Department of Physics and Astronomy
Heidelberg University

Bachelor Thesis in Physics
submitted by

Anna Dunz

born in Stuttgart (Germany)

2023

Implementing a Data Generation Framework for the ATLAS Tile PreProcessor

This Bachelor Thesis has been carried out by Anna Dunz at the
Kirchhoff Institute of Physics in Heidelberg
under the supervision of
Prof. Dr. Hans-Christian Schultz-Coulon

Abstract

In the wake of the High-Luminosity Upgrade of the Large Hadron Collider (LHC), which aims to increase the collider's luminosity, the ATLAS detector will receive its Phase-II Upgrade. As part of this upgrade, the entire front- and back-end electronics of the ATLAS Tile Calorimeter will be replaced. The focus of this thesis is to design a way of testing the new trigger and data acquisition architecture and in particular the Tile PreProcessor (TilePPr) module. For this purpose, data is injected into the module. After it has been processed by the electronics, the output data is recovered and compared to the input data. This way, valuable information will be gained about the behavior and limitations of the new trigger chain. In this work, the necessary firmware to control the data flow is developed on a Field Programmable Gate Array (FPGA). The firmware is able to transmit and receive data at a rate of 9.6 Gbps, fully exhausting the bandwidth of TilePPr. It can be used on a separate board that functions as a data injector or directly on the FPGAs of the TilePPr module. Furthermore, software is designed to generate test data. The software is able to provide a variety of artificial detector signals, to represent signals coming from single channels or entire detector areas. Finally, the functionality of the firmware in combination with the generated patterns is verified.

Zusammenfassung

Im Zuge des High-Luminosity Upgrades des Large Hadron Colliders (LHC), mit welchem die Luminosität des Teilchenbeschleunigers erhöht werden soll, erhält der ATLAS Detektor sein Phase-II Upgrade. Als Teil dieses Upgrades wird die Front- und Backend-Elektronik des Tile Kalorimeters ersetzt. In dieser Arbeit geht es um die Entwicklung einer Methode, die neue Triggerelektronik, insbesondere das Tile PreProcessor (TilePPr) Modul, zu testen. Dazu sollen Daten in das Modul eingegeben werden. Wenn nach Durchlaufen der Elektronik die Daten wieder ausgelesen und mit den anfänglichen verglichen werden, können so Informationen über Verhalten, aber auch Grenzen der neuen Triggerelektronik gewonnen werden. Die zur Regelung der Datenübertragung nötige Firmware wurde auf einem Field Programmable Gate Array (FPGA) entwickelt. Die Firmware ist in der Lage, Daten mit einer Rate von 9.6 Gbps auszugeben und einzulesen und schöpft somit die Bandbreite des TilePPr voll aus. Sie kann auf einem seperaten Board verwendet werden, welches Daten in das Modul einspeist, oder direkt auf den in dem TilePPr verbauten FPGAs ausgeführt werden. Des Weiteren wurde Software zur Erstellung von Testdaten entwickelt. Die Software kann verschiedene künstliche Detektorsignale generieren, die Signale von einzelnen Kanälen oder ganzen Bereichen des Kalorimeters repräsentieren. Abschließend wird die Funktionalität der Firmware in Kombination mit den generierten Daten überprüft.

Contents

1 | Introduction

The High-Luminosity (HL) Upgrade of the Large Hadron Collider (LHC) at CERN aims to increase the instantaneous luminosity from the current peak of $L = 2.26 \times 10^{34}$ cm^{-2}s^{-1} to $L = 7.5 \times 10^{34}$ cm^{-2}s^{-1}. This unprecedented luminosity will offer a wealth of opportunities for new research. The upgrade will present a big challenge for the detectors located at the LHC. Bunches of particles collide every 25 ns and with the higher luminosity, the number of collisions per bunch crossing is estimated to be about $\langle \mu \rangle = 200$. This creates a very demanding environment for trigger and data acquisition systems which need to process and filter the resulting data and ensure that relevant information is saved.

In this setting, processing needs to be extremely fast. Because not all data can be stored, quick decisions about what information is worth keeping for analysis are essential. At the same time, the electronics need to comply with strict latency requirements. Any mistakes in processing might result in the loss of interesting data.

One aspect of building hardware that can fulfill these requirements is the use of Field Programmable Gate Arrays (FPGA). FPGAs are a type of integrated circuit for which the logic is defined and reconfigured by firmware, which makes them very flexible. Another one of their strengths lies in their ability to easily run processes in parallel. This makes FPGAs a vital technology in hardware systems.

For the ATLAS detector, a number of changes are planned, before the HL-LHC starts its operation. Among them is the replacement of the readout electronics of the ATLAS Tile Calorimeter. The new architecture is designed to trigger at higher rates, but also to be more radiation hard and more reliable by means of redundancy.

The TilePPr module is a part of the new trigger chain. Its main purposes are calibration and reconstruction of the energy. TilePPr also provides the interface for transferring data from and transmitting control signals to the front-end electronics of the calorimeter. The module makes use of multiple FPGAs to help accelerate signal processing.

The objective of this thesis is to investigate a way to test the new trigger chain, and this module in particular, by injecting artificial data into it. Reading the data out after it has passed through the chain could help validate the new electronics and provide useful insight into their limitations. For this, firmware, which controls the data flow, and software to provide a variety of test data are developed.

A description of the LHC and the ATLAS detector is given in Chapter 2. The hardware and the control protocol used in this work are introduced in Chapter 3. In Chapter 4, the development of the firmware is described and Chapter 5 gives an overview of the created software. Chapter 6 discusses tests of the firmware. Finally, a summary is presented in Chapter 7.

2 | The LHC and the ATLAS Detector

2.1 The Standard Model of Particle Physics

The Standard Model of particle physics [1] is a theory that describes the elementary particles and their interactions. Three of the four fundamental forces, the electromagnetic, strong and weak forces, have been incorporated.

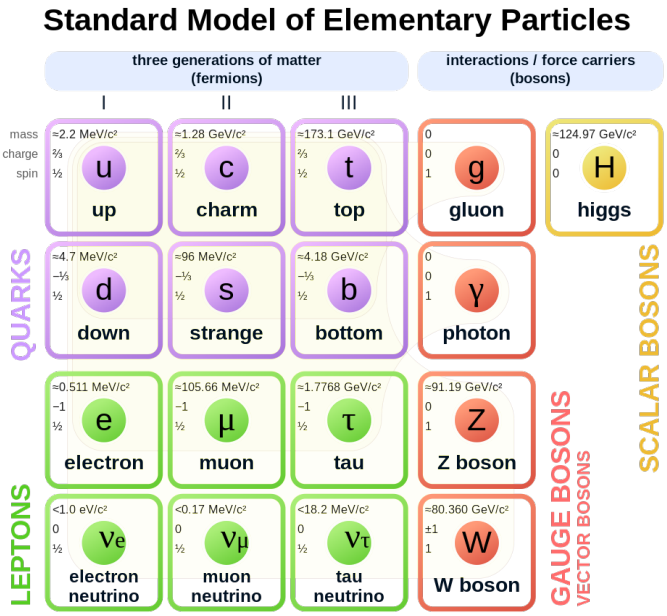


Figure 2.1: Elementary particles in the Standard Model [2]. For each of the matter particles, there is a corresponding antiparticle.

Figure 2.1 shows an overview of the elementary particles. The first group, colored in green and purple, are the twelve matter particles. They are all spin-$\frac{1}{2}$ fermions, divided into six quarks and six leptons.

The leptons are made up of the electron, the muon (μ) and the tau (τ) and their associated neutrinos. The neutrinos do not carry charge or color, meaning they can only interact via weak interactions. The other leptons carry an integer charge and can therefore interact electromagnetically as well. All of the leptons have antiparticles that are charged oppositely.

The six different quarks (up, down, strange, charm, top and bottom) carry both a fractional charge and color. They couple to all three of the forces described by the Standard Model.

An antiquark carries both the opposite charge as well as the corresponding anticolor charge compared to the corresponding quark.

Both kinds of matter particles can be divided into three generations of two particles each, with masses increasing from first to third generations. All ordinary matter is made of particles of the first generation, e.g. atomic nuclei consist of protons and neutrons, which are made of up and down type quarks, and electrons. The particles of second and third generation typically decay quickly.

The second type of particles are the force carriers. In the Standard Model, the three types of interactions are mediated by gauge bosons. The electromagnetic force is carried by the massless photon. The gluons are responsible for strong interactions, are massless as well and carry a color charge. Weak interactions are mediated by the $W^{\pm}$ and Z^0 gauge bosons, which are massive and, in the case of the $W^{\pm}$, electrically charged with a charge of -1 or $+1$.

The last of the elementary particles, the Higgs boson, is an excitation of the Higgs field. Interaction with the Higgs field is what gives mass to the massive elementary particles.

Since its formulation, experimental evidence has given strong credibility to the Standard Model, in particular the discovery of theoretically predicted particles. The Z^0 and $W^{\pm}$ bosons were discovered at CERN in 1982 and 1983 [3], the top quark at Fermilab in 1995 [4]. In 2012, both ATLAS and CMS reported the discovery of a particle with the properties expected of the Higgs boson [5, 6].

However, some observable phenomena are not explained by the Standard Model. Some examples of this include matter-antimatter asymmetry and neutrino oscillations. Nevertheless, the Standard Model has accurately predicted and validated a large number of experimental results.

2.2 The Large Hadron Collider

The Large Hadron Collider (LHC) [7] is the worlds largest particle accelerator. It is designed to collide protons or heavy ions. The collider was built at CERN in a 26.7 km circular tunnel near Geneva. Construction of the LHC finished in 2008. For proton collisions, it is currently able to reach center-of-mass energies of $\sqrt{s} = 13.6$ TeV and a peak instantaneous luminosity of $L = 2.26 \times 10^{34}$ cm^{-2}s^{-1} [8].

The LHC has two beam pipes with counter-rotating beams of protons. The proton beams are structured into bunches, each containing up to $N_b = 1.15 \times 10^{11}$ particles. Particle collisions happen at so-called interaction points, where the two beams share a beam pipe. The LHC has four of these interaction points, as depicted in Figure 2.2. This is where the four major experiments, ATLAS [9], ALICE, CMS and LHCb are located.

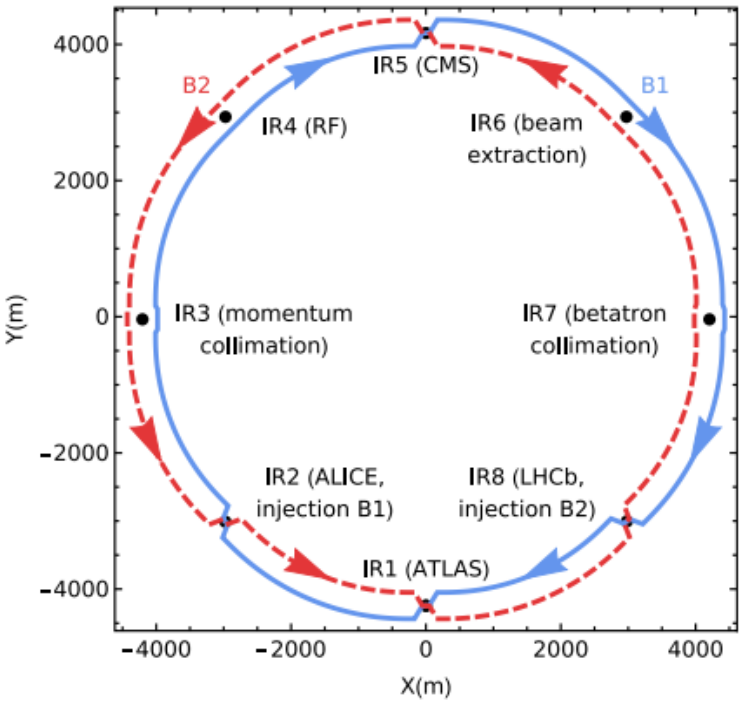


Figure 2.2: Schematic layout of the LHC with exaggerated beam pipe separation [10].

