

MASTER THESIS ON THE TOPIC

CALIBRATION AND OPTIMISATION OF TIMEPIX3 HYBRID PIXEL DETECTORS FOR THE BEAM GAS IONISATION PROFILE MONITOR IN THE PROTON SYNCHROTRON AT CERN

MASTER COURSE:
SUBMITTED AT THE:

SUBMITTED BY:
MATRIKEL-Nr.:

TIME:

SUPERVISORS:

SUPERVISING PROFESSOR:

CHAIR:

ROSTOCK:

ELECTRICAL ENGINEERING
FACULTY OF COMPUTER SCIENCE AND
ELECTRICAL ENGINEERING
LEONARD SEBASTIAN THIELE
216205822
15.08.2022 - 29.01.2023
DIPL.-ING. TIM BROCKMANN,
DR.-ING. MICHAEL RETHFELDT
PROF. DR.-ING. DIRK TIMMERMANN
COMPUTERS IN TECHNICAL SYSTEMS
29.01.2023

MASTER THESIS

FACULTY OF COMPUTER SCIENCE AND ELECTRICAL ENGINEERING

List of Figures	III
List of Tables	V
Acronyms	VI
Task Description	VIII
1 Introduction	1
1.1 Motivation	1
1.2 Thesis Goals	2
2 Theoretical Basis	4
2.1 Context	4
2.1.1 CERN Accelerator Complex	4
2.1.2 Scientific Goals at CERN	5
2.1.3 Basic Accelerator Physics	7
2.1.4 Challenges for Beam Instrumentation in the Context of HL-LHC	9
2.2 Beam Gas Ionisation Profile Monitor	9
2.2.1 Instrument Design	10
2.2.2 Readout Chain	16
2.2.3 Software Stack	18
2.3 Timepix3 Hybrid Pixel Detector	20
2.3.1 Pixel Front End	21
2.3.2 Silicon Sensor	22
2.3.3 Chip Periphery	25
2.3.4 Equalisation of the Pixel Matrix	25
2.4 Zynq MPSoC	30
2.4.1 Processor Architecture	30
2.4.2 FreeRTOS	31
2.4.3 Libmetal	32
2.5 Qt Framework	32
2.6 Pybind11	33
3 Conceptual Design	35
3.1 Calibration of the Timepix3 BGI	35
3.1.1 Equalisation	35
3.1.2 Coarse Threshold Setting Determination for the Equalisation	38
3.1.3 Equalisation Reference Signals	38
3.1.4 DAC Calibration	40
3.1.5 Communication Calibration	41
3.1.6 Pixel Masking	42
3.2 Changes to the Readout Chain	43
3.3 Implications on Software and Gateware Design	43
3.4 Expert GUI Concept	45

3.4.1	Previous Expert GUI	45
3.4.2	Access to the Driver	46
3.4.3	GUI Design	46
3.5	RPU Concept	47
3.5.1	Aims of the RPU Usage	47
3.5.2	Possible Tasks to be Implemented on the RPU	48
3.5.3	Separation of Different Tasks on the RPU	48
3.5.4	Communication between RPU and APU	48
4	Implementation	50
4.1	Equalisation	50
4.2	Expert GUI	51
4.2.1	Restructuring of Driver Code	53
4.2.2	Equalisation Displaying	54
4.3	RPU Implementation	56
4.4	Debugging for APU and RPU Applications	57
4.5	Deployment and Verification	58
5	Evaluation	59
5.1	Performance of Different Software Stacks	59
5.2	Equalisation Performance	61
5.2.1	Equalisation Procedure	61
5.2.2	Performance of Different Reference Signals	63
5.2.3	Verification of Equalisation Improvements for Different Parameters	66
6	Summary and Outlook	69
6.1	Summary	69
6.2	Outlook	70
	Statutory Declaration	73

List of Figures

1	CERN accelerator complex (cropped from [2]).	5
2	ALICE detector at the LHC [7].	6
3	Forces a focusing Quadrupole magnet will enact on the beam [13].	7
4	Cartoon of the Beam Gas Ionisation (BGI) instrument design [4].	10
5	Rendering of the Beam Gas Ionisation (BGI) instrument, adapted from [4]	10
6	Installed Beam Gas Ionisation (BGI) in the PS-tunnel, picture taken from [21]. .	11
7	Removed Beam Gas Ionisation (BGI) with it's magnet during an intervention in 12/21.	12
8	Vacuum tank removed from the magnet.	13
9	In-vacuum electronics of the Beam Gas Ionisation (BGI) instrument, picture taken from [21].	14
10	Timepix3 BLM assembly	15
11	Magnet setup of the Beam Gas Ionisation (BGI) instrument.	15
12	Readout architecture from [4].	18
13	Software stack, simplified after [29].	20
14	Timepix3 pixel frontend, simplified after [31].	21
15	Visualisation of Time of Arrival (ToA) and Time over Threshold (ToT).	22
16	Conduction band energies for different materials from [32].	23
17	PN-junction in semiconductors from [33].	24
18	Visualisation of a P on N silicon sensor, following [4].	25
19	Threshold mismatch between two pixels and the resulting difference in Time of Arrival (ToA) and Time over Threshold (ToT) values from [22].	26
20	Threshold scan for a single pixel	27
21	Histogram of the amount of pixels, which had the 50% occupancy crossing at a certain global threshold value.	28
22	Linear interpolation after the first two threshold scans.	28
23	Histograms during the different steps of the equalisation.	29
24	Zynq Multi-Processor System on Chip (MPSoC) architecture, simplified after [40].	31
25	5 σ environment for non-calibrated and calibrated chip.	36
26	Overshoot after the correction of the local threshold in the first fine-tuning step.	37
27	0x0 and 0xF threshold scans with a coarse global threshold value which is not high enough.	38
28	S-curves for a single pixel with a different amount of pixels being charged at once	39
29	DAC calibration heuristic.	41
30	Redesigned readout chain.	44
31	Example of the communication concept between Real-Time Processing Unit (RPU) and Application Processing Unit (APU) for retrieving of the chip temperature. .	49
32	Principle of the driver changes from [58].	53
33	Class diagram after rework from [58].	53
34	Screenshot of the expert Graphical User Interface (GUI) equalisation monitoring.	54
35	Display of the results tab of the equalisation.	55

36	Comparison between an S-curve with interpolation and moving average applied and a raw S-curve.	62
37	Logarithmic cumulative distance of the 50% crossings from the target value. . . .	63
38	Logarithmic cumulative distance of the 50% crossings from the target value for 4 fine-tuning steps.	64
39	Equalisation with the noise floor and a checkerboard level of 16.	65
40	Equalisation with test pulses and a checkerboard level of 16.	66
41	Examples for masks and Digital to Analog Converter (DAC) values that are not suitable for operational devices.	67

List of Tables

1	Run time difference for the acquisition of a matrix with 256x256 entries, which was run 1000 times	52
2	Software performance for different implementations of the startup sequence. . . .	60
3	Software performance of the startup sequence with and without a direct connection from Python.	60
4	Comparison of the results of different equalisations, all values as global Digital to Analog Converter (DAC) steps	67
5	Comparison of the results of equalisation runs with different parameters, all values as global Digital to Analog Converter (DAC) steps	68
6	Comparison of the results of equalisations run with interpolation and moving average settings, all values as global Digital to Analog Converter (DAC) steps . .	68

AD	Antiproton Decelerator
ALICE	A Large Ion Collider Experiment
AMP	Asymmetric Multi-Processing
API	Application Programming Interface
APU	Application Processing Unit
ASIC	Application-specific Integrated Circuit
AXI	Advanced eXtensible Interface
BGI	Beam Gas Ionisation
BIPXL	Beam Instrumentation PiXeL
BLM	Beam Loss Monitor
BSP	Board Support Package
CCC	CERN Control Center
CERN	Conseil Européen pour la Recherche Nucléaire
CLI	Command Line Interface
CMS	Compact Muon Solenoid
CPU	Central Processing Unit
DAC	Digital to Analog Converter
DDR	Double Data-Rate
DRAM	Dynamic Random Access Memory
EC	Event Count
ELENA	Extra Low ENergy Antiproton
FEC	Front End Computer
FESA	Front End Software Architecture
FPGA	Field Programmable Gate Array
FreeRTOS	Free Real-Time Operating System
GIL	Global Interpreter Lock
GUI	Graphical User Interface
HL-LHC	High Luminosity-LHC
HPD	Hybrid Pixel Detector
iToT	integrated ToT
JSON	Java Script Object Notation
LEIR	Low Energy Ion Ring
LHC	Large Hadron Collider
LHCb	Large Hadron Collider beauty
LINAC	Linear Accelerator
LIU	LHC Injectors Upgrade
LS	Long Shutdown
MPSoC	Multi-Processor System on Chip
OCM	On-Chip Memory
OS	Operating System
PL	Programmable Logic
PLC	Programmable Logic Controller
PS	Proton Synchrotron

PSB	Proton Synchrotron Booster
PSU	Processing Subsystem Unit
RAM	Random Access Memory
RF	Radio Frequency
RPU	Real-Time Processing Unit
RT	Real-Time
SDK	Software Development Kit
SEU	Single Event Upsets
SoC	System on Chip
SP	Super Pixel
SPS	Super Proton Synchrotron
STL	Standard Template Library
TCF	Target Communication Framework
TCM	Tightly Coupled Memory
TMR	Triple Modular Redundancy
TN	Technical Network
ToA	Time of Arrival
ToT	Time over Threshold
TP	Test Pulse
UIC	User Interface Compiler
VM	Virtual Machine
xsct	Xilinx Software Command Line Tools

Topic for a Master Thesis

for Mr. B.Sc. Leonard Sebastian Thiele

Dipl.-Ing. Tim Brockmann

p: +49(0)381 498-7269
m: tim.brockmann@uni-rostock.de
w: <http://www.imd.uni-rostock.de>

Calibration and Optimisation of Timepix3 Hybrid Pixel Detectors for the Beam Gas Ionisation profile monitor in the Proton Synchrotron at CERN

Particle accelerators are used for a wide field of applications. On one hand, there are large research facilities, like the CERN accelerator complex, where the goal is to understand how the universe functions on a very small scale. Another application are medical accelerators where proton or ion beams are used to remove cancerous cells, while minimizing damage to the surrounding healthy tissue.

To operate accelerators it is necessary to measure basic beam parameters such as position, intensity and size. These parameters are measured by instruments by means of either: 1) electromagnetic or 2) direct interaction with the beam. Measurement of the beam profile (or size) is particularly challenging since it typically requires direct interaction with the beam, which disturbs the beam and can damage the instrument itself. Over the past years, a new beam profile monitor has been developed at CERN based on the detection of rest gas ionisation electrons.

The Beam Gas Ionisation (BGI) Monitor (<https://bgi.web.cern.ch/>) is based on the detection of rest gas ionisation electrons with Timepix3 Hybrid Pixel Detectors (HPDs). These detectors are mounted inside the ultra-high vacuum (UHV) of the beam-pipe, where the detectors must tolerate: 1) electromagnetic interference from the beam; 2) radiation and 3) switching on and off of a dipole magnet. Due to this environment, the properties of the HPDs can change over time. Consequently, also the accuracy of the created beam profiles can change. In order to ensure a good measurement long term, certain parameters of the chips and the detector have to be monitored. If a parameter changes outside of its bounds, the instrument has to be recalibrated.

Currently, a new readout chain is in development. The main processing component in this chain will be a Xilinx Zynq UltraScale+ MPSoC. This chip will allow the readout processing, monitoring and control procedures to be split between the FPGA part, the Real-Time core and the Dual-Core ARM processor.

In this Master thesis, the following tasks have to be performed:

- Familiarisation with the BGI, especially the implemented calibration and controlling software
- Implementation of Python bindings for the C++ driver
- Implementation of an expert GUI in PyQt
- Automation and improvement of calibration procedures:
 - DAC voltage adjustments
 - Communication setup
 - Implementation of an algorithm to perform equalization of the response across the pixel matrix
- Conceptualization and implementation of periodic health checks
 - Research of which parameters affect the instruments performance
 - Defining of acceptable boundaries for parameters
 - Implementation of code, mostly in the Zynq Real-Time core, to check the parameters against these boundaries
- Communication within a heterogeneous multi-core processor
 - Implementation of inter-core communication using standard-compliant methods and libraries

Supervisors: Dipl.-Ing. Tim Brockmann, Dr.-Ing. Michael Rethfeldt
Date of issue: 15.08.2022
Date of submission: 02.01.2023

Prof. Dr.-Ing. Timmermann
Supervising Professor

1 INTRODUCTION

1.1 MOTIVATION

In the last decades, major advances were made in fundamental research. Nowadays, many processes in the field of particle physics are well understood. In order to prove the mathematical constructs used in particle physics, one has to be able to observe the processes in the real world. The challenge in this undertaking lies in the fact, that some of the processes are only triggered, when very high energies are present. Additionally, some of the processes to be observed only have a low probability to occur [1].

After the second World War, the Conseil Européen pour la Recherche Nucléaire (CERN) was founded in Switzerland. It was meant to bring together physicists in order to advance the research in the field of particle physics in Europe. For the purpose of the measurement of particles, many accelerators were built over the years. The higher the collision energy needs to be, the bigger the accelerators have to become. In order to reach the current maximum collision energy of 13 TeV, a chain of accelerators is used, which nowadays spans between France and Switzerland [2].

With the advances in collision energy and accelerator technology, the requirements for the beam instrumentation also increased. In the next upgrade for the Large Hadron Collider (LHC), High Luminosity-LHC (HL-LHC), the luminosity of the collisions will be increased. In order to prepare for this upgrade, the last long shutdown incorporated the LHC Injectors Upgrade (LIU). Within the framework of the LIU, the Beam Gas Ionisation (BGI) profile monitor was re-developed and is now installed in the Proton Synchrotron (PS). The BGI is providing continuous non-destructive measurement of the transverse beam profile. It is assisting the previous beam instrumentation, as well as allowing for measurements, which were not possible with the existing equipment [3].

In the next long shutdown, the final upgrades towards HL-LHC will be performed. Alongside with the upgrades to the accelerator, a new type of beam instrumentation should be installed. One of the contenders for this beam measurement devices is a modified version of the PS BGI beam profile monitor. It would allow the operators to take measurements throughout the whole beam cycle. This would in turn help to understand the processes that happen during the acceleration and collision stages of the final acceleration in the LHC. From this understanding, the beam quality can possibly be increased, which would increase the amount of data, which the main LHC experiments would be able to take [4].

In addition to the the usage in fundamental research, particle accelerators can also be used for medical applications. For the treatment of certain types of cancer, accelerated particles can be used to remove the cancerous cells. In a medical environment, the requirements for the beam instrumentation are vastly different from the research application. The beam energies are generally a lot lower, while a bigger emphasis is placed on safety and reliability of the instrument. A possible application of the BGI beam profile monitor would be to serve as a measurement device for such a medical accelerator [5].

1.2 THESIS GOALS

The work in this thesis will focus on the readout chain and the software of the instrument.

In the first chapter, the theoretical basis of the thesis is described. Section 2 starts with a brief introduction to accelerator physics and where the beam instrumentation comes into use. Afterwards, the BGI beam profile monitor hardware is explained, which is important in order to be able to understand the different aspects of the calibration algorithms. In section 2.2.2 the readout chain and the software stack are described.

Afterwards, a closer explanation of the Timepix3 Hybrid Pixel Detector (HPD) is given in section 2.3. The Timepix3 is the central data-taking device in the BGI beam profile monitor. Therefore, the performance of the instrument depends heavily on it. Different parts of the chip have to be calibrated and the idea behind the calibration algorithms is explained. The data-generation in the sensor as well as the periphery of the Timepix3 are illustrated.

Currently, a rework of the readout chain is performed by the BGI team, which will change the computational hardware to consist mainly of a Xilinx Multi-Processor System on Chip (MPSoC). It features multiple different cores, which can be used for different applications. One task in this thesis will be the implementation of firmware for certain cores in this new architecture. The libraries and operating systems are discussed as well as the general architecture of the chip in sections 2.4.3, 2.4.2 and 2.4.1.

The next chapter will outline the concepts behind the implemented software and firmware. One of the goals of the thesis is the improvement of the implemented equalisation. Furthermore it should be made possible to run the equalisation with a different reference signal. Additionally, several other calibration steps should be automatised. The change in the hardware of the readout chain makes several changes in the software stack possible. The approach behind the separation of the readout chain onto the new hardware is explained in section 3.3.

Another section of the thesis is dedicated to the implementation of a new expert Graphical User Interface (GUI). The old expert GUI is implemented with Qt, which is not supported in the CERN Technical Network (TN). Therefore, it needs to be replaced with a new implementation. During this transition, several changes and improvements are implemented, which are outlined in section 3.4.

In section 3.5.2, the concepts behind the implementation of the Real-Time Processing Unit (RPU) firmware are outlined. With the addition of the MPSoC, the software stack can be flexibly run on different hardware devices. The RPU will play a special role within this new environment.

In the next chapter, the details of the implementation of the software and firmware are discussed. An emphasis is put on the implementation of the Python bindings and the displaying of the equalisation procedure. Additionally, the strategies and tools, which were used for the debugging of the different applications, are outlined.

The evaluation chapter focuses on the impact the implemented changes had on the performance of different procedures. Firstly, the performance of the different software implementations of the startup sequence for the detector are discussed in section 5.1. The main emphasis is put on the run-time, as this procedure can only succeed or fail. In section 5.2.3, the impact of the changes to the equalisation are analysed. For the equalisation, both the quality of the resulting mask, as well as the run-time, are relevant. Both are evaluated for different parameter sets and reference signals.

In the end, an outlook is given, which outlines different possible improvements that could be implemented in the readout chain.

2 THEORETICAL BASIS

2.1 CONTEXT

2.1.1 CERN ACCELERATOR COMPLEX

The CERN accelerator complex is located near Geneva on the border between France and Switzerland. While the main site in Meyrin is located mostly in Switzerland, parts of the accelerator as well as supporting buildings are located in France. One of the main goals in the accelerator is to create a large number of collisions of particles with the highest possible collision energy. To achieve this task, a chain of circular accelerators is used, which lead up to the collisions in the LHC. In each part of the chain, the particles get accelerated further until they finally reach the maximum energy in the LHC. For most of the runs, protons the particles to be accelerated in the LHC [2]. All of the CERN accelerators and experiments are shown in figure 1. The circumference of the different accelerators can be seen in the picture.

The path of the LHC protons starts in Linear Accelerator (LINAC) 4. There, negative hydrogen ions are accelerated from rest to an energy of 160 MeV. After this acceleration, the particles get transferred into the Proton Synchrotron Booster (PSB). During this transition, the electrons are stripped from the ion so that only a proton remains in the accelerator [2].

From this point onward all machines in the accelerator chain are circular accelerators. The big advantage of circular accelerators is, that the accelerating elements can be passed multiple times. On the other hand, the particles need to be constantly bent around the circle, which requires a stronger magnetic field at higher energies.

After being accelerated to an energy of 2 GeV in the PSB the particles are transferred into the PS where the particles are accelerated to an energy of 26 GeV. All of the aforementioned accelerators are small enough to be located entirely on the Meyrin site. The reason for the increase in the circumference of circular accelerators lies in the physics of circular accelerators. If the particles move at a higher energy, the magnetic field to keep the particles in the desired circular trajectory also becomes larger. To counteract this fact, the circumference of the accelerator can be increased, so that the bending radius and thus the required magnetic field strength is not as high.

The next accelerator in the chain is the Super Proton Synchrotron (SPS), which spans between France and Switzerland. In this accelerator the energy of the particles is increased to 450 GeV. After this acceleration, the particles are then injected into the LHC.

Up to this point in the accelerator chain, the particles were all in a single beam-pipe, travelling in the same direction. In the LHC, two different beam-pipes exist, where the particles are inserted so that they counter-rotate in the accelerator. This is done in order to make the collisions of the particles in the big experiments Compact Muon Solenoid (CMS), ATLAS, A Large Ion Collider Experiment (ALICE) and Large Hadron Collider beauty (LHCb) possible. In figure 2 the ALICE detector is shown.

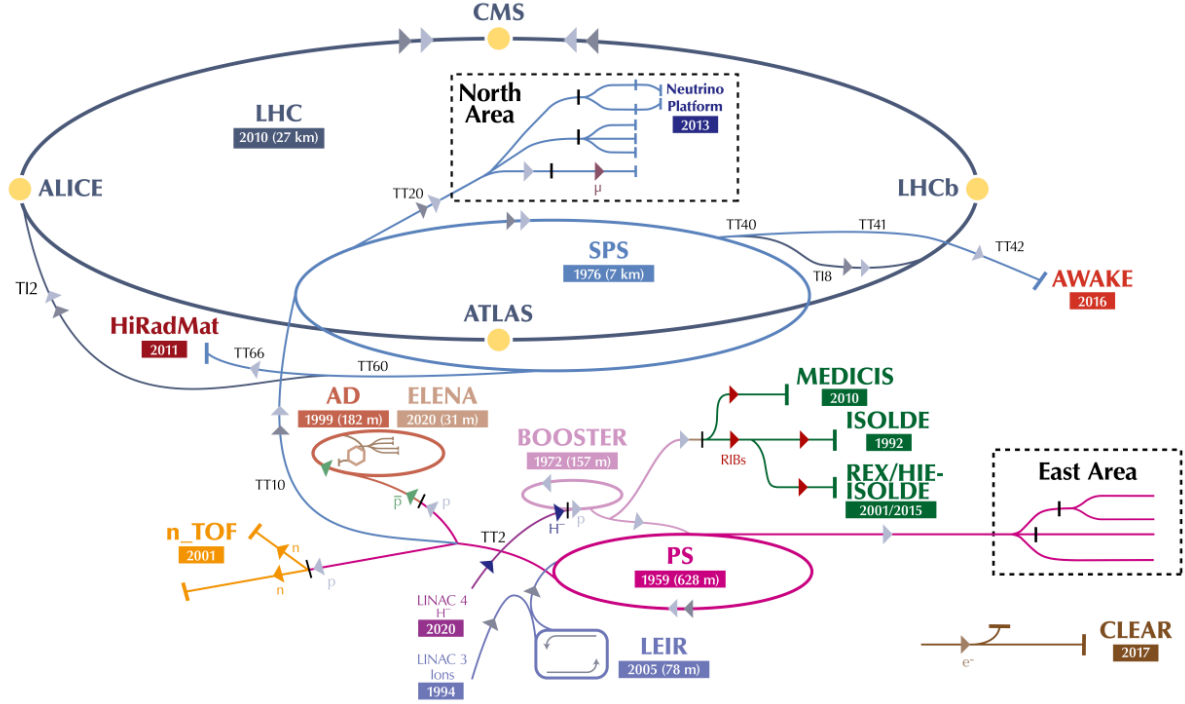


FIGURE 1: CERN ACCELERATOR COMPLEX (CROPPED FROM [2]).

The magnets in all circular accelerators except the LHC are room temperature magnets. Due to the increased requirements on the magnetic field and thus the required current, the magnets in the LHC are superconducting. On the basis of this difference and the increased circumference, the particles can be accelerated from the injection energy of 450 GeV to the collision energy of 6.5 TeV. In the big experiments, the particle beams are then crossed, which causes the particles to collide at an energy of 13 TeV [2].

During standard operation of the LHC, protons are accelerated, starting in LINAC 4 and colliding in the big experiments. Another possibility in the main accelerator chain is to use lead ions for the acceleration and collision. These particles are created in LINAC 3 and then injected into Low Energy Ion Ring (LEIR) where they are concentrated into a smaller space. Afterwards, they follow the accelerator chain from the PS to LHC. The lead ions are mainly used in the ALICE detector to study the Quark-Gluon plasma, which resembles the state of matter that existed shortly after the big bang [6].

