result of the L1 trigger.

Figure 4.5 shows the pulse shapes of a signal passing the front end electronics. A particle depositing energy in the detector results in a triangular form. The tail results from the drift time of the electrons. Collisions happen every 25 ns and it is possible that additional

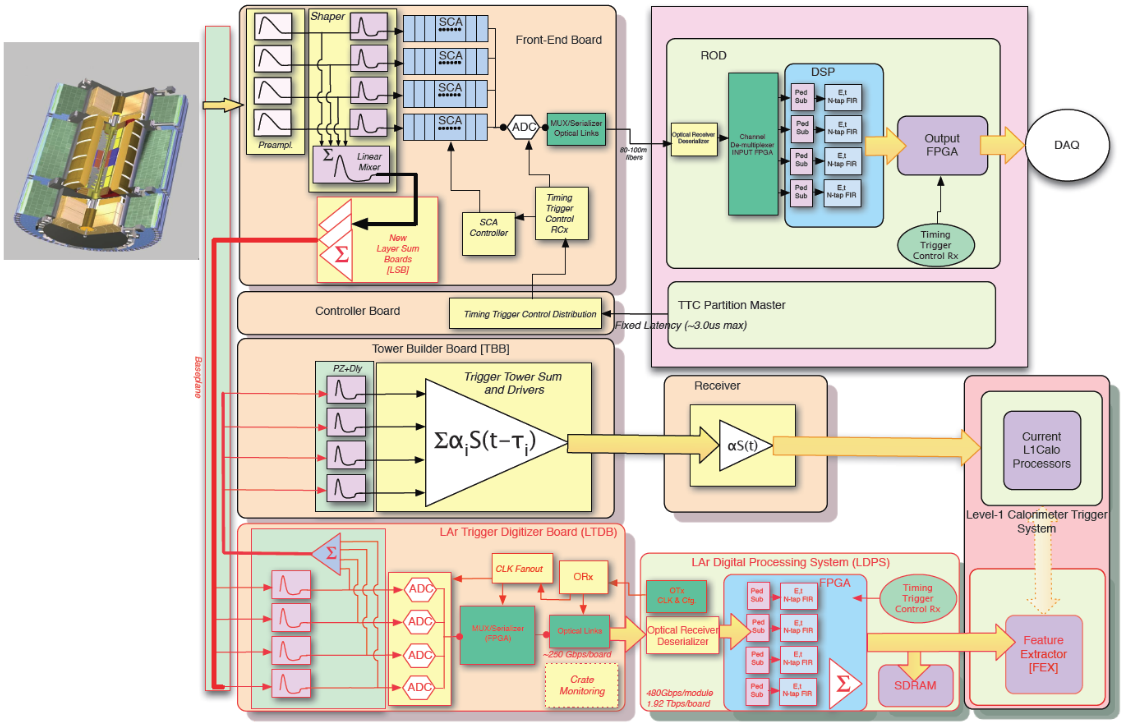

Shaper
Preampl.
SCA
SCA
SCA
SCA
Front-End Board
ADC
MUX/Serializer Optical Links
80-100m fibers
Σ Linear Mixer
New Layer Sum Boards [LSB]
SCA Controller
Timing Trigger Control RCx
Controller Board
Timing Trigger Control Distribution
Fixed Latency (~3.0us max)
ROD
DSP
Optical Receiver Deserializer
Channel De-multiplexer INPUT FPGA
Ped Sub
E,t N-tap FIR
Output FPGA
DAQ
Timing Trigger Control Rx
TTC Partition Master
Baseplane
Tower Builder Board [TBB]
PZ+Dly
Trigger Tower Sum and Drivers
$\Sigma\alpha_i S(t-\tau_i)$
Receiver
$\alpha S(t)$
Current L1Calo Processors
Level-1 Calorimeter Trigger System
LAr Trigger Digitizer Board (LTDB)
CLK Fanout
ORx
ADC
MUX/Serializer (FPGA)
Optical Links
~250 Gbps/board
Crate Monitoring
LAr Digital Processing System (LDPS)
OTx CLK & Cfg.
FPGA
Optical Receiver Deserializer
Timing Trigger Control Rx
SDRAM
480Gbps/module
1.92 Tbps/board
Feature Extractor [FEX]

Figure 4.4.: The readout system of the LAr calorimeter is displayed. New components have a red rectangular. [34]

particles pass through the same LAr cell and add to the existing pulse. This is called "pile up" and makes it difficult to identify the energy of the single particles. Therefore a series of resistors and capacitors in the shaper are used to convert the signal into a bipolar form. First a CR differential step removes the long tail of the detector response. Two RC integration steps limit the bandwidth to reduce the noise [24]. The amplitude of the pulse is then measured every 25 ns by the ADC (Analog Digital Converter). The output is an 12 bit digital signal (ADC counts) for every point.

In the current electronics the analog signals of the elementary cells are collected in the Layer Sum Boards (LSB) and directly sent to the Tower Builder Board (TBB). Here the different layers are further summed to form the TTs. The still analog signal is transfered to a receiver from the back end electronics and from there it is forwarded digitally to the L1 trigger.

Since the improved trigger works with a higher granularity, the LSBs will be replaced for the new readout system. A reconstructed baseline transports the signal to the new

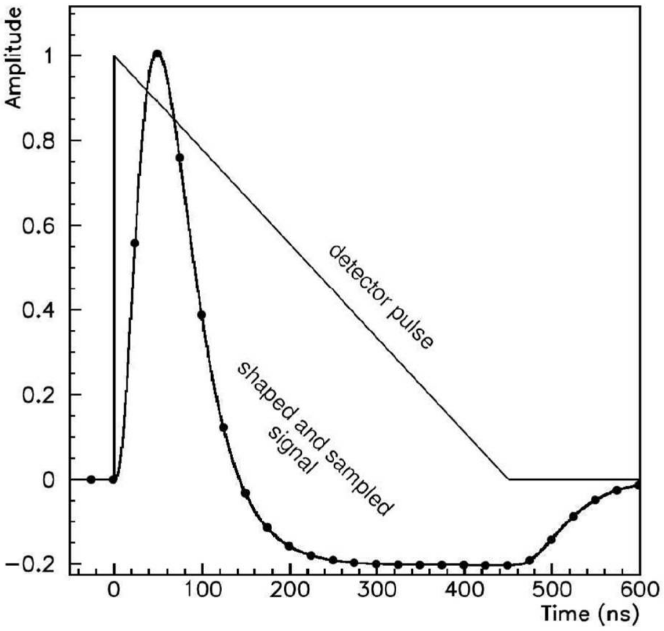


Figure 4.5.: The different stages of the pulse shape is displayed. A triangular signal arrives from the detector. It gets shaped into a bipolar form measured every 25 ns.[34]

LAr Trigger Digitizer Board (LTDB). Here it gets split first, summed and sent to the Tower Builder Board. That way the original trigger path is conserved and can be used to compare the functionality of the new electronics.

To avoid the additional noise, which is generated by sending an analog signal through a 50 m cable, an ADC will be located in the LTDB. The signal is then going to be serialized and transfered via optical links to the back end. The setup of the LAr Digital Processing System (LDPS) will be explained in the next chapter. Its purpose is to convert the digital signal into an energy value and send it to the Feature Extractor (FEX), a new part of the L1 Trigger. The FEX is responsible for the detection of electrons, photons, tau-leptons, jets and energy sum signatures.

4.4. The LAr Digital Processing System

The LDPS electronic board, which will be integrated during Phase-1, has a main ATCA (Advanced Telecommunications Computing Architecture) board and is equipped with a few so called "LAr Trigger prOcessing MEzzanine" (LATOME). There will be 31 LDPS boards for 34,000 SCs. Its task is the calculation of the pulse energy [34]. Its schematic firmware can be seen in Figure 4.6. It represents the blue part of the LDPS in Figure 4.4. The data arrives on the left from the LDTB after the optical receiver and deserializer. The alignment of the signals takes place in the input stage. The remapping module is necessary to put the data in the correct order to simplify the calculation of the sums for the trigger tower. There the unit is a TT and the SC have to be grouped according to detector geometry [37].

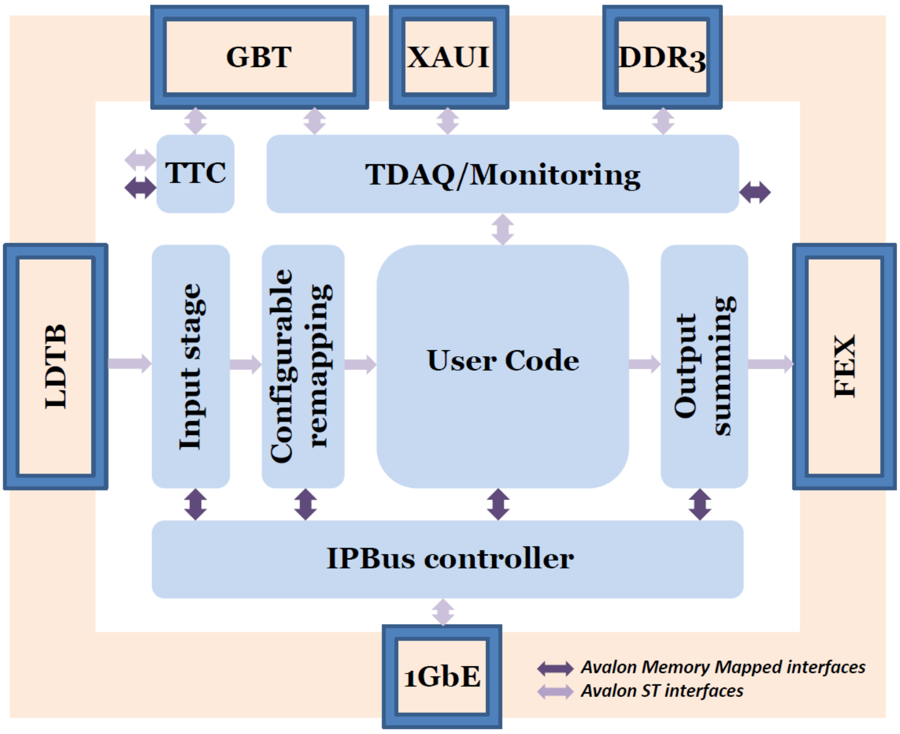


Figure 4.6.: The Layout of the LDPS-firmware with the data flow from left to right is shown. [37]

The next stage is the User Code (UC). Digital filtering algorithms are used to convert the ADC counts to transverse energy [38]. They calculate it by linear combination of multiple samples and special filter coefficients. The UC may also include a saturation detection system. When the deposited energy of a particle is too high, the electronics cannot amplify the signal any further and the linearity of the ADC-energy-ratio is no longer given. This issue will be covered in Chapter 8.5 in more detail.

The energy information continues to the output summing. The signals are again grouped and summed over specific areas. They are then transported to the FEX of the trigger system.

Several additional modules can be seen in Figure 4.6. The Trigger Timing and Control (TTC) receives the LHC clock (alternating signal between zero and one at 40 MHz), BCID (identification number of a BC) and L1 Accept (indicates if a signal is triggered). The Trigger and Data AcQuisition (TDAQ)/Monitoring module stores all data in a circular buffer and the accepted signals are then forwarded to check trigger and filter performances [37].

Finally the IPBus [39] represents the slow control. It connects the other modules to the outer world via an Ethernet connection and enables the user to change the configuration parameters of the system. The two main connections between the different modules are also displayed in Figure 4.6. These so called Avalon Interfaces [40] are discussed in Chapter 5.4.

5. Test Environment

During the first part of the master thesis, parts of the monitoring and IPBus modules were developed and tested. This chapter explains the construction of the test environment, which had to be installed to probe the functionality of the software. In addition the used hardware is shown and different occurring problems and solutions are explained.

5.1. FPGA

The energy identification of the digitized pulse will be done inside the firmware of a Field Programmable Gate Array (FPGA). This Integrated Circuit (IC) mainly consists of input and output ports, logic blocks and interconnections between them. The outstanding property is the possibility to reconfigure the device. This is done by applying a voltage at the interconnections. A simplified representation is shown in Figure 5.1. This reprogramming ability has many advantages. New functions can be implemented fast and faults in the design are easily eliminated. Disadvantages on the other hand are a high power consumption and low radiation resistance.

An alternative are Application Specific Integrated Circuits (ASIC). They have a certain functionality after manufacturing and are not reprogrammable. These ICs are for example used in the front end electronics.

Additional important components of FPGAs are the look-up tables, which realize the logical function with inputs and outputs, and the registers as memory elements. These are controlled via a signal, which constantly switches between zero and one: the clock. This results in a fixed latency and discriminate the FPGAs from conventional processors.

The FPGAs are configured with the aid of special Hardware Description Languages (HDL). An introduction will be given in Chapter 5.2. The procedure is a bit different to common programming languages, since the tasks are not done successively, but multiple processes are running in parallel. This is a main advantage compared to conventional

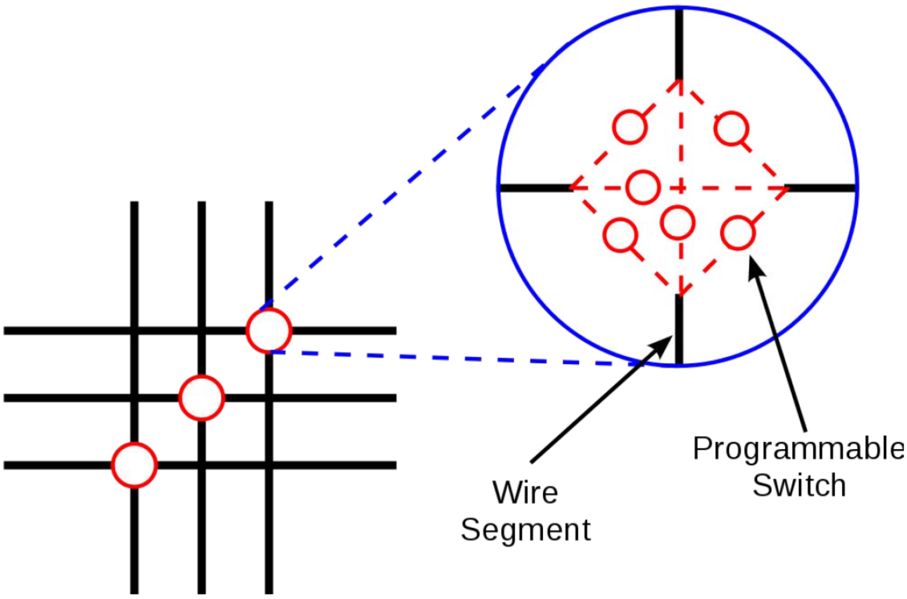


Figure 5.1.: The sketch of the programmable switches inside a FPGA are displayed. The wires are connected to logic blocks, inputs, outputs or registers.

processors and is often used in cryptography or image processing. In the LAr calorimeter many channels have to be treated quickly at the same time, making FPGAs the perfect candidate for these processes.

During the master thesis a Cyclone V FPGA from Altera was used [41].

5.2. Programming of the FPGA

The programming of a FPGA has several steps, which are executed inside a design tool. Altera provides the Quartus II software for this purpose. First the functionality of the final product is described via an HDL. During the master project VHDL (Very high speed integrated circuit Hardware Description Language) was used. The code is then tested for hardware compatibility and compiled. The implementation is carried out in three parts ("Translate", "Map" and "Route"). During this process, the circuit logic is transfered into the FPGA hardware and the individual blocks are connected. Lastly a "Bit"-file is generated and can be programmed on the FPGA. Now the interconnections are charged and the system can be used for the intended purpose.

Similar to other languages, the VHDL libraries are included first. Then generics, in- and output ports are defined. Both can be seen in Code example 5.1. Generics can be compared to global variables from other languages. "std_logic" is a data type, which can for example hold the values "0" and "1". There are also other possibilities, which are not important for the thesis. In addition a "std_logic_vector" is an array of this data type. Worth mentioning is also the common "integer" type. Comments are initiated with "–".

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity TOP is --"TOP" is the name of the module
  generic (
    a: integer := 1
  );
  port (
    in1, in2  : in  std_logic;
    out1      : out std_logic
  );
end TOP;
```

Code example 5.1: This code shows how the library is included. The generic variable and ports are initiated with the data type.

The actual function is located inside the architecture section. Code example 5.2 shows the definition of internal signals and the implementation of simple logic gates.

```vhdl
architecture Behavioural of TOP is
  signal x : std_logic := ’0’; --internal signal declaration
begin
  x     <= in1 and in2 ; --simple signal assignments
  out1  <= not x;
end Behavioural;
```

Code example 5.2: The architecture section contains the functional elements. In this case the inputs are connected via an AND-gate. The resulting signal "x" is then inverted and linked to the output.

Code example 5.3 shows some additional possibilities, which can be included in the architecture section. Components, conditional assignments and procedures are displayed. Further information can be taken from the comments.