The instantaneous luminosity of a collider, which is the rate of possible collisions per area, is defined as

$$L = \frac{f_{orbit} \cdot n_b \cdot N_1 \cdot N_2}{4\pi \cdot \sigma_1 \cdot \sigma_2} \tag{2.1}$$

Here N_1 and N_2 denote the respective number of particles in the colliding bunches, σ_1 and σ_2 their spread and n_b the number of filled bunches.

The frequency of a bunch crossing (BC) is $f_{BC} = 40.07897$ MHz, meaning bunches collide every 25 ns. The orbit frequency, which is the frequency of a full rotation around the collider, is $f_{orbit} = 11.2455$ kHz [11]. By dividing the frequency of an orbit by the bunch frequency, the number of possible bunch crossings in an orbit can be obtained.

$$\frac{f_{BC}}{f_{orbit}} \approx 3564 \tag{2.2}$$

Bunch crossings are assigned numbers from 0 to 3563 that are called bunch crossing identifiers (BCID).

It should be noted that due to technical constraints, not all bunch slots are actually filled with protons. The LHC operates with a maximum number of filled bunches of $n_b = 2,808$. This further means that not in all possible bunch crossing slots, collisions of protons occur.

Currently, there are approximately $\langle\mu\rangle = 51.2$ [12] proton-proton collisions per bunch crossing. The collisions which occur at the same time as the collision of interest, as well

as some other effects that result in additional detector signals, are referred to as in-time pile-up [13]. Out-of-time pile-up is a result of protons colliding immediately before and after the collision of interest, for example in the previous bunch crossing.

2.2.1 The High Luminosity Upgrade

The HL-LHC [14, 15] is an upgrade of the current LHC that will drastically increase the luminosity of the accelerator. It is planned to be in operation by beginning of 2029.

The HL-LHC aims to achieve an instantaneous luminosity of up to $L = 7.5 \times 10^{34}$ cm^{-2}s^{-1}, which means that the number of interactions per bunch crossing will increase from the current average of $\langle\mu\rangle = 51.2$ to about $\langle\mu\rangle = 200$. The integrated luminosity (the integral with respect to time of the instantaneous luminosity) over the runtime of the HL-LHC is estimated to be $3000 - 4000$ fb^{-1}, compared to about 300 fb^{-1} at the end of the runtime of the current LHC.

The increased number of collisions allows for further research into known phenomena as well as potential observation of rarer, not yet observable processes. However, it creates a much more challenging environment for the detectors because pile-up and radiation will be significantly higher.

2.3 The ATLAS Detector

The ATLAS detector [9] is one of two general-purpose detectors at the LHC. It was designed to be sensitive to a variety of different physics processes at the uniquely high luminosity that the LHC could deliver [16]. This promised huge potential for the discovery of new physics.

A cut-away view of the full detector is pictured in Figure 2.3. The first point of detection is the inner detector, which measures charge, direction and momentum of charged particles. Three superconducting toroid magnets and a thin solenoid exert a 2 T magnetic field on the inner detector. For hadronic calorimetry, the detector is equipped with a scintillating-tile calorimeter. Liquid argon sampling calorimeters are employed at multiple parts of the detector. LAr technology is used in hadronic end-caps and forward calorimeters, as well as the electromagnetic calorimeter. Muons are detected by three layers of tracking chambers surrounding the calorimeters.

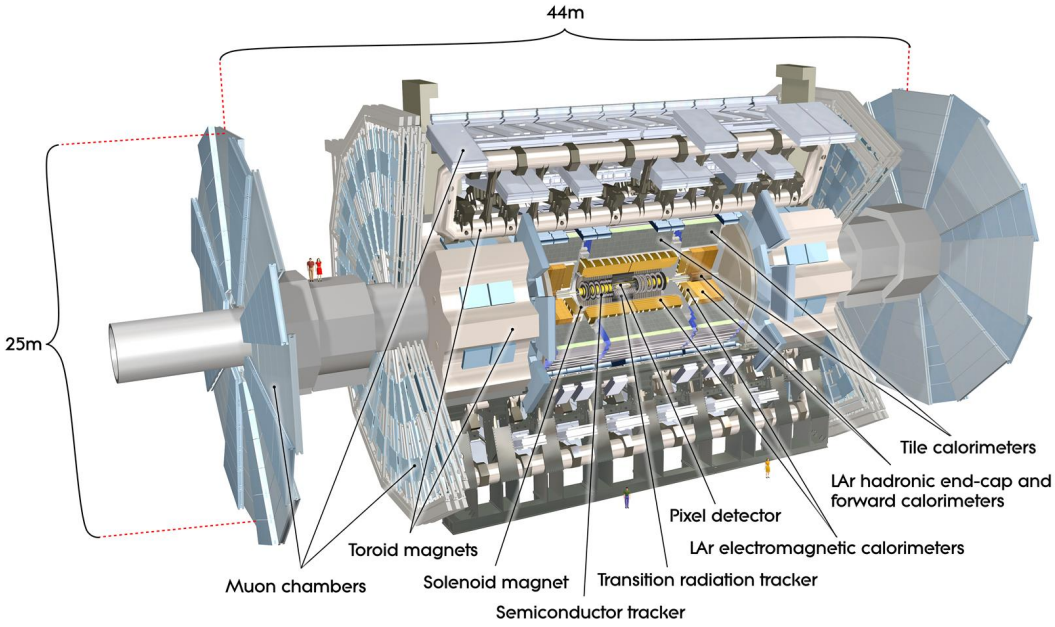


Figure 2.3: Cut-away view of the ATLAS detector [9].

The coordinate system typically used to describe ATLAS defines the point of origin as the interaction point and the z-axis as the beam direction. The azimuthal angle ϕ is measured around the beam axis. Instead of the polar angle θ, the pseudorapidity η is commonly used. It is defined as

$$\eta = -\ln \tan \frac{\theta}{2}. \tag{2.3}$$

Operation of ATLAS began in 2010, during the first data-taking period of the LHC. At that time, the collider delivered a maximum center-of-mass energy of $\sqrt{s} = 8$ TeV and a maximum luminosity of $L = 7.7 \times 10^{33}$ cm^{-2}s^{-1} [17, 18].

In 2012, the LHC was shut down for maintenance. Data taking commenced in 2015, this time at a luminosity of up to $L = 1.9 \times 10^{34}$ cm^{-2}s^{-1} and a center-of-mass energy of $\sqrt{s} = 13$ TeV. It lasted until 2018, when the LHC was shut down again until 2022 [19, 20].

The Phase-I Upgrade of ATLAS was installed during this second shutdown and is currently in operation. The third period of data-taking is supposed to last until 2025, at which point the High Luminosity Upgrade of the LHC will begin [14].

To deal with the challenges presented by HL-LHC, the ATLAS detector will undergo a number of changes during the collider shutdown [21], called the Phase-II upgrade. It includes the replacement of the current tracking system, implementation of a new trigger architecture, as well as new read-out electronics for the muon detector and calorimeters.

2.4 The ATLAS Tile Calorimeter

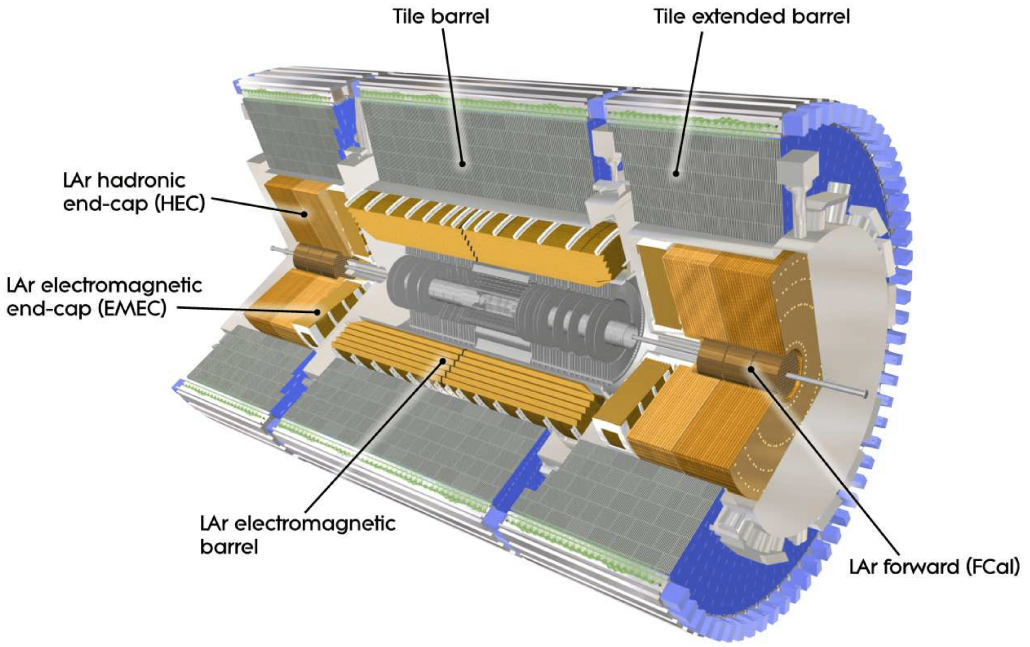


Figure 2.4: Computer Generated image of the ATLAS calorimeter [22].

The ATLAS Tile Calorimeter (Tile) [23], which can be seen in Figure 2.4, is the central part of the hadronic calorimeter of the ATLAS detector and covers an area of $|\eta| < 1.7$ and $\phi = 2\pi$. It consists of four barrels, two central long barrels (labeled "Tile barrel") and two end-cap barrels (labeled "Tile extended barrel"). Tile is a sampling calorimeter, with steel absorbers and plastic scintillators as an active material. It is segmented into 4670 cells, each of which is read out by two Photo-Multiplier Tubes (PMT).

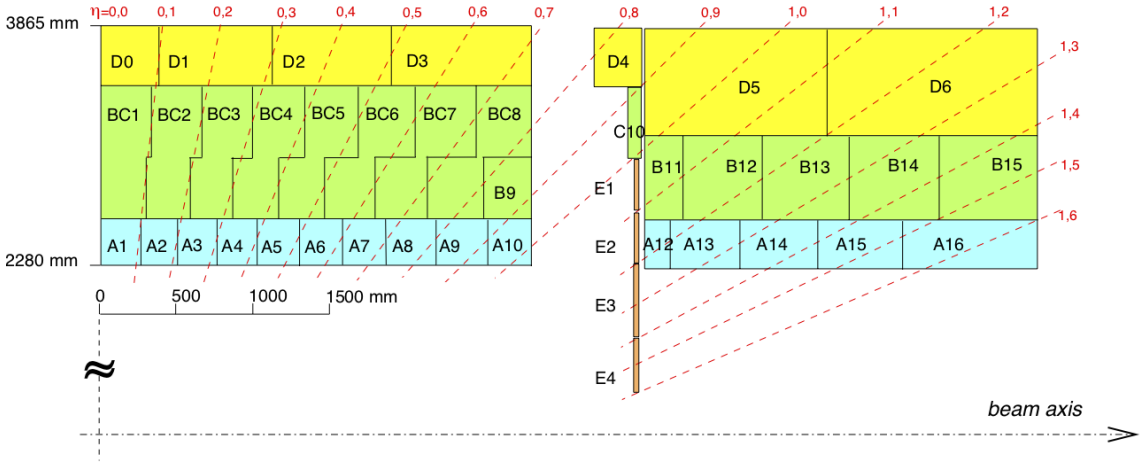


Figure 2.5: Map of Tile cells, the central barrel segmentation is shown on the left and the end cap barrel segmentation on the right [24]

The calorimeter is segmented radially into three layers, labeled "A", "BC" and "D", which are shown in Figure 2.5. The "E" cells in the middle are the "gap" scintillators, which cover the region between central and end-cap barrels and quantify the energy escaping the electromagnetic calorimeter.