2.1.2 SCIENTIFIC GOALS AT CERN

The four big experiments in the LHC are used to study the high energy collisions. During these collisions, processes are triggered, which in turn produce different particles. The trajectory and properties of these particles can then be used to get a better understanding of the interactions in which they were created. Using this general method the big experiments are working on the research towards a more detailed Standard Model of particle physics. Included in this endeavour is the search for new particles and forces to explain not yet understood interactions or prove the existence of theoretical models. During the first run of the LHC the Higgs Boson was discovered

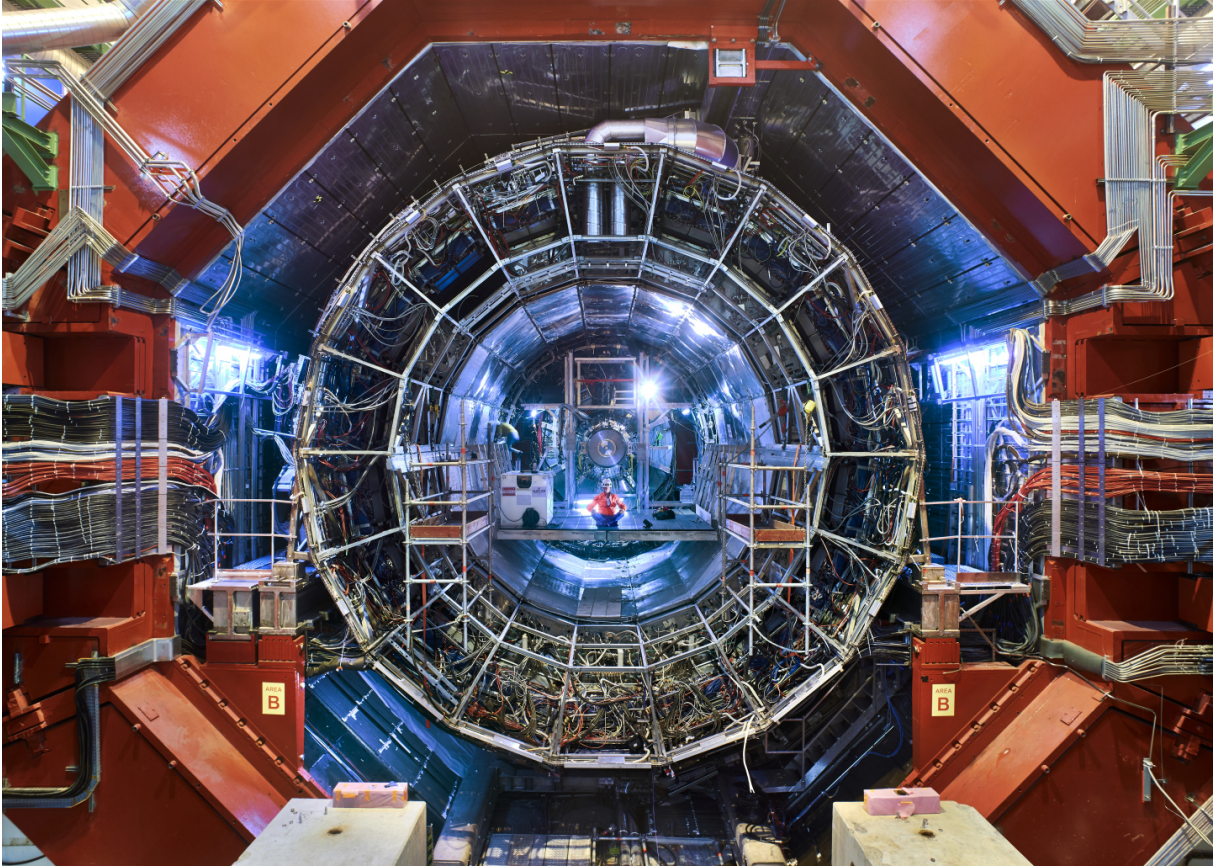


FIGURE 2: ALICE DETECTOR AT THE LHC [7].

in 2012. This particle has a fundamental role within the Standard Model, as it is responsible for the mass of matter [8].

In figure 1 one can see that, besides the main experiments at CERN, there are also numerous smaller experiments and experimental areas. Both the SPS and PS have an experimental area attached to them, in which fixed target experiments are located. In these fixed target experiments the particle beam gets directed into a certain material. From the collision at high energy, processes get triggered, which are then studied [9].

Another experimental complex is the antimatter factory. The beam for this facility is provided by the PS which gets directed into a block of metal. Some of the particles created in this collision are antiprotons, which are collected and fed into the Antiproton Decelerator (AD). The particles cannot be studied in the state at which they enter the AD because they still carry too much energy and are too fast. During the AD-cycle the antiprotons are then decelerated and injected into Extra Low ENergy Antiproton (ELENA). In ELENA, the energy gets reduced even further to 0.1 MeV, which allows the experiments attached to study the properties of antimatter [10].

Additional to high energy physics experiments, scientists at CERN are also researching in different fields. The World Wide Web for example was invented at CERN to enable scientists to communicate with each other over a computer network [11]. The Knowledge Transfer (KT) group at CERN tries to make the expertise gained in the technologies surrounding the accelerator

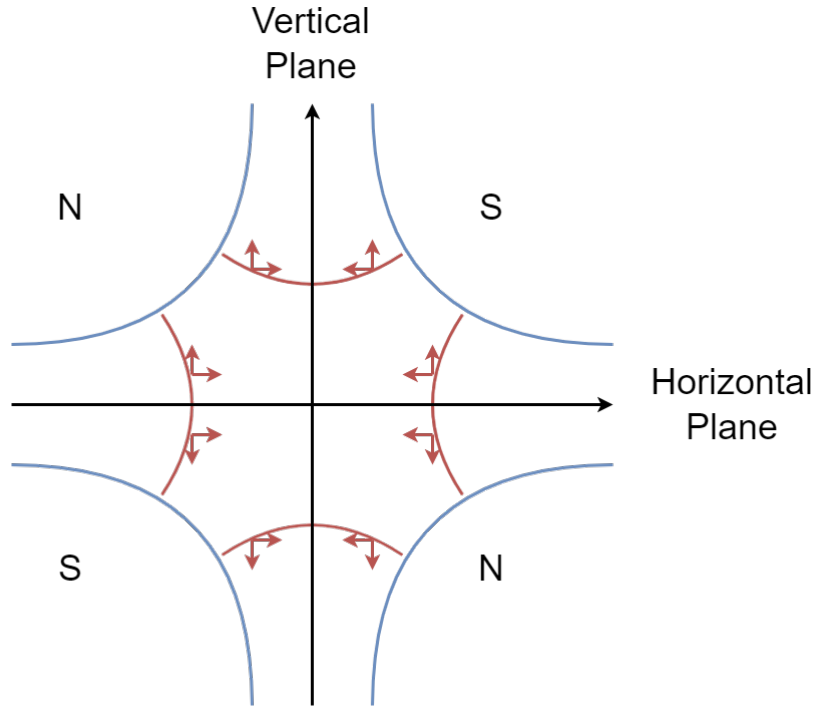


FIGURE 3: FORCES A FOCUSING QUADRUPOLE MAGNET WILL ENACT ON THE BEAM [13].

available to industrial partners or other scientific facilities. This transfer includes technology that can be used in medical accelerators, medical equipment or other applications [12].

2.1.3 BASIC ACCELERATOR PHYSICS

During the accelerator cycle different devices within the accelerator have to be controlled in order to ensure the quality of the beam. These devices consist of different magnets as well as radio frequency cavities. In order to control these devices, different beam instrumentation is in place to monitor the current status of the beam.

One of the forces that need to be controlled in a circular machine is the bending motion. It is provided by a dipole magnet. This magnet provides a static magnetic field, which keeps the particles within a predefined orbit. Due to the Betatron oscillation this orbit will not be the same for all particles. The Betatron oscillation is caused by the magnet acting differently on particles that enter the bending magnet on a different angle. It creates the basis for the transverse motion of the beam [13].

In order to influence the transverse position of the beam Quadrupole magnets are used. These consist of two magnetic pole pairs that are rotated by 90 degrees. The force these magnets enact on the beam particles are shown in figure 3. As it can be seen, the magnet will focus the beam in the horizontal plane, but will defocus it in the vertical plane. Because the magnet is focusing the beam in the horizontal plane, this magnet configuration is called a **focusing** Quadrupole. As the beam needs to be focused in both planes, it is necessary to also have a magnet, which focuses the beam in the vertical plane. These magnets are called defocusing Quadrupoles, which are in essence focusing Quadrupoles rotated by 90 degrees. Because both planes need to be focused, the Quadrupoles need to be installed together with an opposing one. A common configuration

of these magnets is the FODO cell, where a focusing, then a defocusing and then a focusing Quadrupole are combined to focus the beam in both planes [13]. The focusing magnets feature half of the length of the defocusing one in order to keep the difference between both planes minimal. Other magnetic devices that are used in circular or linear accelerators will not be discussed in this thesis.

All instrumentation and controlling elements have to operate with a high precision while working on a small timescale. As most particles move at almost the speed of light, even small changes in the trajectory of the beam can lead to a beam loss in a short time.

If a beam loss occurs it has multiple implications on the instruments and magnets surrounding the beam-pipe. Generally, a beam loss will create a lot of radiation. When an atom is being hit by high energy particles, the particle can either be absorbed by the atom or another particle may be emitted by the atom, both events are possibly making the atom unstable [14]. This radiation will then cause damage to electrical equipment as well as mechanical structures. In addition, radiation can also cause Single Event Upsets (SEU)s in equipment that performs computations, which can invalidate the results.

In the LHC, another problem can arise from the fact that the bending magnets are superconducting. A beam loss event will likely happen at a position, where a magnet is installed. If many high energy particles hit the superconductor, the probability of a quench is greatly increased [15]. When a superconducting magnet is experiencing a quench, the superconducting state is lost in a small area, which causes it to enter a normal conducting state. This will in turn heat the area, which will cause a chain reaction in the whole magnet. A quench would halt the accelerator as the magnet needs to be cooled down again before it can be used [16].

The emittance of a circular beam is defined as the area, in which a certain percentile of the particles is located in the position and momentum phase-space. Therefore, the emittance describes, how much the particles in the beam move in the beam-pipe during the cycle. The emittance is directly correlating to the beam size in a certain plane. In order to reach a high beam density and therefore collision probability of the two beams in the LHC, a small emittance is required [13].

The beam in a circular accelerator is receiving energy from Radio Frequency (RF) cavities. During the acceleration of the beam, it gets automatically separated into bunches. The electric field inside the RF cavities is formed in such a way that it will act differently on particles depending on the longitudinal position relative to the centre of the bunch. Thus the particles will form the bunch around the ideal trajectory of a particle inside the accelerator [17]. The RF cavities will also ensure that the bunches stay apart from each other. Having the beam in these bunches implies that there are gaps in the beam, where no particles are present. When bunches are injected into the next accelerator, the full accelerator can not be filled at once. The positions, where bunches can theoretically be located, are called buckets. The buckets are a consequence of the electric field of the RF system. When filling the accelerator, one can choose to not fill certain buckets and leave a gap between the different bunches. These are among others used for the abort gap. The abort gap is used to ramp up the magnet, which is used to divert the

beam towards the beam dumps. This process takes a bit of time, in which the beam would only partially be deflected and therefore hit the walls of the vacuum tank instead of the beam dump [18].

2.1.4 CHALLENGES FOR BEAM INSTRUMENTATION IN THE CONTEXT OF HL-LHC

The next upgrade to the LHC is called HL-LHC and will be installed in the next Long Shutdown (LS). In 2029, the upgraded LHC is planned to start its operation. The luminosity of the beam is increased by focusing the beams stronger at the collision points as well as by increasing the bunch charge and decreasing the emittance of the beam. By increasing the luminosity in the interaction points, the rate at which physics data can be collected is improved as well, increasing the frequency at which rare processes can be observed [19].

To enable this increased luminosity, several upgrades have to be installed. The upgrading of the injector chain was conducted within the frame of the LIU during LS 2. In this step, many of the accelerators in the chain received upgrades, which are necessary for the increased luminosity. Within this framework, the BGI was installed into the PS [19].

The next step towards HL-LHC will be taken in LS 3. In the frame of these activities, several components in the LHC need to be upgraded. Additionally, new beam instrumentation will be necessary, as the parameters of the beam will change. Currently, there is no instrument in place that can measure each bunch during an entire cycle. This bunch-by-bunch measurement gets even more challenging during the HL-LHC, as the beam energy will be even greater than in LHC [4].

Another challenge within the HL-LHC will be the beam halo. The beam halo describes the energy that is not stored in the centre of the beam profile but rather in the tail of the beam. One side effect of the increased luminosity will be an increase of the energy stored in the beam halo. The risk from having the beam not within its perfect orbit is therefore greatly increased, as the energy in the halos will then be deposited in the walls of the superconducting magnets. This in turn adds an additional risk of causing a quench in the magnets, resulting in a stop of the machine. To address this problem, it is necessary to measure the beam halo to determine the magnitude of the challenge [20].

2.2 BEAM GAS IONISATION PROFILE MONITOR

The description of the BGI beam profile monitor in the following paragraph is based on the description in [4]. The BGI abbreviation will be used as a synonym for the BGI beam profile monitor from here on.

The BGI itself is meant to enable the measurement of the beam profile and emittance of the beam during the whole cycle of the PS. This measurement will enable a better understanding of processes happening during the injection, acceleration and extraction phases of the PS. The development of the BGI was performed in the framework of the LIU.

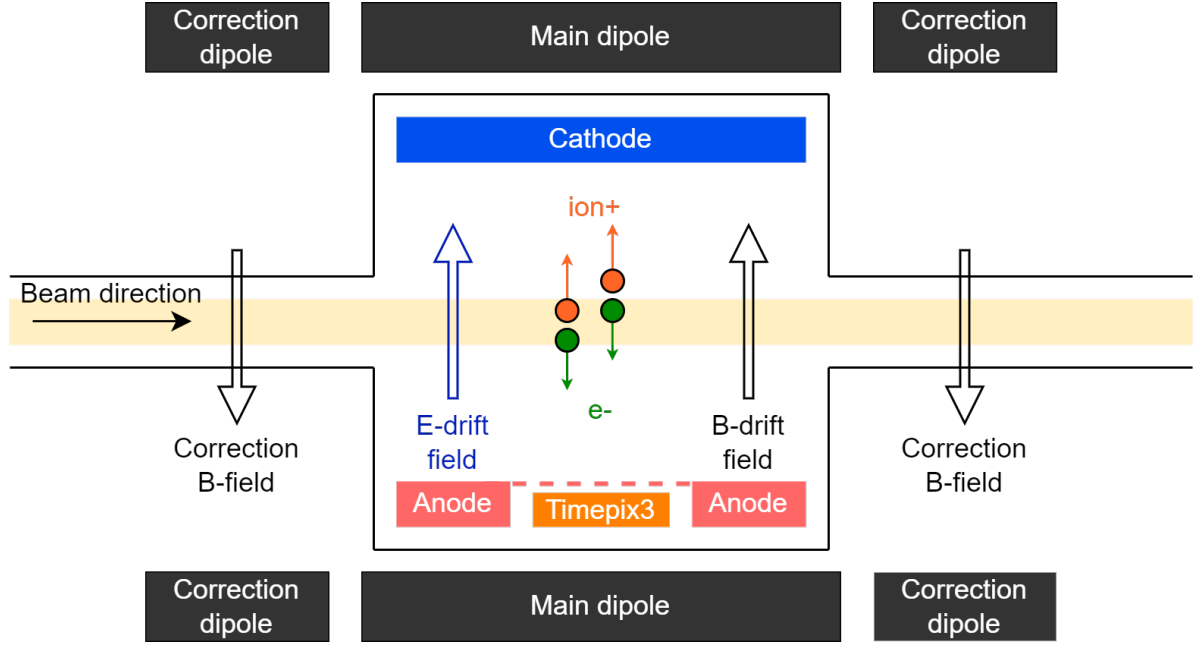


FIGURE 4: CARTOON OF THE BGI INSTRUMENT DESIGN [4].

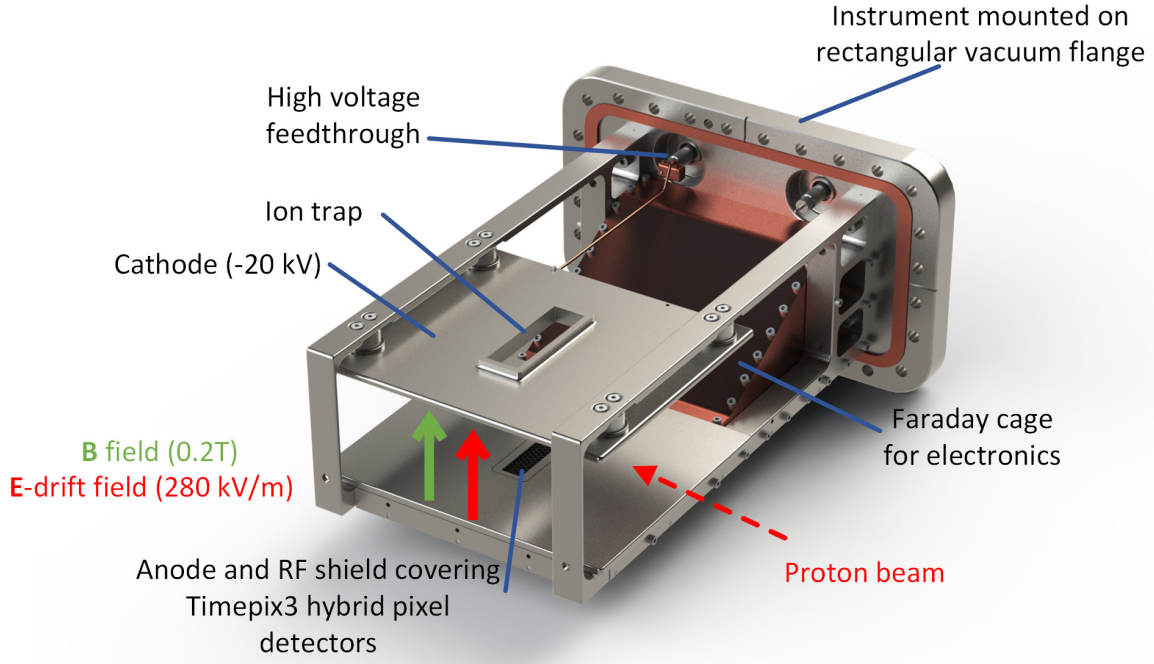


FIGURE 5: RENDERING OF THE BGI INSTRUMENT, ADAPTED FROM [4]

2.2.1 INSTRUMENT DESIGN

The BGI is designed to be a beam profile monitor, based on the detection of residual gas ionisation electrons. The working principle of the instrument is shown in figure 4. The instrument itself is located inside the ultra-high vacuum of the beam-pipe. The pressure in the primary PS vacuum ranges from 10×10^{-8} mbar to 10×10^{-10} mbar. Even though it is a ultra-high vacuum, rest gas atoms are still present. If the beam circulates in the machine and passes the instrument, it will hit these residual gas atoms. Due to the energy of the particles in the beam, the atom will then be ionised, creating a electron-ion pair.

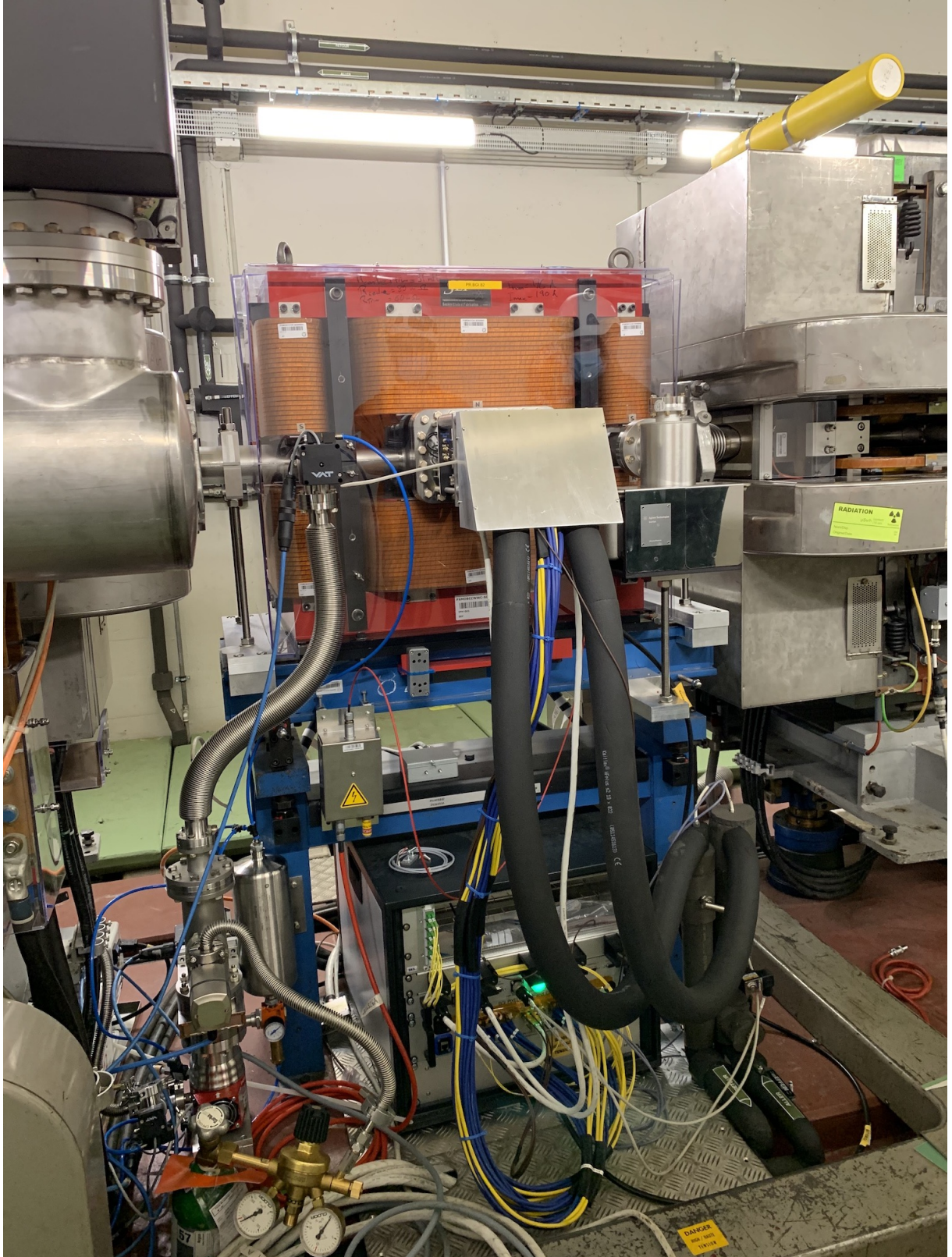


FIGURE 6: INSTALLED BGI IN THE PS-TUNNEL, PICTURE TAKEN FROM [21].

In the main vacuum tank of the instruments two different fields are present. For one, there is an electric field between the Cathode and the Anode. The Cathode is kept at a high negative voltage, which will cause the rest gas ions to be drawn towards it. When the ions finally hit the Cathode, they will create a shower of secondary electrons. To avoid these electrons to interfere

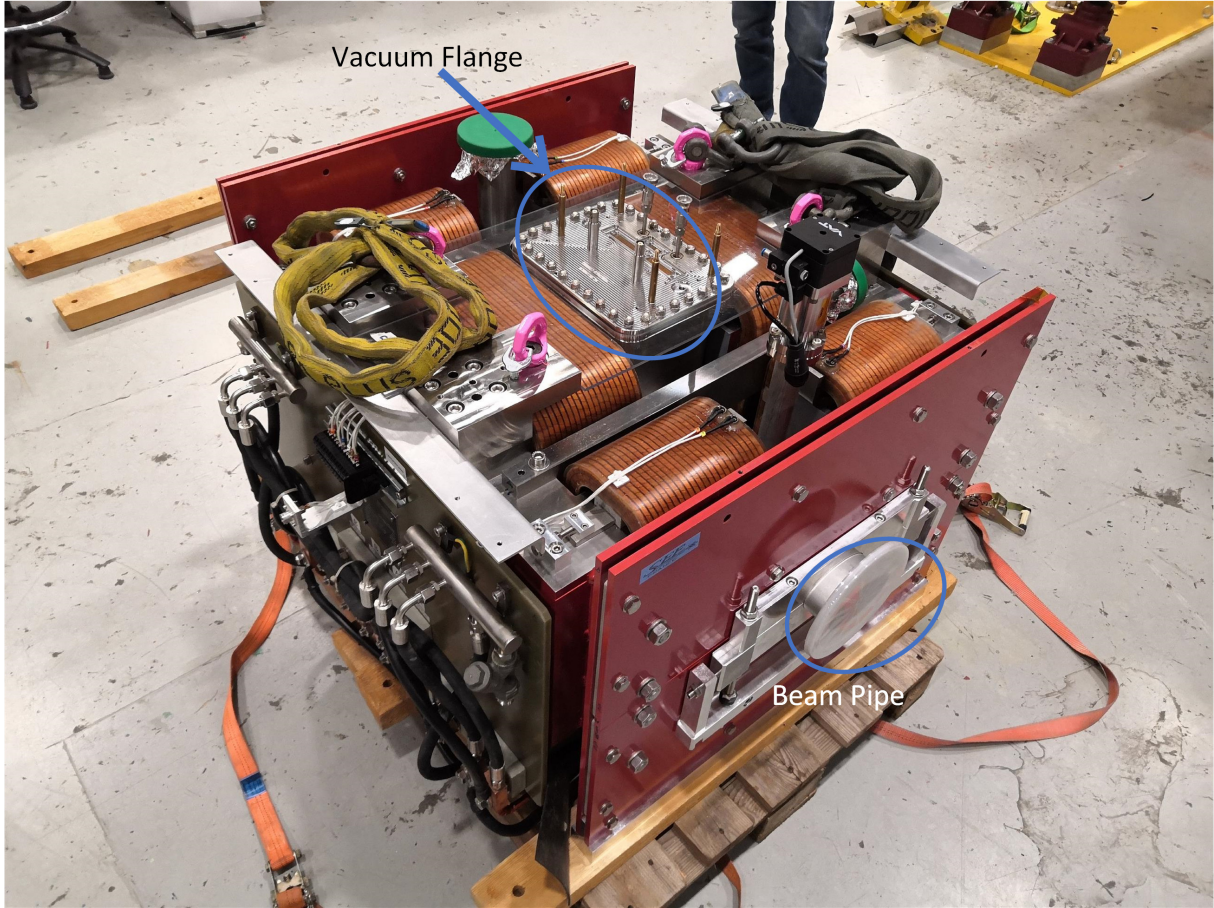


FIGURE 7: REMOVED BGI WITH IT'S MAGNET DURING AN INTERVENTION IN 12/21.

with the measurement of the ionisation electrons, an ion trap is placed above the position where the data taking chips are placed. This trap will cause the ions to impact the Cathode on the back, where the secondary electron shower will not affect the measurement. A rendering of the instrument can be seen in figure 5. The ion trap is visible on top of the instrument inside the Cathode plate.