```
--conditional assignment:
out1 <= '1' when in1 = '1' else
        '1' when in2 = '1' else
        '0';

--initiation of components (to reuse available solutions)
--in this case: entity TOP from before is used in another code
NAND: entity work.TOP --"NAND" is the name of the component
--"work" refers to the current library
Generic map(
  a => 10
) --each generic and port is assigned with a signal
Port map(
  in1   => k,
  in2   => l,
  out1  => m
) ;

--inside of processes the order of assignments matter
--the syntax is different, but closer to conventional programming
process(in1, in2)
begin
  --two methods of conditional assignments:
  if in1 = '1' then ...
  else ...
  end if;

  case in1 is
    when '1'    => ...
    when others => ...
  end case;

  --common loop deklaration:
  for i in 0 to 3 loop
    ...
  end loop;
end process;
```

Code example 5.3: This example shows conditional assignment, components and processes. They can be included in the architecture section of a program.

5.3. SoCKit Development Kit

The SocKit Development Kit is a design platform [42], which is built around the Altera System-on-Chip (SoC) FPGA. DDR3 memory, Ethernet networking and several peripherals are also included and were used during the project. An overview of DDR3 will be given in Section 6.2. The Ethernet connection is explained in Section 5.5. In addition there is an ARM-based Hard Processor System (HPS) inside the FPGA. In Figure 5.2 the layout of the board is shown. One important connection is the Ethernet port, which is directly linked to the HPS. The FPGA is located on the right side of the board and can be programmed via the USB Blaster II port. The DDR3 memory is located close to the Cyclone V and the peripherals like buttons, switches and LEDs are on the bottom of the platform.

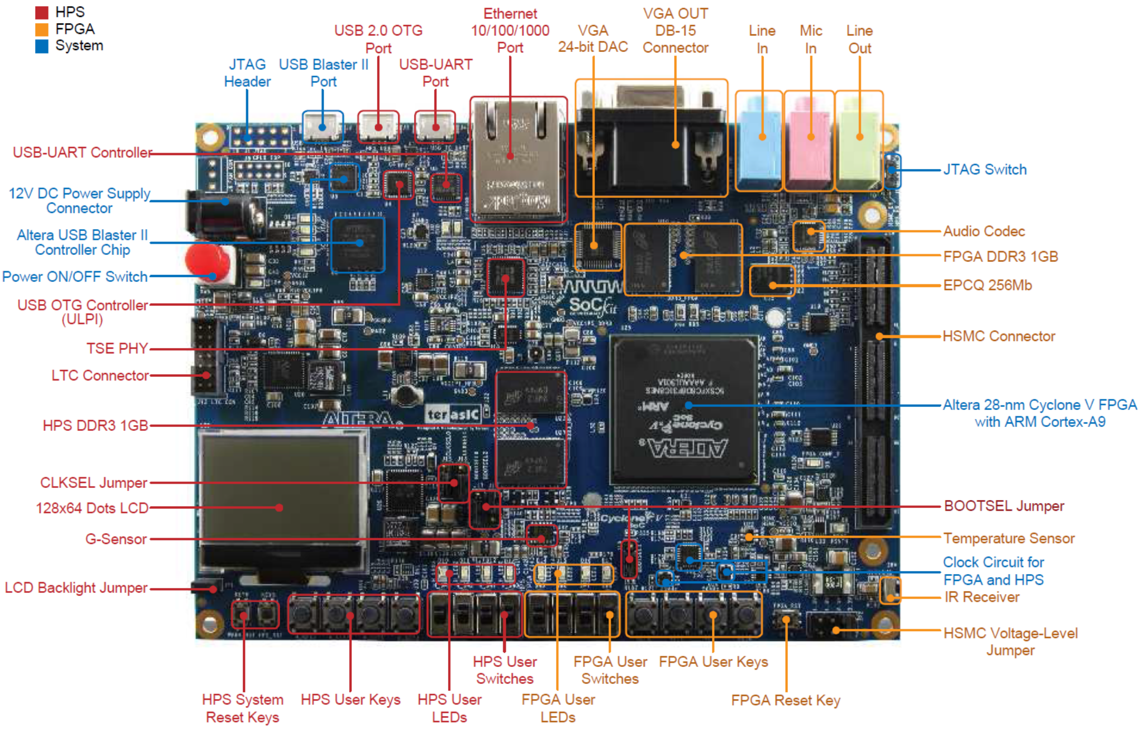


Figure 5.2.: The layout of the SoCKit board is shown. The color legend in the upper left corner indicates the connection between the components. The Ethernet port on the top is linked to the HPS. The USB Blaster II port in the upper left part is used to program the FPGA. [43]

The first task was to transport data from the computer to the FPGA. The Ethernet connection, which is necessary to check the functionality of the IPBus module, was not accessible. The signals would need to be transported through the HPS. Unfortunately this was not possible during the master project. As a solution, an additional extension card for the board was purchased and the IPBus could be tested.

Finally the USB Blaster II port was used to transport the data from and to the FPGA. This connection simulated the data stream, which would arrive from the LAr front end. Therefore Altera provides the "System Console" software. With the aid of tcl-scripts it

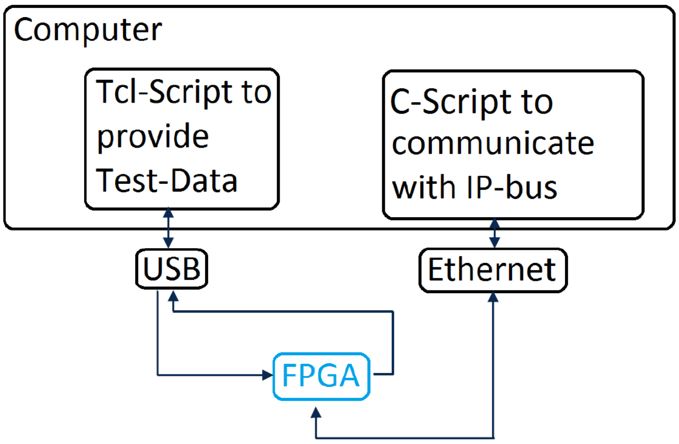

Computer
Tcl-Script to provide Test-Data
C-Script to communicate with IP-bus
USB
Ethernet
FPGA

Figure 5.3.: The connections between Computer and FPGA are shown. The data stream is simulated with a tcl-script and routed into the FPGA via the USB port. The IPBus uses the Ethernet link.

is possible to communicate with the FPGA. Data could be read from a file or generated in the script, sent to the FPGA and afterwards the processed data is sent back to the computer and written into a file. In Figure 5.3 the basic setup can be seen.

5.4. Avalon Interfaces

As mentioned in Chapter 4.4 there are two important standard connections between the modules. The Avalon Streaming (AST) interface [40] is used for the signal transport from the input stage to the output summing. The data flows between the modules from a "source" to a "sink". Two FIFOs (First In First Out) have been used to simulate this behavior. These are small storages, where the first arriving packages are the first to be sent out again. One FIFO handles the incoming data from the USB Blaster and represents a "source". The second FIFO simulates a "sink" interface and sends the data back to the computer.

The slow control and the DDR3 connection are based on the Avalon Memory Mapped (AMM) interface. Both are an important part of the project and therefore this interface is explained more precisely. It works via a master-slave-principle. The IPBus controller is the master and every other module in the firmware is a slave. The DDR3 can be accessed by a memory controller, which has a slave interface. The master sends address-based read or write commands. The slave modules can only answer the commands and not act on their own. A typical transfer can be seen in Figure 5.4.

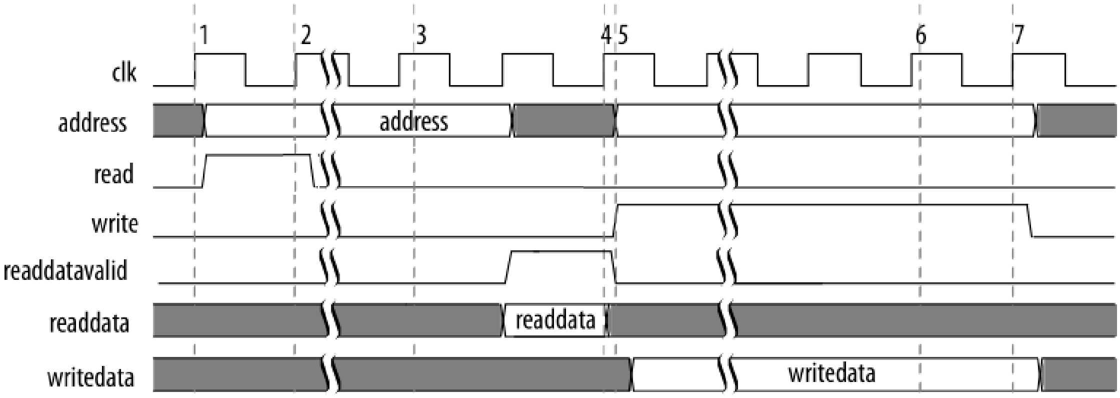


Figure 5.4.: A read and write transfer of the AMM interface is displayed. The signals are listed on the left and the time axis points to the right. The left half of the figure shows a reading process and the right part a writing command.

The names of the signals can be seen on the left side of the figure. The time axis points to the right. The clock (clk) is located on the first line. It switches between one and zero with a fixed frequency. The left part of the figure displays a read command. The master asserts the "read" signal and provides an address. Then it waits until the slave answers by asserting the "readdatavalid" signal and sending the "readdata". The write command on the right half of the figure is similar. The master asserts the "write" signal and provides the address and the "writedata". A response from the slave is possible but not obligatory.

A summary of the test environment can be seen in Figure 5.5. It focuses on the inside of the FPGA from Figure 5.3. The Ethernet can be accessed via the Media Access Control (MAC). AST is shown in red and the AMM-interfaces are displayed in green. The different components can be implemented by "Qsys", an integration tool of the Quartus II software [44]. With this setup it is now possible to test any module of the LDPS-firmware on the SoCKit Development Kit. The FIFOs simulate the data stream and a connection to DDR3 and Ethernet are also available.

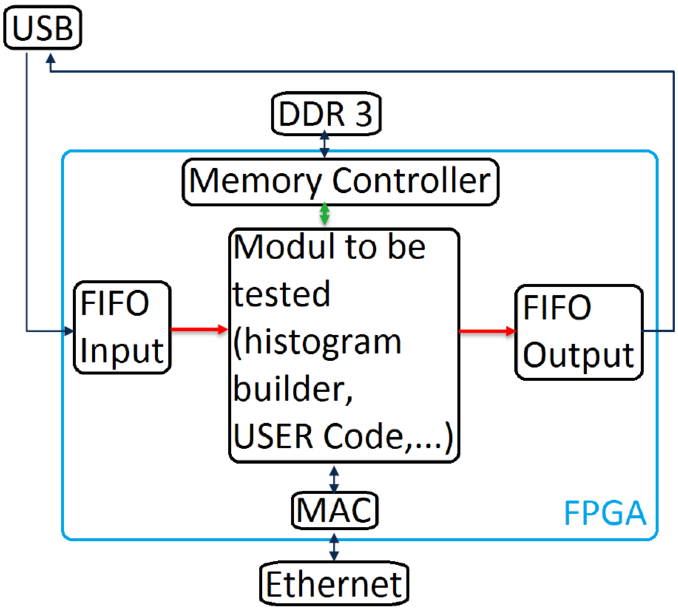


Figure 5.5.: A sketch of the test environment inside the FPGA is shown. The USB port is directly connected to the FIFOs, which provide the AST interface. The DDR3 and Ethernet are also included. Between these connections the module, which should be tested, can be inserted.

5.5. Ethernet connection

This section gives a short overview of the Ethernet connection [45] between FPGA and the computer.

The data transport from one medium to another can be described by envelopes. The sender divides the data into several packages and each is put into multiple successive envelopes. The receiver unpacks these and obtains the original data. Every envelope resembles a protocol and is implemented by placing a "header" in front of the package.

The User Datagram Protocol (UDP) is used as the first "envelope". The header is relatively short and consists only of the source and destination port, the length of the packet and a checksum for errors. UDP is an easy protocol, which goes without further data protection methods. Another important method would be the Transmission Control Protocol (TCP). It is a bit slower, because a connection between the ports is created, which means that every transaction is acknowledged by the receiver.

The next protocol is the Internet Protocol (IP). It makes the transaction outside of network boundaries possible and is known for its important role in the world wide web. The protocol works with special IP addresses. It works without a connection, but still provides some error detection.

The final "envelope" is the Ethernet protocol, which not only consists of a header, but also adds a "trailer" after the data to mark the end of the package. The header contains amongst others the MAC address of the sender and receiver.

The IPBus will introduce an additional protocol, which is covered in Chapter 7.

6. Histogram Builder

After the test environment was completed, the first programming task could be approached. The histogram builder is a tool to inspect the data and would be located inside the Monitoring module. This chapter describes the development of a histogram builder. It is checked with the test environment, which is explained in the last chapter.

6.1. Motivation

All signals are temporary stored in circular buffers or the SCA, but are erased when they are not accepted from the L1 trigger. Considering the huge amount of data ($1\,\mathrm{PB\,s^{-1}}$) this is completely reasonable. It would still be very interesting to have the possibility to analyze untriggered data. Other applications could be noise monitoring and pulse shape studies. The output of the histogram builder sub-module can be used for the calibration of pedestal and filtering coefficients [46] (refer to Section 8.3).

As the name suggests the data is filled into a histogram, before entering the circular buffer. Different parameters or data from SCs can be stored. The conditions are specified during the initialization stage. The histograms are stored inside the DDR3 memory block. The readout would be accessible from the slow control [46].

6.2. DDR3 memory

The full term of the memory is DDR3 SDRAM. They consists of many cells and each stores the values "0" or "1". RAM stands for Random Access Memory, which means, that the time it takes to access the memory is not affected by the location of the data. The storage is lost, when the power is turned down. Dynamic RAM (DRAM) uses less transistors than the Static RAM (SRAM) and is therefore cheaper and smaller. The DRAM uses a capacitor in each cell. They lose charge over time and need a refresh cycle, which results in a slower access time compared to the SRAM [47].

Synchronous DRAM (SDRAM) uses a memory bus clock to synchronize the signals to the memory. This simplifies the logic and reduces memory access latency. DDR3 is the third generation of "Double the Data Rate" (DDR). The increase of speed is accomplished by transferring the data both at the rising and falling edge of the clock. Multiple cells are grouped and each group gets its own address. To simplify the access there is an external hard memory controller on the FPGA chip. It has a slave AMM interface and the FPGA resembles the master.

The idea is now to use each address of the DDR3 as a bin number of the histogram and the data of the address is the content of the bin.

6.3. Data handling process

The handling of the histograms is done with the aid of a state machine. The code consists of multiple states and each has a certain functionality. If particular conditions are fulfilled the program changes the state and therefore has a different task.

After initialization the program is in the "idle" state. When the data arrives it switches to the "read" state, since the content of the bin needs to be known first and then increased by 1 in the "write" state. For example an eight bit data word arrives, resembling the ADC counts, and the aim is to build a histogram of these counts. The incoming data should be divided by the bin size and the new value is the bin number. As mentioned in Section 6.2 this is equal to the address of the DDR3.

Since division is complicated and takes time inside an FPGA, it is better to fix the bin width to a number of the powers of two. Then a simple bit shift to the right fulfills this task. A right shift by one means for instant that the least significant bit is deleted, because each bit is moved one place to the right. This represents a division by 2 and ignores the remainder. A bit shift by two implies a division by four and so on. An example with a bin width of four is presented in Table 6.1.

After the reading and writing task of this address is done, the program switches again to the "idle" state and waits for the next data. The whole procedure can be seen in Figure 6.1. This image was generated by a program called "Signal Tab" [48]. It provides the

Arriving data	(Decimal)	Shifted data	(Decimal)	Address
0101101	45	01011	11	01011

Table 6.1.: The process of the arriving data is shown. It is shifted by two bits, which represents a division by four.

possibility to inspect the state of multiple signals at several times. The figure is therefore not simulated, but in fact shows the time elapsed of the actual signals inside the FPGA.

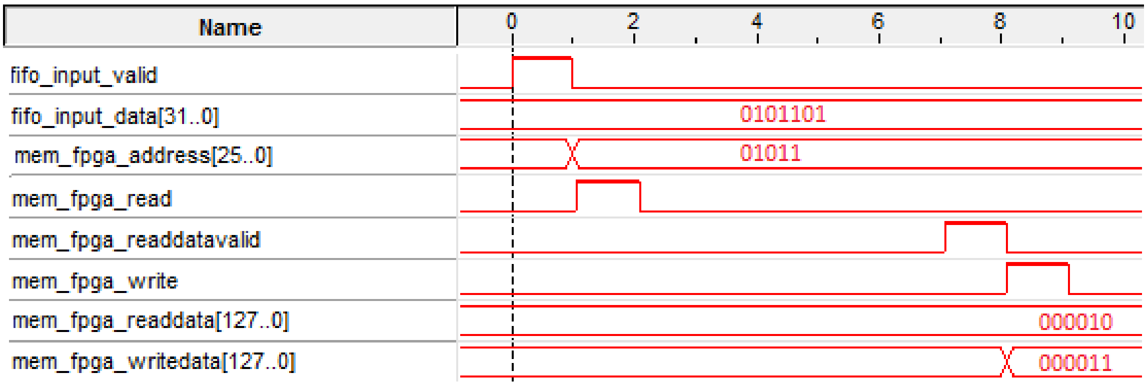


Figure 6.1.: A snapshot of the signals inside the FPGA during histogram building is shown. After the data arrives, it is shifted by 2 bits. A read command is issued and the answer is awaited. The "readdata" is then increased by one and written on the same DDR3 address.

The number on the top of the image are the clock cycles. When the FIFO sends data ("fifo_input_data") it asserts the "fifo_input_valid" signal. The "mem_fpga_address" is the bit shifted value of the data as discussed. The program is now in the "read" state and asserts the "mem_fpga_read" signal. After several cycles of waiting for an answer the memory controller responds by asserting the "mem_fpga_readdatavalid" signal and provides the "mem_fpga_readdata" as the content of the bin. This signal is increased by 1 in the "write" state and set as "mem_fpga_writedata". The "mem_fpga_write" signal is asserted and afterwards the program returns to the "idle" state.

6.4. Result and Discussion

In a test sequence 5000 normally distributed numbers were fed into the FPGA. Each value consisted of 16 bits, which makes the representation of numbers from 0 to 65535 possible. A bin width of 64 was used and therefore a right shift by 6 bits was necessary. This results in 1024 bins. The histogram is visible in Figure 6.2.

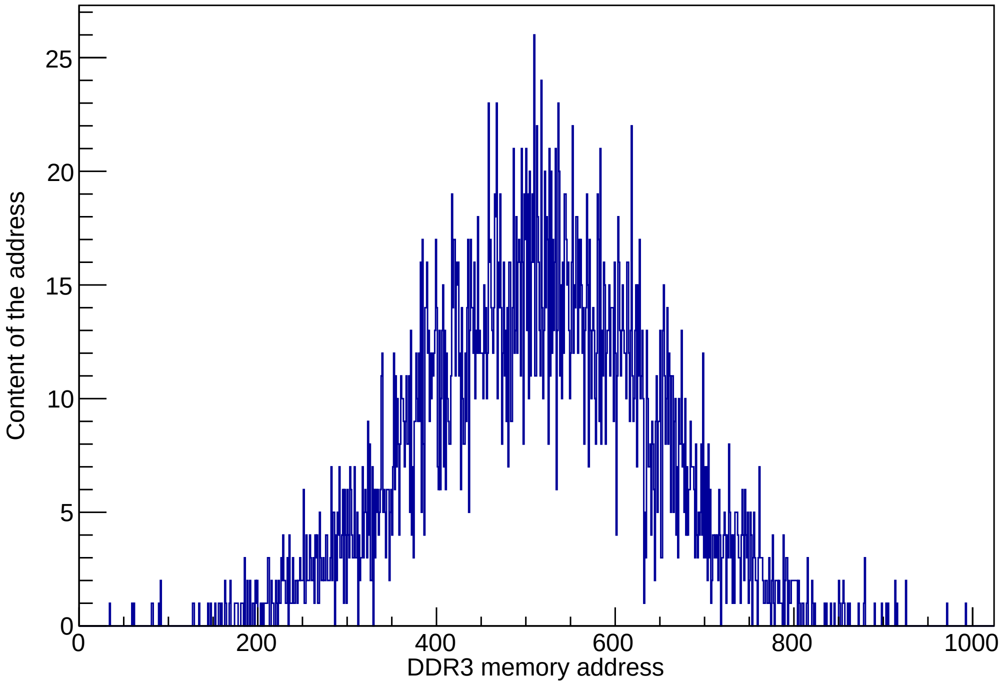


Figure 6.2.: The resulting histogram of 5000 Gaussian distributed numbers is displayed. The DDR3 memory address is equal to the bin number. The bin height is the content of the address.

A big problem of this procedure can be seen in Figure 6.1. It takes ten clock cycles to complete one data input. The histogram builder operates at 200 MHz and the collisions take place with a frequency of 40 MHz. Therefore new signals arrive already after 5 cycles and it should be possible to process multiple channels at once. One possibility to handle the amount of data is to implement a trigger system to select signals.

The DDR3 is not ideal for this task of continuously switching between read and write commands. Figure 6.1 also shows that the "write" state is a lot faster than "read". An alternative idea would be the usage of the DDR3 as a memory dump. Instead of building the histograms the program could just write as much data as possible on the DDR3 until it is full. In addition a trigger could decide which data should be stored. The histograms are built on the computer after the collection of data.

The readout of the histograms or the memory dump is done via the IPBus, which is explained in Chapter 7 in more detail.

7. IPBus

This chapter covers the slow control of the LATOME board. After implementation, the system is used to transfer the histograms from the DDR3 to the computer.

7.1. Structure

The second part of the programming task is about the IPBus system [39]. In CMS it is part of the "Code Archive for the Trigger UpgradeS" (CACTUS) Project. Since the system could fit the tasks in LATOME, it has been decided to use this protocol via Ethernet as transport mechanism for the slow control interface. The purpose is the controlling of all modules of the firmware. The system was originally developed for the CMS experiment and it was necessary to check, if it would fit this task in the LATOME board. The software was designed for Xilinx devices and a compatibility test to Altera products needed to be done.

The IPBus protocol is a simple packet based control protocol. It is the innermost "envelope" of the data package with its own header. The other protocols are covered in Chapter 5.5. Similar to AMM the IPBus works with the master-slave-principle and includes write and read commands. The basic setup can be seen in Figure 7.1.

The data arrives from the Ethernet at the UDP engine on the left. Here most of the headers are removed. The final one is the IPBus protocol. It is processed in the bus master. The bus fabric decides with the aid of the address decoder which slave is the destination of the package. Each slave is a module of the final firmware, for example user code, remapping or the histogram builder. The idea is that each module should have at least a read-only status and control register, which can also be changed by the computer.

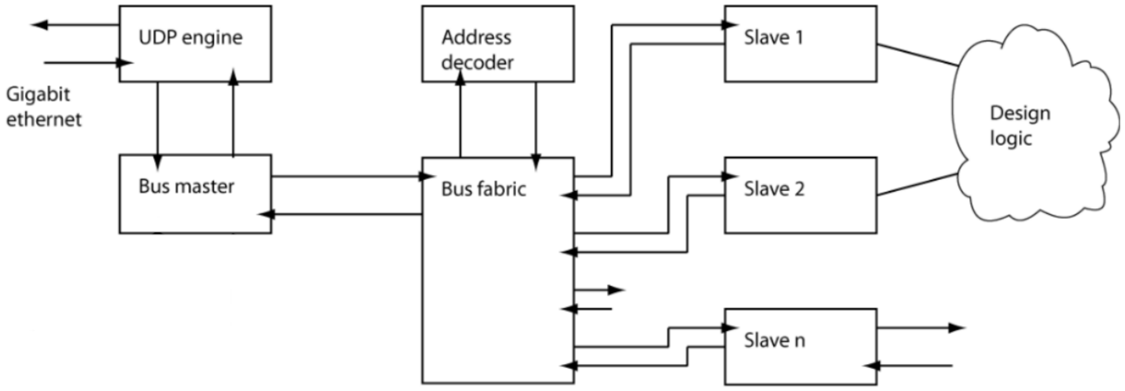


Figure 7.1.: The basic structure of the IPBus system is shown. The Ethernet packet arrives from the left. The bus fabric decides which slave is addressed with the aid of the address decoder [49].

7.2. Implementation into LATOME

The interfaces of the original IPBus are not compatible to the Altera environment. The connection to the different slaves is done via the "wishbone" interface [50], but in LATOME the aim is to use AMM, which was explained in Chapter 5.4. They are similar, but instead of a "read" signal, wishbone uses "strobe". It is always asserted when a command is given. Therefore a "write" command in wishbone consists of both strobe and write signal and the "read" command only consists of the strobe signal. The creation of the adapter, which connects the two interfaces, can be found in [51].

The connection to the Ethernet is also different. CACTUS uses AXI (Advanced eXtensible Interface), while the SoCKit board requires AST. Again the development of an adapter was simple due to the similarities. AST needs however a "start of packet" signal, which only asserts at the rising edge of the valid signal. It is not included in AXI and had to be generated.

An additional module "showcase" was created to demonstrate the connection between the IPBus with the other modules. The other members of the LATOME team could use this as a blueprint. It contains the basis status and control registers and can respond to the IPBus requests.