For the trigger, cells are grouped together as "trigger towers". In Figure 2.5, this is illustrated by the dashed lines. This results in a granularity of 0.1×0.1 in $\Delta\eta \times \Delta\phi$.

2.4.1 Upgrade of the Tile Electronics

For the HL-LHC, the entire front- and back-end electronics of the Tile Calorimeter will be replaced to deal with the conditions resulting from the higher luminosity. The new electronics are designed to be more reliable and more resistant to radiation. The change is also made necessary by the decreasing availability of replacement components for the old system, which was designed more than 25 years ago [23].

The new design for the electronics [25] is pictured schematically in Figure 2.6.

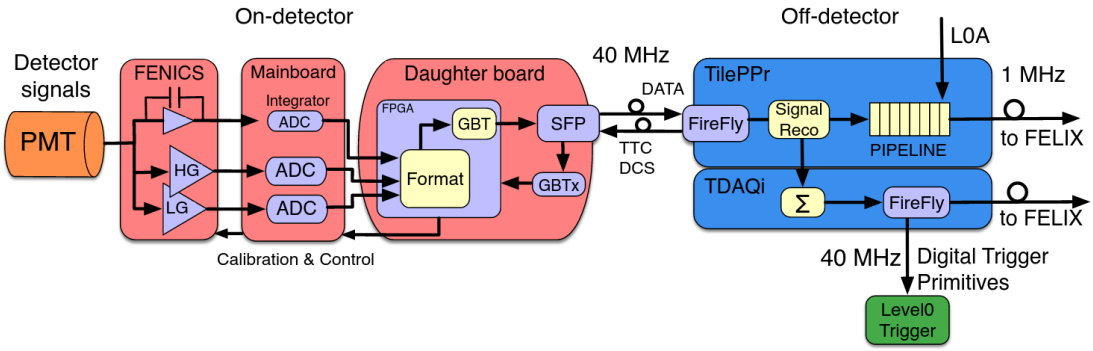


Figure 2.6: Block diagram of the HL-LHC Tile read-out electronics [26].

In the on-detector electronics, the signal from the PMTs is digitized. Every 25 ns, the signal is transferred to the off-detector electronics, which are in a room about 100 m from the detector.

The new trigger and data acquisition architecture will have a trigger rate of about 1 MHz and a 40 MHz digital trigger. It will be able to send all of the Tile cell information to the trigger systems in digitized form. For this reason, bandwidth to the back-end electronics will be significantly increased from 256 links with an 800 Mbps bandwidth to 2048 links, each with a bandwidth of 9.6 Gbps [23]. The overall latency will be increased as well.

To achieve more reliability, the new design is split into more independent units than before. This is a result of the experience with the older system, where a failure in cooling or power distribution could render large parts of the detector unusable.

2.4.2 The Tile PreProcessor

The Tile PreProcessor [27, 28] is a key part of the new read-out electronics for the upgraded Tile Calorimeter for HL-LHC. It reconstructs the energy using the digital ADC samples, performs calibration and transfers the data from the front-end electronics to the global data acquisition and trigger systems. It also provides an interface to pass control signals onto the on-detector electronics. Each of the TilePPr modules covers an area of 1.6×0.4 in $\Delta\eta \times \Delta\phi$, meaning $2 \times 16 = 32$ modules are needed to cover all of Tile.

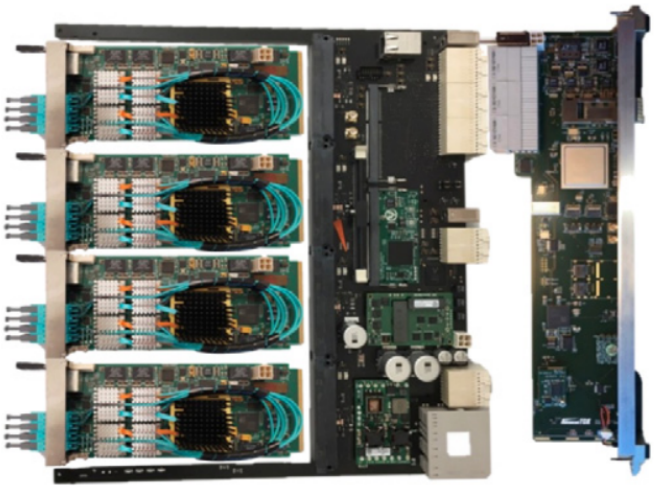


Figure 2.7: Picture of TilePPr with 4 CPMs (left) and a TDAQi (right) [27].

The TilePPr, pictured in Figure 2.7, consists of an Advanced Telecommunications Computing Architecture (ATCA) carrier board, the Trigger and Data Acquisition interface (TDAQi) and four Compact Processing Modules (CPM). The CPMs read the data and reconstruct the energy of the cells. They also transmit signals coming from the trigger systems, as well as the LHC clock, to the front-end electronics.

The TDAQi creates trigger primitives from the data transmitted by the CPMs. These are transferred to the global Level 0 Trigger and the Front-End Link eXchange (FELIX) system, which transmits data packages from the front-end electronics to the data acquisition systems.

The up-link has a data transmission rate of up to 9.6 Gbps, while the down-link, for clock signals, control and configuration, reaches a rate of 4.8 Gbps [23].

3 | Hardware and Software Communication

3.1 Field Programmable Gate Arrays

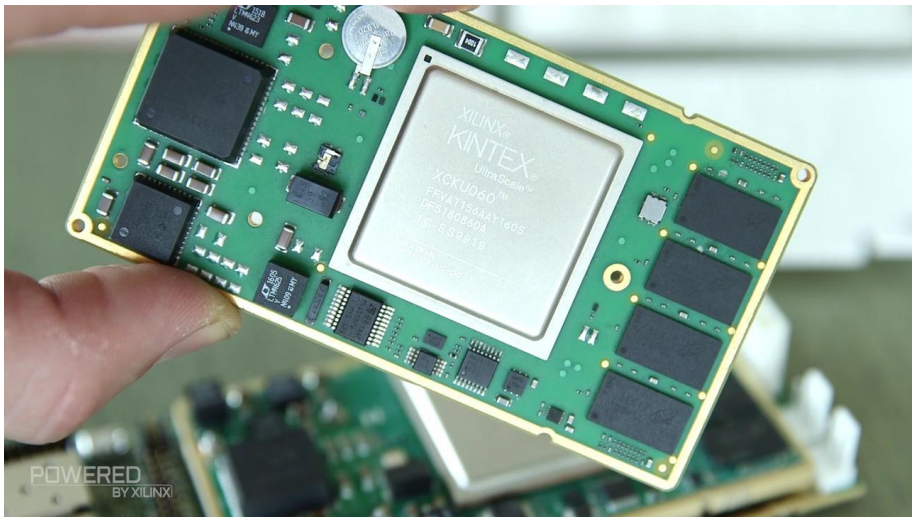


Figure 3.1: An AMD Kintex UltraScale FPGA [29]. An FPGA of the same architecture is used for the CPMs of the TilePPr and for the firmware development in this project.

Field Programmable Gate Arrays (FPGA) [30] are semiconductor devices and a type of integrated circuits. They are called field-programmable, because, unlike, for example, an Application Specific Integrated Circuit (ASIC), they can be modified after the manufacturing process. They can be configured using firmware, written in hardware description languages like VHDL [31]. This makes them advantageous in any application where changing requirements are expected.

FPGAs can easily run multiple applications in parallel, have a small latency and a reasonable power consumption. Because of this, FPGAs can be used for hardware acceleration in certain environments.

In the PreProcessor architecture, where a lot of data needs to be processed quickly and there is very little latency, FPGAs can be employed to accelerate signal processing. Because of parallelization, an FPGA can provide many high speed links in a compact design, which is essential in this application. Additionally, their flexibility allows for modification of

the circuit and correction of potential mistakes in a less complicated manner, since the firmware can be updated.

The TilePPr module also makes use of FPGAs. Each of the four CPMs is equipped with a Kintex UltraScale FPGA, like the one pictured in Figure 3.1. Another is installed in the TDAQi module.

3.2 Hardware

For the firmware development in the context of this thesis, the AMD Kintex UltraScale FPGA Development Board [32] is used. The board provides sufficient resources. In particular, it is equipped with two sets of SFP+ ports, which are able to transmit and receive data at rates of up to 10 Gbps. The FPGA has the same architecture as the one used for the TilePPr CPMs, with reduced resources. In Figure 3.2, the Development Board that was used in this project is shown. The Kintex UltraScale FPGA is covered by the black fan. The SFP+ ports are contained in the silver box in the bottom left corner, where the optical fiber is plugged in.

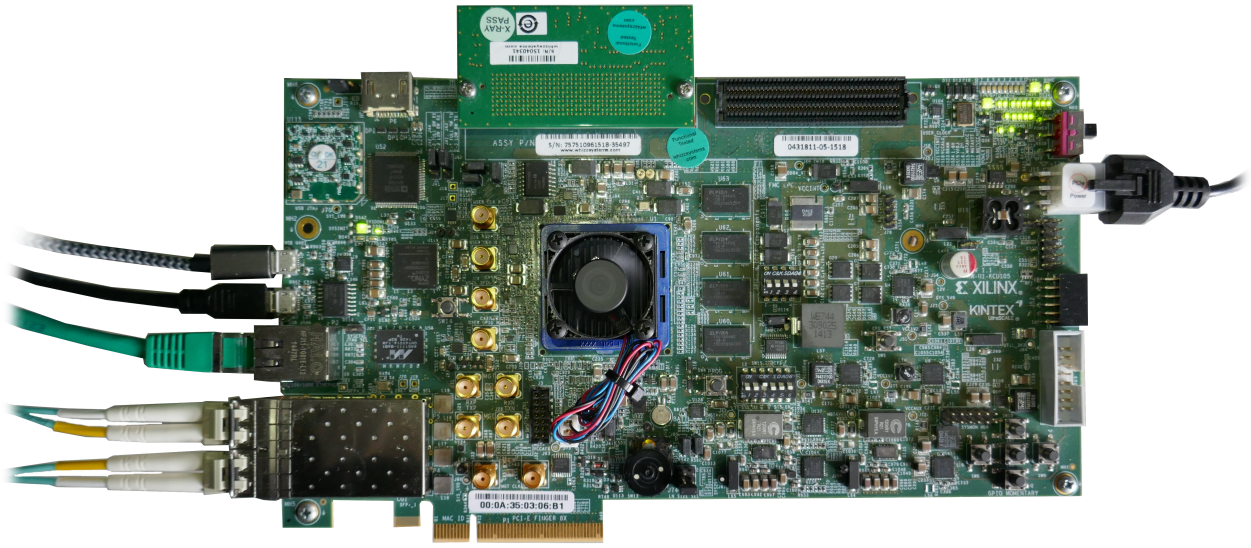


Figure 3.2: The AMD Kintex UltraScale FPGA Development Board used in this work. The FPGA is covered by the black fan. The SFP+ ports can be seen in the bottom left.