The electrons are drawn towards the Anode by the electric field. Below the middle of the Anode plate, four Timepix3 chips are located, which will then detect the electrons. The detection is described in section 2.3.2. One of the challenges in this measurement is the trajectory the electron takes from its creation to the detection in the Timepix3 chips. This trajectory has to be kept as straight as possible in order to avoid a smearing effect on the recorded profile.

One of the main contributors to the change of the electron trajectory is the bunch charge. Each bunch creates an electric field, which will change the direction of the electrons. To counteract this change, a magnetic field of 0.2 T is used. This magnetic field, together with the electric field, will cause the electron to move towards the detector in a helical motion according to the Lorentz Force [22]. The stronger the magnetic field is, the smaller the radius of this motion will be.

Using a magnet for this motion provides a different challenge. The applied magnetic field will not only act on the electrons but also on the trajectory of the beam. To minimise the effect of the instrument on the beam, additional correction magnets are placed on both sides of

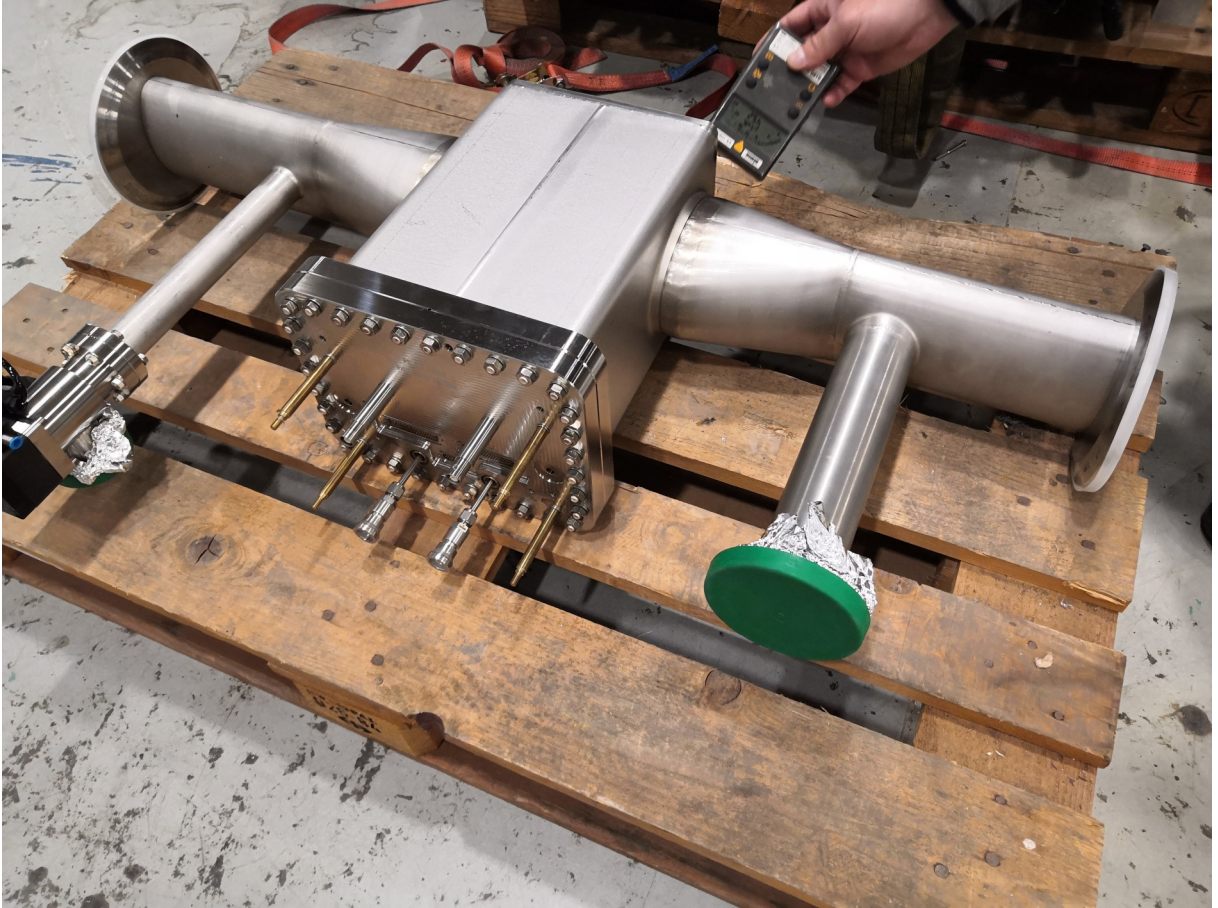


FIGURE 8: VACUUM TANK REMOVED FROM THE MAGNET.

the instrument. These magnets have the same strength but are only half as long as the main magnet with a field in the opposing direction. The beam will therefore not see an alteration in its trajectory. The magnet configuration is shown in figure 11.

Another difficulty lies in the fact that the electrons will experience a different amount of kinetic energy gain, depending on the location where they are created. The closer to the chips the electrons are created, the smaller the amount of acceleration will be, thus reducing the amount of kinetic energy, which they acquire before hitting the sensor. If this energy is reduced, the probability of the electron not penetrating the topmost layer of the sensor is increased, making the electron not detectable by the Timepix3.

To receive a beam profile from this measurement, one BGI device is necessary for the horizontal and one for the vertical plane of the beam. A profile is usually acquired by integrating the measurement over multiple turns of the beam. To decrease the time needed for a usable measurement or to enable bunch-by-bunch measurement, the gas pressure in the instrument can be increased, which directly increases the signal rate that the Timepix3 will receive. To avoid interference with the beam more than necessary, additional pumps are installed next to the instrument, which then remove the added gas.

The design of the instrument brought many challenges with it. The electronics, the Timepix3 chips are connected to, have to be installed inside the main vacuum of the PS, which is being

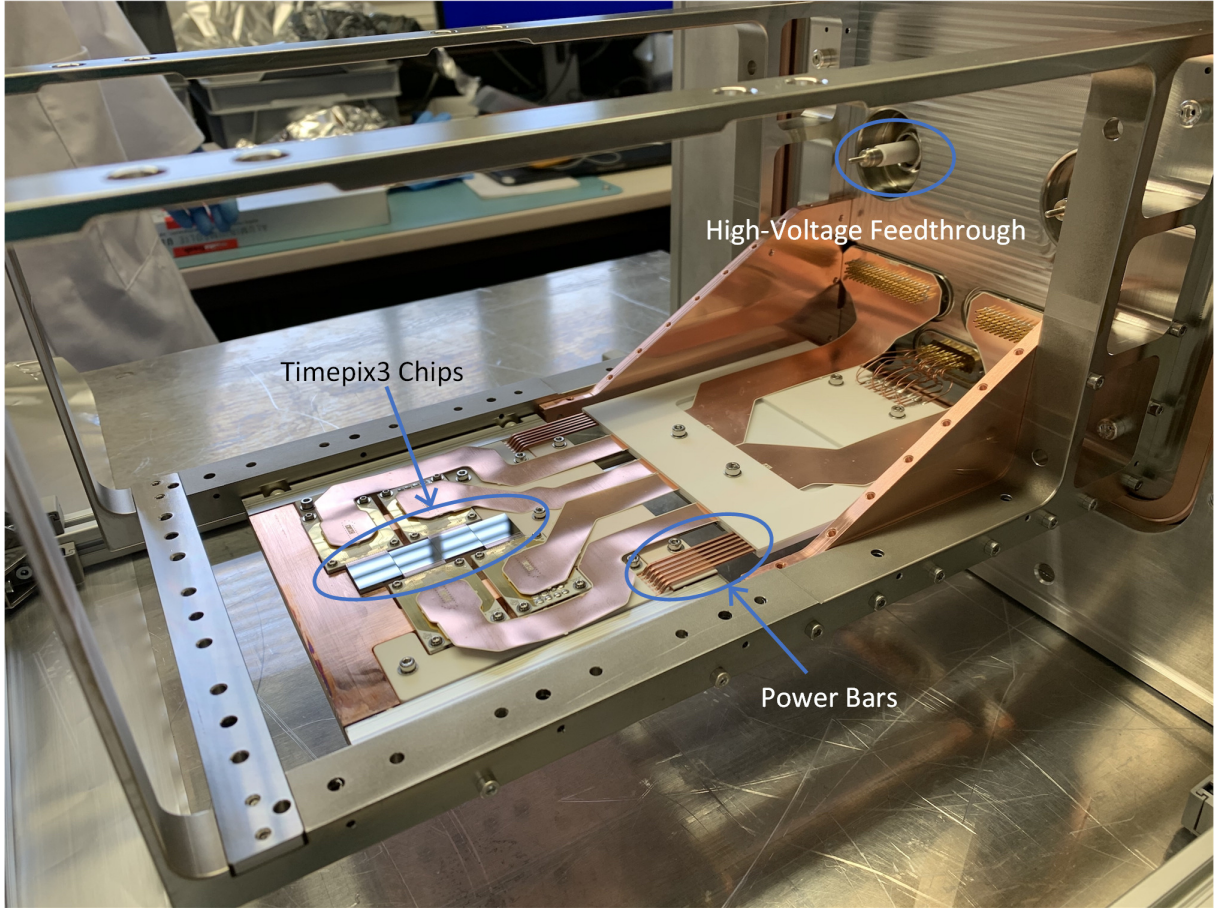


FIGURE 9: IN-VACUUM ELECTRONICS OF THE BGI INSTRUMENT, PICTURE TAKEN FROM [21].

pressured at roughly 10×10^{-8} mbar. Because the beam is circulating in this vacuum, the gas pressure has to be kept as low as possible around the instrument. Introducing additional gas uncontrollably into the vacuum is not acceptable, as it would disturb the beam circulation. Many materials used for electronics design, i.e. FR4, are based on plastics, which will out-gas a lot in this vacuum. Therefore, different strategies had to be applied to the production of these in-vacuum electronics [22].

Due to the beam loss of parts of the beam, the radiation level close to and inside the beam-pipe is significantly higher than the background radiation. Special care has to be taken to any electronics placed in a radiation environment. If, for example, an element performing a calculation is hit by radiation, a SEU can occur, leading to an incorrect calculation result. The ageing of components in this area is also possibly sped up. For these reasons, most of the components used in the tunnel are either radiation hard or radiation tolerant. In components, which were not available as radiation tolerant or hard, special measures had to be taken to ensure the functionality of these components.

One of the main advantages of the BGI is the non-invasiveness of the measurement. As long as no gas is injected, the beam parameters are not changed. Additionally, the instrument is not damaged by the passing beam and thus can also be used with high energy beams, alleviating the main problems with existing beam instrumentation, like the wire scanner. The working principle of the wire scanners is to move a thin wire through the beam. The beam will cause secondary



FIGURE 10: TIMEPIX3 BEAM LOSS MONITOR (BLM) ASSEMBLY. THE CHIP IS LOCATED BELOW ORANGE FOIL, WHICH IS PREVENTING DUST FROM ENTERING THE DEVICE. CONTACTS FOR THE POWER, BIAS VOLTAGE AND DIGITAL TO ANALOG CONVERTER (DAC) READING ARE LOCATED BELOW THE ETHERNET CABLES FOR CONTROL AND DATA CHANNELS.

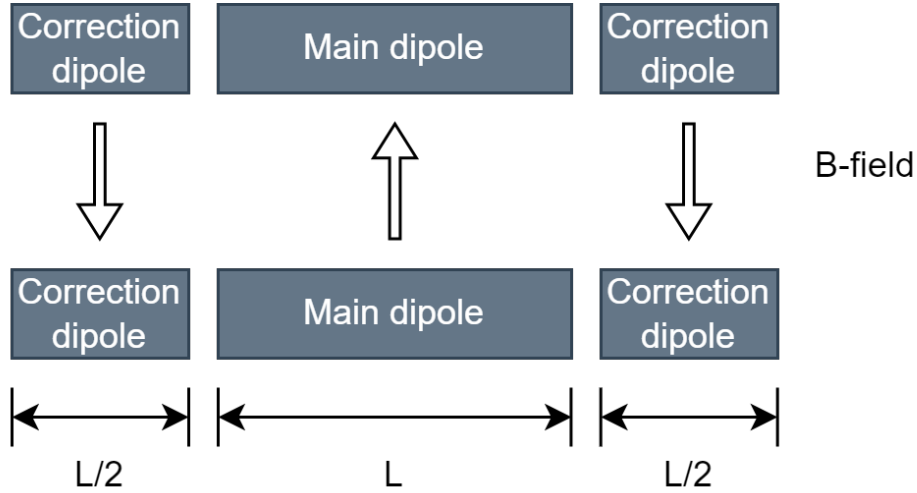


FIGURE 11: MAGNET SETUP OF THE BGI INSTRUMENT.

particles to be emitted from the contact points with the wire. The secondary particles are then detected from outside of the pipe by monitoring elements. By correlating the position of the wire with the amount of secondary particles, one can determine the beam profile. This method will alter the beam trajectory slightly. More importantly the method cannot be applied for high density high energy beams, as the wire would be destroyed too quickly. Other beam instrumentation approaches require the effective destruction of the beam, making them impractical for the use in a physics beam cycle. For a physics beam cycle, the beam has to be kept stable for several hours.

If the instrument is used extensively and takes a lot of data, it is possible that a power glitch occurs. The most likely cause is a drop in the supply voltage, which then causes the chip to not function properly. During the data-taking with a lot of hits, it is possible that the current the

chip draws changes rapidly, causing the voltage to drop [22]. The impact of this glitch is amplified if the instrument is used in the tunnel, and the supply voltage is located a few hundred metres away. The glitch can become problematic in the operational instrument because of the additional heat. As the instrument is placed inside the ultra high vacuum, no heat can be dissipated by convection. The cooling is realised mainly by conduction with cold water, which runs through pipes below the chips themselves. This solution is sufficient during normal operation, but might not be able to sufficiently cool the chips if they are in a reset state for too long.

The installed instrument is shown in figure 6. The orange part is the magnet. In the middle of the magnet, one can see the beam pipe and the vacuum tank, in which the actual instrument is located. Connected to the instrument is the flange board behind the angled plate. The flange board includes signal rerouting from the Ethernet cables in blue and yellow to the flange connector. One can also see the black insulation of the cooling pipes, which are connected to the flange directly. Inside the flange, vacuum feedthroughs are installed to ensure the integrity of both the vacuum and the electrical signals.

In figure 7, one can see the instrument in the magnet. The pictures were taken during a magnet intervention in December 2021. In the front of the picture, one can see the transparent cover, behind which the beam-pipe is located. The instrument and flange can be seen in the middle of the magnet. This instrument is rotated by 90 degrees compared to the previous one, as this picture shows the vertical assembly. In figure 8, one can see the vacuum tank, after it was removed from the magnet. On the sides, the two tubes running perpendicular to the beam-pipe belong to the gas injection system. The flange board hosts the connectors for the high voltage, as well as data and cooling connectors.

The BGI instrument without the Anode or covers installed can be seen in figure 9. In the middle of the picture, one can see the Timepix3 chips. These are then connected with flexible cables to the vacuum flange on the right side. In the top of the flange, one can see the in-vacuum side of the high voltage connectors. In the lower part of the flange, the power connector for the chips is visible. The power is distributed with copper bars, which can be seen exiting the ceramic piece in the middle of the picture.

Apart from the main instrument, which is used directly in the main vacuum of the PS, a smaller Beam Loss Monitor (BLM) device was created. This device is meant to determine the capabilities in the field of the detection of beam loss events with a high precision. The detection of the particles will be performed directly in this device with no additional magnetic or electric fields being used. The BLM consists of one Timepix3 chip, the chip interface and the power converter, which all fit in a small box. A picture of a BLM box is shown in figure 10. The readout chain for the BLM is the same as for the full instrument. This makes the device ideal for software and readout testing in the lab, as it can be easily installed with minimal effort.

2.2.2 READOUT CHAIN

A beam measurement device has to be placed close to the beam and thus some parts will need to be inside the tunnel of the accelerator. The challenge in this environment is that the accelerator

tunnel is a lot more radioactive than the natural radiation. When operating in such an area, one has to be careful about selected components and the selected architecture [22].

If computations are done in a radiated environment, one has to consider that SEUs might happen [23]. In practice, this can invalidate computations or control signals. To reduce the likelihood of this issue, one can perform calculations multiple times in parallel and then compare the results at the end. If the results differ, one of the computing devices suffered from a SEU. In a Triple Modular Redundancy (TMR) setup, the modules are implemented three times. The correct result is defined as the result, which the majority of the modules calculated. Although this technique reduces the probability of an error from happening, it is still possible that the majority of the modules calculated the wrong value.

The readout chain is presented in figure 12. The detector itself is located in the primary vacuum of the beam-pipe. It is connected to the flange board, which is connected directly to the vacuum flange. This flange board is mainly hosting the power converters, which provide the chips with their supply voltage. Additionally, this board relays the electrical connections, which arrive from the chips, onto Ethernet cables, which are then connected to the frontend.

The frontend is located below the magnet of the instrument in the tunnel. It can be seen in figure 6, where it is located in the small rack below the magnet. The flange board is connected to the detector-flange on one side and the frontend on the other side. The connection to the frontend is done with a direct Ethernet connection. The install location of the front as well as the detector requires a reliable hardware configuration as access to the tunnel is limited. While the accelerator is running, no access is possible due to the radiation level. During the technical shutdowns one is able to access the tunnel, so a change to the frontend is technically possible. The technical stops are very limited in length and frequency. The vacuum tank can only be accessed during the shutdowns at the end of the year.

Inside the frontend crate, two different Field Programmable Gate Array (FPGA)s are installed. One is responsible for the control signals, which are sent to the detector. The other FPGA is responsible for the data streams, which are coming from the detector. In these FPGAs, TMR is used to limit the probability of SEUs. For the control FPGA, a ProASIC3 is used [24]. As this is a flash-based FPGA, it does not have to be reprogrammed after a power-cycling. At the same time, it is necessary to reprogram the FPGA with a dedicated programming tool to change the gateway. *Gateway* describes the implementation of the logical gates on an FPGA. As the FPGA is located in the tunnel, this approach limits the flexibility of the gateway. For the data FPGA, a Kintex7 is used. The Kintex7 is not a flash-based FPGA, and thus stores the configuration on volatile memory and has to be reprogrammed after each power cycle. On the other hand, it can be freely reconfigured to change the data-handling [4].

The frontend is connected to the backend with a fibre optics connection. The transceivers used are GBTx radiation tolerant transceivers, which were developed at CERN [25]. The signal is then received by the same transceivers in the backend, which is located outside of the accelerator tunnel. At the receiving end of the fibre optics, variable delays are implemented. These can be adjusted to correct for clock skews between the receiving and sending clock. A clock difference

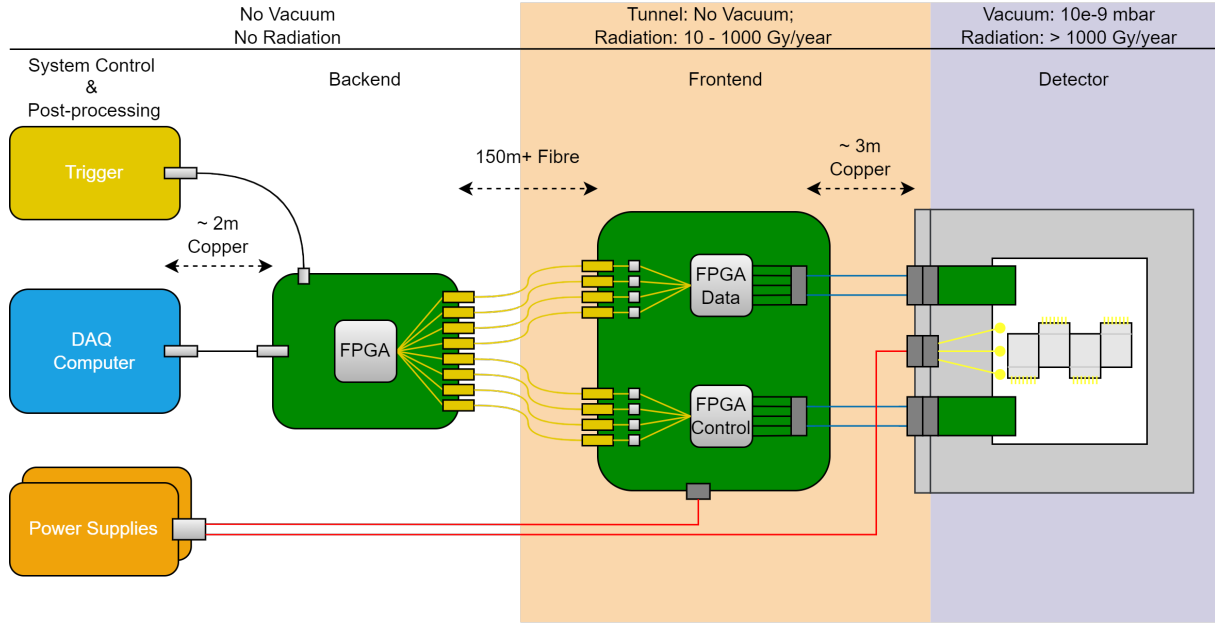


FIGURE 12: READOUT ARCHITECTURE FROM [4].

would lead to an asynchronous receiving and thus data packets are not being read out correctly in the backend.

The radiation level in the backend is not elevated, which allows the use of non-radiation hard components. The backend is then connected to the Front End Computer (FEC) via another Ethernet connection. The backend hosts a Random Access Memory (RAM) to buffer the events received. If the 4 GB of this RAM are filled, newly received events are dropped. The software architecture is explained in section 2.2.3.

2.2.3 SOFTWARE STACK

The software stack has certain CERN-specific requirements, which it needs to comply to. The stack needs to be compatible with the existing infrastructure and software packets used by different users of the instrument. The operators in the CERN Control Center (CCC) for example need to be able to inspect the profiles, measured by the device. To make the operators able to acquire the data, it needs to be published in a Front End Software Architecture (FESA) class. This generalised framework is used for most instruments and control hardware in the accelerator [26]. In order to enable the FESA class to access this data, a C++ Application Programming Interface (API) is provided. The settings for the instrument operation are stored in a common place and are regularly redefined, so that they represent a currently operationally optimal state of the instrument. The GUIs and scripts necessary for the operation of the device will not be discussed, as they are created and maintained by a different section.

In addition to the operational use of the instrument, several other tasks need to be performed. When commissioning new instruments, it is necessary to check if all features of the chips and the instrument work as expected before installing it in the beam-pipe vacuum. For this purpose, a different software is necessary, as the operators do not need to have access to all low level functions.

Both of these functionalities have to be implemented in the TN at CERN. The TN is an internal network, which is used for control and monitoring elements of the accelerator. Only a predefined set of software is supported and available in this network [27].

With these requirements in mind, a modular software architecture is used, where as many components as possible will be shared between different end-points of the information. The stack is displayed in figure 13. Components in red are running on the FEC, while components in blue can run on any computer that has a connection to the TN. On the left side, one can see, for which components the different sections are responsible. Components, which the SY-BI-XEI section is not responsible for, will not be discussed in this thesis. BI stands for the beam instrumentation group, which is organised in the systems department. The XEI section is responsible for **eX**perimental areas, **E**lectron beams and **I**on monitors, while the SY-BI-SW section is responsible for the software in the BI group. BE-OP is the **OP**erational section, which is organised in the **BE**ams department.

The basis for the software is implemented in the Beam Instrumentation PiXeL (BIPXL) Driver, which handles all communication between higher abstraction levels and the device itself. The Panda GUI is then using this driver directly as it is meant to be an expert GUI, which is able to control all features of the detector. Another usage of the driver can be found in the Command Line Interface (CLI) software Grizzly. It can be used to run specific tasks without the need for a GUI. For both of these applications, the driver is executed on the machine running the main application.

The FESA class for the operational use only needs a reduced instruction set of the instrument. This abstraction is done in the BIPXL Software API, which is then used by the FESA classes to interface the BIPXL Driver. The FESA class is running on the FEC, which is connected to the backend FPGA crate.

To implement the communication between the driver and the gateway, the IPbus software is used [28]. It consists of a program on the FEC, which forwards network commands from connected devices to the gateway. In the backend, a gateway module is run on the FPGA, which then translates and interprets the messages and starts the appropriate actions or retrieves data. One of the challenges of using IPbus is the latency. For each read/write request, IPbus will create a new packet, which needs to go through the TN towards the backend and return. If this is done for every data-point individually, the latency is greatly increased and throughput will be greatly decreased. Therefore, a buffer is implemented, which will gather all requests to be sent to the backend. The data in this buffer will then be sent in a smaller number of individual packets, increasing the throughput and lowering the latency for readout and configuration transmissions.