```vhdl
entity showcase is
  generic (
    addr_width: natural := 1
    --number of registers in 2**addr_width
  );
  port (
    clk                    : in    std_logic;
    reset                  : in    std_logic;
    ipctrl_showcase_reg_mm : inout amm_bus_t
  );
end showcase;
```

Code example 7.1: The port of the showcase module is shown. It consists of a clock, the reset signal and a specially constructed data type. It contains all read and write signals of the AMM.

The port is seen in Code example 7.1. "addr_width" resembles the number of registers, which are included in this module. The clock and reset should be present in every component to control latency and to have an accessible restart function of the system. "ipctrl_showcase_reg_mm" is a combined signal, which contains the input and output of the IPBus AMM interface. These can be seen in Code example 7.2. The first four lines contain the input signals and the final two resemble the output.

```vhdl
read          : std_logic;
write         : std_logic;
address       : std_logic_vector(31 downto 0);
writedata     : std_logic_vector(31 downto 0);

readdata      : std_logic_vector(31 downto 0);
readdatavalid : std_logic;
```

Code example 7.2: An overview of the signals of IPBus AMM are given. They are combined in the "amm_bus_t" data type.

Code example 7.3 shows the response part of the architecture section. These lines resemble the answer of the slave. "readdatavalid" is asserted, when there was a command of the master.

"reg" is an array of registers and the integer "sel" indicates which one of these is targeted. The information comes from the address signal of the IPBus AMM. Therefore "readdata" is the addressed register.

```
ipctrl_showcase_reg_mm.readdatavalid <= '1' when
  (ipctrl_showcase_reg_mm.write='1' or ipctrl_showcase_reg_mm.read='1') else '0';
ipctrl_showcase_reg_mm.readdata <= reg(sel);
```

Code example 7.3: The response of the showcase module in the architecture section is displayed. The first signal is asserted when there is an command from the master. The second signal is linked to the addressed register.

This summarized the most important sections of the showcase module. The whole code can be seen in Appendix A.2.

7.3. uHal

The library uHal [39] is necessary to control the slaves using the computer. It stands for "uTCA (Micro Telecommunication Advanced Architecture) Hardware Access Library" and provides the needed commands for a C program. The options are read/write a single register, memory block or FIFO. The structure of a program will be explained in this section.

```
int main () {
  ConnectionManager manager ("file://address_map/connection.xml");
  HwInterface hw=manager.getDevice ("sc.udp.0");
```

Code example 7.4: The beginning of the C program with the opening of a socket to the dedicated device is shown. An important aspect is the call of the "connection.xml".

The first part of the program handles the connection to the device (Code example 7.4). Therefore a special xml file is needed. Afterwards the hardware is defined to be accessible through the program, which is in this case "hw". The "connection.xml" file can be seen in Code example 7.5. The most important part is the IP address and the name of the address table.

```
<connections>
  <connection
    id="sc.udp.0"
    uri="ipbusudp-2.0://192.168.3.62:50001"
    address_table="file://address.xml"
  />
</connections>
```

Code example 7.5: The content of connection.xml is given. The right device is accessed and the "address.xml" is handed over.

"address.xml" contains all the locations in address space of the modules and the different registers inside them. An example can be seen in Code example 7.6. The file is initiated with the "TOP" node. On the next layer are two nodes: "showcase" and "hist". These are two example modules with the base address as an attribute. The final project will then have the User Code, Remapping, TDAQ and so on.

```
<node id="TOP">
  <node id="showcase" address="0x00000000">
    <node id="status" address="0x0" permission="r"/>
      <node id="b_0"  mask="0x00000001" permission="r" />
      <node id="b_1"  mask="0x00000002" permission="r" />
      <node id="b_2"  mask="0x00000004" permission="r" />
      ...
    <node id="control" address="0x1" />
      <node id="b_0"  mask="0x00000001" />
      <node id="b_1"  mask="0x00000002" />
      <node id="b_2"  mask="0x00000004" />
      ...
  </node>

  <node id="hist" address="0x00000010">
    <node id="status" module="file://status.xml" address="0x0" />
    <node id="control" module="file://control.xml" address="0x1" />
    <node id="fifo" address="0x2" mode="port" />
    <node id="memory_block" address="0x3" mode="block" size="16" />
  </node>
</node>
```

Code example 7.6: The content of address.xml is shown. It contains two modules with a status and control register each. "hist" also has a fifo and memory block.

In "showcase" there are two additional nodes on the next layer. "status" and "control" are the two registers. The address is given relatively to the "showcase" location. As mentioned earlier the status register should only be readable by the IPBus and therefore the corresponding node has the suiting permission.

Each of the 32 bits of the registers is listed in the final layer of this module. To save space, only the first three are displayed. Every bit will have its own meaning and therefore it is convenient to be able to address each one individually.

The status and control registers are also present in the "hist" module. The single bits are not listed, but rather transfered to different xml files. This makes the code clearer and shorter, since every module should have these registers.

The last two nodes display the other two options of uHal. "fifo" is specified by the port mode. There it is possible to read or write multiple data from or to the same address.

"memory_block" works similar, but the address changes with every request. Therefore it is important to include the size attribute aside the block mode.

This xml file is also used to generate the address decoder for the IPBus inside the FPGA. A Python program is provided by CACTUS for this task.

The next part of the C program shows different commands to interact with the modules. In Code example 7.7 the reading of a status register is given. A variable "stat" is defined with the result of the read command. It consists of the device "hw" and then the different nodes of the address xml-file, which are in this example "showcase" and "status". "read()" indicates the type and with "hw.dispatch()" the command is sent. The final line displays the "stat" variable as the result.

```
ValWord<uint32_t> stat = hw.getNode("showcase").getNode("stat").read();
hw.dispatch();
std::cout << "status=" << stat.value() << std::endl;
```

Code example 7.7: The example of a reading process is shown. The device is indicated first. Then every node is stated and the command is given at the end.

The example for writing can be seen in Code example 7.8. The process is similar. The dedicated device and nodes need to be chosen. The command is completed with "write()" and the desired data is placed between the brackets. The second part shows the addressing of a single bit. Therefore only an additional node has to be included.

```
hw.getNode("showcase").getNode ("ctrl").write(42);
hw.getNode("showcase").getNode ("ctrl").getNode ("b_0").write(1);
hw.dispatch();
```

Code example 7.8: The example of a writing process is displayed. It is executed similar to the read request with a different command at the end.

The final part of the code shows the other options of uHal. In Code example 7.9 the vector "xx" is defined with an order of ascending numbers. This will be used to write the memory block or FIFO, which is similar to handling a single register. Only the last part of the commands changes to "writeBlock()", with the vector between the brackets. The next line reads this block, including the information of the size in "readBlock". Finally the input and output data is displayed.

```
const size_t N=20000;
std::vector<uint32_t> xx(N);
for(size_t i=0; i!= N; ++i)
  xx[i] = i;

hw.getNode("hist").getNode("memory_block").writeBlock(xx); //or ("fifo")
ValVector< uint32_t > mem = hw.getNode("hist").getNode("store").readBlock(N);
hw.dispatch();