3.3 The IPbus Protocol

The IPbus protocol [33, 34] is a control protocol for FPGA-based hardware and works with virtual A32/D32 busses. Development of IPbus began at CERN in 2009. Since then, the project has been advanced and successfully implemented in several large particle physics experiments. In particular, it has been used in parts of ATLAS and CMS since the second data-taking period.

In this work, the IPbus protocol is used to create the necessary interface between software and hardware, both for writing test data to the firmware and for recovering the resulting signals for comparison.

In the past, the VMEbus (Versa Module Eurocard bus) [35] standard was used for electronics systems. This was true for many physics experiments as well. For example, the current electronics setup for TileCal uses this standard [23]. The VMEbus standard is not widely used anymore, newer electronic components usually follow the xTCA (ATCA or μTCA) [36] specifications. This allows for the usage of industry-standard serial communication techniques, especially Gigabit Ethernet. However, unlike VMEbus, the xTCA standard does not specify a hardware access protocol. Therefore, a new control protocol was needed, which is why IPbus was developed.

The IPbus suite of associated firmware and software consists of the following:

- **IPbus firmware:** Firmware repository with different modules, implemented in VHDL. The firmware components are able to interpret IPbus transactions.
- **μHAL:** The User Hardware Library provides a C++/Python API and can be used to modify IPbus enabled hardware with software applications.
- **Control hub:** Software that implements the IPbus reliability mechanism, which corrects packet loss or duplication, and mediates simultaneous access from multiple clients.

The reliability of the system has been tested extensively, for example at CMS electronics integration center [34].

4 | Firmware Development

To inject data into the trigger chain, firmware is designed that controls the data flow through the physical ports. It is developed on the board described in Chapter 3.2, which is equipped with high speed links. Data can be transmitted and received via optical fiber to test the generated trigger objects in a laboratory environment.

The firmware can be used in two ways. The first is to implement it in an FPGA on a separate board, which can then be used as a data injector. The other way is to add the firmware block in the FPGA design of the TilePPr module directly and transmit data from the module itself to the following trigger chain. The TilePPr is equipped with Xilinx Kintex UltraScale FPGAs [23], the same architecture as the one featured on the development board used in this project, which should reduce the risk of issues with deploying the firmware on the TilePPr directly.

A possibility to easily write to and read from hardware components is desired, which will be realized by using components that can be accessed via software. This way, memory on the FPGA can be read out and modified by a simple program. The IPbus protocol provides the foundation for this, with both the necessary firmware components and the software API.

One of the main challenges of the firmware development is to operate with the high data transmission rates of the trigger chain. The goal is to reach a rate for transmitting and receiving data close to the maximum bandwidth of the TilePPr up-link, which is 9.6 Gbps. 8-bit/10-bit encoding is used because of the optical links, meaning 32-bit words are converted into 40-bit words before being transmitted. The necessary frequency, at which words need to be transmitted, can be calculated:

$$\frac{9.6 \text{ Gbps}}{40 \text{ bit}} = 240 \text{ MHz}. \tag{4.1}$$

Therefore, each 32-bit word needs to be sent out at 240 MHz. The TilePPr module receives and passes on data at a frequency of 40 MHz. At this frequency, six words need to be transmitted to exhaust the maximum bandwidth. For data transmission, the SFP+ ports on the development board are used, which can operate at data rates of up to 10 Gbps.

4.1 Validating the IPbus Dual-Port RAM

The basic idea of the developed firmware is to use a Dual-Port RAM (DPRAM), which generally refers to memory, that can be accessed from two sides, typically allowing both read and write transactions from either. More specifically, for this project, memory that can be accessed both via software and the FPGA fabric, is suitable. The IPbus DPRAM [37] can be found in the IPbus firmware repository. It is designed so that it can be read out and written to using IPbus' software API, but signals coming from the FPGA fabric can also be used for reading and modifying. It is therefore a fitting choice for this application. Furthermore, the software and firmware processes of the DPRAM run with separate clocks. This will be very convenient for accelerating data transmission to the FPGA fabric, while keeping the software access, which cannot be operated with a clock speed that high.

In the first version of the firmware, the IPbus DPRAM was implemented and tested. A schematic view of the firmware is pictured in Figure 4.1. Both sides of the DPRAM use the IPbus standard clock, which runs at 31.3 MHz.

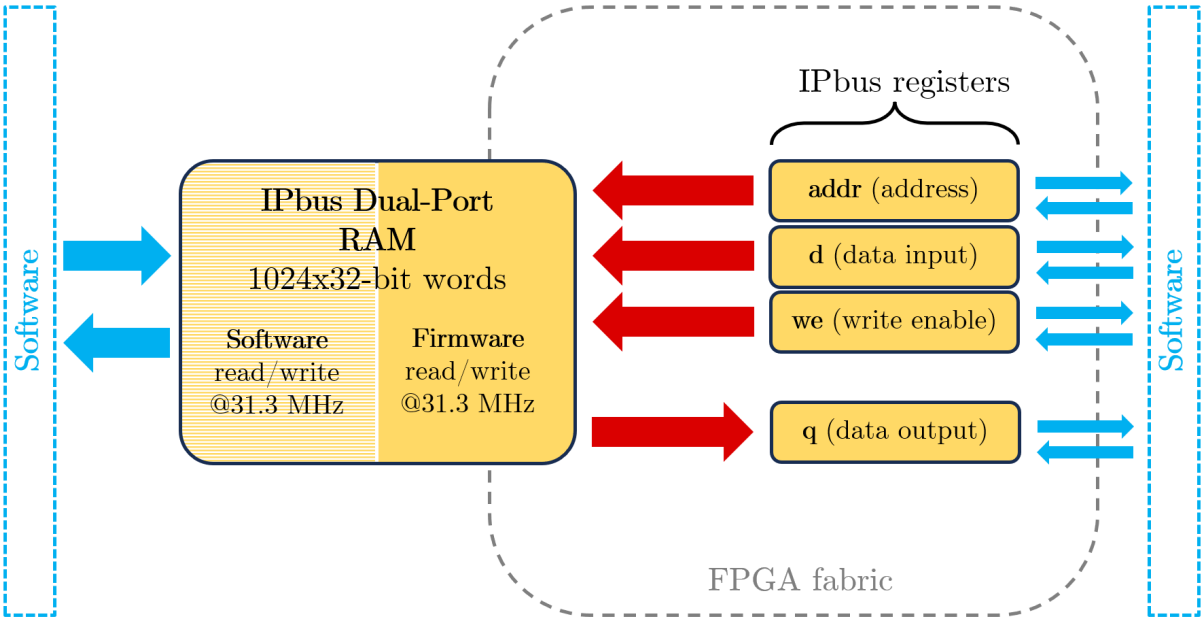


Figure 4.1: Schematic view of the firmware setup for the validation of the IPBus DPRAM. The DPRAM can be read out and modified via the software access or the FPGA fabric, using IPbus registers. Firmware connections are colored red, while software access is denoted by blue-colored arrows. Modules taken from the IPbus firmware repository are in yellow.

The DPRAM has four ports, where variables are received or transmitted to other firmware components. They are connected to IPbus registers which have a single entry that can be changed and read out via the software API.

There are three input variables and one output. The given address determines which

entry of the DPRAM data is read out from and potentially written to. The output data, which is whatever is written at the specified address, is passed onto the FPGA fabric. If the signal to enable writing is given, the input data becomes the new data entry at that address. Because of the additional signal, a data entry is not necessarily modified after it is read out.

With these registers, single data points can be read or modified at a chosen address, just by modifying the registers themselves. This way, the DPRAM can be controlled both using the registers or via software. With the firmware pictured in Figure 4.1, the functionality of the DPRAM could be verified successfully.

4.2 Data Output

The next step is to extend the firmware to read out data from the DPRAM and pass it on to the physical ports of the development board.

The data output from the DPRAM is controlled by the firmware entity *RAM Logic*, as shown schematically in Figure 4.2. Addresses of the DPRAM are counted up and data entries are read out one after the other. Similar to the registers in the previous firmware setup, an address is supplied and the data point at that address is read out. The data point is transferred to the *RAM Logic* entity, from where it will be sent to the SFP+ ports later.

Two registers are implemented that can be used to reset the address loop and to control the maximum address. If the maximum address is reached, the loop restarts from the beginning.

The clock for the firmware side of the DPRAM is changed, so that the data can be transferred to the FPGA fabric at 40 MHz. The size of the DPRAM is increased from 1024 to 3564 32-bit words. The number of data points is chosen because there are 3564 possible bunch crossings in a full orbit of the LHC, as was calculated in Chapter 2.2.

4.2.1 Connecting to High Speed Links

With the previously implemented logic, words are read out of the DPRAM at 40 MHz and transferred to the *RAM Logic* entity. The data is transmitted to another logic block, called *High Speed Link*. This part of the firmware is also responsible for "framing" the data and mapping 8-bit to 10-bit words, which is necessary to ensure the DC balance of the optical transmission. The data is then transmitted via the SFP+ ports.

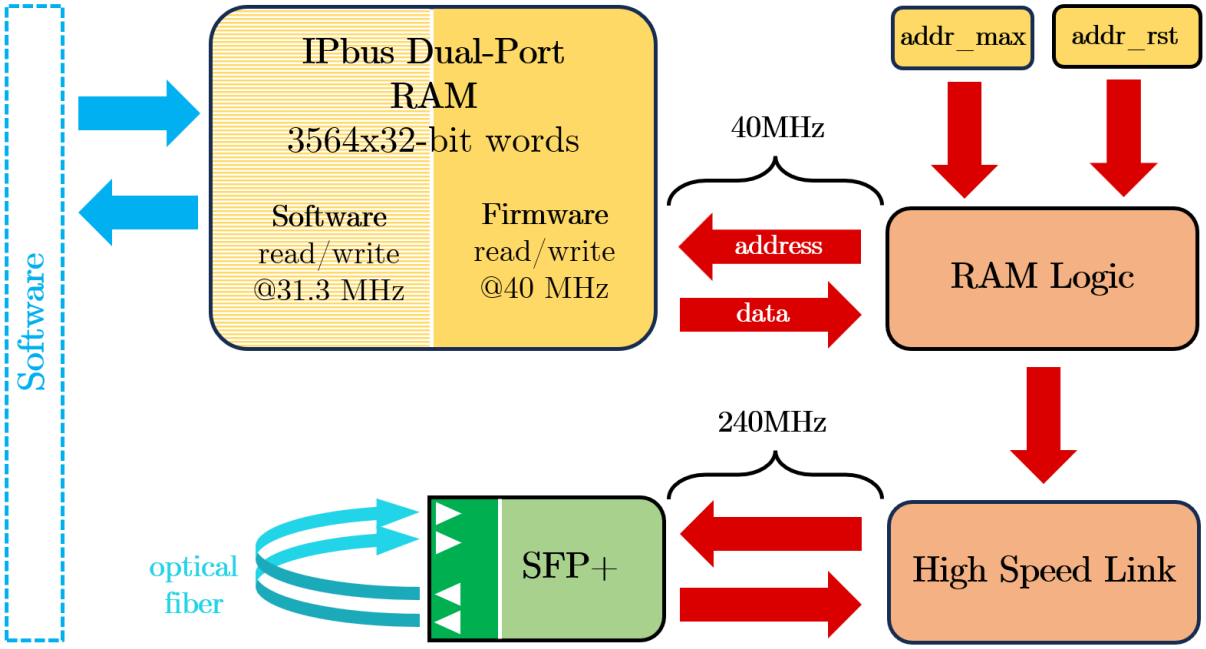


Figure 4.2: Schematic view of the firmware setup with data output. With the two new firmware components data points can be transmitted and received via the SFP+ ports. The red arrows denote firmware connections, blue arrows indicate software access and the turquoise arrows represent the optical fibers.