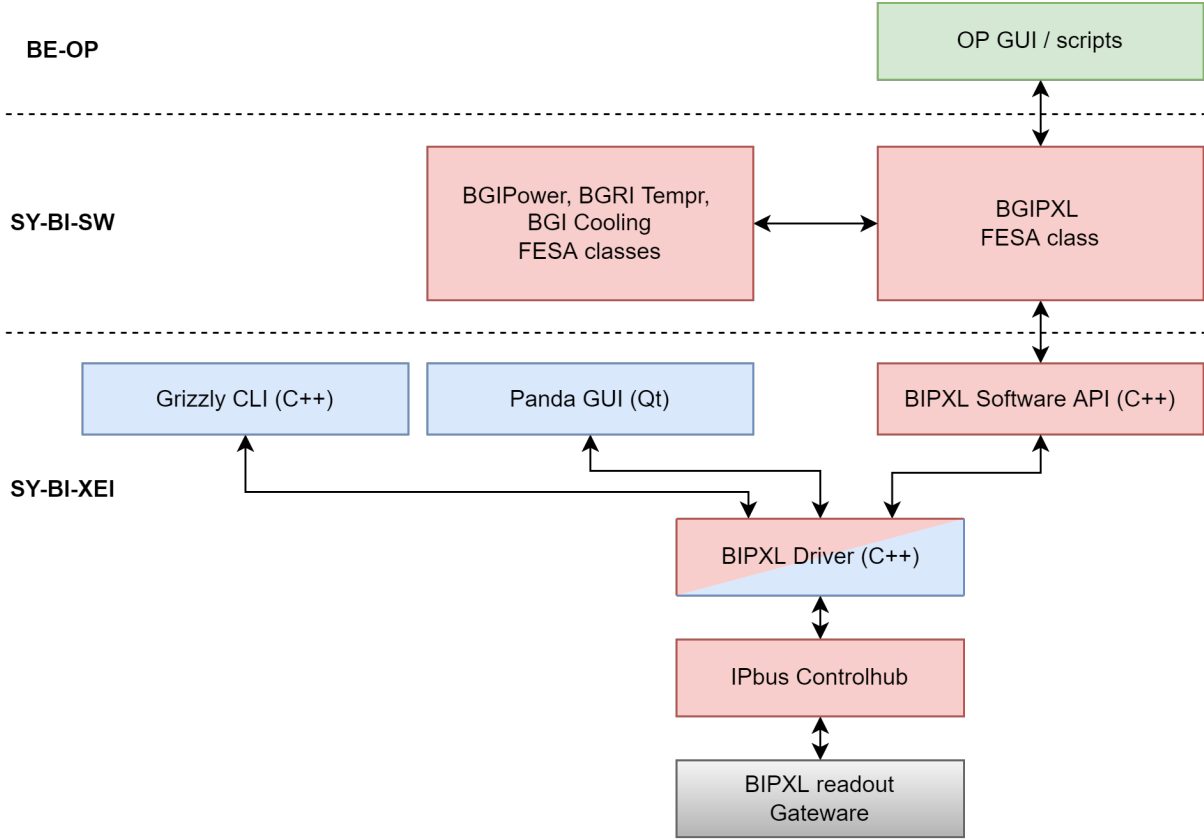


FIGURE 13: SOFTWARE STACK, SIMPLIFIED AFTER [29].

2.3 TIMEPIX3 HYBRID PIXEL DETECTOR

The Timepix3 HPD is an Application-specific Integrated Circuit (ASIC), which can be used to detect a wide array of signals. It can be used for particle detection, X-Ray imaging and other applications, depending on the sensor installed on top of the chip. The chip features a pixel matrix of 256x256 pixels with a pixel pitch of 55 μm . Different aspects of this chip and its usage will be described in this section. The description of the Timepix3 follows the explanations in [30] and [4].

2.3.1 PIXEL FRONT END

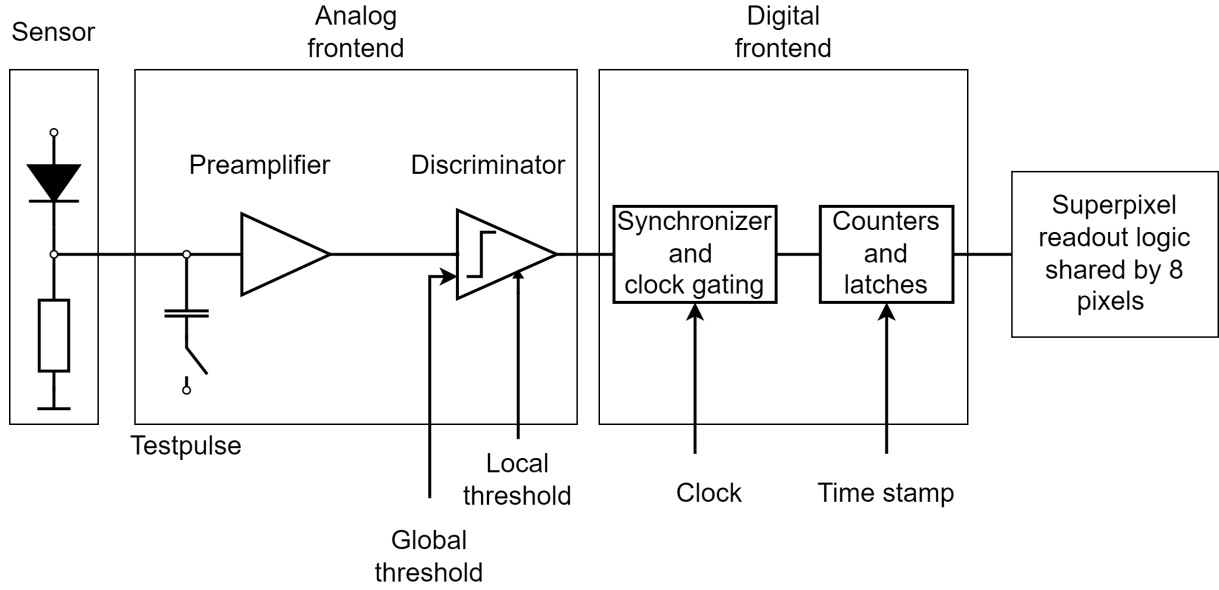


FIGURE 14: TIMEPIX3 PIXEL FRONTEND, SIMPLIFIED AFTER [31].

Each of the pixels in the Timepix3 includes its own frontend with the simplified schematic in figure 14. If a charge arrives at the sensor, it will create a current, which charges a capacitance and gets preamplified. Afterwards, the discriminator will decide if the signal propagates. Only if the current is above a combined threshold of the local and global thresholds, the signal will be transmitted into the digital frontend. The global threshold is a combination of a coarse and a fine setting. Also included in the analog frontend is a Test Pulse (TP). This can be used to charge the capacitor with a known voltage and thus injecting a known charge into the frontend directly.

In the digital frontend, the data of the event is taken by the counters and latches after clocking and synchronisation were applied. The next step of the chip readout chain is shared between eight pixels. This union of pixels is called a Super Pixel (SP). Inside this SP, a FIFO for data transmission as well as a Voltage Controlled Oscillator (VCO) for more precise timing information are included [31].

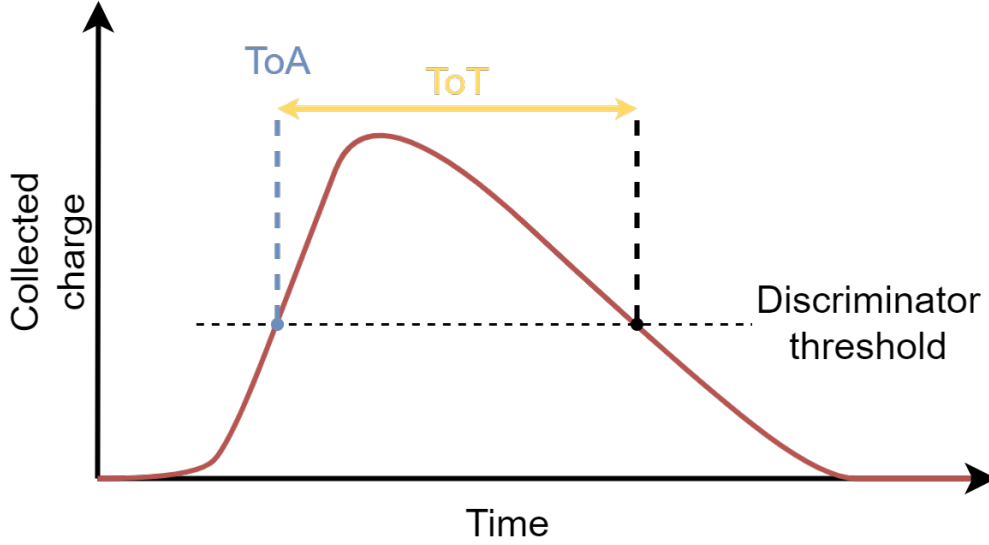


FIGURE 15: VISUALISATION OF TIME OF ARRIVAL (ToA) AND TIME OVER THRESHOLD (ToT).

After the frontend received charge from the sensor, the charge is preamplified and registered by the discriminator. From the triggering times of this discriminator, the digital frontend will then save two values, the Time of Arrival (ToA) and Time over Threshold (ToT). The ToA describes the time, when the charge level first goes over the threshold level of the discriminator. The ToT on the other hand describes the time it took the charge to reach the threshold level again after the first transition. Both values are visualised in figure 15. If another charged particle arrives before the collected charge hits the threshold again, the hits will not be distinguishable. Without a threshold, the digital frontend would pick up a lot of noise.

The pixel frontend will collect different data depending on the acquisition mode selected. One possibility is to use the ToA and ToT mode, in which the data of both the ToA and ToT is collected for each event. Additionally, the chip can be used in only ToA mode, in which the ToT data of an event is not sent to the readout device. Another possibility is the Event Count (EC) & integrated ToT (iToT) mode. This mode is used in the equalisation procedure described in 2.3.4.

2.3.2 SILICON SENSOR

In the BGI instrument, the Timepix3 HPD is used to detect ionisation electrons. To implement the first stage of this task, a silicon sensor is bonded on top of the Timepix3 ASIC. This sensor will convert the energy of an ionisation electron into a usable signal for the Timepix3 chip [4]. For the detection of different particles, the sensor can be substituted for a different one, hence the name, **hybrid** pixel detector.

A semiconductor differentiates itself from a conductor and an insulator in that it is not permanently conducting. In figure 16 one can see the valence band in green and the conduction band in red. For a material to be conductive, it needs to have electrons in the conduction band. The band gap is defined as the difference between the green and the red zone. The maximum energy for the valence band is marked with E_v , while the minimum energy for the conduction band is marked with E_c . In a metal, the electrons have a high probability of being in the

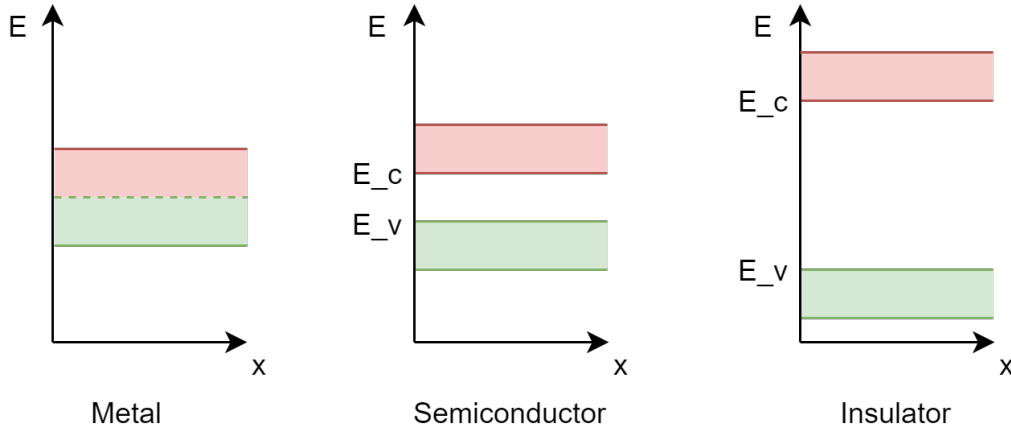


FIGURE 16: CONDUCTION BAND ENERGIES FOR DIFFERENT MATERIALS FROM [32].

conduction band, therefore metal is always conducting. The bandgap is nonexistent and the electrons can freely change between the bands.

In a semiconductor on the other hand, a bandgap exists. The electrons need to be excited by an energy equal to or higher than the bandgap in order to enter the conducting state. As long as a temperature above 0 K is present, the thermal energy will randomly cause electrons to be excited into the conducting band, thus making the material semi-conducting. The amount of electrons to be excited is proportional to the temperature. In an insulator the band gap is larger so that it is not possible to excite the material enough to become conductive.

To change the properties of the semiconductor, one can insert artificial impurities into the material. On one hand donor material can be doped into the semiconductor. The atoms of these elements have an additional electron compared to the atoms of the semiconductor, which acts as a charge carrier. On the other hand an acceptor material can be used, which has one less electron than the semiconductor. The atom is able to accept an electron from the surrounding atoms, resulting in a hole, also acting as a charge carrier.

If one combines the two doping possibilities into a single device, a PN-junction is created. The P area is positively doped, or acceptor material is added, while the N part is negatively doped or donor material is added. The electrons from the N doped area can drift towards the opposing side, where they will recombine with the holes of the P doped area charge carriers present. Due to the present temperatures in the semiconductor, some of the electrons will be in the excited conducting state and also recombine with the charge carriers in the P doped area. This will generate a depletion region in the middle, where no charge carriers are present. The recombination will cause charge difference in the depletion region and therefore an electric field will build up [33]. In figure 17, one can see the principle of the created depletion region. The electrons of the donor atoms recombine with the holes from the acceptor atoms. The remaining ions will then create the depletion zone, where no free charge carriers are present despite the doping.

In figure 18 a schematic display of the sensor is shown. The shown sensor is a P on N sensor, meaning that the P doped region is embedded within the N-bulk of the sensor. In this case, the

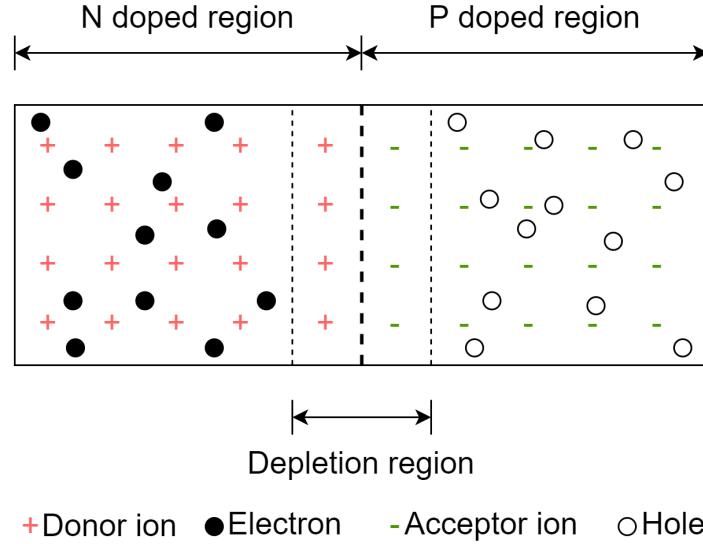


FIGURE 17: PN-JUNCTION IN SEMICONDUCTORS FROM [33].

sensor needs a positive bias voltage to work. This sensor type is discussed in detail in the next paragraph. An N on P sensor has the same working principle, but needs a negative bias voltage. It also collects electrons instead of holes, which alters the polarity of the generated signal. The main performance difference is the higher drift speed of electrons at the same bias voltage [4].

For the operation of the sensor, a reverse bias voltage is applied to the N+ region at the top. This will increase the size of the depletion region between the N+ and the P+ doped regions in the N-bulk of the sensor significantly. If an electron now hits the sensor, it will create secondary electron-hole pairs in this depletion region. The amount of created pairs is proportional to the energy of the electron. The electrons will have an average energy of 10 keV depending on the location where they were generated. From a 10 keV electron, roughly 2800 electron hole pairs are generated [4]. Because of the applied bias voltage, the holes will then be drawn towards the P+ side of the sensor, while the electrons are drawn towards the N+ side. When the hole arrives at the P+ side, the charge will travel through the bump bond into the Timepix3 readout, where it will be detected [4].

If no particles hit the sensor, no electron hole pairs will be generated by this effect. Nevertheless, as the temperature is still above 0 K, charge carriers will still be excited into the conduction band. As an electric field is applied via the bias, these carriers will be drawn towards the detector and possibly detected by the sensor. This effect is the basis for the noise floor reference signal in the equalisation.

One of the challenges is that the charge generation can only happen in the depletion region. Outside of the depletion region, no electric field exists that would pull the generated electron hole-pair apart, and they would therefore recombine in-place. The main challenge is therefore to increase the size of the depletion region as much as possible by increasing the bias voltage. After the depletion region in the sensor occupies the whole volume of a pixel, the field can be increased further. The increased field will make the probability of a recombination less likely, as the electric force pulling the electron-hole pair apart is larger. Generally, one can say the yield of

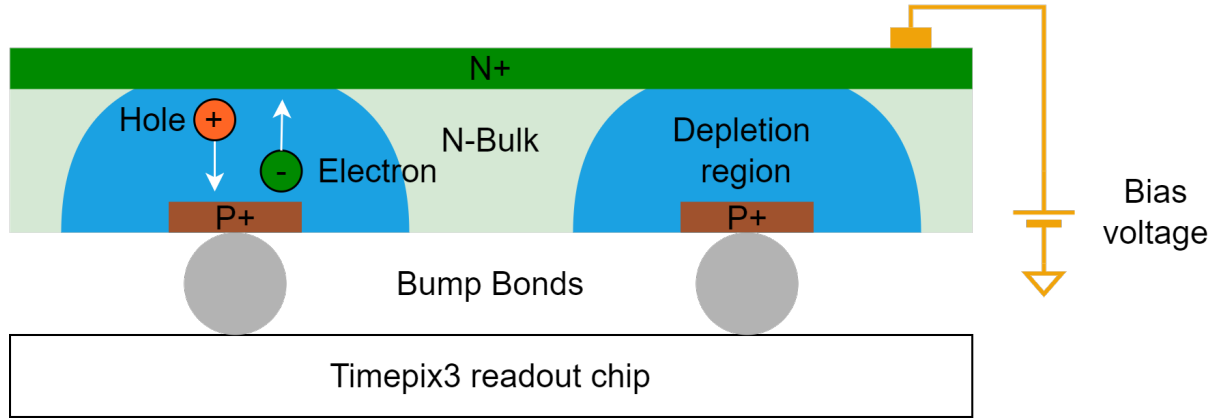


FIGURE 18: VISUALISATION OF A P ON N SILICON SENSOR, FOLLOWING [4].

signal from electrons hitting the sensor is proportional to the applied bias voltage. The drift speed of the charge carriers will be increased by the electric field as well, reducing the latency of the measurement [4].

2.3.3 CHIP PERIPHERY

The Timepix3 publishes the acquired data on a maximum 8 Scalable Low Voltage Signaling (SLVS) output channels. Each of these channels can run at a maximum speed of 320 MHz in Double Data-Rate (DDR). If a channel is activated, the data is transmitted using 8b/10b encoding [34]. Within the 8b/10b encoding of the Timepix3, a comma symbol is transmitted, which can be used for clock recovery [4]. As long as the hit rate stays below 40 Mhits/s/cm² spread out accross the chip, all events can be read out.

The Timepix3 features two different readout modes. For one, there is the sequential mode, which reads out the entire collected data when the chip receives an external trigger. Alternatively, a data-driven mode can be used, in which the data is sent to the readout chain upon creation of a new data event [34].

In addition to the readout elements, the on-chip periphery also includes the Digital to Analog Converter (DAC)s, which are used to control the chip voltages. One of the most important ones is the Vfbk voltage. This voltage serves as a reference to many components on the chip as well as a reference for the threshold voltages in the analog frontend. Voltages of the different DACs can be read out from an output pin on the chip. Another important feature in the periphery is the possibility to read the chips temperature. While the Timepix3 has no direct temperature measurement, the voltages of two DACs can be measured in an alternating fashion. From the difference and a conversion factor one can determine the approximate temperature on the chip.

2.3.4 EQUALISATION OF THE PIXEL MATRIX

The explanations in this section follow the descriptions in [4]. During the production of the chip, each of the 65536 pixels will have slightly different properties. Additionally, the ageing of the on-chip electronics will affect different areas and thus different pixels differently. In a particle accelerator, high energy particles hitting the chip will further change the parameters of

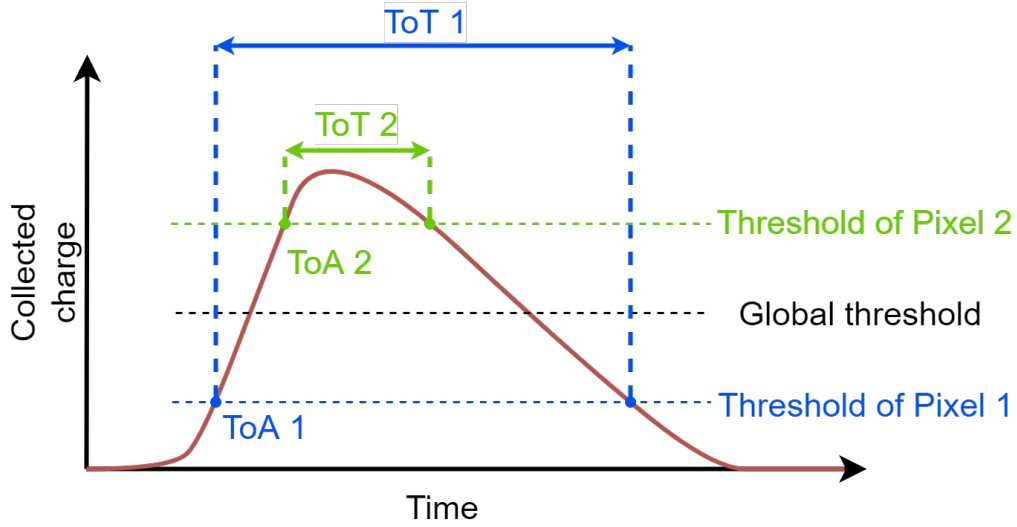


FIGURE 19: THRESHOLD MISMATCH BETWEEN TWO PIXELS AND THE RESULTING DIFFERENCE IN ToA AND ToT VALUES FROM [22].

the analog pixel frontend. The additional radiation present in the operational environment is also affecting the chip's long-term performance.

One of the most important components, in which these effects induce a problem, are the discriminators. If the threshold of the discriminators in the different pixels is not at the same value for a given global threshold, a mismatch in both measured ToA and ToT will be visible. The bigger the differences in the discriminators are, the bigger the differences in the measured data will be. If the difference gets larger and larger, one pixel might not detect charge, which is clearly detected by a different pixel. The effects of the different threshold values of different chips are shown in figure 19.

To reduce the effect of these differences, each pixel includes a 4 bit local threshold DAC. The procedure to adjust these local thresholds is called equalisation. The procedure described follows the description in [35].

Before explaining the equalisation itself, it is important to be familiar with some prerequisites. The explanations in the following paragraph are given for a P on N sensor. For an N on P sensor, the increasing and decreasing operations are inverted but the principle stays the same. The standard building block of the equalisation is the so-called threshold scan. A threshold scan is run in the EC & iToT mode of the chip. This mode is used, as only the total time above the threshold is relevant and not the ToT value of the last registered hit. During a threshold scan, the local threshold of all pixels is kept at a constant value. The global threshold is then increased, while opening the shutter for each value. After the shutter opening finished, all events are read out in sequential readout mode. This is done to make effective use of the iToT value. A data-driven readout mode would deactivate the pixel for some time, which would reduce the effectiveness of the equalisation. Additionally, the mode minimises the amount of readout operations per shutter opening. When increasing the global threshold, the chip will get more and more sensitive. At some point, the chip will reach a sensitivity level, which is inside the thermal noise floor of the sensor, which was described in section 2.3.2. At the beginning of this scan the

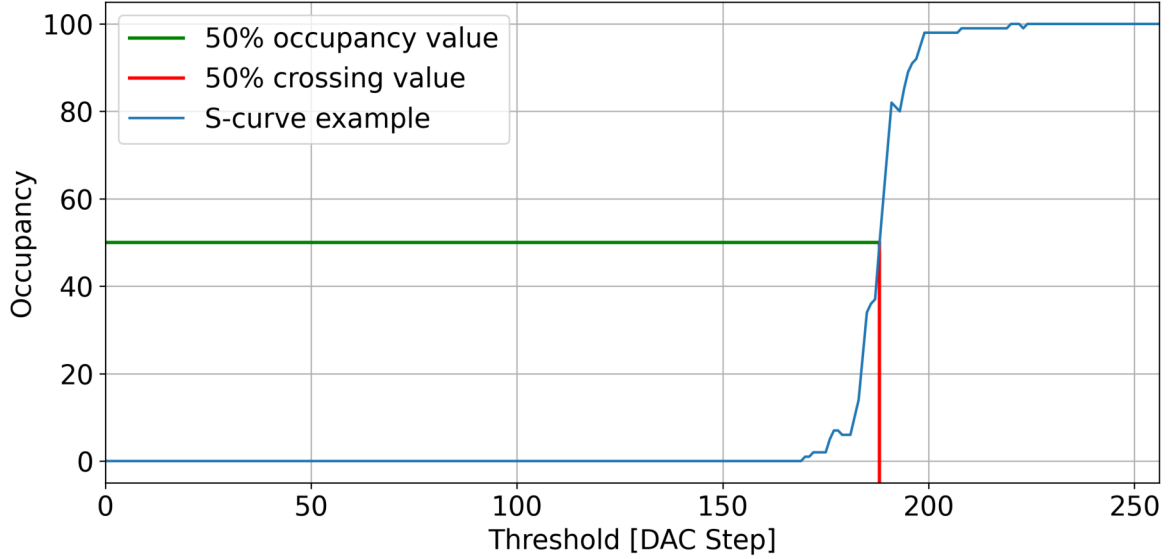


FIGURE 20: THRESHOLD SCAN FOR A SINGLE PIXEL. THE RED LINE MARKS THE VALUE, WHERE THE PIXEL HAS HIT AN OCCUPANCY OF 50%.

iToT value of each pixel is usually at a level close to zero, and will rise to 100% of the shutter opening time by the end of the scan. The occupancy is defined as

$$occupancy_{pixel} = \frac{iToT_{pixel}}{T_{shutter}} \quad (1)$$

in order to ease the calculations inside the equalisation and to allow for an easy change of the shutter opening time.