for(size_t i=0; i!= N; ++i)
  std::cout << "xx␣=␣" << xx[i] << "␣mem␣=␣" << mem[i] << std::endl << std::endl;
```

Code example 7.9: The handling with FIFOs and memory blocks is given. The first section defines a vector. It is written to the FIFO/memory plot in the second part and read again. The last section compares the two arrays.

With this setup the change of registers or bits was successfully tested and the functionality of the IPBus system proved. The next task is to combine this with the histogram builder. In order to test the capability of the system, it is necessary to readout the whole DDR3 memory and send the histograms to the computer.

7.4. Reading of the DDR3 via IPBus

This section describes the readout of the DDR3 memory and shows several signal tap snapshots. The used code can be seen in the Appendix A.3.

There are multiple possibilities to read the DDR3 through the IPBus. The usage of the memory block is one of them. Some space is needed to be cleared on the FPGA to store a part of the content from the DDR3. After reading this memory by the IPBus the next section from the DDR3 is been written on the FPGA. Step by step the software transports the whole data to the computer. This takes a lot of time and also requires space on the FPGA, so it was not implemented.

The alternative possibility is the usage of the FIFO option. The IPBus only reads one address and this location provides one continuous stream of data points. Figure 7.2 shows a snapshot of the beginning of a readout with this technique. The figure can be evaluated the same way like the last Signal Tap image. On the left are the signals, sometimes combined in vectors with the quantity inside the brackets. The time axis points to the right.

The process starts when the IPBus sends a read command and the address "12h". The "h" indicates that a hexadecimal number is displayed. All other values in this image are

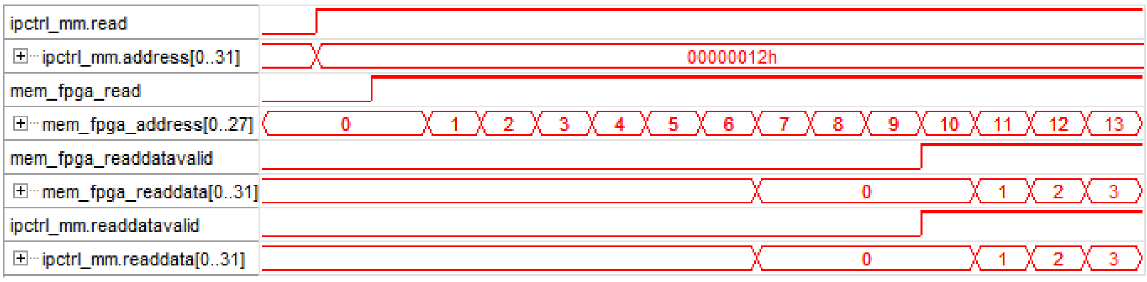


Figure 7.2.: The beginning of a readout cycle, captured with Signal Tap, is shown. The read command is given to the DDR3 as soon as the request from the IPBus arrives. The address is increased with each cycle. The response of the memory controller arrives after a short period of time.

unsigned decimals. First the vector shows which module is addressed. The "1" on the second to last position means that the command is forwarded to the histogram builder, while a "0" would stand for the showcase module (refer to the Code example 7.6). The last digit indicates the register. "0" would be the status register, "1" the control register and "2" points to the FIFO port. As mentioned in Section 7.3, this option provides a continuous data stream without the necessity of the IPBus to change the address. Therefore this signal stays the same for the whole process.

As soon as the command from the IPBus arrives, the module initiates a read process of the DDR3 memory. The answer after a "read" command takes several cycles (refer to Section 6.3). Fortunately the DDR3 possesses "burst"-reading. This means, that the commands can be given continuously and after the some time the response is also continuous. Thanks to this feature the readout is rather quick. The address needs to be increased with each command.

In oder to check the validity of the readout, each location of the DDR3 memory was filled with its own address. Therefore the "mem_fpga_readdata" signal is an ascending sequence. The output of the DDR3 memory controller is directly connected to the response of the module to the IPBus. Figure 7.2 shows that both pairs of signals are the same.

A problem which needed to be addressed is the course of action when the connection is interrupted. For example sometimes during the read burst of the DDR3 the memory controller asks for a rest period via the "mem_fpga_waitrequest_n". This can be seen in Figure 7.3. This signal is usually "1", but if a problem occurs it becomes "0". Therefore it is active low and has the "_n" at the end of the name.

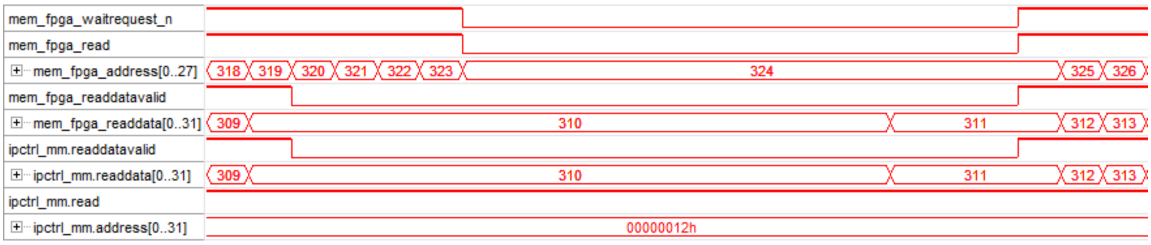


Figure 7.3.: An assertion of the wait signal of the memory controller is given. The read command is immediately interrupted and the increasing of the address stops. Both continues when the wait request is set to one again.

This does not result in conflicts in the final code. The module simply stops the read request and keeps the address constant for the time the wait signal is asserted. The output to the IPBus does not cause problems either. It is still connected to the respond of the memory controller and continues to forward data from the DDR3. The address of the IPBus command and the read signal do not change as well.

The most common reason for a disconnection is the IPBus protocol itself. Its packages can only contain 255 words. Figure 7.4 shows that after reading 255 registers (address: from 0 to 254) the acceptance of data from the IPBus stops. By setting the "ipctrl_mm.readdata" signal immediately to zero, no information can be sent for a few cycles.

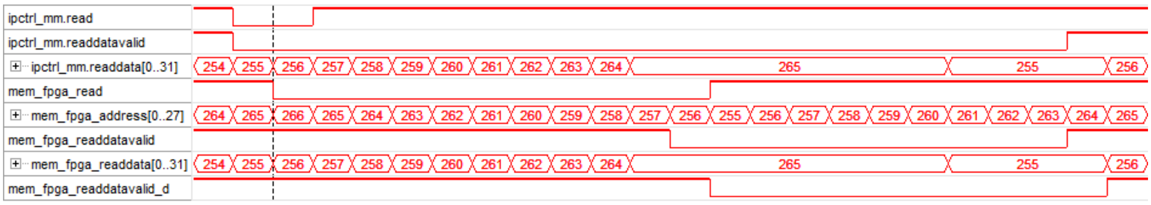


Figure 7.4.: A snapshot of signals between the IPBus packets is displayed. The read command directed at the DDR3 memory controller is interrupted and the program waits until the "readdatavalid" is zero. During this time the address decreases and the readout starts again at the right location.

The problems are about 10 additional registers, which have already been requested from the module. The program waits until these tasks are processed and the "mem_fpga_readdata" signal returns to zero before sending a new command (asserting "mem_fpga_read"). In addition the address was already increased too much at the time of the end of the packet, visible in the beginning of the fifth line in Figure 7.4. The following cycles are used to decrease the address again, so it has the right value at the beginning of the new packet.

The remaining problem is how the program distinguishes between continuing with the current address or starting the readout from the beginning, when a new request arrives. Therefore the first bit of the control register is defined as the reset of the address. Beginning with a new readout request of the DDR3 by the IPBus, this bit is set to "1" first. The procedure is shown in Figure 7.5.

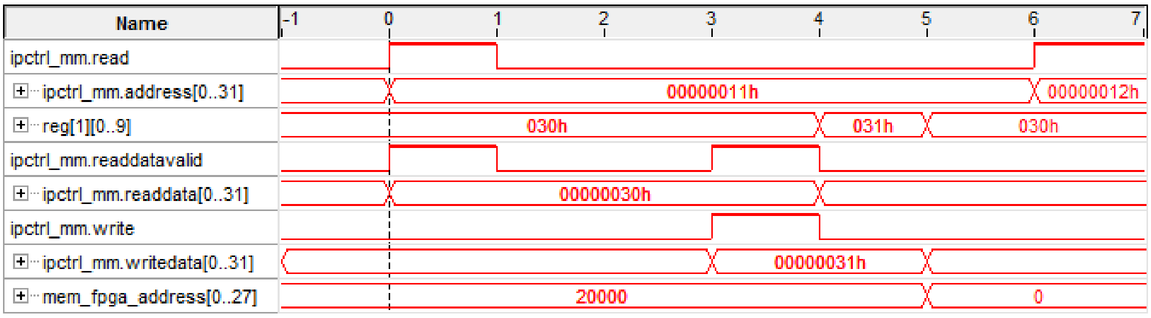


Figure 7.5.: The process of resetting the address counter is shown. First the control register is read, the special bit changed and the whole register written again.

The addresses of the IPBus are 32 bit based and single bits cannot be changed independently. Therefore the corresponding register is read first. The "11" of the address signal in the second line indicates that the control register of the histogram module is being contacted. The third line shows the control register. This information is sent to the IPBus via the "readdata" vector and by asserting the "readdatavalid" signal. There the single bit is changed and the write command is started.

The new content of the control register can be seen in the third line after the fourth cycle in Figure 7.5. As soon as the first bit is changed, the module resets the bit again and the control register has the same content as in the beginning of the process. The last line shows the reset of the "mem_fpga_address" signal after the fifth cycle. After this procedure the readout process from Figure 7.2 is started.

It is possible to readout the whole 1 Gb of the DDR3 in 17 seconds. Since every location of the memory contained the value of its own address, the possibility of the skipping of an register could be excluded and the readout without error was proven.

This IPBus connection and the histogram builder itself could then be combined to gain the final program. A sketch of the test environment including the histogram builder and the IPBus can be seen in Figure 7.6.

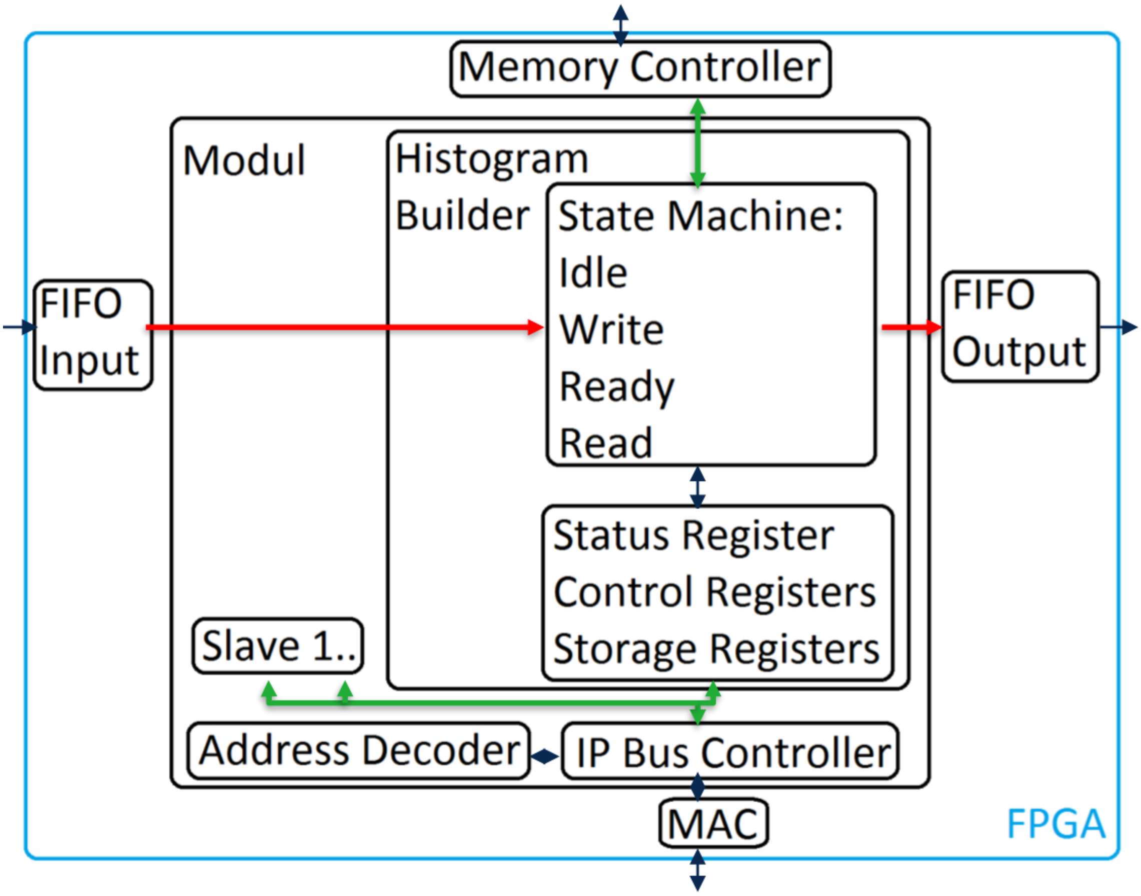


Figure 7.6.: A sketch of the final test environment is displayed. The module consist of the IPBus and the Histogram Builder. The handling of the data, which arrives from the FIFO input and is sent to the FIFO output, is done via a state machine. Several registers are the connection to the IPBus. Figures 5.3 and 5.5 show the connections to the PC.

8. Calibration

This chapter describes the analysis of current calibration data from the LDPS demonstrator system installed on the LAr calorimeter in ATLAS. The results are then compared to simulation. The simulation tool SPICE [52], which generates this data, and its configuration is widely used. A validation of the simulation process or the identification of differences to the calibration data could help improving the software implementation.

8.1. Introduction

A demonstrator was installed in the LAr calorimeter readout system during the Phase-0 upgrade in 2012. The goal was to show that the system works and does not disturb the current readout. The new components are LSBs, two backplanes, two LTDBs, which cover an area of $1.767 < \phi < 2.160$, $0 < \eta < 1.4$, and two LDPSs. The LTDBs on the front end can handle up to 320 SC signals. After digitization and serialization, the data streams are transported via optical fibers to the Back End electronics [53].

The two LDPSs are called "ABBA" (ATCA Board for a Baseline of Acquisition), a prototype of LATOME. They receive the data from 40 optical fibers and process 320 SCs. Similar to LATOME the calculated energies are stored in a circular buffer to wait for L1 trigger acceptance.

After the verification, that the new readout chain does not introduce additional noise [53], the system was calibrated. A DAC (Digital Analog Converter) charges a capacitor to generate a pulse, which is feed into the readout chain close to the LAr detector. The signal is sampled every 25 ns with an 12 bit ADC. The digital values cover an interval of $2^{12} = 4096$ ADC counts. The value without a pulse is called pedestal. In addition it is possible to shift the phase of the same pulse by a small delay (e.g. 1 ns) and sample it again. Therefore the signal can be digitized very accurately.

The two important collections of data, which were analyzed during this master thesis are "ABBA_278938" (in the following: Run 1) from the 9th of September 2015 and "ABBA_286202" (in the following: Run 2) from the 18th of November 2015. The differences between the runs will be explained throughout the chapter.

It is possible to simulate the detector signals with SPICE. This tool is able to build numerical models of electrical components. The aim is now to compare these simulations with the measured calibration data and find mistakes or opportunities for improvements.

8.2. Signal shape

A run consists of many events, in which the capacitor is charged according to a certain DAC value. The discharge results in the exponential pulse. This mimics the triangular shape of a real ionization pulse. A description of the triangular pulse and the reshaping can be found in Section 4.3. As mentioned before every event has a particular delay number. Each combination of variables (DAC value and delay number) is repeated 200 times to be able to form the average. Table 8.1 summarizes the different values of the two runs.

A special equation is needed to construct the signal shapes from the different events, which are taken at different time delays.

$$Time = \Delta t_{\mathrm{TTC}} \cdot \left(i_{\mathrm{sample}} + 1 - \frac{i_{\mathrm{delay}}}{N_{\mathrm{delays}}} \right)$$

In this formula Δt_{TTC} is the sampling time and equals to 24.95 ns. $N_{\mathrm{delay}} = 239$ is the number of possible delays per clock cycle. Table 8.1 shows the different i_{delay} values. i_{sample} starts at 0 and ends at $N_{\mathrm{samples/event}} - 1$. The result is displayed as the x-axis in the following plots with signal shapes. The y-axis is the output of the ADC and is stated in "counts". Figure 8.1 shows an example of a Run 2 calibration pulse.

Run	$N_{\mathrm{samples/event}}$	N_{DAC}	N_{delays}	$\Delta i_{\mathrm{delay}}$	Last i_{delay}	N_{samples}
1	10	5	24	10	230	240
2	40	14	3	100	200	120

Table 8.1.: The different values for the two runs are shown. Specific DAC values will be given later. The first i_{delay} is always zero.

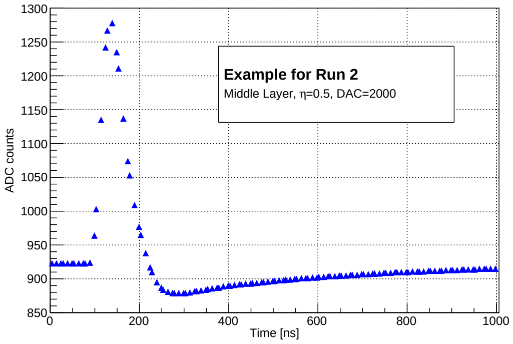


Figure 8.1.: A sampled calibration pulse is displayed. The 120 digital values were measured during Run 2.

The time axis reaches until 998 ns, because there are 40 samples per delay. Since Run 1 has only 10 samples (compare Table 8.1), the axis ends at 249.5 ns. In this case the tail is missing and a comparison with the SPICE simulation is not possible. Therefore additional runs have been taken with the same setup, but with a different latency of the trigger. With each set the sampling was done 250 ns later compared to the previous one. When referring to Run 1 it is actually a combination of four different sets (278938, 278943, 278945, 278949). Unfortunately due to electrical problems the latency is not perfectly adjustable yet and can result in a shift of data points, which is displayed in Figure 8.2. Fortunately that happened only to a small number of SCs. In this study these kind of signals will not be used.