From the ports, 40-bit words are given out at 240 MHz, meaning the transmission rate is 9.6 Gbps. The data that is received via the SFP+ ports is transferred to the *High Speed Link* entity, where it is processed, so that it can be sent to the FPGA fabric.

The firmware setup with the *High Speed Link* and *RAM logic* entities is pictured in Figure 4.2. It also shows that the size of the DPRAM has been changed and that a different clock for the firmware side is used. In this test setup, the optical fibers, indicated by the turquoise arrows, are looped right back into the board.

Using internal logic analyzers, it was found that the data is transmitted and received correctly. However, the returning data is not stored and easily accessible yet.

4.2.2 Data Framing

Data points are transmitted from the *RAM Logic* entity at a frequency of 40 MHz, but from the *High Speed Link* entity, words should be sent out at a frequency of 240 MHz. In order to achieve this, as well as signal the beginning of an actual data point, each data entry is framed by five other words, as listed in Table 4.1. A data frame thus consists of six 32-bit words.

word index	word content
	32-bit
0	0000 0000 0000 0000 & K.28.0 & K.28.5
1	data input
2	0xBEEFCAFE
3	00 00000 00000 00000 00000 00000 00000
4	00 00000 00000 00000 00000 00000 00000
5	00 00000 00000 00000 00000 00000 00000

Table 4.1: The six 32-bit words that make up a data "frame". K.28.0 and K.28.5 are control words that signal the beginning of the frame, while the hexadecimal number "BEEFCAFE" is used as a marker.

K.28.0 and K.28.5 are control words, which mark the beginning of the sequence. For the data recovery, the firmware searches for these signal words and then recovers the data entry that follows. This way, the frame can be deconstructed again.

The hexadecimal number "BEEFCAFE" is also used as a marker, after the actual data entry. This marker is human readable and exists mainly for debugging purposes.

At the moment, the rest of the frame is made up of zeros, which act as placeholders. However, in order to send multiple data entries in one frame, the firmware needs to be changed to provide more than one word every 40 MHz to the *High Speed Link* entity.

4.3 Data Recovery

The final step is to implement a way to recover the data received via the SFP+ ports and store it for later comparison with the input data. For this, a second IPbus DPRAM is used. Data is received from the FPGA fabric at a high rate, but the recovered data should be easily accessible using software. Therefore, the IPbus DPRAM is a suitable choice.

In Figure 4.3, the finalized firmware setup is pictured. For the DPRAM blocks, the favored direction of data flow for the software access is marked in a darker color.

The data frames are deconstructed in the *High Speed Link* entity before the data is passed on, so only the actual data entries are stored. This is why data is written to the second DPRAM at 40 MHz and not 240 MHz.

The data path, in the firmware setup as pictured in Figure 4.3, can be summarized as follows. Data can be written to a DPRAM using the software access, for example with a simple Python script. The entries in that DPRAM are read out and passed onto the fabric of the FPGA. This happens continuously at a frequency of 40 MHz. If the final address is reached, counting starts from the beginning again. In the *High Speed Link* entity, each

data point is framed with signal characters and zeros, with the total frame being six 32-bit words long. Each word of the frame is transmitted through the SFP+ ports at 240 MHz. Considering 8-bit/10-bit encoding, this means that a data transmission rate of 9.6 Gbps is indeed reached, as was calculated before. The data received from the SFP+ ports is recovered in the *High Speed Link* entity. The frames are deconstructed and the data points are written to a second DPRAM. The recovered data can be accessed via software and compared to the input.

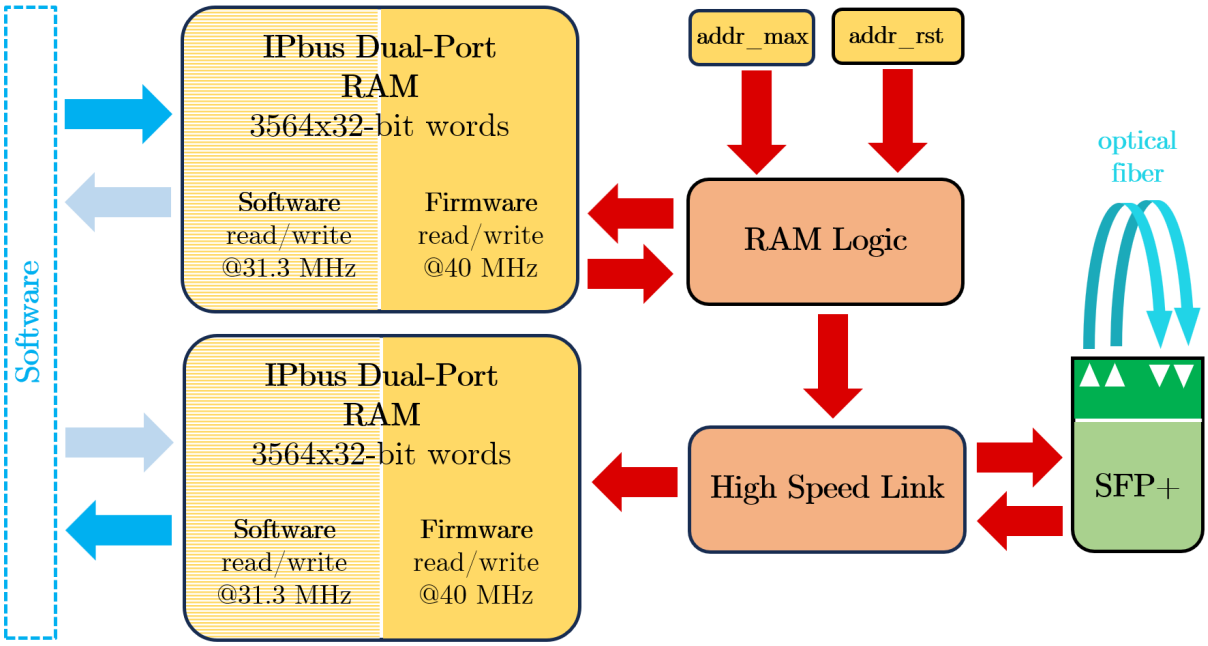


Figure 4.3: Schematic view of finalized firmware setup. Words are transmitted and received via optical fiber at a rate of 240 MHz. The input data can be written to an IPbus DPRAM. A second one stores the recovered data. The favored direction of the software access is marked in a darker color. Red and turquoise arrows denote firmware connections and optical fibers, respectively.

4.4 Outlook

While the firmware transmits data at a rate of 9.6 Gbps, only a single data entry per six-word frame is sent out. It can therefore only simulate the data coming from a single channel.

The main goal of future development of the firmware will be sending multiple data words at 40 MHz. This way, multiple channels can be simulated. As a first step, the placeholder data words in the frame could be replaced. The second approach would be to reduce the size of data points to 16-bit, meaning every 32-bit word could hold two data points. Reducing the bit size to 16 is justifiable because the real ADC data consists of 12-bit words.

Illustrated in Table 4.2 is an example of how the data frame could be used with those changes, in comparison to the current frame as shown in Table 4.1. Further upscaling can be realized by using multiple optical links.

word index	word content	
	16-bit	16-bit
0	channel 1	K.28.0 & K.28.5
1	channel 3	channel 2
2	channel 5	channel 4
3	channel 7	channel 6
4	channel 9	channel 8
5	channel 11	channel 10

Table 4.2: Example of modified data frame to fit more words. The word size has been reduced to 16-bit and the "BEEFCAFE" marker, which is not strictly necessary, no longer exists. Now, the six 32-bit word frame is enough to transfer 11 data points.

5 | Software Development

To test the TilePPr module and validate the trigger chain, the method considered in this thesis is the injection of artificially generated data. Therefore, data for testing is needed.

This is achieved by developing a custom software package to generate such data. The aim was to generate a number of different patterns representing the signals in the trigger chain. The created data can then easily be loaded into the firmware using the IPbus protocol.

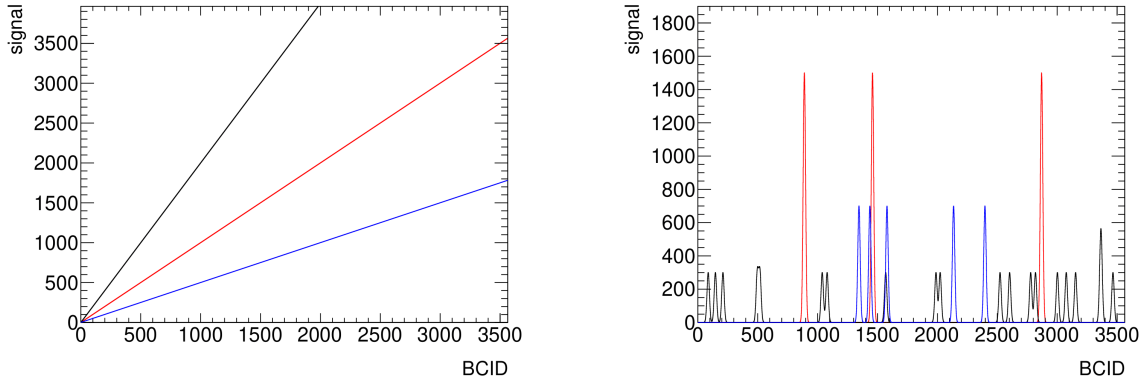
Every TilePPr module is capable of reading out 64 channels, each corresponding to a Trigger Tower, which covers an area of $\Delta\eta \times \Delta\phi = 0.1 \times 0.1$. The first step is to create artificial data as a stand-in for the signal coming from a single channel, later to generate data for entire detector areas, meaning multiple channels at once.

The software should generate both patterns designed to resemble actual data coming from the detector and patterns that do not look like real signals. On the one hand, data that resembles the signals generated by actual events in Tile could provide useful information about how the trigger chain will respond to the detector signals. On the other hand, data that looks very different from the signals generated in Tile offers the possibility of experimenting with signal patterns that are not typically received. Finding out what happens to meaningless signals can give interesting results as well, especially when it comes to testing the limits of the electronics and can be used for commissioning and integration purposes, such as verifying the mapping of all the channels.

In the following, the process of developing the software is described and examples of generated patterns are shown.

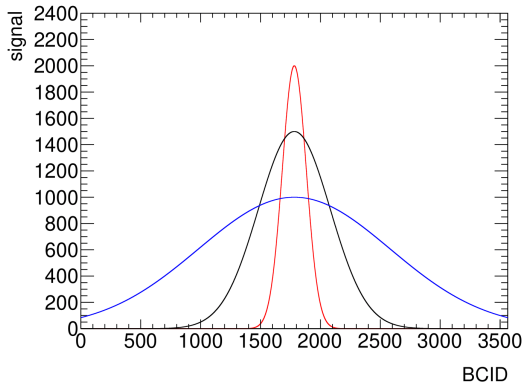
5.1 Pattern Generation for a Single Channel

Firstly, software to generate data representing signals coming from a single channel is designed. It is decided to base the default length of the patterns on the number of possible bunch crossings in an orbit, which is 3564. The data is made up of integer values, emulating the input from the front-end electronics described in Chapter 2.4.1. A number of possible patterns were implemented. All of the patterns have two variable parameters, like the slope and offset of a ramp or the width and amplitude of a Gaussian distribution. Some examples of patterns with varied parameters are shown in Figure 5.1. The output results of the data generation are written into **.txt**-files.