To compare the different pixels with each other, a reference point is necessary. It is chosen as the global threshold value, which is closest to the crossing point of the 50% occupancy of the pixel. The reference point is displayed in figure 20. The 50% value is chosen, because the maximum will only be reached asymptotically, making it a lot harder to detect than the 50% value. To display the crossing points on the whole chip, the result is summarised in a histogram, containing the amount of pixels, which had their 50% crossing at a certain value of the global threshold. An example of this histogram can be seen in figure 21.

The equalisation procedure involves four main steps. First a threshold scan is conducted at the minimum local threshold value of each pixel. Secondly, another scan is performed at the maximum of the local threshold. The midpoint of the mean values of the two resulting histograms is chosen as the target value. The goal is to change the local threshold values of all pixels to a value, so that they have their 50% crossing value at this global threshold. A linear interpolation is conducted for each pixel to determine the local threshold value, which is closest to this target. The interpolation is shown in figure 22. After the first two steps, the crossing values of each pixel at a local threshold of 0x0 and 0xF are known. These are the values resulting from the 4 bit resolution of the local threshold DAC. Assuming a linear increase in the crossing

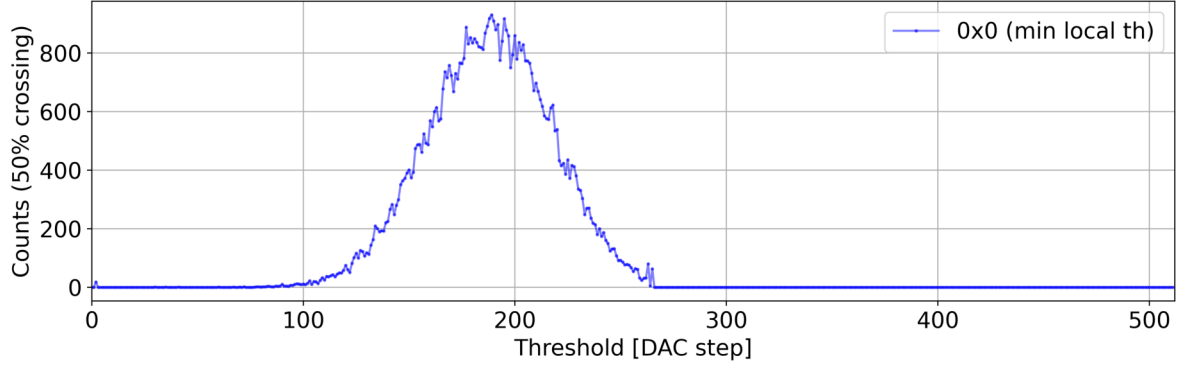


FIGURE 21: HISTOGRAM OF THE AMOUNT OF PIXELS, WHICH HAD THE 50% OCCUPANCY CROSSING AT A CERTAIN GLOBAL THRESHOLD VALUE.

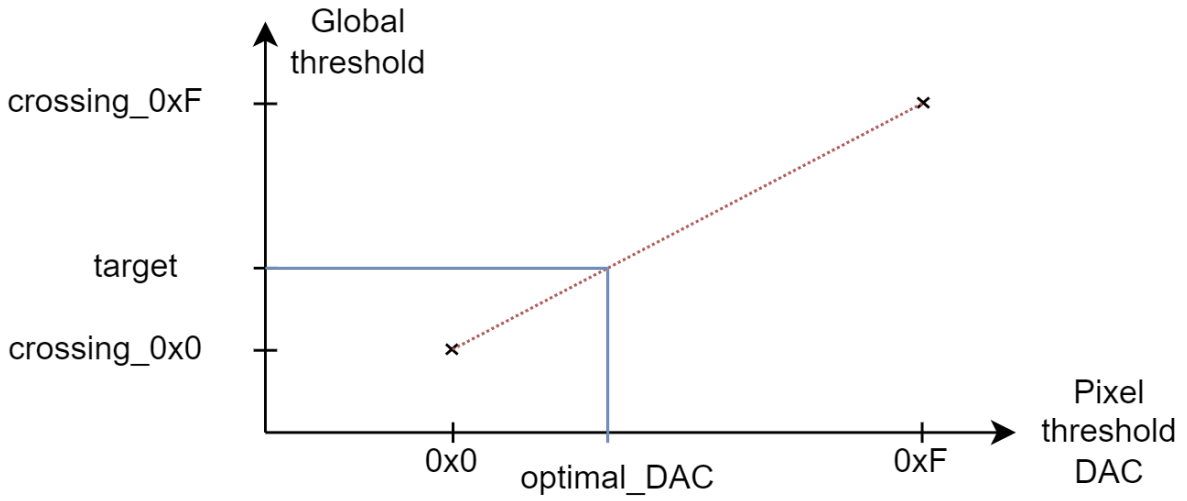


FIGURE 22: LINEAR INTERPOLATION AFTER THE FIRST TWO THRESHOLD SCANS.

value, one can then determine the slope of the crossing value increase with

$$\alpha_{crossing} = \frac{crossing_0xF - crossing_0x0}{0xF} \quad (2)$$

After having calculated the slope, one can estimate the optimal pixel DAC value for a set target with

$$\begin{aligned} target &= \alpha_{crossing} * optimal_DAC + crossing_0x0 \\ \Rightarrow \\ optimal_DAC &= \frac{target - crossing_0x0}{\alpha_{crossing}}. \end{aligned} \quad (3)$$

The calculated value will in reality not result in an optimal value, as the threshold does most likely not follow a linear curve.

Afterwards, in the fourth step, a fine-tuning is conducted, in which an algorithm tries to get the pixel's crossing value closer to the selected target. In figure 23 the most important steps of the equalisation are shown with their respective histograms.

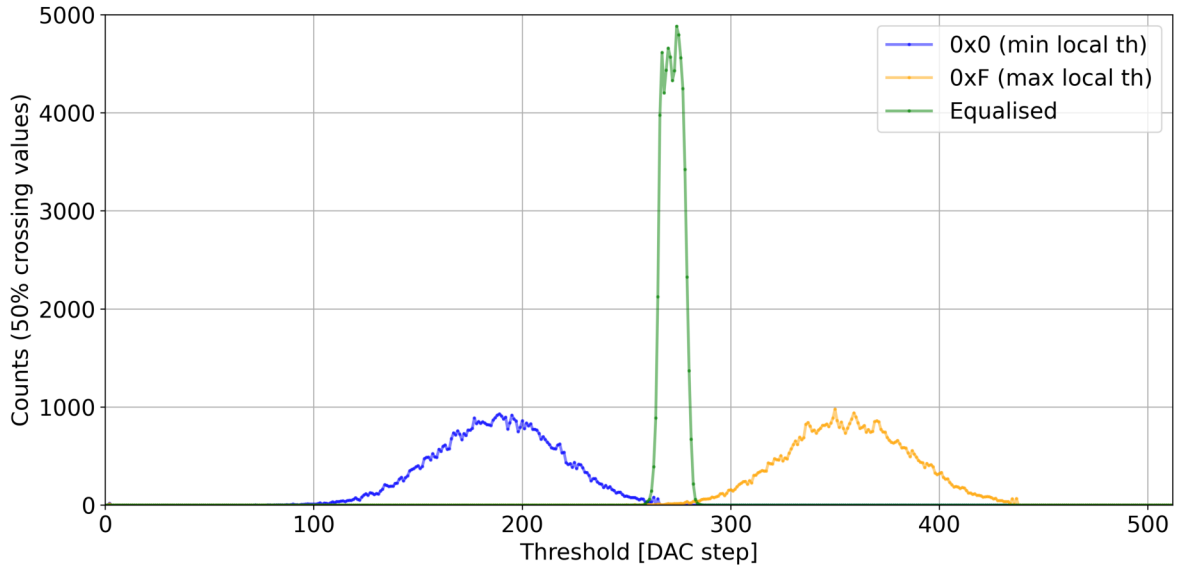


FIGURE 23: HISTOGRAMS DURING THE DIFFERENT STEPS OF THE EQUALISATION.

During the fine-tuning stages, the algorithm will check each pixel's crossing position in comparison to the target. If the crossing point is below the target, the local threshold value will be increased or decreased by one, if possible.

A big challenge during the equalisation is the temperature stability of the chip. Because the noise floor is used as a reference, the chip needs to be in a stable state throughout the procedure. If the temperature of the chip changes, the noise floor and as such also the reference of the equalisation will be changed. When this passes a certain amount, the equalisation result will not be valid or not optimal. The power consumption of the chip will naturally change during the equalisation, as it consists of constant data-taking from the chip. Therefore, the discriminators and timers will need additional current, which in turn increases the temperature of the chip. To minimise the severity of this effect, only a part of the pixel matrix is equalised at once, reducing the amount of current drawn, thus making the temperature of the chip more stable. The configuration value associated with this is the checkerboard level. A checkerboard level of 2 implies that half of the matrix is equalised at once.

To determine the quality of the result of the equalisation, the standard deviation of the crossing histogram is analysed at the end of the procedure. A large standard deviation corresponds to a bad equalisation result, as the crossing points of the pixels still vastly differ. An ideal equalisation result would have all crossing points at a single value of the global threshold.

One of the main limitations of the equalisation is the resolution of the local pixel DAC. It forms a lower limit to the reachable equalisation accuracy. Another limitation is the non-linearity of the discriminator for different energy levels. This limitation could be addressed by doing an energy calibration, which will not be discussed in this thesis [35].

During the equalisation, the bias voltage of the sensor needs to be turned on. If the bias voltage is not turned on, no reference voltage for the noise is defined and the sensor potential is floating. This would prevent the reliable detection of noise signal from the sensor.

2.4 ZYNQ MPSoC

2.4.1 PROCESSOR ARCHITECTURE

The Zynq MPSoC from Xilinx is a single chip with multiple different processing cores inside. The simplified architecture of the chip is displayed in figure 24. It incorporates general purpose Application Processing Unit (APU), a Real-Time Processing Unit (RPU) and Programmable Logic (PL) section. The APU consists of a quad-core ARM Cortex-A53, which can be used to run a standard Linux. The RPU consists of two ARM Cortex-R5 cores, which can be programmed using Free Real-Time Operating System (FreeRTOS). Another option is to create a bare-metal program on these devices. FreeRTOS is described in section 2.4.2 [36].

The Linux distribution, which will be used in this project, is PetaLinux [37]. It is provided by Xilinx and includes an easy-to-use framework for the configuration and building of an Operating System (OS) to be used on the Zynq MPSoC as well as other Xilinx devices. Alongside with the OS itself, a Software Development Kit (SDK) can be created, which is used during the implementation of software for this specific system. PetaLinux is also able to embed gateway for the PL as well as firmware for the RPU in the build process, so that they can be loaded automatically during the startup of the Processing Subsystem Unit (PSU). *Software* is used to describe the implementations on the APU or any other general purpose Central Processing Unit (CPU), while *firmware* is used to describe the implementation on the RPU.

The advantage of this architecture is that the different components can communicate efficiently and fast with each other, as they are located on the same chip and connected to the same Advanced eXtensible Interface (AXI) bus. It allows for a more flexible separation of different sub-tasks of an algorithm into different hardware. Also, hardware acceleration of certain computations can be implemented more easily and efficiently [36].

One of the advantages of having the RPU on the chip as well is the gained level of predictability. This can be very helpful for controlling and monitoring tasks. The RPU is connected both to the Dynamic Random Access Memory (DRAM) and Tightly Coupled Memory (TCM). While the TCM is implemented as On-Chip Memory (OCM), the DRAM consists of one external DDR3/4 module. The TCM can only be accessed by the RPU by default, but can also be mapped into the general address space. The size of the TCM is limited to 128 kB per RPU core [38].

The RPU has two different operating modes. The cores can either run in Lock-Step mode or in free-running mode. While in locked step, one of the cores is disabled, while the other one is running normally. In this mode, the TCM of the non-running core is added to the running core. The modes cannot be changed while the processor is not in reset state. In the free-running mode, the scheduler will independently schedule jobs on both cores of the RPU. This can mean, that a task with a lower priority is run while a task with a higher priority is run on a different core, adding to the complexity of the implementation [39].

The programming and configuration of the different parts of the MPSoC is done in multiple stages during the boot sequence. During the boot sequence, the APU, RPU and PL can be initialised. The RPU can either be configured to run individually or in tandem with the APU.

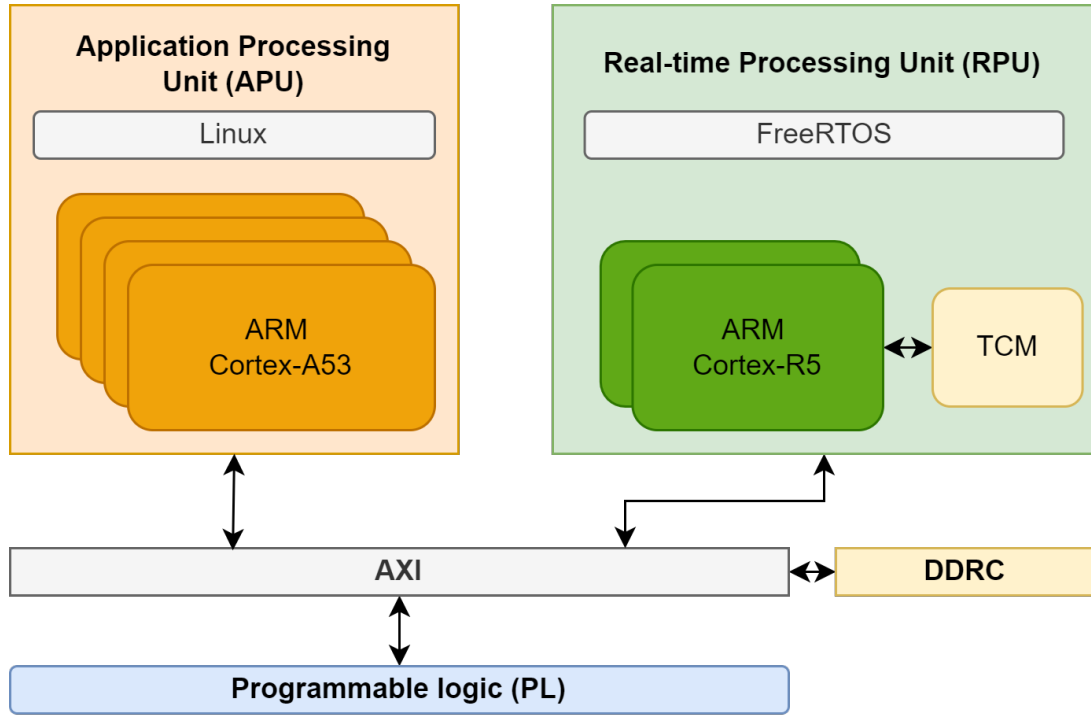


FIGURE 24: ZYNQ MPSoC ARCHITECTURE, SIMPLIFIED AFTER [40].

After the booting sequence, the RPU firmware can be loaded manually from the APU through the remoteproc framework. The PL gateware can be loaded during the startup of PetaLinux [41].

2.4.2 FREERTOS

FreeRTOS is built on the basis of a Real-Time (RT) kernel, and it can be used as an operating system for a micro-controller or CPU. The main advantage of FreeRTOS is that all tasks can be specifically programmed, without any default tasks running in the background. The task scheduler can be freely configured, which leads to deterministic program sequences and priorities. This can greatly improve the safety and predictability of a program compared to the program running on a general purpose kernel. In the general purpose kernel, background processes are running and the scheduler can more freely decide on which programs to run [42].

One of the limitations of the RT core is, that the memory, which can be used for the program, is limited. The RT-core in the MPSoC technically has access to both the TCM and the DRAM. One of the challenges with using the DRAM is that other hardware components can also write to it, which can lead to problems with data allocation within the program. In order to address this issue one would need to predefine a memory area in the other components. When making changes to this allocation, it would mean to also change the other components necessarily, making the approach a lot less flexible. Another issue with the DRAM is the speed of the connectivity. While the TCM is connected directly to the RPU, the DRAM has to be accessed through the DDR controller. It therefore has a big advantage to load the program into the TCM and not the DRAM. The main limitation in this approach is the size of the TCM, which the program size cannot exceed. The TCM has a size of 128 kB per RPU core, which is combined to 256 kB in lock-step mode.

Due to the limited amount of memory available to the RPU program size, it is also important to use a lightweight operating system. For this purpose, FreeRTOS is a good choice, as the binary size will typically be between 6 kB and 12 kB for the kernel, leaving enough space for the task definition and program to run [43]. If more memory is needed during run-time, the DRAM can also be used, with the limitations stated before. Another limitation of using FreeRTOS or bare-metal on the RPU is the fact that drivers for peripherals or accessories only have a limited availability. The access to all peripherals is possible on a hardware level, but has to be implemented as a driver for most devices.

FreeRTOS is also usable in the Eclipse based Vitis software development environment by Xilinx. As Vitis is the recommended SDK of Xilinx to create software for both the APU and RPU on the MPSoC, many software packages and libraries are readily available and configurable. The debugging of FreeRTOS programs can also be done in the software [44]. The debugging procedures are described in section 4.4.

2.4.3 LIBMETAL

The Libmetal library can be used to communicate between two different devices via a shared memory or for inter-processor communication in an Asymmetric Multi-Processing (AMP) environment. In the case of the MPSoC, the devices, which will communicate with each other, are the APU and the RPU. The shared memory will be defined inside the DRAM of the board, as the memory region needs to be known and accessible to both units at the start of the program [45]. In the case of the MPSoC, the memory region needs to be defined in the device tree of the PetaLinux. In FreeRTOS, the memory region can be defined in a simple header-file. PetaLinux is a Linux distribution designed by Xilinx for the use on different System on Chip (SoC) platforms.

In addition to the shared memory, an interrupt needs to be defined on each side. The interrupt will be pulled, as soon as the opposing side transmits data, so that the receiving device knows the data was transferred into the shared memory. Afterwards, the data can be copied into an internal buffer and appropriate action can be taken [45].

2.5 QT FRAMEWORK

The Qt framework was used for the implementation of the previous expert GUI, the Panda GUI [46]. Within this thesis, the Python variant of Qt, PyQt, will be used in order to create an expert GUI, which is able to run in the TN. PyQt is very similar to Qt, as it is implemented as a binding to the C++ functions of Qt.

The GUI design with Qt is separated into two main steps. First, one creates the definition of the forms to be used in the graphical interface. These definitions are saved using an xml-like format in a *.ui* file. This *.ui* file is then translated by the User Interface Compiler (UIC) into the target language, i.e. Python. The *.ui* files can be created in the Qt-designer, which generally eases the design phase of the graphical frontend. One can also directly implement a *.ui* file or write the definitions directly into a Python source file. While possible, this approach generally takes longer, as it takes more time to visualise the result of the definitions. Within this thesis, the Qt-designer is used to create the user interface.

The basic building block of the graphical elements is the widget. A widget can be a simple push button, but it can also be a more complicated plot. With the addition of external widgets into Qt, many more possibilities beyond the default widgets can be implemented.

All widgets from the *.ui* file are represented by their respective classes in the compiled file. The compiled file is a regular source file in the respective language. In order to add functionality to the created widgets, a source file is created, which initialises the class created in the compiled *.ui* file. If an element is now clicked, an event will be created, which possibly emits a signal. For the most actions, such as clicking a button, these events and signals are already implemented. The signal can then be connected to a function, which gets called when the signal is emitted.

2.6 PYBIND11

Pybind11 is a set of macros and functions, which can be used in order to make C/C++ code accessible in Python [47]. The software is available on github [48]. It allows for a wide range of standard C++ features to be translated into Python. The bindings are compiled into a Python library, which can then be imported into a regular Python source file. The compilation of the library can be implemented with cmake as well as Python setuptools. In this thesis, the setuptools approach is used, as the cmake approach proved to be more difficult to implement in the TN.

In order to create a binding, one writes a preferably separate C++ header file. This header file will include the source code of the function to be bound. Afterwards, a function or class can be bound by referencing it in the Pybind11 function call. The compiler will then make the binding accessible in the package in such a way that the C++ source code is not necessary for the user of the package. The function call allows for different additional arguments to be added, which will alter how the function is translated. This includes the alternation of return types, as well as function parameters to fit the task. Documentation can be added to the functions and classes and will be available for the help function and in the Python package. The implementation of default arguments is also possible as well as the translation of callback functions between the languages. Data-types will automatically be converted to their counterparts. For custom data-types, one can implement custom conversions.

As an example, the binding for the logging class is shown.

```
1 #include <pybind11/pybind11.h>
2 #include <pybind11/iostream.h>
3
4 // source file inclusion
5 #include "../bipxl-software-drivers/logging.cpp"
6
7 // this function will be called by the generator module
8 // the module is a reference to the object, which will
9 // be the basis of the library
10 void initLogging(py::module_ &module){
11     // class definition
12     py::class_<bipxl::Logging> Logging(module, "Logging", "#DOC");
13
14     // enum definition, which will be available as member
15     // of the class in python
16     py::enum_<bipxl::Logging::LogLevelType>(Logging, "LogLevelType",
17         "Defines which logging data is processed")
18         .value("kDebug", bipxl::Logging::LogLevelType::kDebug)
19         .value("kData", bipxl::Logging::LogLevelType::kData);
20
21     // definition of the initializer for the class,
22     // which uses default arguments.
23     Logging.def(py::init<const bool, const bool, const std::string,
24         bool, const bipxl::Logging::LogLevelType>(),
25         py::arg("enabled") = true, py::arg("print_to_cerr") = true,
26         py::arg("path_to_log_file") = "",
27         py::arg("append_date_to_path") = false,
28         py::arg("log_level") = bipxl::Logging::LogLevelType::kError);
29
30     // simple function definition without default arguments
31     Logging.def("setLoggingEnabled",
32         &bipxl::Logging::setLoggingEnabled);
33 }
```

3 CONCEPTUAL DESIGN

3.1 CALIBRATION OF THE TIMEPIX3 BGI

The calibration of the Timepix3 chip and the readout chain consist of different parts, which are explained in this section.

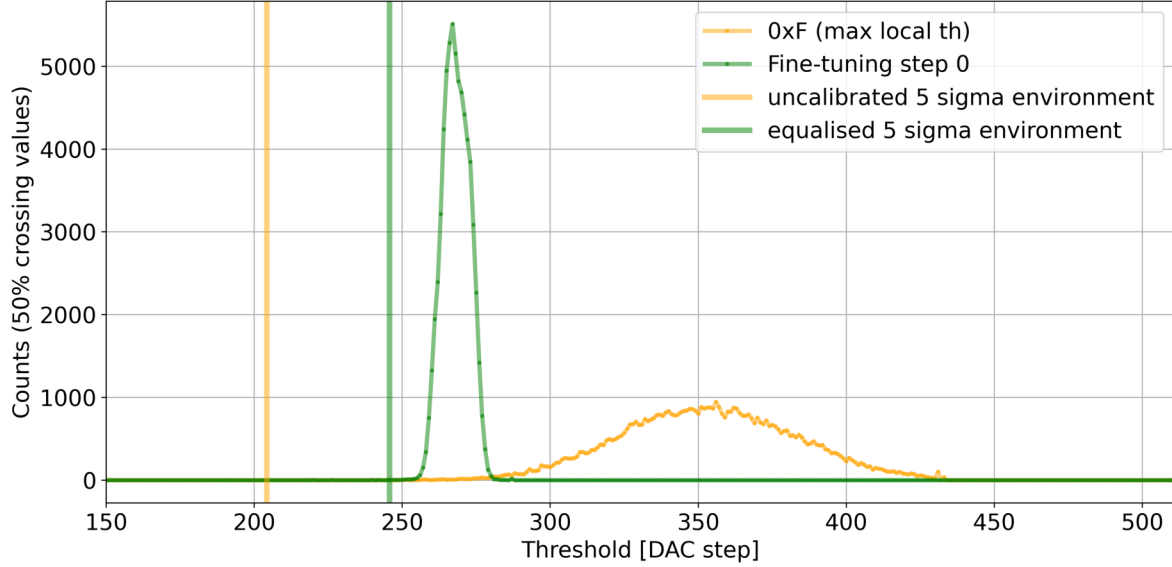
3.1.1 EQUALISATION

The general procedure of the equalisation is explained in section 2.3.4. The equalisation related task in this thesis is the optimisation of the algorithm and the use of TPs for the equalisation.