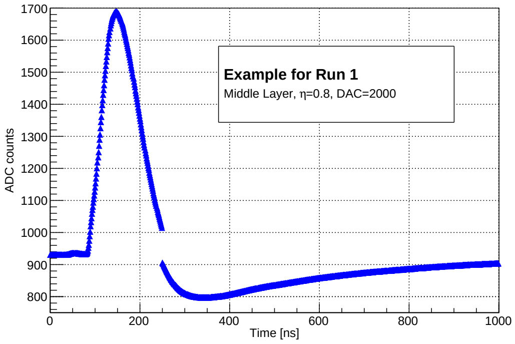
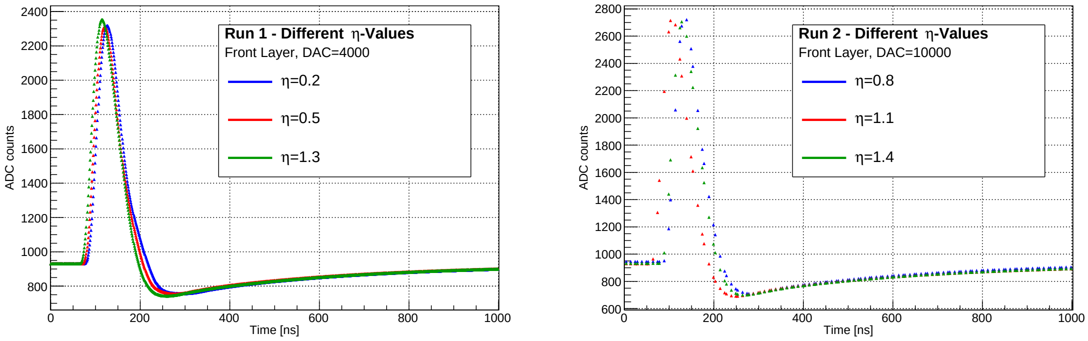


Figure 8.2.: A signal shape with an unwanted step is shown. Here four runs are combined and the pulse is not connected due to problems at setting a specific latency.

Figure 8.3 displays plots of different η regions. Each elementary cell is charged with the same DAC value during calibration. Therefore the difference for various η values is minimal. The peak amplitude is slightly shifted due to different cable lengths. The pulse height and consequently the undershoot vary as well, but are not directly related to η. The reason lies probably in the change of absorber thickness. Therefore several conversion factors will be introduced to take this effect into account. In Figure 8.3(b) the red curve is slightly shifted to the left. This also appears at different DAC values in the same η region. As mentioned before, the reason is the improper latency setting for this SC.



(a) Run 1

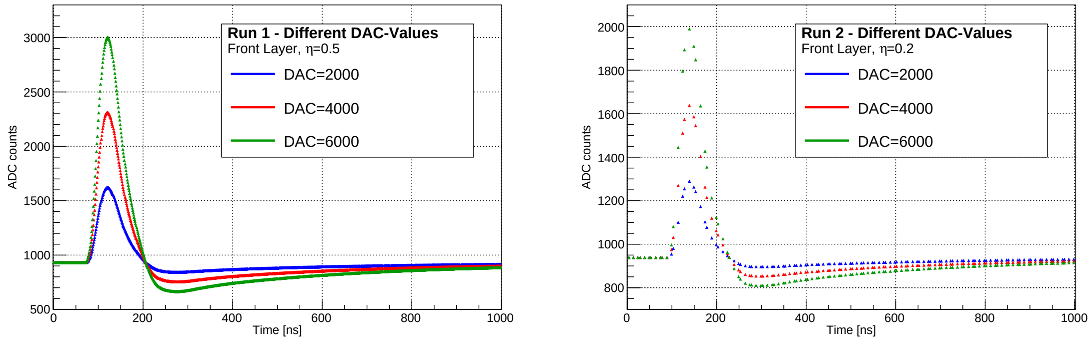
(b) Run 2

Figure 8.3.: Signal shapes for different η values.

The Figure 8.4 shows, that a higher DAC value results in a larger signal height as expected. This is very important, since the calibration simulates the energy deposit of the particle and the pulse height is used to identify that energy. The linear relation between the pulse height and the energy, which is an crucial aspect of calorimetry, will be demonstrated later. The plots also show that a higher pulse also yields a larger undershoot.

If the signal shapes are compared to Figure 4.5 a difference in the tail becomes visible. The reason can be found in the calibration system. The discharge of a capacitor generates an exponential curve, as shown in Figure 8.5, and not a triangular one. Therefore the undershoot of the bipolar signal has no plateau and vanishes slowly with a small slope. To be able to compare the calibration results with the simulation, SPICE was fed with the exponential curve from Figure 8.5 and not a triangular shape. The dots are the measurement from an oscilloscope and the red line is the exponential fit.

The comparisons can be seen in Figure 8.6. Instead of using ADC counts for the y-axis,



(a) Run 1 (b) Run 2

Figure 8.4.: Signal shapes for different DAC values.

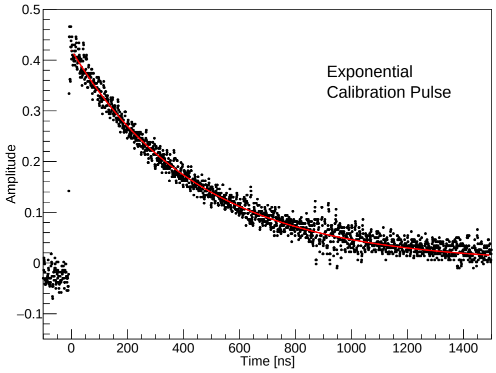
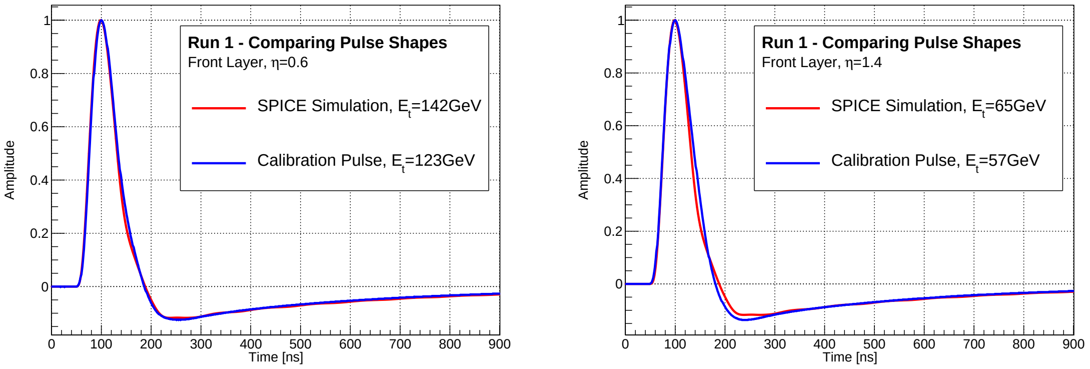


Figure 8.5.: The discharge of an capacitor is displayed. The measured data points are shown in black. The red exponential fit serves as the input for the SPICE simulation.

the pulse was normalized to a pulse height of 1 with a pedestal value of 0. In addition the shape was shifted in time to set the maximum amplitude to 100 ns, which makes the differences more visible. Some examples (Figure 8.6(a)) show that the simulation with exponential shaped pulses fits the measured calibration pulse very accurately. The falling slope differs slightly and in some cases the smallest amplitude is lower in simulation. This appears in particular at higher η regions (Figure 8.6(b)). The energy value of the calibration pulse is calculated from the DAC value (refer to Section 8.3). Please note, that the simulated energy can differ slightly from the measured data since the same energy was not always available.



(a) (b)

Figure 8.6.: Comparison of signal shapes between calibration and SPICE

8.3. Detector geometry

The height of the pulse will be set in relation with the DAC value. Therefore the pedestal value is calculated by forming the average of the first values before the signal rises. Then the pedestal is subtracted from the maximal value of the pulse and this determines the height. The idea can be seen in Figure 8.7. In Run 2 the number of data points is rather low and therefore a Gaussian fit was applied to the data points around the peak to get a more accurate maximum value.

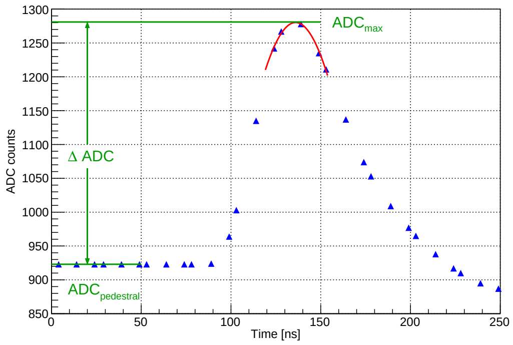
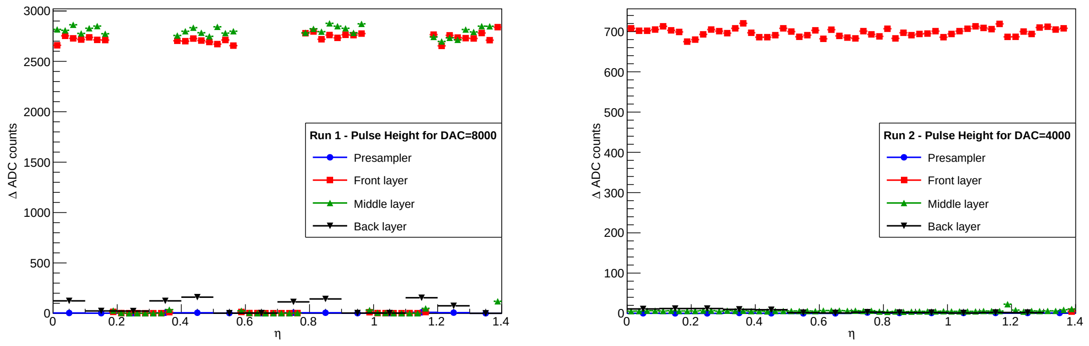


Figure 8.7.: The determination of the pulse height is shown. The Gaussian fit is drawn in red.

This section is used to give an insight into the geometry of the cells connected to the demonstrator readout. Each LTDB covers a ϕ slice and the η region from 0 to 1.4. One ϕ slice consists of 14 SCs in Presampler and back layer and 56 SCs in the front and middle layer. This sums up to 140 signals for each LTDB. A special mapping table, which is needed to relate the channels to the right SCs, was provided by the LAr Calorimeter group. Due to complications in the readout only the data from one LTDB was collected.



(a) Run 1

(b) Run 2

Figure 8.8.: The pulsed SCs are displayed. The expansion of each cell is shown by the error bars.

In addition not every cell was pulsed during the calibration. Figure 8.8 shows to which SCs a signal was injected. During Run 1 only parts of the front and middle layer were pulsed. Run 2 was only used to investigate the pulse shapes coming from the front layer. Every data point is a single SC and the error bars show their expansion in the η direction. In the front layer 8 elementary cells form 1 SC. During Run 2 only every second elementary cell was pulsed and therefore the corresponding DAC value needs to be multiplied by two in relation to Run 1.

It is also important to know which DAC value represents which deposited energy. The proportional relation for each SC can be seen in Figure 8.9. The values were measured during former calibration runs and consist of different factors, which will be covered in the next paragraph. A mentionable fact is the higher conversion factors for the middle layer compared to the front layer. The visible jump at $\eta = 0.8$ arises mainly due to the design of the LAr calorimeter. This can be seen in Figure 8.10, which shows the detector thickness dependence on η.

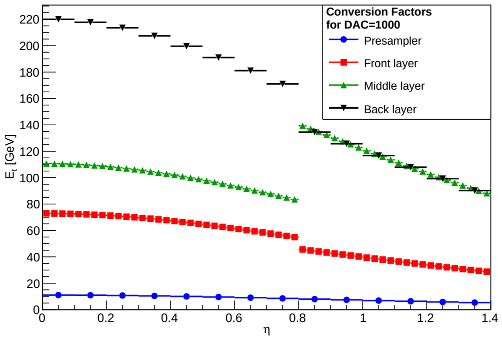


Figure 8.9.: The conversion factors for each SC are displayed. The jump at $\eta = 0.8$ is due to a change in absorber thickness and related electronics.

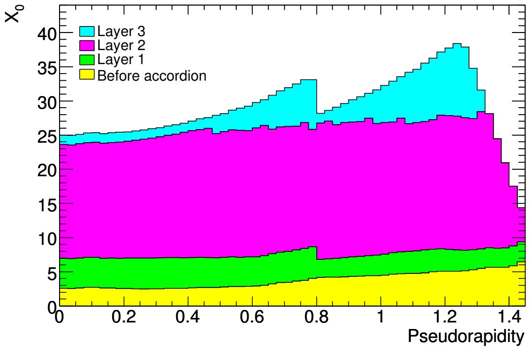


Figure 8.10.: The barrel thickness in radiation length depending on η is displayed.[24]

The conversion factors can be broken down into two parts:

$$E_{\text{cell}} = F_{\mu\text{A}\to\text{MeV}} \cdot F_{\text{DAC}\to\mu\text{A}} \cdot DAC$$

The factor $F_{\text{DAC}\to\mu\text{A}}$ converts the DAC value to a current. It is a property of the calibration board and depends on the resistors in operation[54]. $F_{\mu\text{A}\to\text{MeV}}$ is obtained from test beam studies and converts the current to an energy value. It is affected on the one hand by the absorber thickness causing the production of secondary particles which in turn ionize the LAr and on the other hand on the design of the readout system and the connected electrodes.

The final energy calculation will be explained briefly. It is the linear combination of several samples of a pulse and multiple factors [55]:

$$E_{\text{cell}} = F_{\mu\text{A}\to\text{MeV}} \cdot F_{\text{DAC}\to\mu\text{A}} \cdot \frac{1}{\frac{M_{\text{phys}}}{M_{\text{cali}}}} \cdot G \cdot \sum_{j=1}^{N_{\text{samples}}} a_j (s_j - p)$$

In this equation, s_j are the samples, p is the already mentioned pedestal value and a_j are the filter coefficients. These parameters can be identified or controlled with the histogram builder (refer to Section 6.1). A common value for N_{samples} would be 5. This sum gives the amplitude of the pulse in ADC counts [56]. This topic is widely covered in [57].

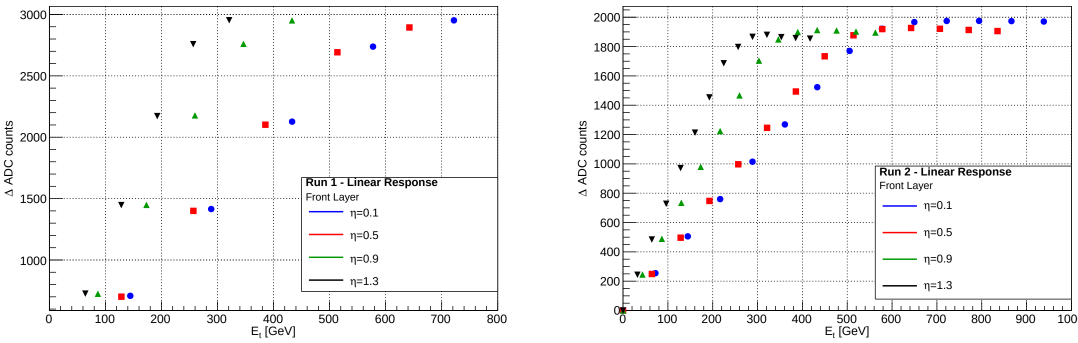
The gain G represents the conversion from voltage to ADC values and is computed by DAC ramp runs (refer to Section 8.4). The factor $\frac{M_{\text{phys}}}{M_{\text{cali}}}$ takes the different injection pulses into account. As mentioned before the calibration pulse has an exponential curve as origin and therefore the peak amplitude is a bit different for the triangular ionization signal.

8.4. Linear Response

With the information about the pulse height and the energy deposit, it is now possible to find a relation between the two values. Therefore correlation plots of these parameters parameter are shown in Figure 8.11. During Run 1 only 5 DAC values were used (2000, 4000,..., 10000), while Run 2 reached a maximum value of 26000 with the same spacing, but with 14 data points, including a measurement without pulsing. Each value was converted into transverse energy and is visible on the x-axis. The y-axis displays ΔADC, obtained from the signal shapes.

In the low energy range of the plot a linear dependence is visible. Figure 8.11(b) shows that ΔADC settles to a maximum amplitude at higher energies, which is only partly visible in Figure 8.11(a). This effect is called saturation and is caused by the analog electronics. At high energies the amplifier is not able to increase the voltage further and the linear relation is lost. The effect of saturation on the signal shapes is examined in Chapter 8.5.

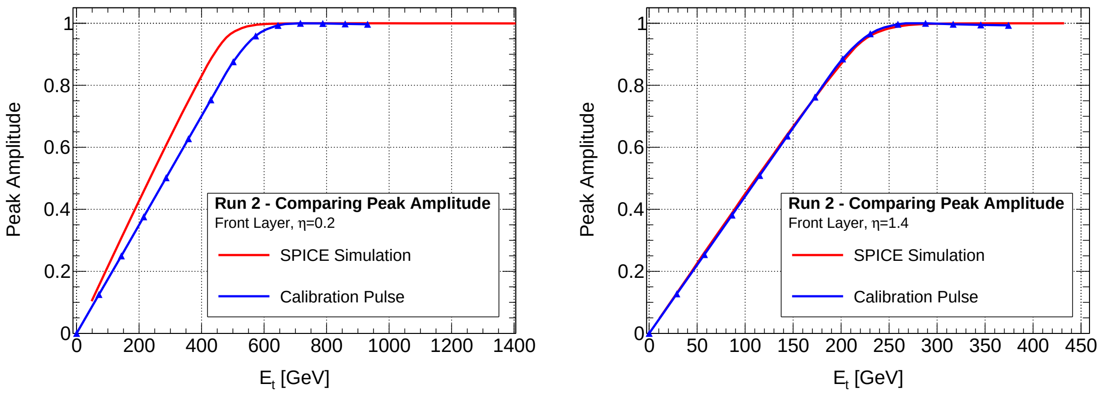
Another observation is the earlier saturation at higher η. The reason is the choice of the x-axis. The ratio of transverse to total energy gets smaller in higher η regions, but the signals saturate at about the same total energy. The curves look roughly the same, if the x-axis would display the total energy.



(a) Run 1 (b) Run 2

Figure 8.11.: Relation between transverse energy and ΔADC

It is easier to compare the linear response of the simulation with Run 2 data, since there are more data points available. The plots can be separated into two groups. Both are visible in Figure 8.12. Unfortunately the data from the simulation is only available in volts and not ADC counts. Therefore the maximum of the curves is scaled to 1.

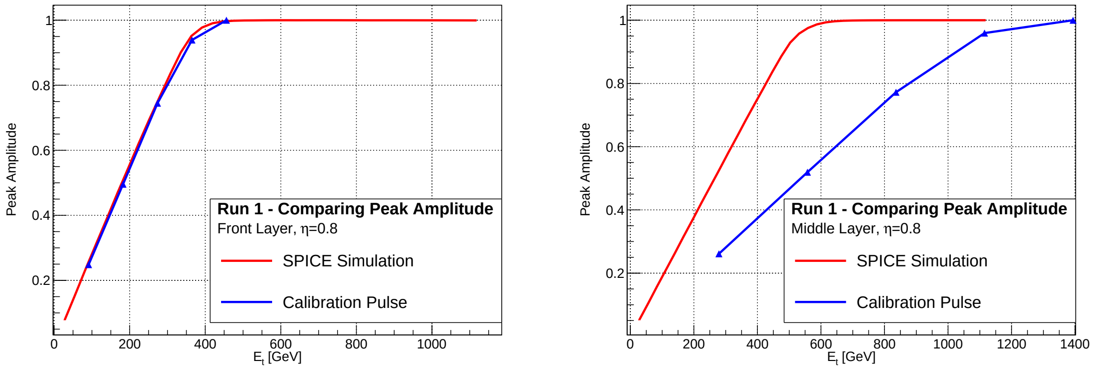


(a) Lower η region (b) Higher η region

Figure 8.12.: Comparison of the linear response between simulation and calibration from Run 2

In the η region between 0.8 and 1.4 the curves agree very well with each other (Figure 8.12(b)). There are some differences in the middle of the barrel (Figure 8.12(a)). The saturation occurs earlier in the simulation and therefore the slope is steeper. It seems the jump of the conversion factors at $\eta = 0.8$ was not included in the simulation. The values of the involved electrical components have to be checked and if necessary updated.

This effect is stronger visible, when the middle layer is analyzed, since the jump in Figure 8.9 is greater. Unfortunately there are only 5 DAC data points in Run 1, but the much earlier saturation in the simulation is still visible (Figure 8.13(b)). The results for the front layer are similar to Run 2.



(a) Front layer (b) Middle layer

Figure 8.13.: Comparison of the linear response between simulation and calibration from Run 1

8.5. Saturation

This section describes the effect of saturation on the signal shape. Figure 8.14 shows several curves in the higher energy region. It is only the beginning of saturation and therefore only different pulse heights are visible. Although the change in the DAC value is the same, the maximum of the largest DAC input has only a few more ADC counts, indicating the loss of linearity.

Since the same DAC input results in a higher energy value in the middle layer, the effect of saturation is more visible in Figure 8.15. A plateau is formed at the level of the pedestal before the undershoot. The plateau becomes larger with higher DAC value. In a bigger η

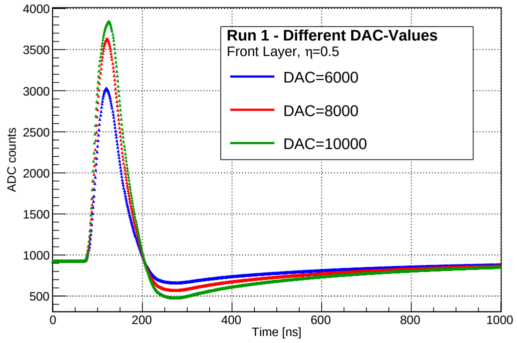
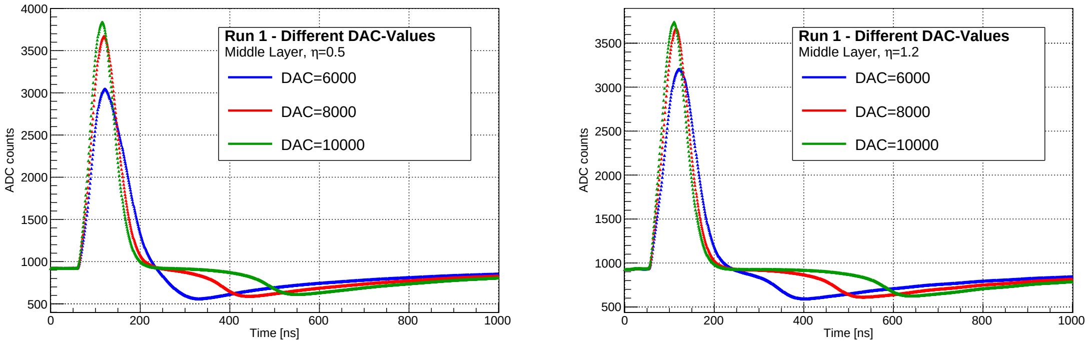


Figure 8.14.: Signal shapes in the high energy region

region this saturation effect is even visible at 6000 DAC counts. These different shapes confirm the conversion factors of Figure 8.9.



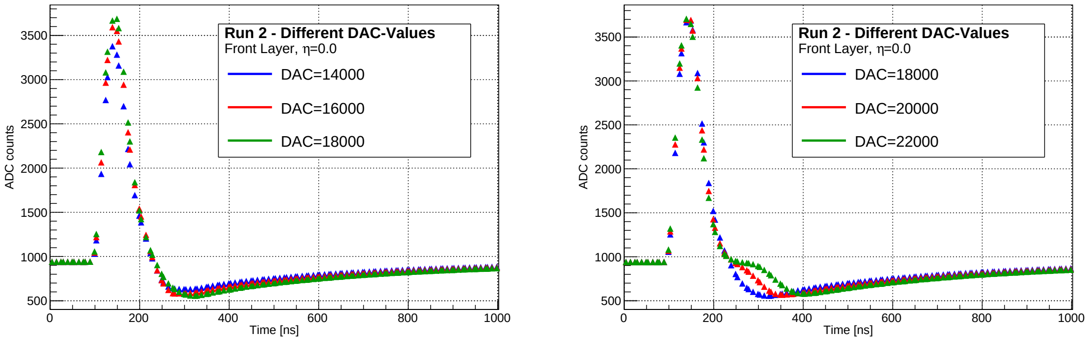
(a) Lower η region

(b) Higher η region

Figure 8.15.: Signal shapes from large DAC values in the middle layer

In Figure 8.16 some examples of Run 2 are shown. The saturation occurs at higher DAC counts, because not every elementary cell was pulsed. It is interesting that the plateau formation is slow compared to the middle layer.

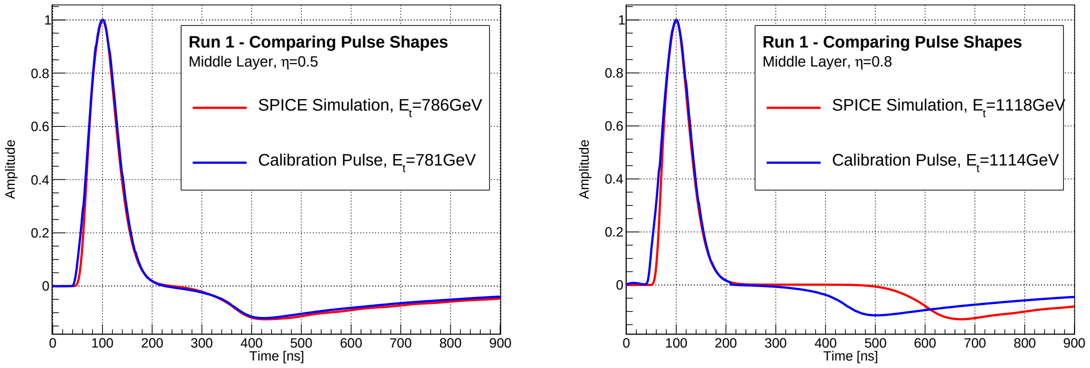
First the middle layer is compared to the SPICE simulation, as displayed in Figure 8.17. The lower η region shows equal or at least similar behavior. The simulation predicts larger effects of saturation at equal energies. This is consistent with the linear response plot. Figure 8.13(b) indicates a different situation in the higher η region. Unfortunately



(a) First set of DAC values (b) Second set of DAC values

Figure 8.16.: Saturation signal shapes of Run 2

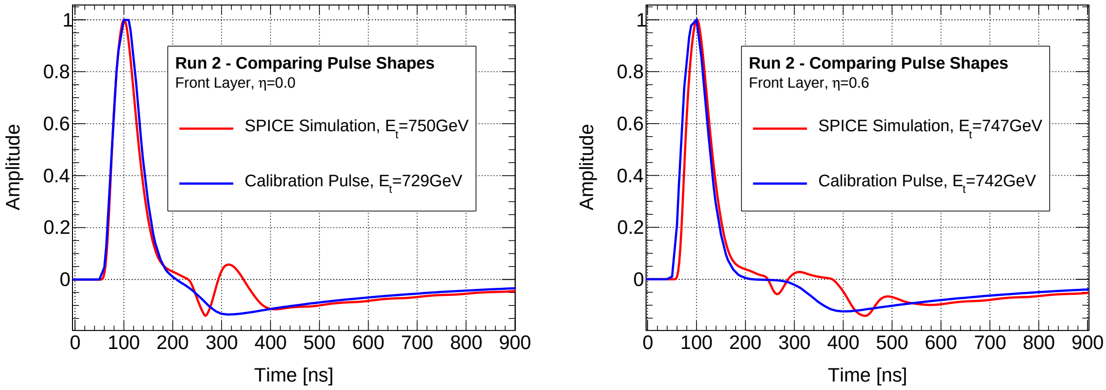
there was not much simulation data available in this η region, but Plot 8.17(b) shows a strong difference between simulation and calibration data.