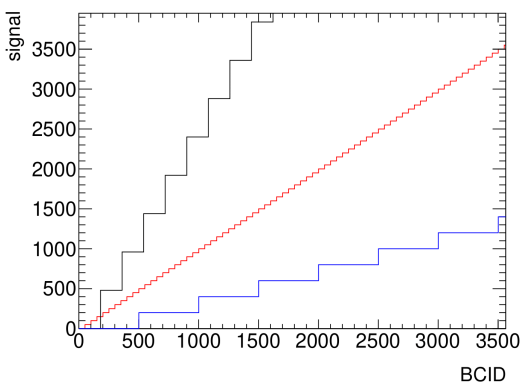


(a) Ramps

(b) Randomly spaced peaks



(c) Gaussian distribution



(d) Steps

Figure 5.1: Examples of generated patterns with varying parameters in blue, red and black.

While the default length of the patterns is 3564, they can be repeated at arbitrary intervals. In Figure 5.2, ramp patterns repeated after a varying number of steps are shown. The maximum value of a pattern can also be adjusted.

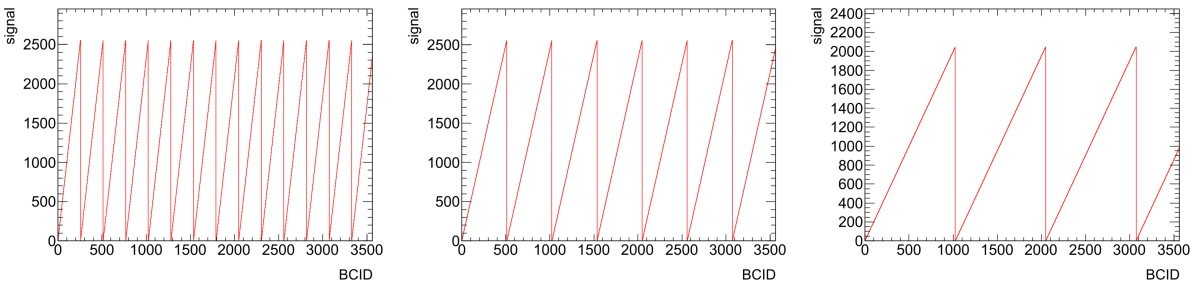


Figure 5.2: Ramp patterns, repeated every 256, 512 and 1024 BCID.

Next, the possibility to add noise to the patterns is implemented. It is generated by taking random samples from a normal distribution that are added after the pattern is created.

The noise level, meaning the standard deviation of the distribution, can be adjusted. In Figure 5.4, some examples of patterns with and without noise are pictured.

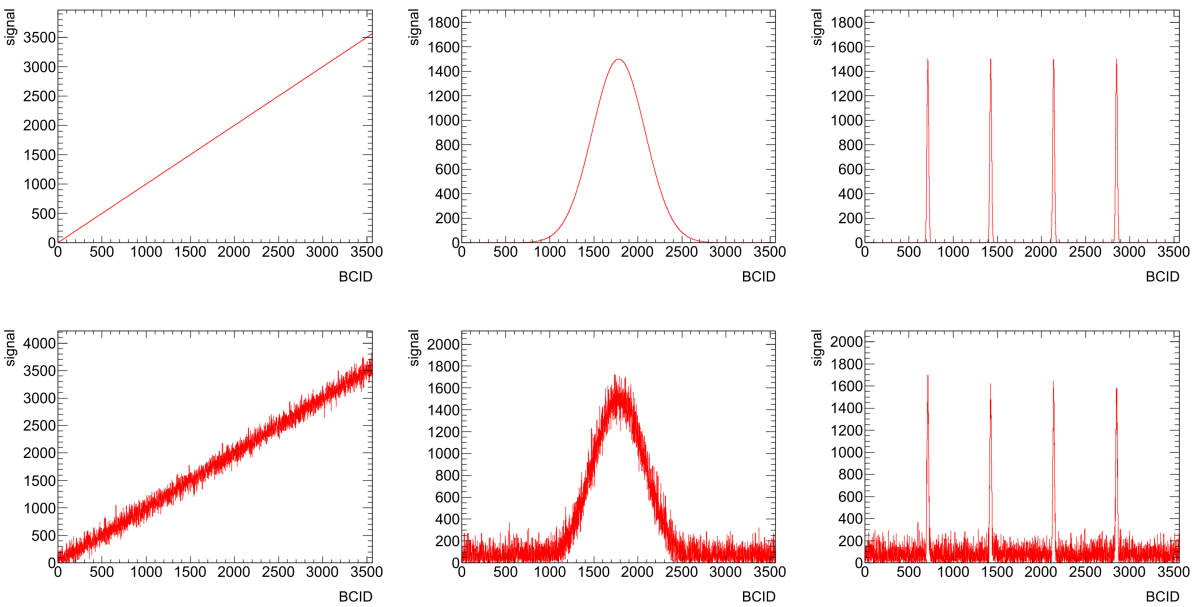
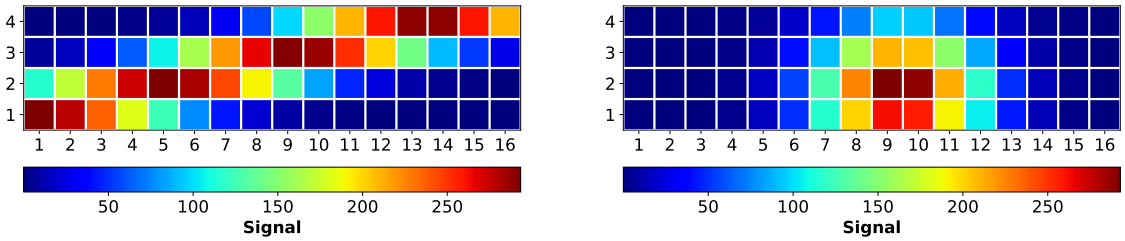


Figure 5.4: Examples of generated patterns with and without noise added.

5.2 Pattern Generation for Multiple Channels

The next step is generating correlated data for multiple channels, representing signals from entire detector areas. For this reason, artificial detector signals were designed. For a two-dimensional grid, the amplitude values in each cell are computed.

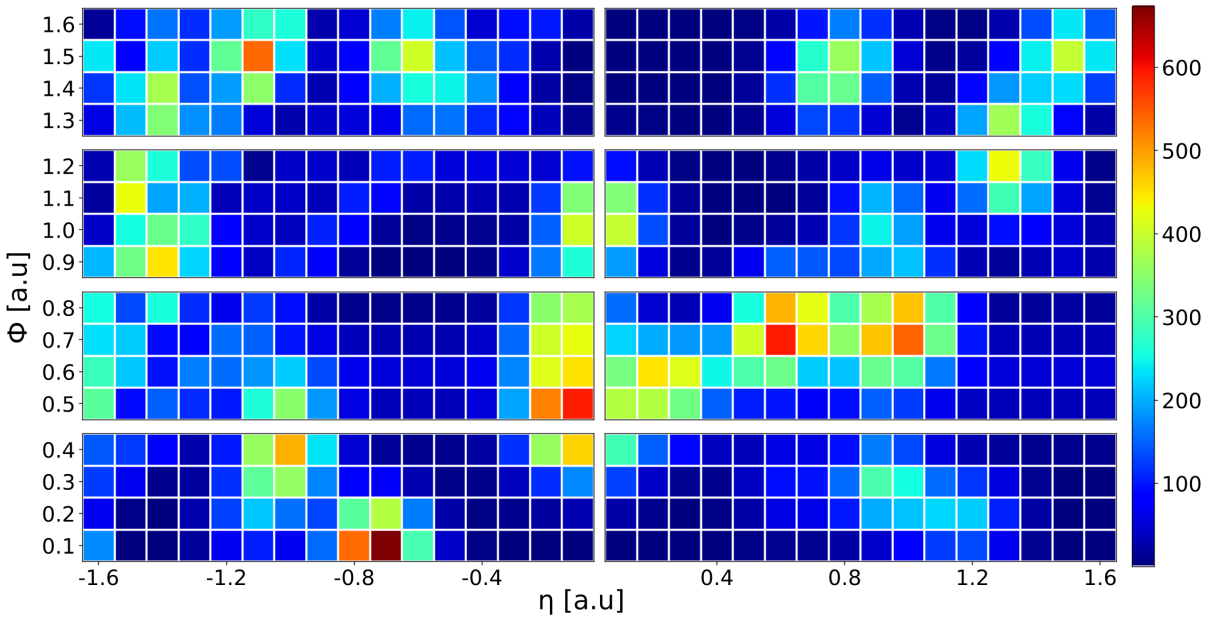
There are two types of patterns currently implemented. The first one is a spread out line, resembling the detector signal resulting from a minimally ionizing particle, like a muon, as shown in Figure 5.5a. The second one is an elliptical Gaussian, similar to the detector signal created by a hadronic or electromagnetic shower, pictured in Figure 5.5b.



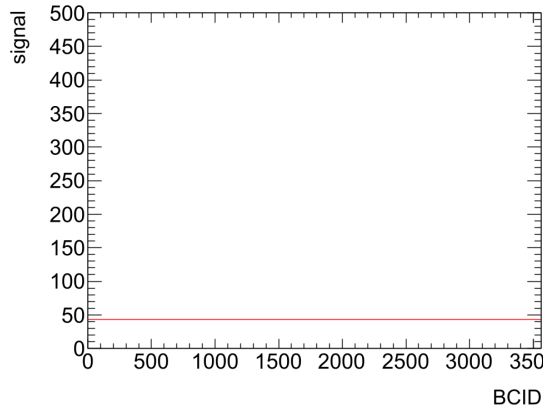
(a) Spread out line **(b)** Elliptical Gaussian

Figure 5.5: Two different two-dimensional patterns for $4 \times 16 = 64$ grid cells. This is the area covered by one TilePPr module.

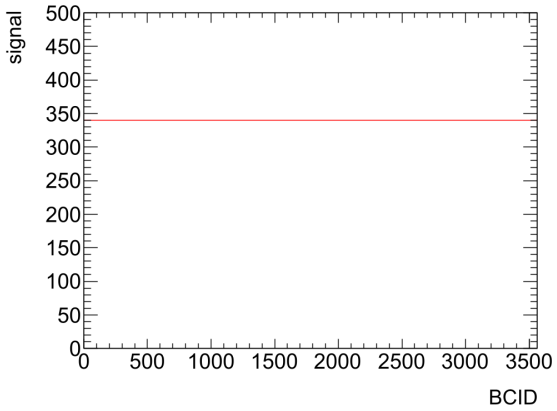
To design a two-dimensional pattern, multiple artificial detector signals of both types are generated at random positions in the $\eta \times \phi$ map, each with a randomized amplitude and, in the case of the line, randomized slope. The maximum amplitude can be adjusted, as can the number of detector signals. Overall, an image like the one pictured in Figure 5.6a can be created, representing the sum of all signals that occurred in one orbit. It is possible to create patterns with over hundreds of detector signals.



(a) $\eta \times \phi$ map with the summed up amplitude of the generated signals. The amplitude is indicated by the color.



(b) Channel: $\eta \times \phi = -1.6 \times 1.0$



(c) Channel: $\eta \times \phi = 0.1 \times 1.1$

Figure 5.6: Generated pattern for $\Delta\eta \times \Delta\phi = 3.2 \times 1.6$ with examples of the generated signal per bunch crossing for two different channels pictured bellow. For the $\eta \times \phi$ map, multiple detector signals get generated at random positions. The amplitude in each cell is the sum over all of the artificial signals and is used for the generation of the data for the corresponding channel.

The size of the generated pattern can be changed as needed. In the examples shown here,

the geometry is based on the granularity and segmentation of the channels in Tile. In Figure 5.6a, an area, which would be covered by 8 TilePPr modules, is shown. It is also possible to generate an image corresponding to the full detector area.