Several different improvements could be thought of in the implementation. The equalisation is a lengthy procedure, which takes approximately 10 minutes. One possible optimisation lies in the speed of this procedure. When optimising for the speed, it is also important to implement the changes in such a way that the reliability and repeatability are not worsened. Another aspect that could potentially be optimised is the quality of the result of the equalisation. As explained in section 2.3.4, this is correlating with σ , the standard deviation, of the resulting histogram.

When optimising for the speed of the procedure, one has to first identify, which part takes the most time. Currently, this is the data transmission to the software and the data taking on the chip. During the equalisation, each threshold scan will involve multiple openings of the shutter and consequently a readout operation afterwards. As this data transmission has to be done through the TN by using IPbus, it should be limited to the least amount necessary. If one inspects the S-curves for a single pixel, it is evident that the scanning over all values of the global threshold is not necessary. The occupancy values themselves are superimposed by a big fluctuation due to noise. At the same time, the S-curve will be measured multiple times during the fine-tuning, which will decrease the impact of this noise. Additionally, the resolution of the local threshold DACs limits the necessary resolution in the threshold scans. Even if the threshold scans were more precise, it would not help in order to find the optimal local DAC setting. In order to reduce the data, which has to be taken, it is therefore possible to only take data on every second value of the global threshold. The value in between two measured ones is defined as their average value. To smooth the data out using this method, a moving average window is used. The interpolation of the measured values was already implemented in the context of the previous Panda GUI.

Another possible improvement is the global threshold value, on which to start and stop the threshold scans. During the first two scans at 0x0 and 0xF DAC values respectively, the location of the crossing points cannot be narrowed down to a smaller value range. After the linear interpolation, the crossing points should be surrounding the target, which was set during the interpolation. The starting point of the threshold scan can therefore be moved in the proximity of the expected crossing points. With this procedure, it is possible that some pixels will not have their crossing value within the scanned range and therefore will be wrongly declared as dead pixels. The amount of these pixels proved to be very small or nonexistent in practice.

FIGURE 25: 5σ ENVIRONMENT FOR NON-CALIBRATED AND CALIBRATED CHIP.

To further decrease the duration of the threshold scan, the stop point of the equalisation can be modified as well. Instead of always conducting a threshold scan from the minimum of the global threshold value to the maximum, one can stop as soon as the majority of the pixels have reached an occupancy of over 90%. This practice has the potential downside of marking pixels as dead, which have not reached their 50% crossing value by that global threshold value. In practice, this only happens during the first two scans. The crossing value is then interpreted as the value, which is closest to the 50% crossing. In the fine-tuning steps, the crossing points of the pixels are generally closely together, which causes the 50% crossing value to have a much smaller spread, causing less values to miss their 50% value. This practice was already used in the previous implementation of the equalisation.

The improvement of the quality of the equalisation is important for the performance of the instrument. The quality is measured by the standard deviation of the 50% crossing values histogram after the equalisation. When taking data, the global threshold has to be set in such a way that the pixels detect the ionisation electrons, but are not inside the noise floor. This task is especially difficult, as the signal of the electrons is not significantly larger than the noise floor. If one can improve the quality of the equalisation, one can move the threshold closer to the noise floor and will be able to detect more ionisation electrons. In figure 25 the 5σ environment is shown. This represents the rough position of where the global threshold could be positioned. If the threshold would be placed closer to the mean of the histogram, one would risk to have pixels which were equalised correctly appear noisy.

To improve the quality of the equalisation, different measures can be implemented. In the originally implemented equalisation, the algorithm tries to get the crossing values closer to the target by increasing or decreasing the local threshold while at the same time trying to adjust the target value to better suit the data in each step. This approach bears some issues, as the local threshold has a limited resolution in comparison to the global threshold. Consequently, an overshoot of the crossing value is very likely, when increasing and decreasing the value. The

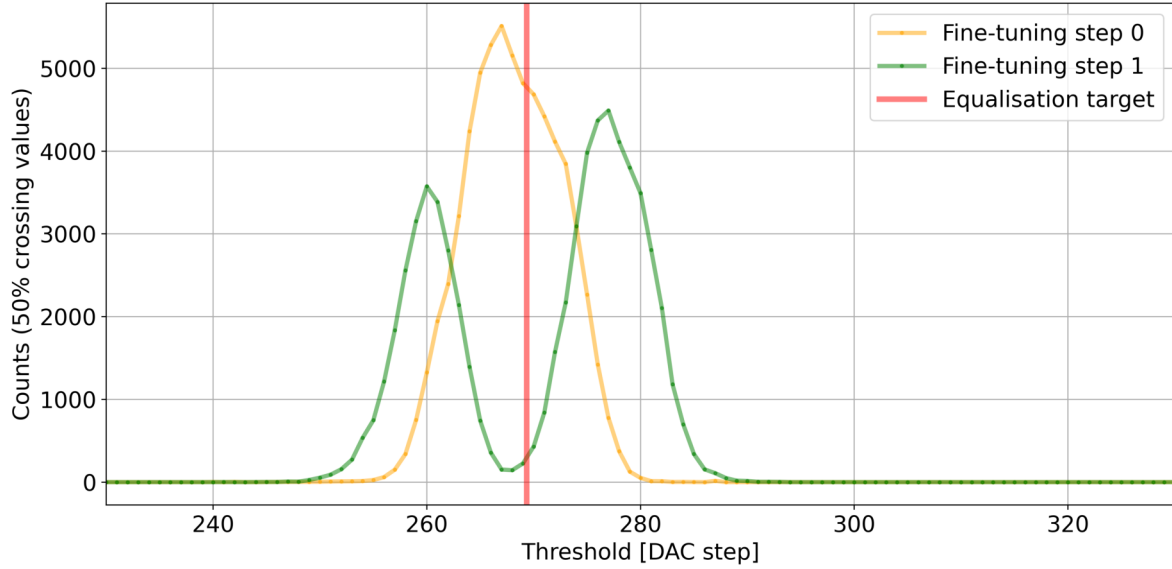


FIGURE 26: OVERSHOOT AFTER THE CORRECTION OF THE LOCAL THRESHOLD IN THE FIRST FINE-TUNING STEP.

effect of this can be seen in figure 26. After the first fine-tuning step, the distribution gets wider instead of narrower, also it is seemingly split into two distributions. This shows that in practice, the overshoot indeed happens and can cause significant difficulties in the equalisation. It also leads to the necessity to always have an even number of fine-tuning steps. More precision can be reached if the minimum distance to the target is recorded for each step of the fine-tuning. The minimum of these distances is applied at the end of the equalisation. For this approach to properly work, it is necessary to keep the target value constant and not change it after each fine-tuning step. Otherwise, the distance to the target would change between different fine-tuning steps.

Another possibility to create a more accurate local threshold map is to deactivate additional pixels. Due to the production differences, several pixels will end up dead or noisy. Dead pixels do not react to any input given to the pixel while noisy pixels are constantly putting out data, regardless of the input data. Both can be identified easily during the equalisation, as they either have no crossing value or a crossing value at the beginning of the scan. This detection was already used in the previous implementation of the equalisation.

For most of the pixels in the matrix, a solution for the equalisation problem can usually be found. Some of the pixels will end the equalisation with a local DAC value at one of the extremes of the range. This can indicate that a good equalisation of said pixel is not possible. The pixel might have its crossing value relatively far away from the equalisation target, but cannot be moved closer. When a low global threshold is now chosen, these pixels will either become noisy or dead respective to the measurement to be taken. Therefore, these pixels should be additionally masked to decrease the probability that an additional pixel masking is necessary after the equalisation.

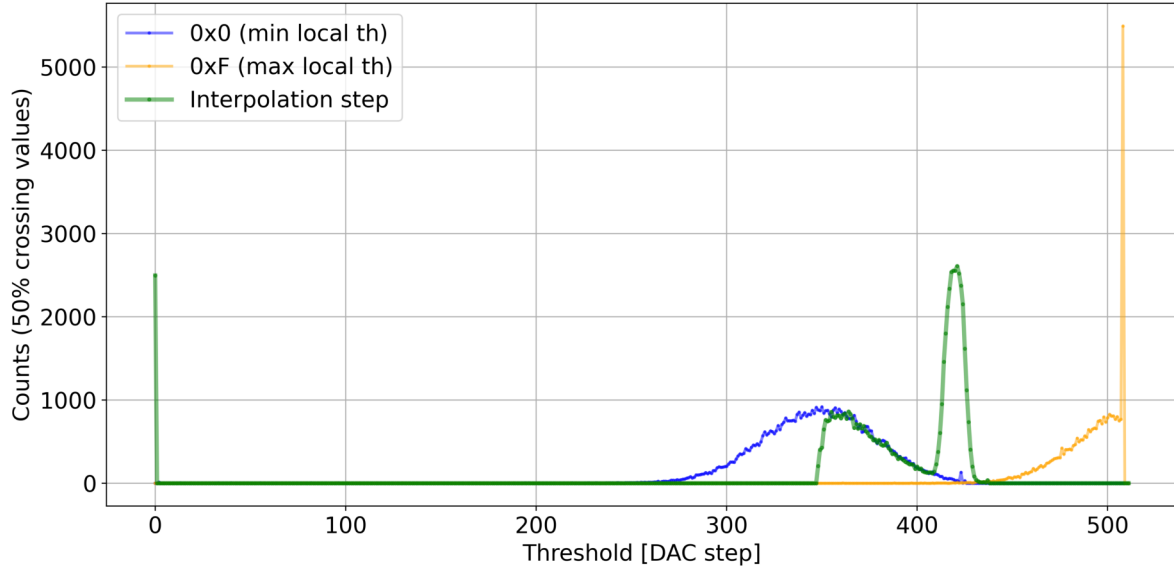


FIGURE 27: 0x0 AND 0xF THRESHOLD SCANS WITH A COARSE GLOBAL THRESHOLD VALUE WHICH IS NOT HIGH ENOUGH.

3.1.2 COARSE THRESHOLD SETTING DETERMINATION FOR THE EQUALISATION

As explained in section 2.3.1, the global threshold consists of a coarse and a fine setting. In order to run the equalisation, one has to select an appropriate coarse threshold. If the correct one is not chosen, the 0x0 and 0xF scans will not be possible with the same coarse setting. The fine threshold has a limited range of 512 values for each coarse value. In case of a too low coarse threshold, the distribution will be cut off. This effect can be seen in figure 27. It is also visible that this causes issues in the interpolation step. This can be explained by the maximum global threshold being defined as the crossing point. Some pixels might not reach their crossing points at all, which results in them being masked or taking a lot more fine-tuning steps. The result from this equalisation might therefore not be optimal or valid.

In order to determine the appropriate coarse value, a scanning of all possible coarse values is conducted. The coarse threshold has a resolution of 4 bit. For each value of the coarse threshold, the voltage at the midpoint value of the fine threshold is measured. The optimal coarse value will be determined by comparing this voltage to the threshold gap voltage. The global threshold voltage, which is closest to the optimal threshold gap, will be set as the optimal coarse value.

3.1.3 EQUALISATION REFERENCE SIGNALS

When running the equalisation, some reference signal needs to be provided to identify how different pixels take the data from it. An ideal source would be uniform on the whole chip and have an energy, which resembles the energy of the data signal.

A signal source, which is always available on the chip, is the thermal noise, which is distributed roughly equally across the whole chip. One of the issues with this reference is that it will change during the equalisation due to the usage of the chip. The temperature of the chip can also be changed by a beam induced heating. In order to avoid this issue, a different reference can

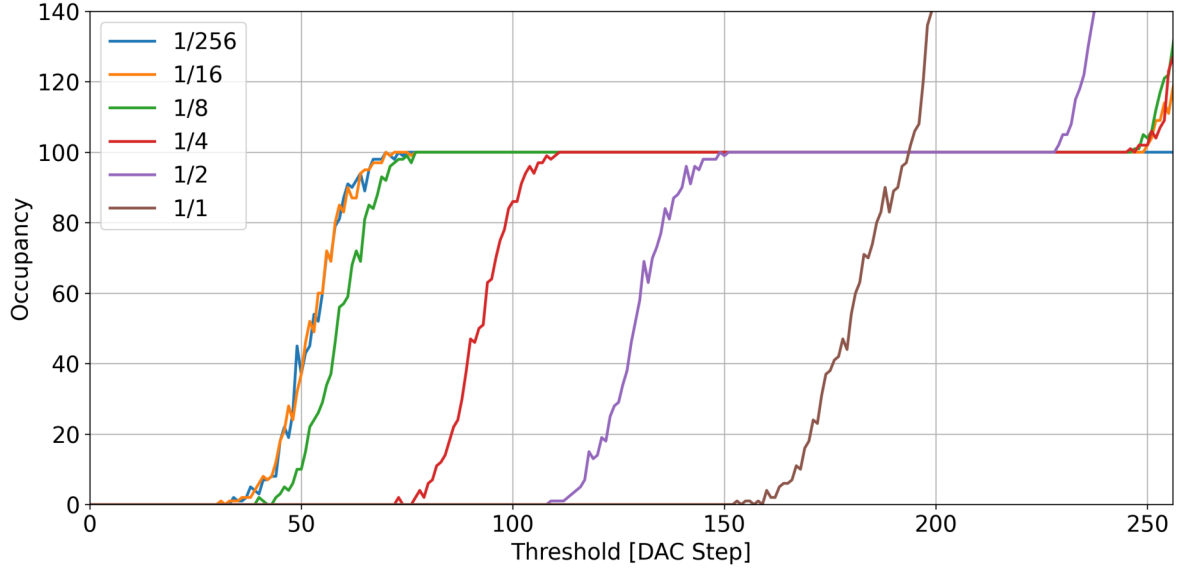


FIGURE 28: S-CURVES FOR A SINGLE PIXEL WITH A DIFFERENT AMOUNT OF PIXELS BEING CHARGED AT ONCE

be used, which is independent of the temperature. Theoretically, it would be ideal to have an external electron source, which could emit electrons with the amount of energy also present during data-taking. Creating such a source would require significant changes to the detector and is therefore outside of the scope of this thesis.

A functionality that is available on the Timepix3 chips are the TP. Every pixel has a small capacitance, which can be charged by a selectable voltage. The energy stored in this capacitance is then released into the pixel frontend directly as soon as the shutter is opened. Each capacitance has a design value of 3 fF. Due to the production differences between pixels, slight differences in this value are expected. As long as these differences are smaller than the resolution of the local threshold, the impact on the result of the equalisation will be small. The details of the TP usage were discussed with the Timepix3 team.

One of the big advantages of using the TPs is that they are more independent from the temperature of the chip than the noise floor. The energy held by the capacitances can be freely chosen by altering the voltage, the TPs are charged with. Therefore, the energy can be set to a value, which differs from the noise floor and thus improves the result of the equalisation as well as increasing the reliability against temperature differences during the procedure.

The use of TPs incurs some challenges. As outlined earlier, the capacitances get discharged automatically into the pixel frontend when the shutter is opened. To ensure that the energy, which reaches the frontend, is as similar as possible, one has to fully charge all capacitances prior to the shutter opening. In total, there are 65536 pixels in the whole matrix, all of which include a capacitance that needs to be charged. Therefore, the charge time as well as maximum charge capabilities of the chip need to be investigated in order to check, how the pixel TPs can be used reliably in the equalisation.

In figure 28, S-curves for a single pixel are visible. The amount of pixels charged at once is then varied, while the charging voltage and time are kept constant. Despite the same charging voltage, the 50% crossing of the pixel changes drastically with the amount of pixels being charged at once. This strongly indicates that the capacitances of the pixels are not fully charged when the full matrix is used at once. The difference of charge deposited in each individual pixel is most likely random and can vastly differ. This in turn gives the reference signal a bigger variance, making it unusable for the equalisation. The difference between a $1/256^{th}$ of the matrix and $1/16^{th}$ of the matrix is negligible such that a checkerboard level of 16 will be used for the equalisation with TPs.

For higher values of the global threshold, one can see in figure 28 that the occupancy goes above 100. This is caused by the noise floor interfering with the TPs for higher threshold values. As the occupancy is defined as received hits divided by given charge cycles of the TP, the noise floor can cause this value to be above 100. A detailed description is given in section 4.1.

A possibility to make more of the matrix able to be equalised at once would be to increase the charge time of the capacitance. The TP generator has two parameters, one for the number of TPs to be generated and one for the period of the TPs. By increasing the charge time, more pixels can be charged at once. In this approach, the load on the charging network will be a lot higher than when the capacitances are not charged in the first place. This would in turn increase the risk of a power glitch, which would crash the chip and thus making the equalisation result incorrect. Therefore, it is not practical to increase the charge time in order to increase the amount of pixels, which can be charged simultaneously.

3.1.4 DAC CALIBRATION

Some of the DAC values of the chip periphery need to deliver a certain voltage in order for the chip to work correctly. The Vfbk voltage for example has a significant impact on the chip detection performance. The Vfbk serves as a reference voltage for the threshold gap as well as a baseline voltage of the preamplifier output [49]. The threshold gap is defined as the voltage difference between the Vfbk and Vthreshold voltages. Both of these voltages will determine, which amount of energy will be detectable by the frontend. With these voltages, the ToT and ToA values of the data will also significantly differ. As multiple chips are involved in each instrument, the chips also have to be able to detect the same ToT and ToA values for an equal energy, making the calibration of the DACs even more important.

As the DACs will naturally deliver different voltages on each chip for a given digital value, they need to be adjusted to the same output voltage. Due to the radiation and ageing, the digital input values necessary to reach a certain voltage will also change over time. It would therefore be advantageous to implement an algorithm, which would be able to automatically calibrate the DACs.

For this purpose, a heuristic was implemented, which is shown in figure 29. The heuristic is based on an iterative approach. A variable DAC step is used to further approach the target value. This DAC step is then applied to the current DAC value, either added if the voltage is

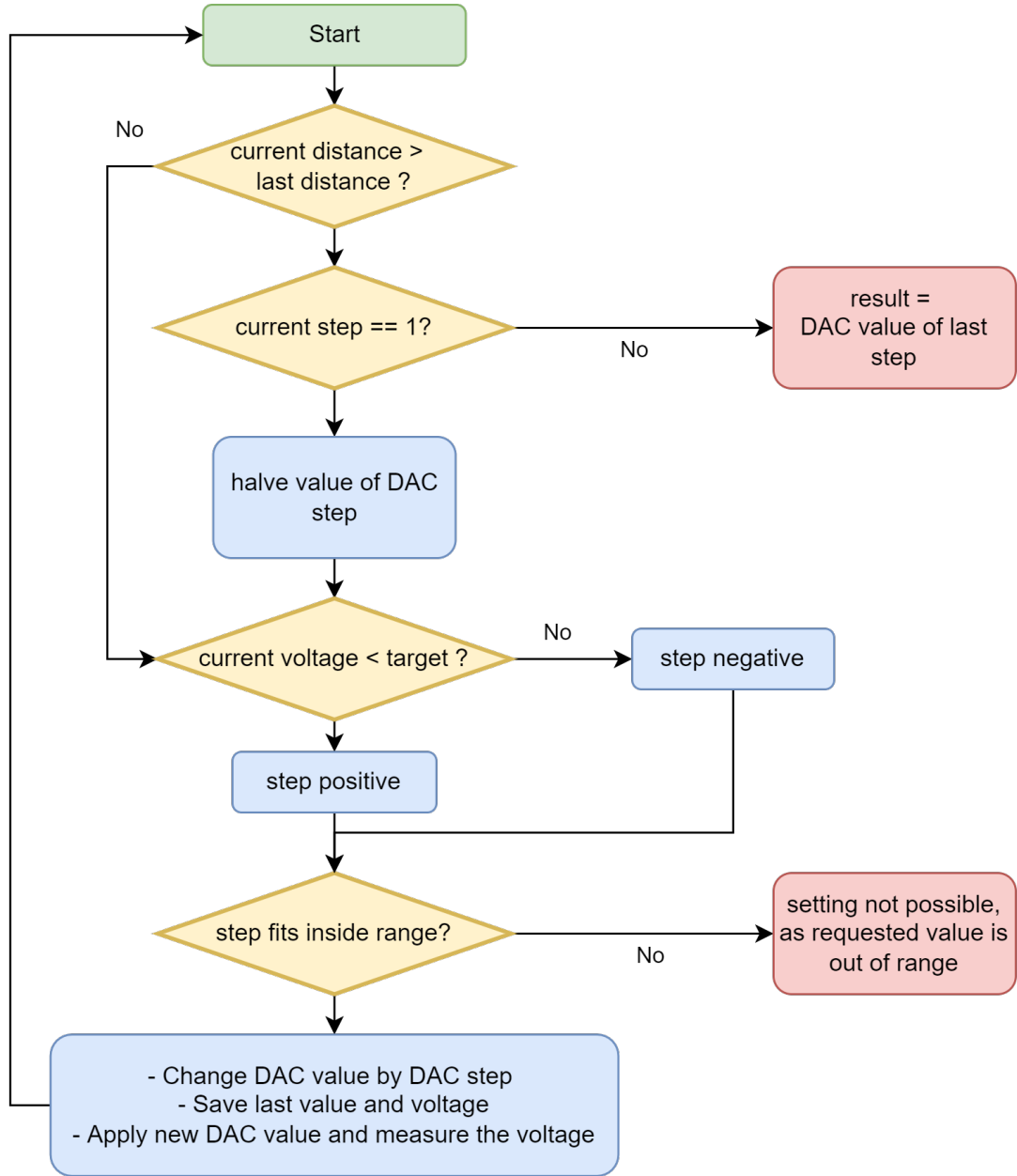


FIGURE 29: DAC CALIBRATION HEURISTIC.

below the target or subtracted if its above. Afterwards, the voltage is measured. Before applying the new value, the old DAC value as well as the voltage are saved. After the application, the algorithm checks if the resulting voltage after applying the DAC step was further away from the target than the voltage with the previous value. This indicates an overshoot of the target value. In this case, the algorithm tries to decrease the step value. If the step value reaches 1, the last result is returned as an optimal value, as no further improvement can be reached within the resolution of the DAC. The algorithm is repeated until either an optimal value is found, or the DAC step would have set the DAC value outside of the boundaries.

3.1.5 COMMUNICATION CALIBRATION

As discussed in section 2.2.2, each chip has 8 channels for data-output. These electrical signals are connected through the flange board and to the frontend, where they are converted to optical

fibre. Due to degradation of the different components or production variations of the components, a clock skew can occur between the Timepix3 output channel clock and the receiving clock in the backend [4]. In this case, the comma symbols will not be detectable and thus the pickup of the data will not happen correctly. This will result in valid packets not being readable or lots of noisy packets appearing to arrive in the backend.

To offset this clock skew, delay modules are used in the Kintex-7. The delay modules are also available in the Zynq MPSoC [50]. These can be adjusted with digital values and will then delay the received signal, so that the clocks align. As the parameters of the chip and frontend will change over time due to radiation and ageing, this task has to be performed regularly. To provide an automatised solution for this procedure, a heuristic is proposed. The heuristic is based on the data-rate reading in the backend.

At the beginning, each channel is turned off. Thus, the measured data-rate in the backend will only originate from the channel to be opened. The detector is setup for normal operation with the shutter being closed. Then the channels will be opened one after the other. The data-rate is averaged over multiple readout periods. If the data-rate is above a certain threshold, the delay value is increased. When the maximum delay value is reached and the channel is still noisy, the channel gets disabled. This procedure can take time, as the RAM in the backend needs to be reset between each measurement to ensure that no packets of the last connection are left over and counted towards the data-rate of the current measurement.

A faster approach can be chosen if the delays are not adjusted and only the noiselessness of the currently applied delays is checked. In the case of the channel sending too much data, it gets disabled. This approach might decrease the chip readout rate below what is technically available, as the alteration of the delay can change the usability of the channel.

3.1.6 PIXEL MASKING

During the equalisation of the chip, a lot of data is taken, which will already indicate that some pixels of the matrix are dead or noisy. One of the challenges with the resulting mask is that the pixel matrix will degrade further over time, especially in a radiated environment. Due to these changes, additional pixels might become noisy. In this case, the noisy pixels will generate a lot of data, filling up the RAM in the backend and potentially causing real data events to be dropped.

One solution might be to regularly run the equalisation. As described in section 3.1.3, this would be very challenging, as the result of the equalisation will be potentially changed by a passing beam. It would therefore be necessary to wait for the next beam stop to run the equalisation. Consequently, a different algorithm, in which the noisy pixels can be determined, is implemented. A simple function, which masks pixel above a certain count, ToT or ToA value within a given data-set, was already implemented in the driver.

The general idea of the additional masking algorithm is to open the shutter for increasing amounts of time and then check for pixels, which fired more than the expected amount of time. Before starting the pixel masking, equalised DAC values need to be loaded to the pixel matrix. The reason behind increasing the time is to also be able to mask pixels, which are not constantly noisy, but rather are only noisy in random periods of time.