(a) Lower η region (b) Higher η region

Figure 8.17.: Comparison of the saturation between simulation and calibration data in the middle layer

The calibration pulses from Run 1 show no saturation effects in the front layer. The more interesting pulses can be seen in Run 2. Figure 8.18 shows some examples of the comparison. It seems that there are strong oscillations predicted in the simulation at the beginning of the saturation (Figure 8.18(a)). In higher energy regions the response of the pulse differs even more from the measured data (Figure 8.18(b)). A reason could be a simulated cable reflection. This effect only occurs in the front layer and therefore the SPICE simulation input of both layers have to be compared to find the differences, which lead to this behavior.



(a) Beginning of saturation

(b) Higher energy region

Figure 8.18.: Comparison of the saturation between simulation and calibration data in the front layer

9. Summary and Outlook

The rareness of interesting physical processes in LHC proton-proton collisions requires a higher luminosity in future experiments. Therefore LHC developed an upgrade plan, which is implemented via two main phases. The increasing luminosity results in a higher data bandwidth for high p_T signatures in the individual detectors. A better trigger system is therefore mandatory. At the moment, the L1 trigger of the ATLAS LAr calorimeter only applies thresholds to the deposited energy and uses additional thresholds as an isolation criteria. In the future special shower reconstruction algorithms are able to deliver a better criteria to distinguish signal and background.

An implementation of these algorithms demands a high trigger granularity. The present TTs are replaced by SCs, which will improve the granularity of the trigger readout by a factor of 10. This requires a new readout system of the LAr calorimeter, which will be implemented during Phase-1 of the upgrade plan. The LATOME board is a core component of the new readout system and has the task to reconstruct the energy of the individual digital pulses.

First a test environment was developed to check the functionality of a LATOME module with the standard ALTERA Avalon interfaces. This test-bench setup made the design of a histogram builder possible. It will be used in the LATOME firmware to monitor untriggered data during normal operation. The histograms are saved on the DDR3 RAM. The setup was successfully implemented, but also revealed a timing problem. As an alternative, the memory is planed to store the raw data stream and the histograms can be build after downloading by the computer. In the future, it has to be evaluated which method is optional for the LAr calorimeter operation.

In the second part, the IP Bus system was tested. It provides control over the individual modules and is a crucial part of the LATOME firmware. It is connected to an external computer via Ethernet. Each module will be equipped with status and control registers, which are under control of the IP Bus. The functionality of the system was successfully tested. In addition it was possible to read out the whole DDR3 memory via Ethernet. Studies of the stability of the system need to be done with future studies.

To verify the correct functionality of the readout, a demonstrator system has been installed during Phase-0. This master thesis covered the analysis of the latest calibration data from this setup. The signal shapes and the pulse heights were determined. In addition the linear response between the amplitude of the signal and the used DAC value were verified and the effects of saturation was studied. It was possible to compare the results with SPICE simulations. The disagreements can be traced back to different points of saturation in some η regions and unexplained oscillations in the front layer. As a solution the simulation configuration of the front and middle layer should be compared, because the simulation of middle layer is free of the oscillation effects.

List of Figures

2.1. Production Cross Section Measurements 5

3.1. CERN's Accelerator Complex 8
3.2. Structure of ATLAS 9
3.3. Interaction of different particles with the detector sections 10
3.4. Structure of LAr Calorimeter 12
3.5. Accordion Geometry 13

4.1. LHC Timeline 15
4.2. Simulation regarding the L1 Trigger Upgrade 17
4.3. SC Upgrade for the LAr Calorimeter 18
4.4. New readout system for the LAr calorimeter 19
4.5. Display of the different stages of the pulse shape 20
4.6. Layout of the LDPS-firmware 21

5.1. Sketch of the programmable switches inside a FPGA 24
5.2. Layout of the SoCKit board 27
5.3. Sketch of Connections between computer and FPGA 28
5.4. Read and write transfer of the AMM interface 29
5.5. Sketch of the test environment used during the master thesis 30

6.1. Snapshot of signals inside the FPGA during histogram building 35
6.2. Resulting histogram of 5000 Gaussian distributed numbers 36

7.1. Structure of IPBus system 40
7.2. The beginning of a readout cycle captured with Signal Tap 46
7.3. Assertion of the wait signal of the memory controller 47
7.4. Snapshot of signals between the IPBus packets 47
7.5. Process of resetting the address counter 48
7.6. Sketch of the final test environment 49

8.1. Signal shape of a sampled calibration pulse 53
8.2. Flawed signal shape of a sampled calibration pulse 53
8.3. Signal shapes for different η values 54
8.4. Signal shapes for different DAC values 55
8.5. Calibration Pulse 55
8.6. Comparison of signal shapes with SPICE 56
8.7. Determination of the pulse height 56
8.8. Display of the pulsed SCs 57
8.9. Conversion factors for each SC 58
8.10. Display of the barrel thickness in radiation length 58
8.11. Relation between transverse energy and ΔADC 60
8.12. Comparison of the linear response between simulation and calibration from Run 2 60
8.13. Comparison of the linear response between simulation and calibration from Run 1 61
8.14. Signals shapes in the high energy region 62
8.15. Signal shapes from large DAC values in the middle layer 62
8.16. Saturation signal shapes of Run 2 63
8.17. Comparison of the saturation between simulation and calibration data in the middle layer 63
8.18. Comparison of the saturation between simulation and calibration data in the front layer 64

B.1. Different η values of the front layer during run 1 79
B.2. Different DAC values of the middle layer during run 1 80
B.3. Different DAC values of the calibration data 81
B.4. Saturating pulse shapes with different DAC values 82
B.5. Comparing the calibration data from the middle layer with the simulation 83
B.6. Comparing the calibration data from run 2 with the simulation 83
B.7. Comparing the data at the beginning of saturation 84
B.8. Linear response plot of the middle layer 84
B.9. Comparing the linear response curves from run 1 85

List of Tables

2.1. Overview of the fermions 4

6.1. Processing of the data inside the histogram builder 35

8.1. Used sample and delay parameters for the different runs 52

A. Compilation of the Produced Code

This appendix shows the written code.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity on_fpga is
  port (
    -- clock and reset signal
    clk, rst      : in    std_logic;
    -- signal to interface the buttons on the board
    key           : in    std_logic_vector (3 downto 0);
    -- connection to the DDR3 memory
    DDR3_a        : out   std_logic_vector(14 downto 0);
    DDR3_ba       : out   std_logic_vector(2 downto 0);
    DDR3_CK_p     : out   std_logic_vector(0 downto 0);
    DDR3_CK_n     : out   std_logic_vector(0 downto 0);
    DDR3_CKE      : out   std_logic_vector(0 downto 0);
    DDR3_CS_n     : out   std_logic_vector(0 downto 0);
    DDR3_DM       : out   std_logic_vector(3 downto 0);
    DDR3_RAS_n    : out   std_logic_vector(0 downto 0);
    DDR3_CAS_n    : out   std_logic_vector(0 downto 0);
    DDR3_WE_n     : out   std_logic_vector(0 downto 0);
    DDR3_RESET_n  : out   std_logic;
    DDR3_DQ       : inout std_logic_vector(31 downto 0) := (others => '0');
    DDR3_DQS_p    : inout std_logic_vector(3 downto 0)  := (others => '0');
    DDR3_DQS_n    : inout std_logic_vector(3 downto 0)  := (others => '0');
    DDR3_ODT      : out   std_logic_vector(0 downto 0);
    DDR3_RZQ      : in    std_logic
  );
end entity on_fpga;

architecture arch of on_fpga is
  -- interfacing the DDR3 memory controller
  signal mem_fpga_address           : std_logic_vector (27 downto 0)  := (others => '0');
  signal mem_fpga_readdatavalid     : std_logic;
  signal mem_fpga_readdata          : std_logic_vector (31 downto 0);
  signal mem_fpga_writedata         : std_logic_vector (31 downto 0)  := (others => '0');
  signal mem_fpga_read              : std_logic := '0';
  signal mem_fpga_write             : std_logic := '0';
  signal mem_fpga_waitrequest_n     : std_logic;
  -- interfacing the fifos
  signal fifo_input_valid           : std_logic;
  signal fifo_input_data            : std_logic_vector(31 downto 0);
  signal fifo_input_channel         : std_logic_vector(7 downto 0);
  signal fifo_input_error           : std_logic_vector(7 downto 0);
  signal fifo_input_startofpacket   : std_logic;
  signal fifo_input_endofpacket     : std_logic;
  signal fifo_input_empty           : std_logic;
  signal fifo_input_ready           : std_logic;
  signal fifo_output_valid          : std_logic;
  signal fifo_output_data           : std_logic_vector(31 downto 0);
  signal fifo_output_channel        : std_logic_vector(7 downto 0);
  signal fifo_output_error          : std_logic_vector(7 downto 0);
  signal fifo_output_startofpacket  : std_logic;
  signal fifo_output_endofpacket    : std_logic;
  signal fifo_output_empty          : std_logic;
  signal fifo_output_ready          : std_logic;
  -- stores the addressed bin for reading and writing
  signal bin                        : std_logic_vector(27 downto 0);
  -- counter to delete memory content
  signal addr                       : integer := 0;

  -- state machine
  type states_T is (idle, cleaning, increasing, stopping);
  signal state : states_T := idle;
begin
  -- ddr3 memory controller interface
  -- read when data arrives
  mem_fpga_read <= fifo_input_valid;
  -- write when result of reading arrives or when memory is deleted
  mem_fpga_write <= '1' when (state = increasing and mem_fpga_readdatavalid = '1')
    or state = cleaning) and mem_fpga_waitrequest_n = '1' else '0';
  -- writedata = readdata + 1 or zeros, when memeory needs to be deleted
  mem_fpga_writedata <= (others => '0') when state = cleaning else
    std_logic_vector(unsigned(mem_fpga_readdata) + 1);
  -- address is inherited from the counter during deletion or shifted vector
  mem_fpga_address <= std_logic_vector(to_unsigned(addr, 28)) when state = cleaning else bin;

  -- shift the data by 6 bits and save the vector until new data arrives
  bin (27 downto 16) <= (others => '0');
  bin (15 downto 0) <= "" & std_logic_vector(unsigned(fifo_input_data(15 downto 0)) srl 6)
    when fifo_input_valid = '1' else bin (15 downto 0);

  -- fifo interface
  -- decides when new data is accepted (fifo waits until current procedure is processed)
  fifo_input_ready <= '1' when state = idle and mem_fpga_waitrequest_n = '1'
    and fifo_output_ready = '1' else '0';
  -- data is transfared from input to output
  fifo_output_data <= fifo_input_data;
  fifo_output_valid <= fifo_input_valid;

  -- process checks the currents state and changes it, if the conditions are met
  state_machine: process(clk,rst)
  begin
    if rst = '0' then
      state <= idle;
    elsif rising_edge(clk) then
      case state is
      -- wait for input (fifo: new data; key: delete memory)
      when idle =>
        if key(2) = '0' then
          state <= cleaning;
        elsif fifo_input_valid = '1' then
          state <= increasing;
        end if;
      -- write zeroes to every memory location (up to address 2000)
      when cleaning =>
        if addr = 2000 then
          state <= stopping;
        end if;
      -- waits until the result of reading command arrives (then back to idle)
      when increasing =>
        if mem_fpga_readdatavalid = '1' then
          state <= idle;
        end if;
      -- waits until the buttons are not pressed anymore (then back to idle)
      when stopping =>
        if key = "1111" then
          state <= idle;
        end if;
      end case;
    end if;
  end process;

  -- addr counter is increased every clock cycle
  addr_counter: process(clk, rst)
  begin
    if rst = '0' then
      addr <= 0;
    elsif rising_edge(clk) then
      if state = idle then
        addr <= 0;
      elsif state = cleaning and mem_fpga_waitrequest_n = '1' then
        addr <= addr + 1;
      end if;
    end if;
  end process;

  ddr3: entity work.DDR3_Qsys
  port map(
    -- this component handles the connection to the fifos and DDR3, but there are
    -- too many signals and therefore are not displayed here
  );
end arch;
```