The next step is to design a way for the software to not just generate an $\eta \times \phi$ map, but to use that map to create data files for each of the channels.

In the first implementation of this, the amplitude in each channel, as generated in the two-dimensional pattern, was used to generate stand-in data for that channel. The data for each channel is the assigned amplitude as a constant function of the BCID, as shown in Figures 5.6b and 5.6c. It is also possible to use the value as a parameter of a more complicated pattern, like using it as the amplitude of a Gaussian distribution.

Generated is a data folder containing a **.txt**-file with the two-dimensional pattern, as well as files with stand-in data for every channel.

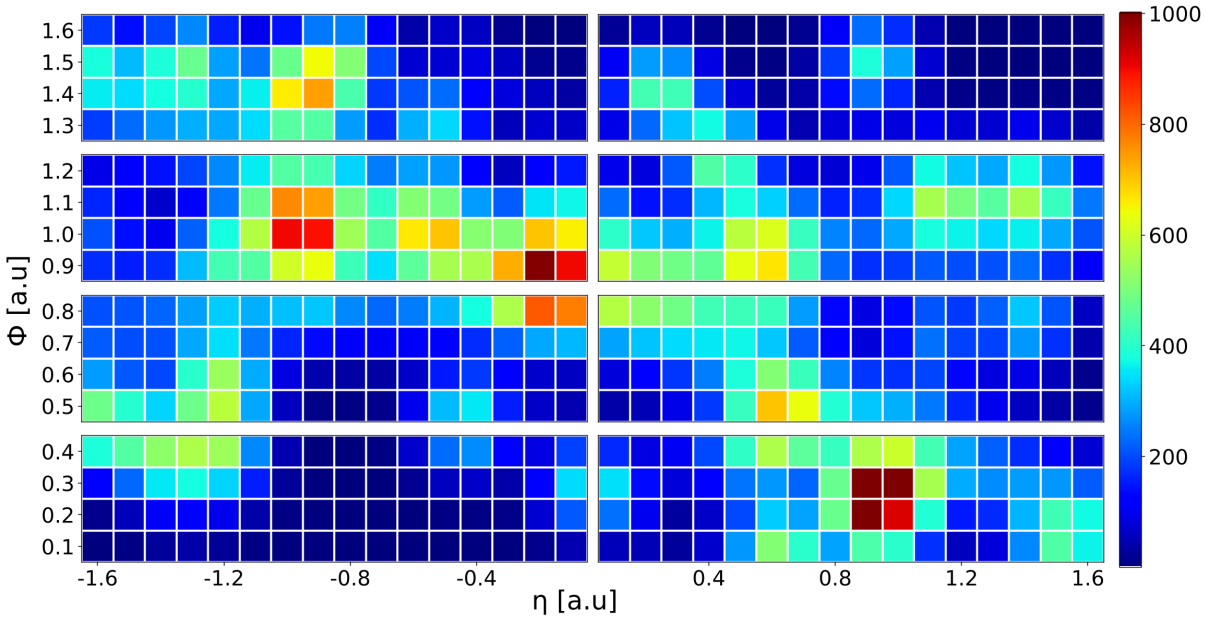
5.3 Timed Patterns

While the previous design of the pattern generation can generate data for all channels, the data was very much simplified. For a more realistic approach, a channel should register detector signals in different bunch crossings.

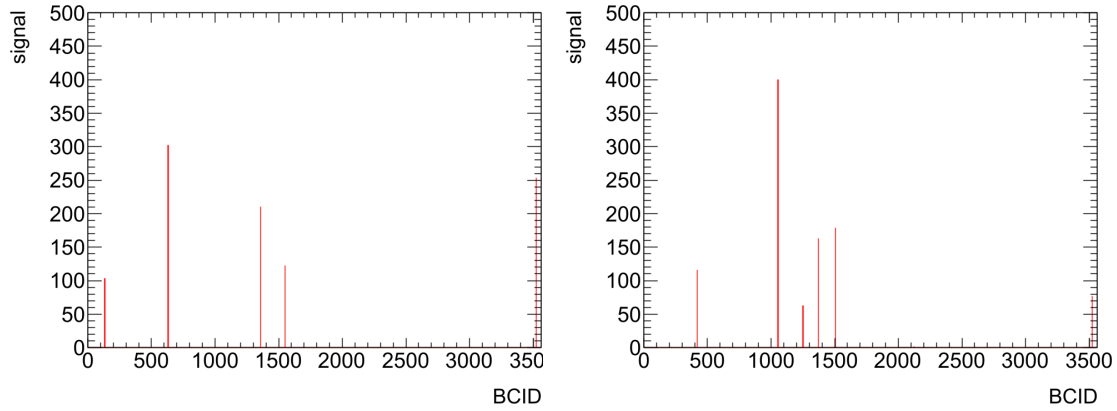
To implement this, the software is reworked such that the artificial detector signals are assigned a random BCID when generated. They are then written to the files for the channels in which they produce a signal at that BCID. This is done subsequently for all generated signals. The two-dimensional pattern made up of the summed up amplitudes is generated at the same time. The data for each channel is now made up of isolated detector signals of a certain amplitude occurring at a certain BCID. The patterns of the channels are correlated, meaning, for neighboring channels, the same signal is registered in the same bunch crossing.

As an example, Figure 5.7a shows the summed up amplitude in each channel after 3564 bunch crossings. In Figures 5.7b and 5.7c, which show the data generated for two different channels, it can be seen that the total amplitude is the sum of detector signals that occur separately in the data for each of the channels.

Using this approach, the generated data resembles more closely the behavior expected of actual detector signals. The first version, with the constant signal for each channel, remains available.



(a) $\eta \times \phi$ map with detector signals summed up over 3564 bunch crossings. The amplitude is indicated by the color.



(b) Channel: $\eta \times \phi = -0.2 \times 0.9$ **(c)** Channel: $\eta \times \phi = 0.9 \times 0.3$

Figure 5.7: Generated pattern for $\Delta\eta \times \Delta\phi = 3.2 \times 1.6$ with examples of the generated signal per bunch crossing for two different channels pictured bellow. In the $\eta \times \phi$ map, the sum of all signals is pictured. The data in each channel is now generated by having different detector signals at different BCIDs, resulting in multiple peaks of varying amplitude.

5.4 Outlook

What is not yet considered, especially in the latter approach to pattern generation, are factors like noise and pile-up. Also, the typical pulse, as generated by the PMTs, spans more than one bunch crossing. This pulse shape is not yet considered in the generation of the test data, where a signal lasts for exactly one bunch crossing. All of this could be incorporated into the data generation in the future.

Using real data or Monte Carlo samples to obtain more realistic test data was discussed.

This idea will be investigated further in the future.

Another idea, which came up during the development of the software, was to use the already existing Tile Pulse Simulator [38]. The algorithm is meant to simulate typical pulses from the PMTs after digitization and is used, for example, in pile-up studies. The height of the pulses is sampled randomly from the statistical distribution of signal amplitudes in each channel, recorded in the case of $\langle\mu\rangle = 1$.

The Tile Pulse simulator has two modes of operation. The first one simply generates single pulses. In the second, which is probably more interesting for this project, pulses are generated continuously, with the amplitude of the previous pulse being added to the current bunch crossing. The resulting data resembles multiple real pulses coming from single detector cells. Potentially, cell data could be added up to get simulated data for an entire Trigger Tower, to make the data generated by the simulator usable for this project.

However, the Tile Pulse simulator only generates data for single cells and cannot simulate multiple cells with correlated amplitudes. It could therefore be more complicated to obtain sensible data for entire Trigger Towers than to simply sum up simulated cells.

Another issue is that for events with more than $\langle\mu\rangle = 1$, the simulator does not have statistical distributions of the pulse height. This means that for $\langle\mu\rangle = 200$, which would be the expected amount of collisions for HL-LHC, it does not sample the pulse height from a distribution recorded at that number of collisions, but instead samples the one for $\langle\mu\rangle = 1$ 200 times.

Nevertheless, utilizing the Tile Pulse simulator will give interesting insights into the behavior of pile-up and pulse shape. As mentioned previously, a realistic approximation of these effects would be a significant step towards generating test data that resembles actual detector signals more closely.

6 | Verifying Correct Data Output & Recovery

For the verification and validation of the data output and subsequent recovery, the optical fibers are looped back into the development board, as depicted in Figure 4.3. Data patterns are written into the memory on the FPGA and passed through the logic chain. In this setup, the same data should always be received as transmitted.

Multiple different patterns are tested. Data resembling a digital signal is loaded into the first DPRAM and transmitted through the SFP+ ports. The second DPRAM, which stores the received data, is read out. Figure 6.1 shows examples of patterns that are used. The input is shown in red, while the output data is depicted in black.

Because the data stream is continuous, the signal that is read out of the second DPRAM starts at an arbitrary point, as can be seen in Figure 6.1. To compare the data, they need to be aligned first. Here, this was done manually.

This test was conducted with multiple generated patterns of varying complexity. After aligning the input and output data, there were no discrepancies between the two in any of the cases.

In the future, software will be developed that aligns and compares the data automatically. Software that can test multiple data files one after the other will also be helpful in conducting more systematic tests.

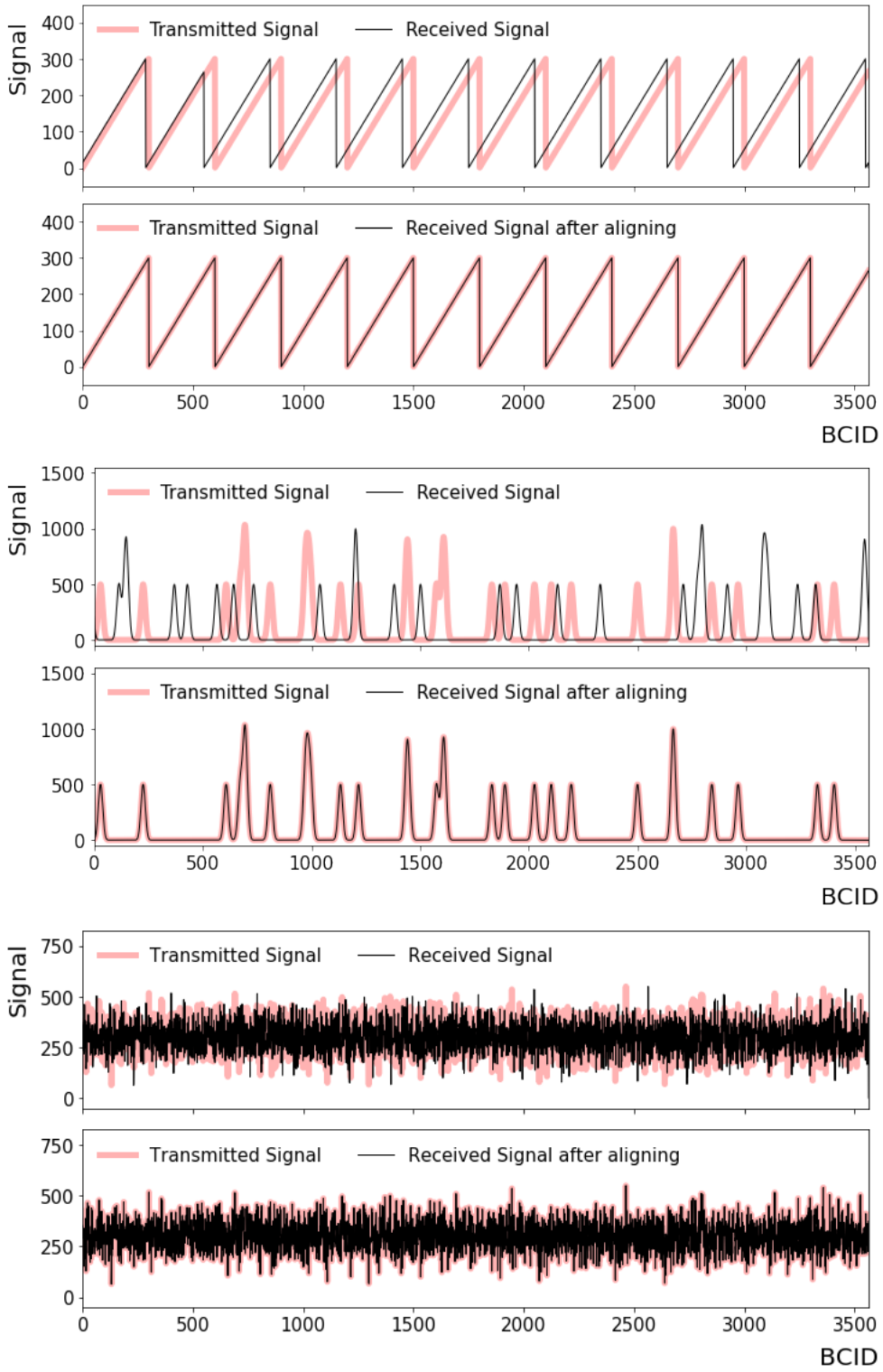


Figure 6.1: Transmitted and received signals for functionality test. For two different test data, transmitted (red) and received (black) signals are compared. In all cases, there are no discrepancies between the input and output data.