3.2 CHANGES TO THE READOUT CHAIN

The readout chain, which was explained in section 2.2.2, included some components, which could be iterated upon. For one, non-radiation hard components were still in use in the frontend. If one of the components in the frontend would break, it would not be possible to fix these components until the next technical stop. This could possibly cease the operation of the instrument for several weeks.

Additionally, the computations on the FPGAs in the frontend make the radiation hard design of the gateway by using TMR necessary, which requires a lot more computational resources than otherwise needed. In addition, a TMR design can only reduce the probability of a wrong computation from an SEU, but not completely eliminate it. Having the computations split between the front- and backend also requires additional communication overhead between the two. Furthermore, the splitting also makes it necessary to maintain and implement multiple gatewares for the different FPGA architectures. This makes a change to one of the parts a lot more complicated than it would be with a single device.

In order to address these issues, the readout chain is being significantly modified by the BGI team. The new readout chain can be seen in figure 30. The FPGA in the backend was substituted by a Xilinx Zynq MPSoC. This allowed for the computations previously done in the frontend to be now done in the backend. By now having these computations done in the backend, the TCM can be removed from the gateway implementation, as the risk from radiation induced SEUs is greatly reduced. By now performing all computations on the same chip, resources, which had to previously be available in different locations, can now be shared and therefore saved. The MPSoC is currently used on the Xilinx evaluation board, which might be substituted for a custom board in the future.

One of the big advantages of this approach is that the amount of parts in the radiated environment is now smaller. The only components left in the readout chain are the optical transceivers, which are radiation hard by design. This also reduces the amount of material, which gets radiated and is not reusable afterwards. The only necessary task of the frontend is to convert the signal from optical to electrical. This opens up the possibility to remove the optical based frontend completely for lab or testing operation.

3.3 IMPLICATIONS ON SOFTWARE AND GATEWARE DESIGN

As explained in the previous section, all necessary computations are now performed in the backend. This allows for potential changes in the software stack and gateway. Also, due to the added computational units in the MPSoC, the tasks can be divided into different domains. All units have different advantages and disadvantages. The decisions of which tasks are to be implemented in which domain were made together with the BGI team.

The PL can be used to do a lot of repeatable computations in a short amount of time. In the PL, it is possible to implement highly optimised gateway algorithms to do very specific tasks. The interfacing with certain hardware parts is also only possible in the PL. The GBTx transceivers for example need to be controlled by the FPGA. The downside of the PL gateway is that the

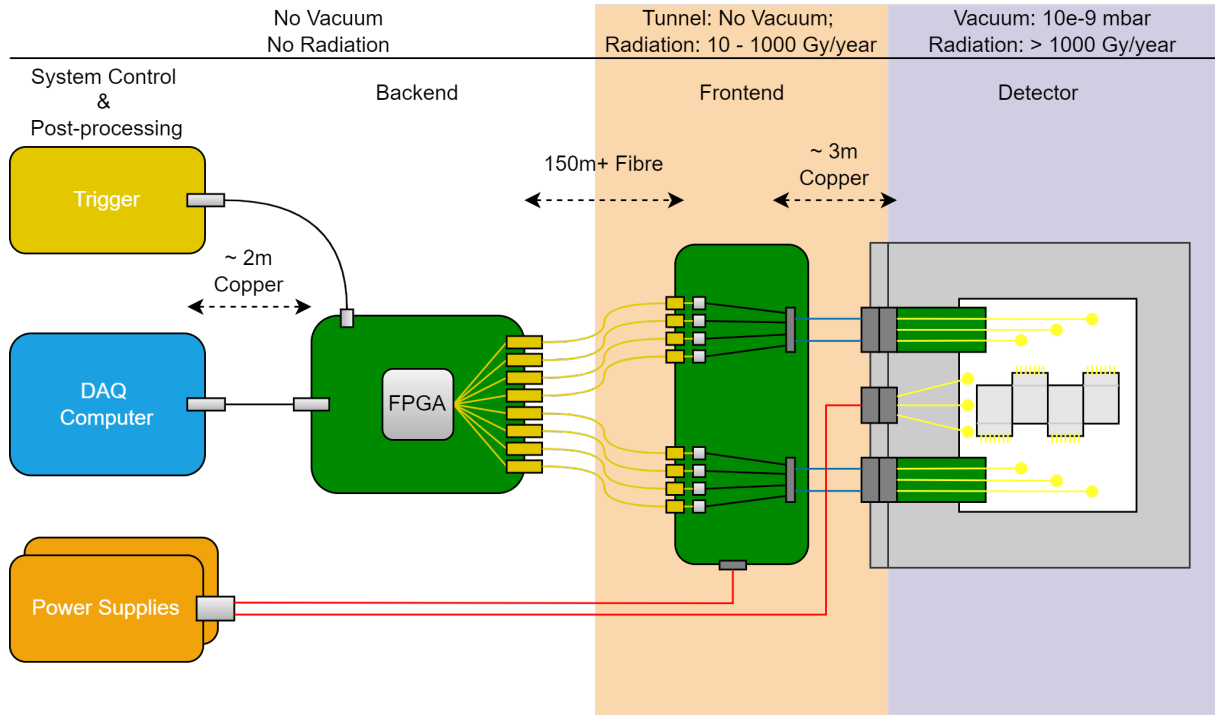


FIGURE 30: REDESIGNED READOUT CHAIN.

development effort is usually high compared to regular software. High-level communication with the network or other devices is therefore generally more time consuming to implement on FPGAs if no built-in modules can be used.

The RPU can be used for time-critical operations. In the implementation, one can freely advise the scheduler on which tasks to prioritise. Because of this, one can also run safety critical operations with a high priority on this device. One of the downsides is the increased development effort, as more care has to be taken for the details of the FreeRTOS implementation and architecture of the tasks. Another disadvantage is the limitation in the size of the program. As pointed out in 2.4.1, the program should be placed inside the TCM, which has a capacity of 256 kB for the application binary. Additionally, the RPU does not have access to the file system of the SD-card, as this is managed by the APU. Any file operations on the SD-card therefore require the data to be read by the APU and then sent to the RPU via the libmetal communication.

The APU on the other hand can be used for general purpose tasks. It is able to access the file system and has no relevant restrictions on the binary size of the application. As drivers for the Ethernet connection are readily available in PetaLinux, it can also easily communicate with other devices in the CERN TN. Via this connection, control data as well as configuration data can easily be transmitted. Computations done in the APU generally take longer than in a highly optimised PL gateway. Compared to FreeRTOS, the general purpose kernel of PetaLinux lacks the configurability of the scheduler for the different threads and tasks.

By distributing the tasks between the different units, a good balance between required resources and time on one side and required development effort on the other can be found. Tasks, which need to communicate with the TN, should be implemented in the APU, as it has the

easiest access to high level network communication. This includes all tasks for displaying data or publishing data to the CCC. Tasks, which need to regularly access files, should also be performed on the APU, as it has the easiest access to the file system and is not limited in binary size. The loading of pixel masks for example requires to load the pixel DAC values, which consist of $65536 * 4$ bits. Also, algorithms, which include a lot of different program steps and are therefore more complicated to implement should be performed on the APU. This includes for example the equalisation. Additionally, in the equalisation, one needs a lot of run-time memory, as all of the data from two threshold scans need to be saved. This results in $65536 * 512 * 32$ bits per threshold scan. It is also important to save the results of the equalisation to the file system, which the APU has direct access to. Long term, it would be beneficial to have the driver run on the APU itself to remove the network traffic and latency for the large amounts of data, which need to be exchanged between the backend and the driver.

If a task needs to perform a lot of computations and has to be done repetitively, it should be performed in the PL. The benefit of implementing dedicated hardware for certain tasks can be very high, especially if a high throughput is required. These tasks include the encoding and decoding of the data. Configuration data as well as measurement data needs to be decoded from 8b10b encoding and in most cases also gray encoding. This can be performed significantly faster with dedicated PL gateware. For the same reason, the PL will also be responsible for the communication through the fibre optics to the chips. Additionally, the GBTx chips need to be configured by an FPGA logic.

Details on the tasks, which will be performed on the RPU, are laid out in section 3.5.2.

3.4 EXPERT GUI CONCEPT

The expert GUI is meant to be used by members of the XEI section to be able to configure and calibrate the instrument. The settings resulting from this calibration will then be used as default settings, when the operators in the CCC are using the instrument for beam measurements. The operators do not need direct access to most of the settings of the BGI.

3.4.1 PREVIOUS EXPERT GUI

The GUI previously used for this task was the Panda GUI. Within Panda, all the settings of the detector and the readout chain were accessible. The GUI was based on Qt, which is a C++ GUI framework. The access to the driver was therefore easily available. One of the challenges with Panda was to run it in the TN. As no external libraries can be easily installed onto the TN virtual machines, a Docker container was necessary to run the GUI. Using a Docker container might degrade the performance and it is also an additional layer to be maintained and installed when compared to running the code natively in the TN. In addition, both Qt and Docker are not currently supported for the TN Virtual Machine (VM)s, which can lead to dependency problems during updates. Additionally, no support will be given in case of compatibility issues with existing software packages. The BI-SW section has implemented an application launcher, which allows one to start all expert GUIs of the group for the different instruments. In this launcher, only Python and Java GUIs are supported.

As the instrument needs to be accessible from the TN, the new GUI needs to be implemented in the PyQt framework, as this is currently the recommended and supported GUI framework for this environment. The package installation for Python packets is also supported for a predefined set, which makes dependency handling easier than in Qt. In the case of a different packet being necessary for the GUI, it is also possible to have these packets added to the index of packets that are available in the TN.

3.4.2 ACCESS TO THE DRIVER

As described in the last paragraph, the new expert GUI will be based on PyQt [51]. Because the GUI needs to communicate with the driver in order to manipulate and retrieve the settings and the status of the detector, a language transition is necessary. The driver is currently written in C++ and consists of a lot of code, which should not be rewritten in Python, as it has been used for some time, is reliable and well tested in an operational environment.

Consequently, it is necessary to create bindings from the C++ driver to a Python module. For the generation of these bindings multiple software toolkits are available. Pybind11 is available in the TN and also has a lot of features necessary for the binding of the driver. Parts of the driver use Standard Template Library (STL) types, which can be easily translated into Python types by Pybind11. Another necessary feature is that functions need to be able to accept custom data types, which Pybind11 also supports. Additionally, all features are described in a good documentation with many examples, which will ease the implementation of the bindings. Pybind11 is described in section 2.6.

3.4.3 GUI DESIGN

Most of the tasks, which will now be implemented in the expert GUI, were previously accessible through the Panda GUI. As a new GUI and a language transition is necessary, the driver interface and the GUI layout can be improved at the same time.

The old GUI was centred around the hardware location of the components in the readout chain. With this approach, configuration options that need to match were available in different tabs, making it more tedious to make changes from a user perspective. One of the goals of the new design is to change it to a more user centred layout, which groups elements that have to be changed together.

One example of this change is the configuration of the communication between the backend and the chip. Each chip has 8 communication channels from the detector to the backend. They can be disabled either by the receiving device in the backend or by the output periphery on the Timepix3. In the case that these settings are different, it is possible that the Timepix3 tries to send data on a channel that the backend does not receive on, in which case the data is lost. If a channel is enabled in the backend and disabled on the Timepix3 side, the input of the channel is floating, which can create a large amount of noisy data reaching the backend and blocking the readout chain. To make it easier for the user of the GUI, both elements will be placed next to each other with the option to lock the channels, so that the mismatch is prevented by the GUI itself.

In addition to the configuration, it should also be possible to take data for a functionality test of the instrument. This functionality was previously available in the Panda GUI. When testing the configuration and taking data with a non-perfect pixel mask, one will see noisy pixels in the matrix. In Panda, one would have to go into a separate tab to be able to mask the pixels and either type in the exact pixel to mask, or start a heuristic with set parameters to determine, which pixels to mask. An improvement over this method would be that one could click on the pixel, which would mask it directly, reducing the time this calibration step takes.

As the Panda GUI was not meant to be run in the TN, the access to several parts of the control system was not implemented. The control of the power supplies for example is done by a different GUI, which is implemented in Java and runs in the TN. As the program now has access to the TN and also the APIs that control these supplies, it is possible to add the monitoring and controlling into the GUI. One of the issues during the operation of the instruments can be an unwanted change into a reset state, which causes the power consumption and thus the temperature to rise significantly. If one would be able to see this change in the expert GUI and shut off or reset the chips, it would prevent such an event from being unnoticed.

Another element to be added to the GUI is the supervision of the equalisation. As explained previously, the equalisation is a long procedure, which takes several minutes to complete, even after the mentioned optimisations. It would be beneficial to get some feedback of the level of progression as well as the quality of the equalisation up to the current point. Based on this feedback, it should also be possible to stop the equalisation at any time.

To increase the usability of the GUI, some tabs will be split into two tabs. The controlling of the DAC values of all chips for example will be implemented in a separate tab from the other settings of the Timepix3. In the DAC tab, the algorithms for the calibration of the DAC values will also be available.

3.5 RPU CONCEPT

3.5.1 AIMS OF THE RPU USAGE

The advantage of the usage of the RPU versus the APU is that it can be configured in a predictable way and is therefore more suitable for control and configuration tasks. One of the main challenges is the limited storage space for the binary in the TCM. Another challenge is that programming on the RPU is more complicated than on the APU as more parameters, like the memory for the heap and the stack, need to be defined by the programmer. At the same time, software on the RPU is faster to test and implement than gateware for the PL, which would be potentially more reliable and exhibit a shorter run-time. The debugging of the RPU and APU software is described in 4.4.

The goal is therefore to implement tasks, which would benefit from the additional predictability, while avoiding tasks, which are long from an algorithmic standpoint. Tasks that have a strict performance requirement on fast computation time should also be avoided in the RPU implementation, as the raw computational power is lower than on the APU. Configuration tasks, which need a large amount of data, are not suitable, as the data needs to be transmitted with libmetal.

3.5.2 POSSIBLE TASKS TO BE IMPLEMENTED ON THE RPU

Time-critical or safety related tasks should be mainly implemented on the RPU as the scheduler and therefore the priority of the tasks can be fully controlled. The temperature monitoring will be run with a high priority on the RPU. If the temperature changes quickly it indicates, that the chip crashed or got into the reset state. When the temperature rises, the additional heat is difficult to dissipate, as the vacuum does not allow for the heat to be dissipated with convection.

To increase the predictability of certain configuration tasks, these should be done through the RPU as well. The startup sequence for example is relatively short and can be used after a power glitch to quickly recover the Timepix3 from its reset state. On the APU, many tasks will be implemented, which deal with data processing in one way or the other. These tasks can take a lot of computing resources, which can lead to delays on urgent configuration tasks.

3.5.3 SEPARATION OF DIFFERENT TASKS ON THE RPU

The separation of the tasks is very important, as it will change, which tasks will be run at which priority in the RPU. Each task will be assigned a different priority. As soon as one task goes into a blocking or idle state, the scheduler will check if another task is available to be run. From these available tasks, the one with the highest priority is chosen and given the right to execute until it enters a blocking state again.

The most important task is the temperature monitoring. If the temperature changes drastically, one can immediately take measures to reset or reconfigure the chip. The temperature monitoring is done by alternating the sense DAC setting between two different DACs. The sense DAC setting chooses, which DAC output is readable on the corresponding output pin on the Timepix3. From the voltages of these, the temperature can be calculated. This task will receive the highest priority.

On a lower priority, the APU communication related tasks will be placed. These are usually not safety critical. This includes the startup sequence. The main reason, why they are placed on the RPU, is the predictability of the configuration tasks, which is kept if the priority is lower.

3.5.4 COMMUNICATION BETWEEN RPU AND APU

Different tasks, like the configuration tasks, will need to communication between the RPU and APU. The values for the configuration of the detector are stored on the SD-card of the Zynq-Board or available from the TN. To use these values, access to either the file system or the Ethernet is necessary, both of which are a lot easier to perform from the APU.

The communication will be implemented using the libmetal library, which was described in section 2.4.3. As a basis for the communication itself, the Xilinx example for the implementation of a libmetal communication is used [52]. One main communication task is created, which will then notify a different data-gathering task. This task will then receive the desired value or run the desired task and then notify the communication task back. This separation is done in order to not block the scheduler for too long. On each context change via notifications, the scheduler

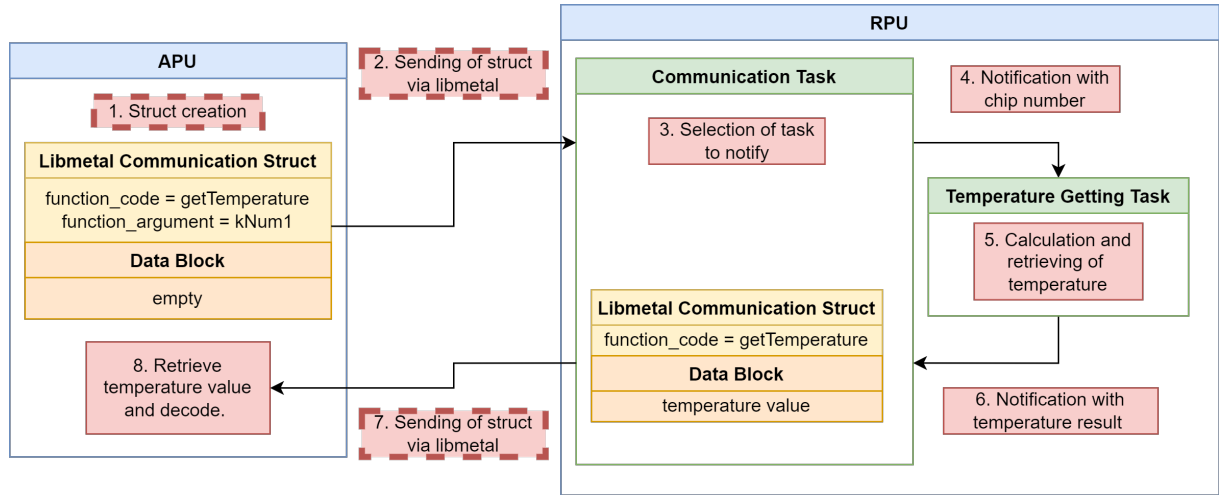


FIGURE 31: EXAMPLE OF THE COMMUNICATION CONCEPT BETWEEN RPU AND APU FOR RETRIEVING OF THE CHIP TEMPERATURE.

can choose to run the task, or run a different task with a higher priority. If everything would be implemented in the communication task, a context change and therefore working on high priority tasks during the communication would not be possible.

The data communication itself is done with enums. These allow the definition of different tasks on different enum values. Another enum is defined for the passing of different arguments for the tasks. If data needs to be shared between the RPU and APU, it can be done with the data field associated with the transmission in libmetal. In this field, any data can be copied, as long as the layout of the memory is known to both communication partners.

In figure 31, the different steps for the retrieving of the chip temperature from the APU are displayed. The steps are marked in red, while the steps with the dashed outline were adopted from the Xilinx example. At the start of the transmission, the communication struct is created and filled with the predefined enum values. After it has been sent to the RPU, the struct is read out by the *Communication Task* and the *Temperature Getting Task* is notified with the number of the chip temperature to be retrieved. The sending of the data includes the copying to the shared memory and the pulling of the respective interrupt for the RPU. The *Temperature Getting Task* retrieves the temperature itself and then notifies the communication task, which then creates a struct for the data sending back to the APU. While the tasks are waiting for a notification or an interrupt, a higher priority task can be selected by the scheduler. If a configuration task is performed, the data is transferred in the data block of the libmetal transmission. In this case, the return data block would be empty.

4 IMPLEMENTATION

4.1 EQUALISATION

For the automatic supervision of the equalisation, a separate thread is started in the C++ driver at the beginning of the procedure. This supervisor checks if the temperature does not suddenly change. Such a behaviour would indicate, that the chip crashed and the equalisation is no longer valid. A sudden temperature change can also potentially damage the chip if the heat cannot be dissipated in the ultra high vacuum. The user is also able to predefine a maximum temperature for the chip during the equalisation, at which the procedure is stopped as well.

Another task for the supervisor is the checking of the communication channels. If a communication channel stops to work during the procedure it can lead to corrupted or dropped packets, possibly invalidating the result. When one of these two conditions fail, an atomic boolean is set. This boolean is checked within the threshold scan function. Additionally, the reason for the stopping of the equalisation is published via an atomic enum value. The equalisation will then throw a run-time exception with an explanation according to the enum value, which can be caught by the code that called the equalisation.

In order to increase the speed of the equalisation over the previous implementation, additional features of the Timepix3 were used during the threshold scan. Previously, the algorithm waited for an arbitrary amount of time before increasing the threshold scan value. In the new implementation, the algorithm waits for the packet that gets sent after the shutter opening has finished. Then, the sequential readout is started. The end of the readout also gets confirmed by the Timepix3. After this confirmation message, or a wait time of one second, the algorithm will change the global threshold value. Especially for scan values, where little to no data is collected, this approach is a lot faster. At the same time, it ensures that all data has been read when a higher data-rate is produced. The maximum wait time is implemented for the case of a confirmation packet being corrupted or missing.

To make the results of the equalisation more comparable, a Java Script Object Notation (JSON) format for saving the most important parameters and results was implemented. As it was already used for other saving purposes in the driver, a JSON header-only library was used [53]. The comparison of different equalisation results also allows the monitoring of the long-term changes of the instrument's performance.

For the implementation of the TP, additional prerequisites were necessary. The previously highest checkerboard level was 8, meaning, one eighth of the matrix could be equalised at once. As discussed in section 3.1.3, a division of $1/16^{th}$ should be used for the TP equalisations. An important detail with the checkerboard implementation is the placement of the pixels to be equalised in one go. The Timepix3 consists of SPs, which share pixel-periphery for 8 pixels. To optimise the utilisation of the chip, it is therefore important to minimise the amount of pixels used per SP. The selection of the pixels was already done in the checkerboard level 8 implementation.

The occupancy will be calculated differently for the equalisation with TPs. The TPs can be injected multiple times with the same energy in short succession. This is done in order to increase the robustness of the TP equalisation against a difference in charging currents between the charging operations. To then determine the occupancy from this signal, a different value is used. In the EC & iTOT mode, each pixel will count how many times the signal was observed in the EC value. The occupancy for the TPs is defined as

$$occupancy_{pixel} = \frac{EC}{number_of_injections}. \quad (4)$$

This will in turn increase the run-time of the equalisation.

4.2 EXPERT GUI

As described in section 3.4.2, Pybind11 will be used for the connection of the PyQt functions with the driver. Some of the *ui* files can be at least partly reused from the Panda GUI, i.e., the tab for configuring the Timepix3 settings. An important difference to the old GUI is the plotting framework. The old GUI used QCustomplot [54]. As this is a C++ Qt-widget, it is not available in PyQt without significant modifications. For plotting in PyQt, different frameworks can be used.

One possibility to use is Matplotlib [55], which requires adaptations when it comes to the exact positioning of the plots as well as placement of legends. It is also not natively supported for the use in PyQt, making it possibly more difficult to integrate into the GUI. Another challenge with using Matplotlib would be the performance. As Matplotlib was originally meant to be used for the creation of publication-style plots, it has limited capabilities, when a graph needs to continuously be replotted. This replotting places a time constraint on the frame generation, which is nonexistent for publication-style plots. Especially for the data-plotting of the pixel matrix, the performance might be a limiting factor.

Another candidate for plotting is the PyQt specific widget PyQtGraph [56]. It is specifically meant to work within the framework of PyQt, thus removing most positioning and legend placement challenges. Additionally, it is meant to be used in an interactive environment, which will be very useful in the implementation of some features. PyQtGraph allows for the creation of all plotting elements that are needed. The elements consist of line plots for single values like the temperature or voltage of a certain hardware. Another task is the plotting of the pixel data in an image plot. One feature missing is the ability to plot the pixel data on a logarithmic scale. This issue can be circumvented by taking the logarithm of the data before plotting it, creating a similar display but invalidating the values presented on the axes.

Due to the usage of Pybind11, some care has to be taken when using the driver functions. One issue within the implementation of the bindings is that the calling of a bound function will always lock the Global Interpreter Lock (GIL) [57]. As the GUI runs in the same Python process, the GIL will be locked for all GUI tasks as long as the bound function is executed in C++. This gets especially problematic if multiple threads are involved, as all threads will get locked by the GIL. To circumvent this problem, one can manually disable the GIL locking on

Time	Single value acquisition	Full array acquisition
Real time	57.37s	03.25s
CPU time	57.01s	2.93s
System CPU time	0.65s	0.57s

TABLE 1: RUN TIME DIFFERENCE FOR THE ACQUISITION OF A MATRIX WITH 256X256 ENTRIES, WHICH WAS RUN 1000 TIMES

specific Pybind11 functions. This change has been added to functions that get regularly called. It is implemented in the function that flushes the sending buffers and reads back all the data from the backend. If many data packets are arriving or much data is to be sent, the GUI would otherwise be blocked for a long period of time.

Another important reason behind the unblocking of the GIL has to do with the equalisation monitoring. As the equalisation is a Pybind11 function, it will naturally block the GIL. If the GIL is not unblocked, the GUI would not be able to show any of the steps of the equalisation. The monitoring would thus not be possible, defeating the purpose of this feature.