Code example A.1: The Histogram Builder with state machine

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use work.amm.all;

entity showcase is
  generic (
    addr_width: natural := 1 -- number of registers in 2**addr_width
```

```vhdl
  );
  port (
    clk                     : in    std_logic;
    reset                   : in    std_logic;
    -- this special type combines all IPBus signals
    ipctrl_showcase_reg_mm  : inout amm_bus_t
  );
end showcase;

architecture rtl of showcase is
  -- definition of multiple registers (e.g. status, control, ...)
  type reg_array is array(2**addr_width-1 downto 0) of std_logic_vector(31 downto 0);
  signal reg  : reg_array;
  -- selected register
  signal sel  : integer;
begin
  sel <= to_integer(unsigned(ipctrl_showcase_reg_mm.address(addr_width - 1 downto 0)))
    when addr_width > 0 else 0;

  -- ipbus response:
  -- has to be asserted even after a write command (transactor needs signal)
  ipctrl_showcase_reg_mm.readdatavalid <= '1' when (ipctrl_showcase_reg_mm.write='1' or
    ipctrl_showcase_reg_mm.read='1') else '0';
  -- selected register is sent to IPBus
  ipctrl_showcase_reg_mm.readdata <= reg(sel);

  -- handle registers:
  -- at the moment only overwrites the selected register after a write command
  process(clk)
  begin
    if reset='1' then
      reg <= (others=>(others=>'0'));
    elsif rising_edge(clk) then
      if ipctrl_showcase_reg_mm.write='1' then
        reg(sel) <= ipctrl_showcase_reg_mm.writedata;
      end if;
    end if;
  end process;
end rtl;
```

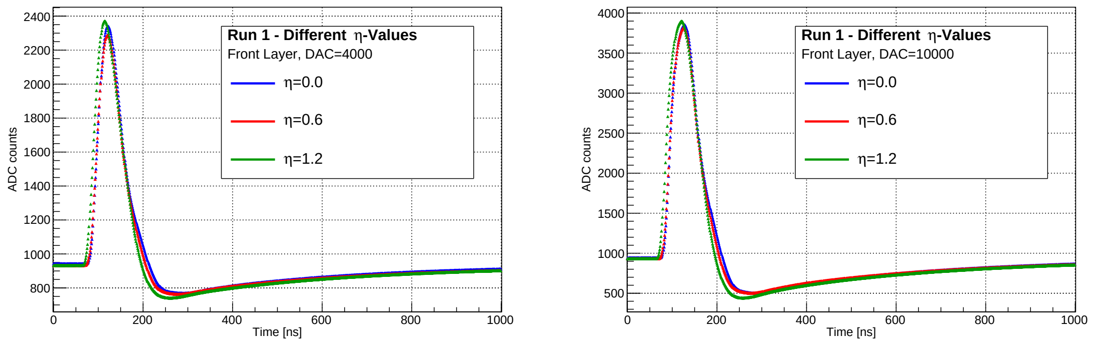
Code example A.2: The showcase module

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
use work.amm.all;

entity ipbus_ctrlreg is
  generic(
    -- same address width like above
    addr_width : natural := 1
  );
  port(
    clk: in std_logic;
    reset: in std_logic;
    ipctrl_showcase_reg_mm : inout amm_bus_t;
    -- signal to interface the buttons on the board
    trigger                     : in  std_logic_vector(2 downto 0);
    -- signal to interface the DDR3 memory controller
    mem_fpga_waitrequest_n      : in  std_logic;
    mem_fpga_beginbursttransfer : out std_logic := '0';
    mem_fpga_address            : out std_logic_vector(27 downto 0) := (others => '0');
    mem_fpga_readdatavalid      : in  std_logic;
    mem_fpga_readdata           : in  std_logic_vector(31 downto 0);
    mem_fpga_writedata          : out std_logic_vector(31 downto 0) := (others => '0');
    mem_fpga_read               : out std_logic := '0';
    mem_fpga_write              : out std_logic := '0';
    mem_fpga_burstcount         : out std_logic_vector(2 downto 0)  := "001"
  );
end ipbus_ctrlreg;

architecture rtl of ipbus_ctrlreg is
  -- this part is similar to the showcase module
  type reg_array is array(2 ** addr_width - 1 downto 0) of std_logic_vector(31 downto 0);
  signal reg                      : reg_array;
  signal sel                      : integer;
  -- address counter
  signal addr                     : integer   := 0;
  -- readdatavalid signal shifted by 1 clock cycle
  signal mem_fpga_readdatavalid_d : std_logic := '0';

  -- definition of the state machine
  type reg_state_T is (idle, writing, reading, stopping);
  signal reg_state : reg_state_T := idle;
begin
  -- address select
  sel <= to_integer(unsigned(ipctrl_showcase_reg_mm.address(addr_width - 1 downto 0)))
    when addr_width > 0 else 0;

  -- ip bus answer
  -- sends content of DDR3 when the port register is selected, otherwise the selected register
  ipctrl_showcase_reg_mm.readdata <= mem_fpga_readdata when sel = 2 else reg(sel);
  -- compare with showcase module; in addition is asserted when reading of DDR3 is completed
  ipctrl_showcase_reg_mm.readdatavalid <= '1'
    when ((ipctrl_showcase_reg_mm.write = '1' or ipctrl_showcase_reg_mm.read = '1') and sel /= 2)
      or (ipctrl_showcase_reg_mm.read = '1' and mem_fpga_readdatavalid = '1' and reg_state = reading)
    else '0';

  -- ddr3 interface
  -- assert the command depending on the current state
  mem_fpga_read <= '1' when reg_state = reading and mem_fpga_waitrequest_n = '1' else '0';
  mem_fpga_write <= '1' when reg_state = writing and mem_fpga_waitrequest_n = '1' else '0';
  -- writing is only used to fill every DDR3 location with its own address;
  mem_fpga_writedata <= std_logic_vector(to_unsigned(addr, 32));
  -- transform the address counter to vector of signals
  mem_fpga_address <= std_logic_vector(to_unsigned(addr, 28));

  -- process checks the currents state and changes it, if the conditions are met
  process(clk,reset)
  begin
    if reset = '1' then
      reg_state <= idle;
    elsif rising_edge(clk) then
      case reg_state is
      -- wait for input (port register is addressed: start reading; trigger: store addresses to DDR3)
      when idle =>
        if trigger(0) = '1' then
          reg_state <= writing;
        elsif ipctrl_showcase_reg_mm.read = '1' and sel = 2 and mem_fpga_readdatavalid = '0' then
          reg_state <= reading;
        end if;
      -- fills each location of the DDR3 with its own address
      when writing =>
        if addr = 200000000 then
          reg_state <= stopping;
        end if;
      -- readout the DDR3 as long as the read command of the IPBus is asserted
      when reading =>
        if ipctrl_showcase_reg_mm.read = '0' then
          reg_state <= idle;
        end if;
      -- wait until the button is not pressed anymore
      when stopping =>
        if trigger = "000" then
          reg_state <= idle;
        end if;
      end case;
    end if;
  end process;

  --address counter:
  process(clk,reset)
  begin
    if reset = '1' then
      addr <= 0;
      mem_fpga_readdatavalid_d <= '0';
    elsif rising_edge(clk) then
      -- address is reset when a special bit of the control register is switched
      if reg(1)(0) = '1' then
        addr <= 0;
      -- increase address during reading and writing
      elsif (reg_state = reading or reg_state = writing) and mem_fpga_waitrequest_n = '1' then
        addr <= addr + 1;
      -- decrease counter when to many addresses have been read (use shifted signal)
      elsif mem_fpga_readdatavalid_d = '1' and mem_fpga_waitrequest_n = '1' then
        addr <= addr - 1;
      end if;
      -- shift readdatavalid by one clock cycle
      mem_fpga_readdatavalid_d <= mem_fpga_readdatavalid;
    end if;
  end process;

  --registers:
  process(clk,reset)
  begin
    if reset = '1' then
      reg <= (others=>(others=>'0'));
    elsif rising_edge(clk) then
      -- overwrite selected register
      if ipctrl_showcase_reg_mm.write='1' then
        reg(sel) <= ipctrl_showcase_reg_mm.writedata;
      end if;
      -- bit to reset address; flip it back after assertion
      if reg(1)(0) = '1' then
        reg(1)(0) <= '0';
      end if;
    end if;
  end process;
end rtl;
```

Code example A.3: The histogram module for readout testing

B. Calibration Result and Comparison

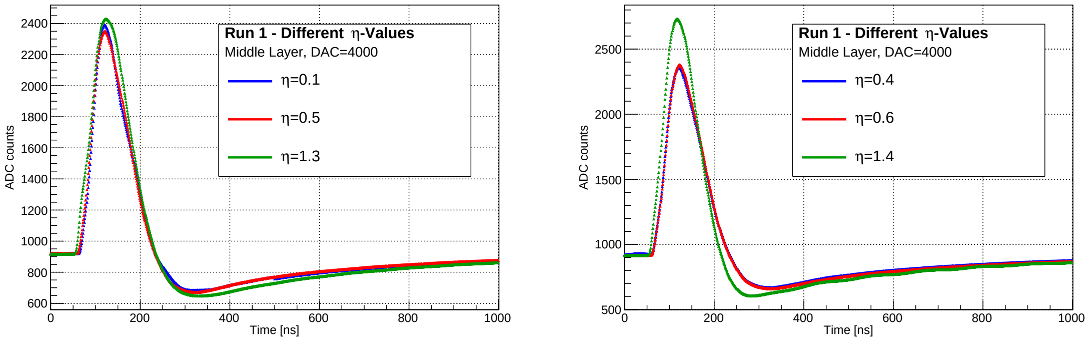
This section displays additional plots of the calibration. First the pulse shapes with different η and DAC values are shown. Examples with saturation effects are included. These are then compared with the SPICE simulation. Finally the same is done with the linear response plots.



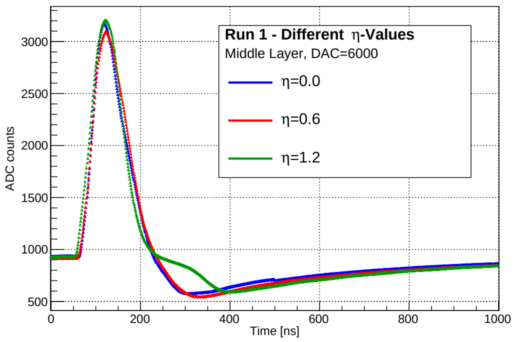
(a) Lower DAC value

(b) Higher DAC value

Figure B.1.: The plots show curves from different η values. The position on the x-axis stays the same with various DAC values. In the saturation region the pulses become wider and the peak amplitudes align to each other.

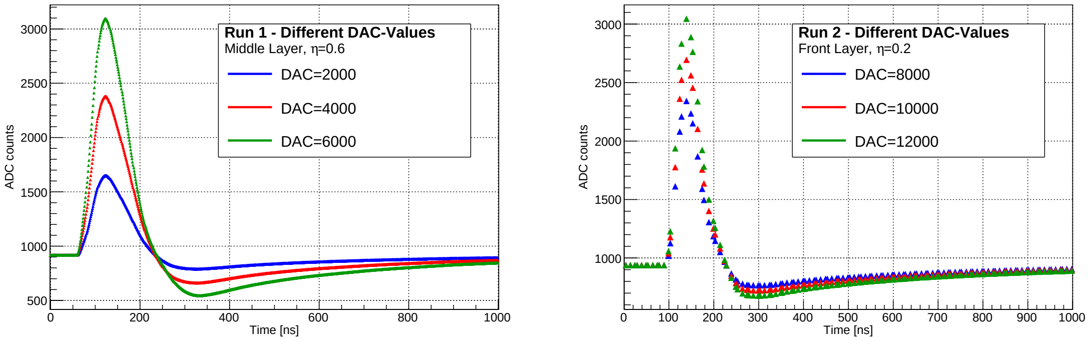


(a) Similar pulse shapes (b) Different pulse shapes



(c) Partly saturating pulse shapes

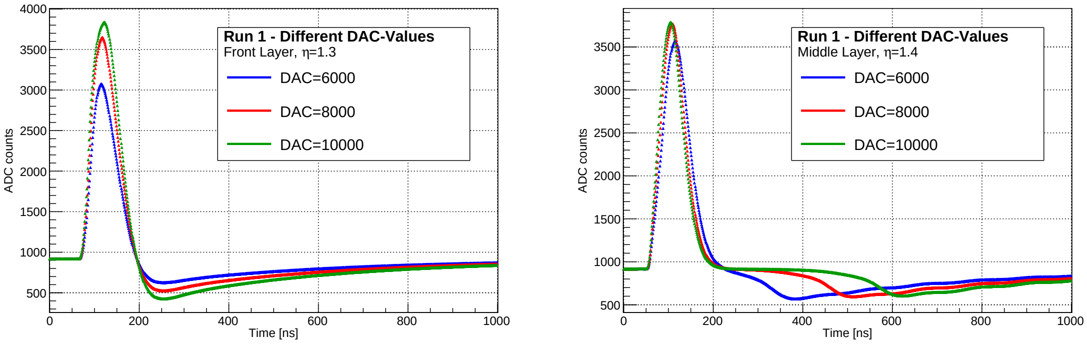
Figure B.2.: The middle layer shows similar results when various η regions are compared with each other. The upper right figure displays an interesting behavior at a certain SC. It is probable that the electrical components differ at the end of the barrel, because the absorber thickness becomes smaller in this η region. The lower plot shows an interesting DAC value. The cells with an η below 0.8 are still linear, while the green curve is already saturating.



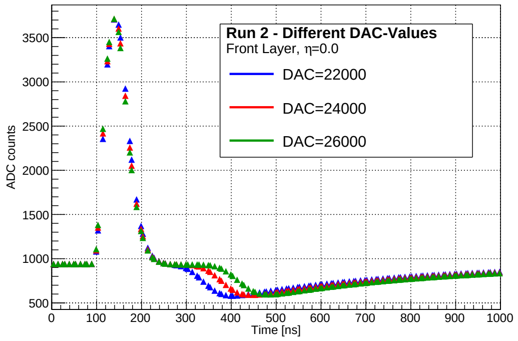
(a) Run 1 middle layer

(b) Run 2 front layer

Figure B.3.: The plots show curves from different DAC values. On the left side are pulses from the middle layer and the right side shows another set of DAC inputs of run 2.

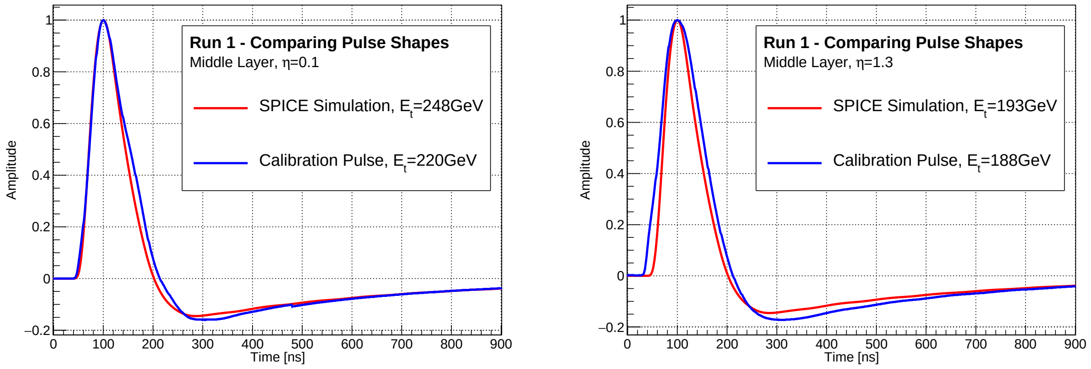


(a) Run 1 front layer (b) Run 1 middle layer



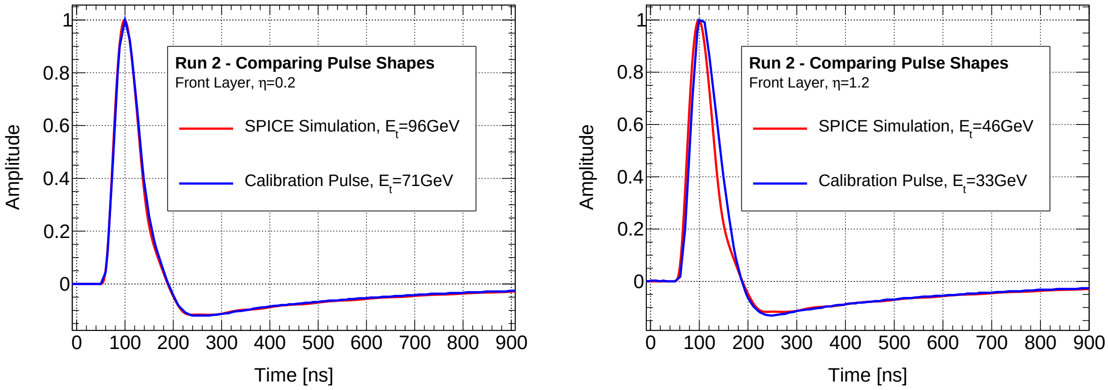
(c) Run 2 front layer

Figure B.4.: These figures show saturating pulses in a higher η region. There is still no plateau visible in the front layer. The plot of the middle layer has an interesting effect. There are small oscillations in the tail of the curve. The last figure shows the highest DAC values of run 2. The plateau becomes larger as expected.