7 | Summary and Conclusion

For the Phase-II Upgrade of the ATLAS Tile Calorimeter, the trigger and data acquisition electronics will be replaced. The TilePPr module is a key part of the new architecture. This thesis describes the development of a framework to test the new trigger chain and in particular the TilePPr module. To this end, artificial data is generated, injected into the trigger chain and read out after being processed. Comparing the output to the initial data gives insight into the behavior of the trigger electronics.

The firmware necessary to control the data output and recovery was developed on a commercial FPGA board with high-speed ports that can be connected to other hardware via optical fiber. It was designed around the IPbus DPRAM, which mediates the transition from software to firmware. The finalized version of the firmware developed in the context of this thesis is able to control data flow through the physical ports of the development board. Data can be written to one DPRAM via the software access provided by IPbus and is given out continuously via the SFP+ transmitters. The data recovered via the receivers is collected in a second DPRAM, that can be read out using software as well. Therefore, data input and output can be compared in a convenient way.

The software package developed in this thesis is capable of providing a variety of artificial data representing signals coming from both single channels and entire detector areas. The data is formatted in a way that is easy to use with the implemented firmware, but could easily be updated to provide data suitable for future firmware setups, for example by changing the number of data points. For single channel data, different patterns can be created with varying length and parameters and an adjustable level of noise can be added. When generating data samples for multiple channels, artificial detector signatures that span multiple channels are created. This way, an $\eta \times \phi$ map of variable size is obtained, which is used to generate signals for every single channel. The option to simply have a constant signal with the summed up amplitude for each channel is available. It is also possible to generate signatures that occur at different bunch crossings. The resulting data for each channel is made up of different peaks at varying BCID.

Finally, the firmware was tested to verify that signals are sent out and recovered correctly. This was done by loading patterns into the firmware, transmitting them through optical fiber back into the board and checking the recovered data. For multiple examples, perfect agreement between the initial data and the received signal was observed.

List of Tables

List of Figures

Bibliography

[1] Mark Thomson. *Modern particle physics*. Cambridge [u.a.]: Cambridge University Press, 2013. ISBN: 978-1-107-03426-6.

[2] *Standard Model*. URL: https://en.wikipedia.org/wiki/Standard_Model.

[3] CERN Courier. *Finding the W and Z*. URL: https://cerncourier.com/a/finding-the-w-and-z/.

[4] The CDF Collaboration. "Observation of Top Quark Production in $p\bar{p}$ Collisions". In: *Physical Review Letters* 74.14 (1995), pp. 2626–2631. DOI: 10.1103/physrevlett.74.2626. URL: https://doi.org/10.1103%2Fphysrevlett.74.2626.

[5] The CMS Collaboration. "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC". In: *Physics Letters B* 716.1 (2012), pp. 30–61. ISSN: 0370-2693. DOI: https://doi.org/10.1016/j.physletb.2012.08.021. URL: https://www.sciencedirect.com/science/article/pii/S0370269312008581.

[6] The ATLAS Collaboration. "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC". In: *Physics Letters B* 716.1 (2012), pp. 1–29. DOI: 10.1016/j.physletb.2012.08.020. URL: https://doi.org/10.1016%2Fj.physletb.2012.08.020.

[7] Lyndon Evans and Philip Bryant. "LHC Machine". In: *Journal of Instrumentation* 3.08 (2008), S08001. DOI: 10.1088/1748-0221/3/08/S08001. URL: https://dx.doi.org/10.1088/1748-0221/3/08/S08001.

[8] URL: https://atlas.web.cern.ch/Atlas/GROUPS/DATAPREPARATION/PublicPlots/2023/DataSummary/figs/peakLumiByFill.png.

[9] The ATLAS Collaboration. "The ATLAS Experiment at the CERN Large Hadron Collider". In: *JINST* 3 (2008). Also published by CERN Geneva in 2010, S08003. DOI: 10.1088/1748-0221/3/08/S08003. URL: https://cds.cern.ch/record/1129811.

[10] R Bruce et al. "Reaching record-low β^* at the CERN Large Hadron Collider using a novel scheme of collimator settings and optics". In: *Nucl. Instrum. Methods Phys. Res., A* 848 (2017), pp. 19–30. DOI: 10.1016/j.nima.2016.12.039. URL: https://cds.cern.ch/record/2274861.

[11] B G Taylor. "Timing distribution at the LHC". In: (2002). DOI: 10.5170/CERN-2002-003.63. URL: https://cds.cern.ch/record/592719.

[12] URL: https://atlas.web.cern.ch/Atlas/GROUPS/DATAPREPARATION/PublicPlots/2023/DataSummary/figs/mu_2022_2023.png.

[13] Zachary Marshall and the ATLAS Collaboration. "Simulation of Pile-up in the ATLAS Experiment". In: *Journal of Physics: Conference Series* 513.2 (2014), p. 022024.

DOI: 10.1088/1742-6596/513/2/022024. URL: https://dx.doi.org/10.1088/1742-6596/513/2/022024.

[14] Béjar Alonso et al. *High-Luminosity Large Hadron Collider (HL-LHC): Technical design report*. CERN Yellow Reports: Monographs. Geneva: CERN, 2020. DOI: 10.23731/CYRM-2020-0010. URL: https://cds.cern.ch/record/2749422.

[15] *New technologies for the High-Luminosity LHC*. URL: https://home.cern/science/accelerators/high-luminosity-lhc/technologies.

[16] The ATLAS Collaboration. *ATLAS: technical proposal for a general-purpose pp experiment at the Large Hadron Collider at CERN*. LHC technical proposal. Geneva: CERN, 1994. DOI: 10.17181/CERN.NR4P.BG9K. URL: https://cds.cern.ch/record/290968.

[17] The ATLAS Collaboration. "Performance of the ATLAS muon trigger in pp collisions at $\sqrt{s} = 8$TeV". In: *The European Physical Journal C* 75.3 (2015). DOI: 10.1140/epjc/s10052-015-3325-9. URL: https://doi.org/10.1140%2Fepjc%2Fs10052-015-3325-9.

[18] The ATLAS Collaboration. "Performance of the ATLAS Trigger System in 2010". In: *The European Physical Journal C* 72.1 (2012). DOI: 10.1140/epjc/s10052-011-1849-1. URL: https://doi.org/10.1140%2Fepjc%2Fs10052-011-1849-1.

[19] The ATLAS Collaboration. *Luminosity determination in pp collisions at $\sqrt{s} = 13$ TeV using the ATLAS detector at the LHC*. Tech. rep. Geneva: CERN, 2019. URL: https://cds.cern.ch/record/2677054.

[20] *The third run of the Large Hadron Collider has successfully started*. URL: https://home.cern/news/news/cern/third-run-large-hadron-collider-has-successfully-started.

[21] Collaboration ATLAS. "Letter of Intent for the Phase-II Upgrade of the ATLAS Experiment". In: (2012). Draft version for comments. URL: https://cds.cern.ch/record/1502664.

[22] *Computer Generated image of the ATLAS calorimeter*. URL: https://cds.cern.ch/images/CERN-GE-0803015-01.

[23] *Technical Design Report for the Phase-II Upgrade of the ATLAS Tile Calorimeter*. Tech. rep. Geneva: CERN, 2017. URL: https://cds.cern.ch/record/2285583.

[24] URL: https://twiki.cern.ch/twiki/pub/AtlasPublic/ApprovedDetectorReferenceFiguresAndSchematics/TileCal_Cells_and_Rows_color.png.

[25] Fernando Carrio Argos. *The Data Acquisition System for the ATLAS Phase-II TileCalorimeter Demonstrator*. Tech. rep. Geneva: CERN, 2021. URL: https://cds.cern.ch/record/2775389.

[26] URL: https://cds.cern.ch/record/2855963/files/tile_upgrade_readout_sketch.pdf.

[27] Antonio Cervelló et al. “The TileCal PreProcessor interface with the ATLAS global data acquisition system at the HL-LHC”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 1043 (2022), p. 167492. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2022.167492. URL: https://www.sciencedirect.com/science/article/pii/S0168900222007847.

[28] Fernando Carrio Argos and Alberto Valero. *The PreProcessor Module for the ATLAS Tile Calorimeter at the HL-LHC*. Tech. rep. Geneva: CERN, 2020. DOI: 10.1016/j.nima.2019.162487. URL: https://cds.cern.ch/record/2666967.

[29] URL: https://cf-images.us-east-1.prod.boltdns.net/v1/static/17209957001/d583bee8-6638-4591-8e1f-75d69f903561/fa8e52d7-6844-41fa-8de4-7240b8be5667/1280x720/match/image.jpg.

[30] *What is an FPGA?* URL: https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html.

[31] “IEEE Standard for VHDL Language Reference Manual”. In: *IEEE Std 1076-2019* (2019). DOI: 10.1109/IEEESTD.2019.8938196.

[32] *KCU105 board user guide*. URL: https://www.xilinx.com/content/dam/xilinx/support/documents/boards_and_kits/kcu105/ug917-kcu105-eval-bd.pdf.

[33] Robert Frazier et al. “Software and firmware for controlling CMS trigger and readout hardware via gigabit Ethernet”. In: *Phys. Procedia* 37 (2012), pp. 1892–1899. DOI: 10.1016/j.phpro.2012.02.516. URL: https://cds.cern.ch/record/2104841.

[34] C. Ghabrous Larrea et al. “IPbus: a flexible Ethernet-based control system for xTCA hardware”. In: *JINST* 10.02 (2015), p. C02019. DOI: 10.1088/1748-0221/10/02/C02019.

[35] *FAQ on VME history and basic technology*. URL: https://www.vita.com/VMEbus-FAQ.

[36] *AdvancedTCA Overview*. URL: https://www.picmg.org/openstandards/advancedtca/.

[37] URL: https://github.com/ipbus/ipbus-firmware/blob/master/components/ipbus_slaves/firmware/hdl/ipbus_dpram.vhd.

[38] *TileDigitsFromPulse*. URL: https://twiki.cern.ch/twiki/bin/viewauth/Atlas/TileDigitsFromPulse.

Erklärung

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

Heidelberg, den 13. September 2023,