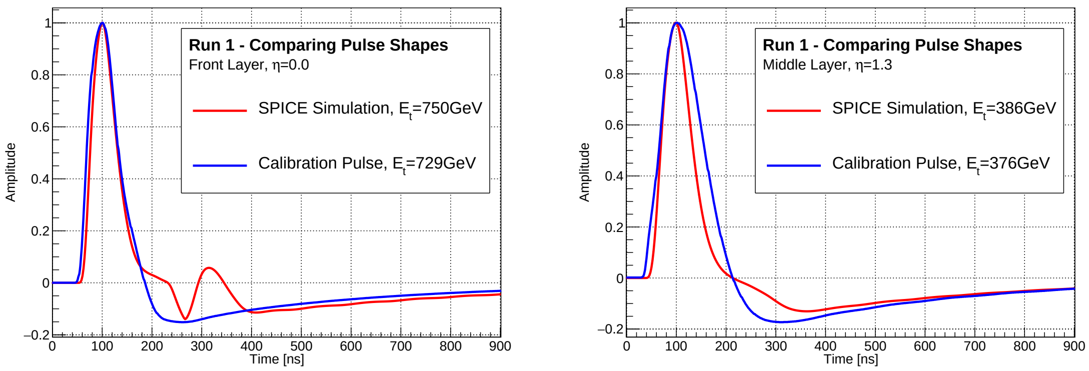
(a) Lower η region (b) Higher η region

Figure B.5.: The calibration data from the middle layer does not match as well to the simulation as the front layer. The left figure shows differences in the falling part of the curve. The little bump at 480 ns is a result of the latency problem. In the higher η region the measured pulse also rises earlier compared to the simulation, resulting in a wider curve.



(a) Lower η region (b) Higher η region

Figure B.6.: Even though the curve is not as well sampled as in run 1, the comparison between the calibration data and the simulation shows similar results as the front layer of run 1. The lower η region displays well matched plots, while there are some differences in the end of the barrel.



(a) Front layer (b) Middle layer

Figure B.7.: These plots show the beginning of the saturation region. The simulation always starts earlier and in the case of the front layer the oscillating effect is very strong, while the calibration data shows not plateau at all.

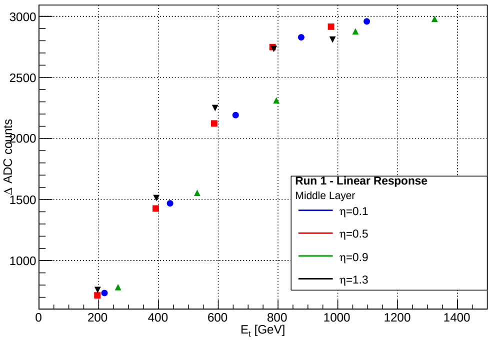
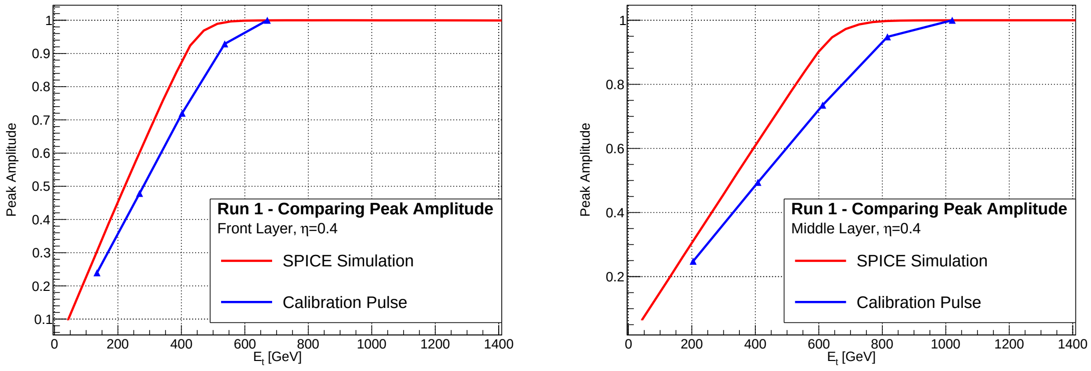


Figure B.8.: This figure shows the linear response of the middle layer. There are two effects, which are resulting the data points to be roughly at the same position. Firstly the curve moves to the left with higher η, because the transverse energy is plotted at the x-axis. Secondly the jump in the conversion factors at $\eta = 0.8$ moves the curve again to higher energies.



(a) Front layer (b) Middle layer

Figure B.9.: These plots show the linear responses of the other lower η regions. In both cases the saturation starts a little bit earlier in the simulation. A sophisticated statement is not possible because of the lack of data points in this run.

Bibliography

[1] A. Pich, *Status after the first LHC run: Looking for new directions in the physics landscape*, Tech. Rep. arXiv:1507.01250. IFIC-15-34. FTUV-15-0705, Jul, 2015. `https://cds.cern.ch/record/2031542`.

[2] B. Dutta, *Dark Matter Searches at Accelerator Facilities*, Tech. Rep. arXiv:1403.6217, Mar, 2014. `https://cds.cern.ch/record/1689941`.

[3] *The matter-antimatter asymmetry problem*, `http://home.cern/topics/antimatter/matter-antimatter-asymmetry-problem`.

[4] T. Berry, *Searches for gravitational effects at the TeV scale with the ATLAS detector*, Tech. Rep. ATL-COM-PHYS-2013-1585, CERN, Geneva, Dec, 2013. `https://cds.cern.ch/record/1632902`.

[5] V. D. Shiltsev, *High Energy Particle CCollider: Past 20 Years, Next 20 Years and Beyond*, `http://arxiv.org/ftp/arxiv/papers/1205/1205.3087.pdf`.

[6] H. Wiedemann, *Particle Accelerator Physics.* Springer, 2007.

[7] L. Evans and P. Bryant, *LHC Machine*, Journal of Instrumentation **3** (2008) no. 08, S08001. `http://stacks.iop.org/1748-0221/3/i=08/a=S08001`.

[8] A. Djouadi, *The anatomy of electroweak symmetry breaking: Tome I: The Higgs boson in the Standard Model*, Physics Reports **457** (2008) no. 1–4, 1 – 216. `http://www.sciencedirect.com/science/article/pii/S0370157307004334`.

[9] S. L. Glashow, *Partial Symmetries of Weak Interactions*, Nucl. Phys. **22** (1961) 579–588. `http://inspirehep.net/record/4328`.

[10] A. Salam and J. C. Ward, *ON A GAUGE THEORY OF ELEMENTARY INTERACTIONS*, Nuovo Cim. **19** (1961) 165–170. `http://inspirehep.net/record/12287?ln=de`.

[11] S. Weinberg, *A Model of Leptons*, Phys. Rev. Lett. **19** (Nov, 1967) 1264–1266. `http://link.aps.org/doi/10.1103/PhysRevLett.19.1264`.

[12] D. Griffiths, *Introduction to Elementary Particles.* WIlEY-VCH, 2008.

[13] *Neutrino Oscillations*, `https://www.nobelprize.org/nobel_prizes/physics/laureates/2015/advanced-physicsprize2015.pdf`.

[14] P. R. M. Drees, R. Godbole, *Theory and Phenomenology of Sparticles.* World Scientific Publishing Company, 2004.
`http://www.worldscientific.com/worldscibooks/10.1142/4001`.

[15] G. Belanger, *Dark matter and the LHC*, tech. rep., CERN, Jul, 2009.
`https://cds.cern.ch/record/1188295`.

[16] *The High-Luminosity LHC*, `http://home.cern/topics/high-luminosity-lhc`.

[17] *Summary plots from the ATLAS Standard Model physics group*,
`SummaryplotsfromtheATLASStandardModelphysicsgroup`.

[18] C. Collaboration, *Missing transverse energy performance of the CMS detector*, 2011.
`http://arxiv.org/pdf/1106.5048v1.pdf`.

[19] J. Haffner, *The CERN accelerator complex*, .
`http://cds.cern.ch/record/1621894`.

[20] *Radiofrequency cavities*,
`http://home.cern/about/engineering/radiofrequency-cavities`.

[21] LHCb Collaboration, A. A. Alves et al., *The LHCb Detector at the LHC*, J. Instrum. **3** (2008) no. LHCb-DP-2008-001. CERN-LHCb-DP-2008-001, S08005.
`https://cds.cern.ch/record/1129809`.

[22] T. A. Collaboration, *The ALICE experiment at the CERN LHC*, Journal of Instrumentation **3** (2008) no. 08, S08002.
`http://stacks.iop.org/1748-0221/3/i=08/a=S08002`.

[23] T. C. Collaboration, *The CMS experiment at the CERN LHC*, Journal of Instrumentation **3** (2008) no. 08, S08004.
`http://stacks.iop.org/1748-0221/3/i=08/a=S08004`.

[24] T. A. Collaboration, G. Aad, et al., *The ATLAS Experiment at the CERN Large Hadron Collider*, Journal of Instrumentation **3** (2008) no. 08, S08003.
`http://stacks.iop.org/1748-0221/3/i=08/a=S08003`.

[25] A. S. Cornell, *Some theories beyond the Standard Model*, Tech. Rep. arXiv:1506.05602, Jun, 2015. `https://cds.cern.ch/record/2028622`.

[26] J. Pequenao, *Computer generated image of the whole ATLAS detector*, Mar, 2008.

[27] *Illustration of particle detection in the subsystems of the ATLAS detector*, 2007. `http://www.interactions.org/cms/?pid=2100&image_no=CE0155`.

[28] C. W. Fabjan and F. Gianotti, *Calorimetry for Particle Physics*, Rev. Mod. Phys. **75** (Oct, 2003) 1243–1286. 96 p. `https://cds.cern.ch/record/692252`.

[29] D. Green, *The Physics of Particle Detectors.* Cambridge University Press, 2000.

[30] J. Pequenao, *Computer generated image of the ATLAS Liquid Argon*, Mar, 2008.

[31] *The first LHC protons run ends with new milestone*, Cern press release, December, 2012. `http://press.web.cern.ch/press-releases/2012/12/first-lhc-protons-run-ends-new-milestone`.

[32] *The HL-LHC project*, Cern. `http://hilumilhc.web.cern.ch/about/hl-lhc-project`.

[33] *CERN releases analysis of LHC incident*, Cern press release, October, 2008. `http://press.web.cern.ch/press-releases/2008/10/cern-releases-analysis-lhc-incident`.

[34] M. C. Aleksa, W. P. Cleland, Y. T. Enari, M. V. Fincke-Keeler, L. C. Hervas, F. B. Lanni, S. O. Majewski, C. V. Marino, and I. L. Wingerter-Seez, *ATLAS Liquid Argon Calorimeter Phase-I Upgrade Technical Design Report*, Tech. Rep. CERN-LHCC-2013-017. ATLAS-TDR-022, CERN, Geneva, Sep, 2013. `https://cds.cern.ch/record/1602230`. Final version presented to December 2013 LHCC.

[35] M. Capeans, G. Darbo, K. Einsweiller, M. Elsing, T. Flick, M. Garcia-Sciveres, C. Gemme, H. Pernegger, O. Rohne, and R. Vuillermet, *ATLAS Insertable B-Layer Technical Design Report*, Tech. Rep. CERN-LHCC-2010-013. ATLAS-TDR-19, CERN, Geneva, Sep, 2010. `https://cds.cern.ch/record/1291633`.

[36] ATLAS Collaboration, T. A. Collaboration, *ATLAS Phase-II Upgrade Scoping Document*, Tech. Rep. CERN-LHCC-2015-020. LHCC-G-166, CERN, Geneva, Sep, 2015. `https://cds.cern.ch/record/2055248`.

[37] N. Checillot, B. Dinkespiler, N. Dumont-Dayot, Y. Enari, R. Hentges, K. Johns, I. Wingerter, and V. Zhulanov, *ATLAS LAr Calorimeter trigger electronics phase I upgrade: LATOME Firmware Specifications*, September, 2015.

[38] S. Staerz and A. Straessner, *Energy Reconstruction and high-speed Data Transmission with FPGAs for the Upgrade of the ATLAS Liquid Argon Calorimeter at LHC.* PhD thesis, Dresden, Tech. U., Feb, 2015. `https://cds.cern.ch/record/2030122`. Presented 19 May 2015.

[39] *CACTUS*, `https://svnweb.cern.ch/trac/cactus`.

[40] *Avalon Interface Specifications*, `https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/manual/mnl_avalon_spec.pdf`.

[41] *Cyclon V Device Datasheet*, 2015. `https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/cyclone-v/cv_51002.pdf`.

[42] *SoCKit - the Development Kit for New SoC Device*, `http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=816`.

[43] *SoCKit User Manual*, .

[44] *Qsys*, `https://www.altera.com/products/design-software/fpga-design/quartus-prime/features/qts-qsys.html`.

[45] *Ethernet Working Group*, `http://www.ieee802.org/3/`.

[46] B. Dinklespiler, N. Dumont-Dayot, Y. Enari, R. Hentges, K. Johns, and V. Zhulanov, *ATLAS LAr Calorimeter AMC Firmware Specifications*, March, 2015.

[47] L. Fischer and Y. Pyatnychko, *FPGA Design for DDR3 Memory*, Worcester polytechnic institute, March, 2012. `https://www.wpi.edu/Pubs/E-project/Available/E-project-031212-183607/unrestricted/FPGA_Design_for_DDR3_Memory.pdf`.

[48] *SigSignal II with VHDL Designs*, `ftp://ftp.altera.com/up/pub/Tutorials/DE2/Digital_Logic/tut_signaltapII_vhdlDE2.pdf`.

[49] *Notes on Firmware Implementation of an IPbus SoC Bus*, May, 2012. `https://svnweb.cern.ch/trac/cactus/browser/trunk/doc/IPbus_firmware_notes.pdf`.

[50] *WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores*, `http://cdn.opencores.org/downloads/wbspec_b3.pdf`.

[51] R. Huebscher, *Aufbau und Datenanalyse mit dem Demonstrationssystem des ATLAS LAr Kalorimetertriggers*, Master's thesis, TU Dresden, 2015. `http://iktp.tu-dresden.de/IKTP/pub/15/Masterarbeit_Rico_Huebscher.pdf`.

[52] L. W. Nagel and D. Pederson, *SPICE (Simulation Program with Integrated Circuit Emphasis)*, Tech. Rep. UCB/ERL M382, EECS Department, University of California, Berkeley, Apr, 1973. `http://www.eecs.berkeley.edu/Pubs/TechRpts/1973/22871.html`.

[53] N. Dumont Dayot, *Performance of the Demonstrator System for the Phase-I Upgrade of the Trigger Readout Electronics of the ATLAS Liquid Argon Calorimeter*, .

[54] on behalf of the ATLAS collaboration, N. Nikiforou, *Performance of the ATLAS Liquid Argon Calorimeter after three years of LHC operation and plans for a future upgrade*, Tech. Rep. arXiv:1306.6756, Jun, 2013. `https://cds.cern.ch/record/1558820`.

[55] ATLAS Collaboration, G. Aad et al., *Electron and photon energy calibration with the ATLAS detector using LHC Run 1 data*, Eur. Phys. J. C (Jul, 2014) 74. 51 p. `https://cds.cern.ch/record/1744017`.

[56] Z. Meng, *Performance of the ATLAS Liquid Argon Calorimeter*, Tech. Rep. ATL-LARG-PROC-2010-017, CERN, Geneva, Nov, 2010. `https://cds.cern.ch/record/1304279`.

[57] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing.* Pearson Prentice Hall, 2007.

Danksagung

Ich möchte mich an dieser Stelle ganz herzlich bei allen bedanken, die mich während des Studiums und der Masterarbeit unterstützt haben.

Ein besonderer Dank geht an Prof. Dr. Arno Straessner, welcher immer ein offenes Ohr für Fragen hatte und bei Problemen neue Anregungen gab.

Desweiteren möchte ich mich bei meinem Betreuer Rainer Hentges und der Elektronikgruppe des IKTP bedanken. Die interessanten Mittags-Gespräche und gute Arbeitsatmosphäre haben die Masterarbeit sehr positiv beeinflusst.

Vielen Dank an die fleißigen Korrekturleser Maximilian Hils, Rainer Hentges, Rico Hübscher und Johannes Krause. Ohne euch wäre diese Arbeit fast unlesbar.

Ich möchtige mich auch bei Robert Wolf bedanken. Vorallem im zweiten Teil meiner Arbeit habe ich ihn oft zurate gezogen und er konnte mir immer weiterhelfen.

Ein großer Dank geht auch an alle meine Freunde. Ihr habt die letzten Jahre zu den besten meines Lebens gemacht.

Der letzte und wichtigste Dank geht an meine Familie. Ihr wart immer an meiner Seite und ich bin unbeschreiblich froh euch zu haben.

Versicherung

Hiermit versichere ich, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der von mir angegebenen Quellen angefertigt zu haben. Alle aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche gekennzeichnet. Die Arbeit wurde noch keiner Prüfungsbehörde in gleicher oder ähnlicher Form vorgelegt.

Dresden, 29. März 2016

Philipp Horn