Each call of a Pybind11 function will also result in a context change. If one needs to a lot of single data points, it is probably faster to summarise them into a data-structure, which can then be retrieved by one function call. Additionally, each single call will lock the GIL. Although the GIL can be unlocked within the function body, this is a manual procedure and requires additional calls [57]. One example for this is the acquisition of the pixel-matrix data. Instead of calling a function for each individual pixel, it is faster to call a single function, which returns the whole matrix at once. In table 1, one can see the difference between the single value acquisition and a full matrix acquisition. For the case of all data being needed in an algorithm, it is a lot faster to acquire the full matrix.

As the data-getting function may take a long time to return, the GUI functionality is split into two different threads. One thread is responsible for the data sending and updating of the GUI elements, while the other thread is responsible for receiving commands from the GUI and adding any actions to the message queue.

To enable custom plotting functionality, like the masking of pixels by clicking the plot, additional features of PyQt as well as PyQtGraph need to be used. In order to notify a custom function from an event in the plot, the signal of said event needs to be connected to the function with *signal.connect(function)*. The signal itself will be emitted by a PyQtWidget, which will then call all connected functions. For the highlighting of the pixels to be masked, the *sigMouseMove* signal of the scene of the plot is used. When it is emitted, it calls the function with the current position of the cursor. The masking itself is connected to the *sigMouseClicked* signal, which also gets emitted with the according position of the cursor.

In order to mark the correct pixels, the coordinates first need to be transformed from the *ViewBox* coordinates to the coordinates of the plot. The function *ViewBox.mapSceneToView(position)* can be used for this task. The visualisation of the pixels that will be masked requires some modification of the plot generation. In PyQtGraph it is not

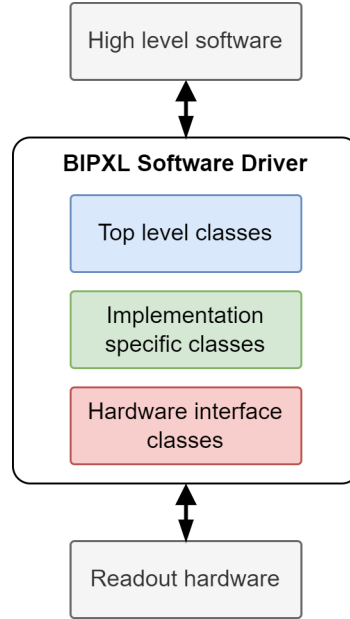


FIGURE 32: PRINCIPLE OF THE DRIVER CHANGES FROM [58].

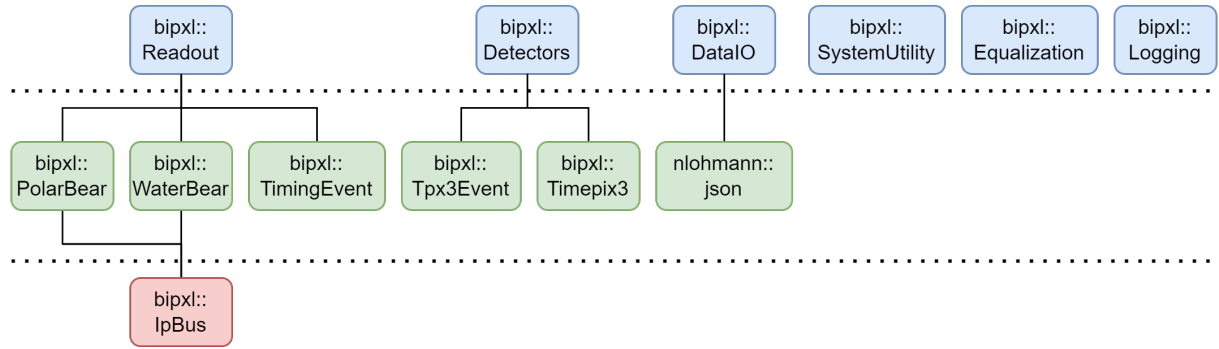


FIGURE 33: CLASS DIAGRAM AFTER REWORK FROM [58].

possible to change the alpha value of individual pixels, which is the preferred way of visualising the pixel. To implement this alpha changing, the code for the colour map generation was extracted from the open source code and recreated in the GUI code. During the creation one has access to the RGBA (Red Green Blue + Alpha) map and is able to change the required values accordingly.

4.2.1 RESTRUCTURING OF DRIVER CODE

Before the bindings were written, the code was restructured significantly by the BGI team. The driver code was extended during the different stages of the instrument and had a hardware centric approach. Different classes for each of the components in the readout chain, as well as many helper classes were added over time. No hierarchy was used during the implementation and therefore each class had many dependencies. Additionally, the user of the driver does not need access to all functions of all classes, as many functions are used for internal data-conversion. A refactoring of the code was done with the aim to separate the driver into different layers, of which only the top one is meant to be accessed by the user of the driver. The principle of the refactoring is displayed in figure 32. By using this general approach, the classes can be separated depending on whether the user needs them to be available.

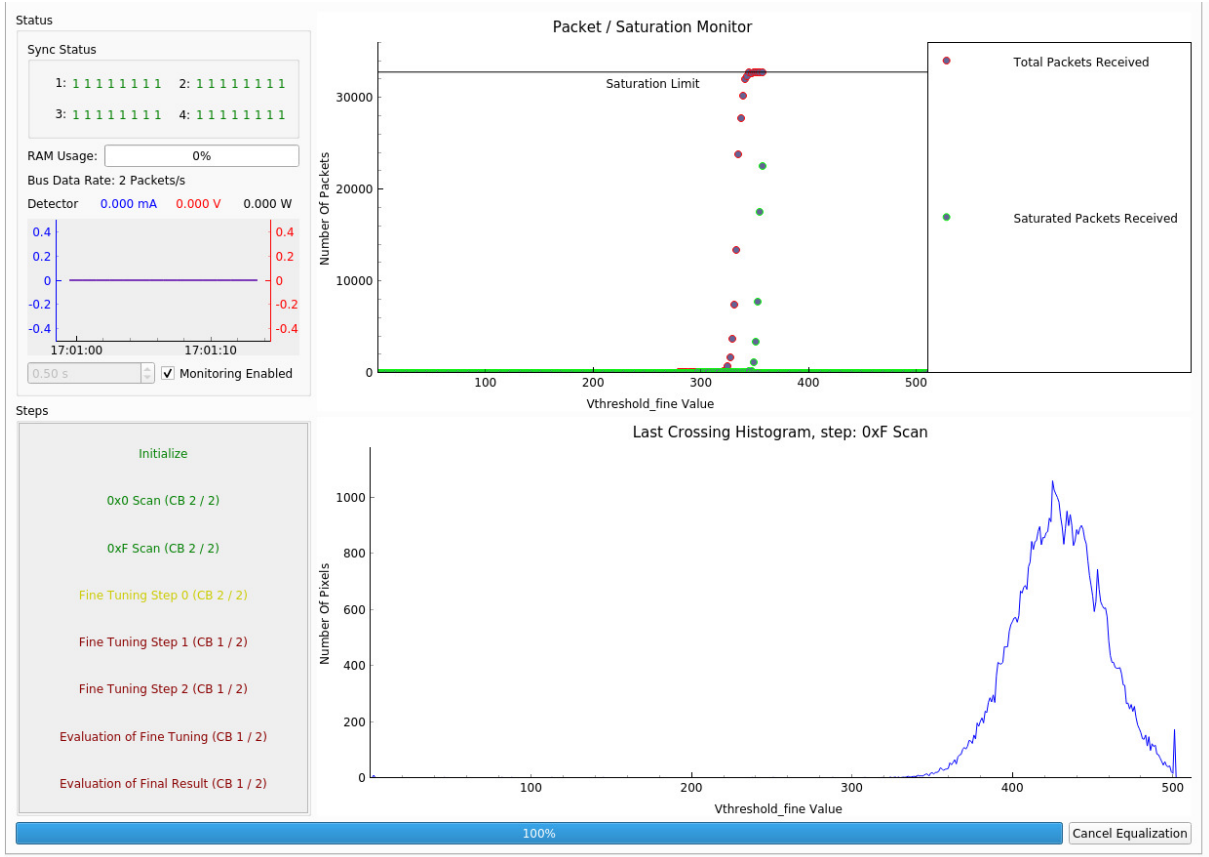


FIGURE 34: SCREENSHOT OF THE EXPERT GUI EQUALISATION MONITORING.

Another advantage of this approach is that more abstraction layers can be introduced in the different layers of the driver. This also means that the uppermost layer is less affected by changes in the hardware of the readout chain. Because only the lowest layer will handle the direct hardware communication, only this layer will need to be adapted to any hardware changes. The class diagram of the finished changes is display in figure 33. Only the classes in blue are accessible by the user.

With the implementation of the bindings in mind, this change has an additional advantage. As the number of classes that need to be accessed by the user is significantly reduced, the amount of classes and functions to be bound is also significantly reduced. The bindings also do not have to be altered a lot in the case of hardware changes, as they do not access the low level functions.

4.2.2 EQUALISATION DISPLAYING

The communication between the GUI and the driver during the equalisation is implemented with callback functions. These callback functions are connected to PyQt slots, which will then handle the displaying of the data.

The displaying elements, which were implemented, are shown in figure 34. In the top right, one can see a display of the currently running threshold scan, where both the saturated and the received pixels are shown. A pixel is saturated, when its occupancy is above 95%. If a pixel does not receive a single hit, it will not send a packet. The pattern seen in the figure is typical for a fine-tuning step as the pixels first start to receive some data and will then saturate after a

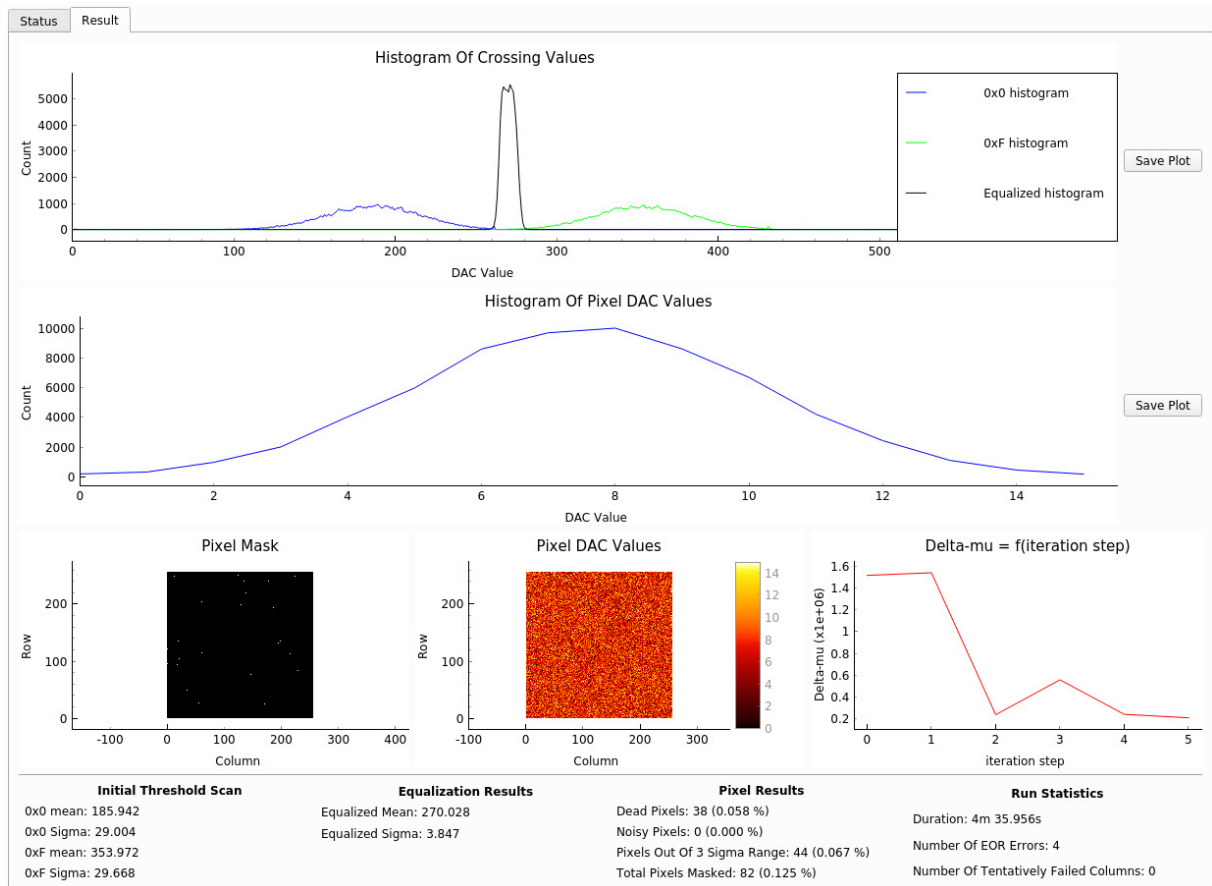


FIGURE 35: DISPLAY OF THE RESULTS TAB OF THE EQUALISATION.

short transition. The saturation limit describes the amount of pixels that should be saturated before the threshold scan stops. If a large gap between the saturated packets and the limit persists, it indicates that a lot of pixels are not working correctly or are masked. In this case, the equalisation should be restarted.

On the bottom right, one can see a plot of the last crossing histogram. By checking this histogram, one can see if the fine-tuning steps are working in narrowing the histogram. Another detail to check is that the histogram roughly resembles a Gaussian shape. If the number of pixels gets bigger in the tails, like it does on the right side of the plot, it indicates a problem. High count values in the tails usually indicate that the coarse global threshold is not in the correct value, thus causing pixels to not hit their crossing values in a reasonable range. If this gets detected, the equalisation should be stopped and the coarse threshold should be altered.

In the top left, one can see the current status of the communication channels. For the case that a channel crashes during the equalisation, it is very likely that data gets lost on said channel or noisy data will appear. The equalisation should then be restarted. Additionally, one can see the power consumption of the detector. In case of a sudden rise in power, the chip might have crashed, in which case it needs a reset or a power cycle. No values are currently visible, as this screenshot was taken in the lab, where the power supplies are not controlled by a Programmable Logic Controller (PLC). The PLC would publish the values to a FESA class, where the values can be accessed by the GUI.

In the bottom left, one can see which steps have already been performed. CB refers to checkerboard, which indicates how much of the matrix is equalised at once. In the current equalisation half of the matrix is processed. Steps that were completed successfully are displayed in green, current ones in yellow and non-completed steps in red. At the bottom, a progress bar is shown to estimate the current progression of the scan. It is currently at 100 as the equalisation in the screenshot was cancelled by the button in the bottom left.

After the equalisation finished, the display will automatically switch to the results tab. The results tab is shown in figure 35. At the top of the tab, one can see the different steps of the equalisation. Afterwards, the distribution of the DAC values is plotted. If these do not resemble a Gaussian shape, it indicates that the equalisation failed. The delta-mu plot displays the cumulative distance of all pixels to the target. The lowest value in this plot should be the last one, as it represents the result. At the bottom, the resulting mask and DAC values are plotted. If a clear pattern is visible on one of these, it indicates that there was a problem during the equalisation. The data at the bottom of the tab can be used to easily compare the results with previous results of the same chip or the results from other chips.

One common issue when equalising the matrix is that parts of a column do not get equalised correctly. Therefore, a warning gets issued when the algorithm detects that a configurable percentage of the pixels in one column was masked. This percentage is used in relation to the checkerboard value, as each new checkerboard run will reconfigure the matrix, and therefore poses a new risk of a misconfiguration of the mask in a given column.

4.3 RPU IMPLEMENTATION

The RPU application as well as the corresponding counterparts on the APU were implemented in Vitis [44]. Within Vitis, the gateway as well as the SDK from the PetaLinux compilation can be integrated to create a development environment that is aware of the interfaces of the system. Vitis also includes a cross compiler for both the ARM-A53 and ARM-R5 processors used in the APU and RPU, respectively.

Due to the differences in binary size requirements, the cross compilers have different default settings. The R5 cross compiler is heavily optimised for binary size and memory usage. One example for this is the usage of the flag `-fshort-enums`. This flag allows the compiler to pick the smallest possible data-type for an enum [59]. As this flag is disabled on the APU, some challenges can arise in the communication between RPU and APU. Therefore, it has to be ensured, that the structs only hold values with a size-defined data-type, for example `uint32_t`. Enabling this compiler flag on the APU as well might not bring the desired effect, as the shortest possible length might be determined differently by the cross compiler. It would also possibly come with an unnecessary performance loss on the APU, because the binary size requirements are not as stringent as on the RPU.

The general layout of the structs as well as possible combinations of function code and arguments need to be known on both sides of the libmetal communication. On the APU side, the combinations are checked before the transmission of the struct, on the RPU side, the

communication task will emit an error message if an unknown combination is detected. The combinations are saved in a separate header file, which has to be included in both the RPU and APU software. The Vitis projects are set up in such a way that both sides include a symbolic link to the original file. This ensures that changes in one source code are immediately available in the other part.

The tasks are implemented as infinite *for* loops. Within these loops, the task waits for a notification or an interrupt. If a notification is received, the task is performed. In the case of the communication task, the interrupt from the libmetal library serves as a task interrupt. Other tasks, like the temperature supervisor, are implemented with a delay in the main loop, as the task has to be performed every 0.5 s.

All projects for the RPU firmware are created in Vitis. An application project generally needs multiple parent projects to work properly. First, a platform needs to be created, which the project will be based on. For the RPU project, this project needs to contain the compiled gateway to be able to determine the layout of the gateway. In the APU projects, the SDK is additionally necessary for the platform initialisation. The platform will define a Board Support Package (BSP). In the BSP, one can define the settings for the compilation of the FreeRTOS, as well as libraries to include and compiler settings to be taken for the compilation of the FreeRTOS [60].

Within this platform, one can then create the software project. In this project, one can decide on libraries to be linked against during the software compilation, as well as the compiler options to be used. When compiling the application, the system project is automatically compiled as well. The compiler chosen for this task is a cross compiler, which is provided by Vitis for the specific architecture of the chip [60].

4.4 DEBUGGING FOR APU AND RPU APPLICATIONS

The RPU debugging was performed with the use of additional hardware. The Xilinx SmartLynq Data Cable [61] is directly connected to the JTAG header of the Zynq evaluation board. It hosts its own hardware server, which is then able to control various aspects of the board. With the SmartLynq, one can perform firmware, hardware and gateway debugging of the PL through Vivado as well as RPU debugging through Vitis.

The SmartLynq is connected to the general network and has its own host name. With this host name, one can then define a launch configuration for the board in Vitis. Instead of starting its own hardware server, the debugger then connects to the one running on the SmartLynq. The hardware server is used in Xilinx debugging tools as an interface between the debugging software and the actual hardware. After the connection, a normal software debugging workflow for the RPU software is possible. As the whole processing system can be controlled through the hardware server, one can also reset all processors or the RPU cores before starting the debugging.

The debugging for APU code uses the Target Communication Framework (TCF) in connection with the Eclipse debugging framework. TCF is available inside of Vitis as well as PetaLinux on the evaluation board. As the board is not directly connected to any CERN network, the port for

the TCF needs to be forwarded in the FEC. After the forwarding, a new target configuration can be created. Then, the debugging from Vitis is available for usage like other available graphical debuggers in Eclipse.

4.5 DEPLOYMENT AND VERIFICATION

In the new readout chain, a lot of different software and hardware elements need to work with each other and need to be integrated into each other. The compiled PetaLinux for the production system needs to already include the RPU firmware, while the driver needs the SDK from the PetaLinux. In order to ensure that the building always includes the latest release of each part of the readout chain, multiple GitLab CI/CD (Continuous Integration/Continuous Deployment) pipelines were implemented for the different projects. This also ensures that the implemented software meets predefined standards and can be run and tested in a test environment in the lab.

To ensure that the compiling is able to take place on the runner, a dedicated machine is used to host the GitLab runner. On this machine, the Xilinx tool chain is installed, as well as other necessary software like the IPbus. The Xilinx tool chain for this project includes Vitis and Vivado together with the Xilinx Software Command Line Tools (xsct).

By creating tcl scripts for xsct, one can setup the workspace for a Vitis project. This setup includes the linking of various other software and gateway components of the readout chain to the project. For example, in the creation of the RPU firmware, the compiled gateway is necessary. From the gateway, information of the periphery and the memory layouts is acquired, which is then used in the creation of the BSP. The APU software on the other hand needs the compiled PetaLinux. It requires the sysroot of the target system to be known by the cross compiler, which is available in the SDK and gets generated after the compilation of PetaLinux.

By having all steps executed as CI/CD pipelines, they can then be published within GitLab as well as pulled from other projects as references. This increases the repeatability of the compilation process. The created binaries and images can be automatically released and published on GitLab, making them easily available for the user or other projects and pipelines. Additionally, one can ensure code quality and correctness with automatic testing and linting within the pipelines.

Within the framework of this thesis, pipelines for the expert API, the expert GUI and the RPU firmware were implemented. The pipeline for the gateway and PetaLinux were introduced at the same time by the BGI team.

5 EVALUATION

All evaluations in this chapter are of a qualitative nature and not a quantitative one. A quantitative analysis would require a lot more effort and studying of the processes involved. In the case of the equalisation, a quantitative performance analysis is difficult, as the processes involved are inherently probabilistic. The same issue applies to the software run-time tests. The machine in use for all runs was the same TN VM, but the background tasks could not be monitored to ensure the same system load.

5.1 PERFORMANCE OF DIFFERENT SOFTWARE STACKS

In this section, the performances of the different implementations of the startup sequence are compared. The startup sequence is a procedure that is frequently run and has a run-time in the order of seconds. Therefore, one can take a sufficient number of samples to get a better understanding of the run-time on average. At the same time it is long enough, so that the software run-time occupies the majority of the application run-time as opposed to the initialisation of the Python interpreter. As most of the tested software stacks are run on a general purpose OS, the run-time will not be deterministic but rather take a different time each time the program is started. The background tasks are not being controlled in any way and will interfere with the task scheduler, which will be different for each program run.

The three variants that will be compared are the Python implementation using Pybind11, the original C++ implementation and the newly implemented RPU variant. The Python implementation will be run on a TN VM, which has standardised hardware and is able to interface with the detector through the TN and the FEC. The C++ application will need to be run in a docker container on the same TN VM. The underlying hardware will stay the same. The speed of the RPU firmware will be tested directly on the Zynq evaluation board, which will in the future be accessible from the TN. This possibility is currently not implemented. The RPU will be interfaced via an internal C++ driver running on the APU of the MPSoC. This driver will build the basis of the future communication setup with the TN.

One of the main differences for the RPU implementation will therefore be, that the IPbus does not have to be used for data transmission. The other implementations will base their communication on IPbus. A comparison will also be made between the direct connection mode and the indirect connection mode. For the direct connection mode, data will immediately be sent to the detector. In the indirect connection mode, the data is first collected and then sent in a packaged manor. This should make the communication more efficient.

In order to monitor the run-time of the procedure, it will be run 1000 times from a *for* loop inside the program. The program itself will consist of all the necessary setup steps, like the creation of the class instances involved in the startup sequence. The timing of the approaches will be performed using the time command in Linux [62].

As one can see in table 2, the results differ greatly between the implementations. Real time refers to the wall clock time that the command took to be executed. CPU time refers to the

Time	C++ in Docker	Python	RPU
Real time	13m 00.42s	12m 57.08s	3m 50.40s
CPU time	2m 55.89s	2m 59.12s	3m 49.70s
System CPU time	1m 5.78s	0m 55.61s	0m 0.14s

TABLE 2: SOFTWARE PERFORMANCE FOR DIFFERENT IMPLEMENTATIONS OF THE STARTUP SEQUENCE.

Time	Python direct connection	Python indirect connection
Real time	14m 01.25s	13m 58.89s
CPU time	3m 2.58s	3m 1.86s
System CPU time	2m 11.75s	2m 10.02s

TABLE 3: SOFTWARE PERFORMANCE OF THE STARTUP SEQUENCE WITH AND WITHOUT A DIRECT CONNECTION FROM PYTHON.

time the process spent in the application itself, while system time refers to the time the process spent in kernel space.

From the results, one can see that the difference between the C++ implementation and the Python implementation is very small. The overhead of the Pybind11 bindings in comparison to the native C++ calls does not seem to be large. Additionally, the overhead of the docker container is not present for the Python bindings. These effects partially balance each other out and as such, it is not possible to determine, which of the two implementations performs better.

In comparison to the programs running in the TN, the RPU software solution has a lower wall time duration. The main contribution to this difference is probably the difference in the data transmission. On the RPU, the data does not have to be transmitted back and forth over the TN and is processed directly on the chip, avoiding the additional latency and run-time from IPbus. The CPU time for both processes on the other hand is very similar, which supports the suspicion that most of the duration of the TN implementations was actually caused by the process waiting on results or network delays. In the future, a communication protocol between the TN applications and the Zynq board needs to be implemented. As this interface is not yet implemented, a direct comparison of the times on a quantitative level is not practically relevant. The communication implementation into the TN will add wall time to the RPU implementation, the amount depending on the actual implementation and communication framework chosen.

Generally, one can conclude that the performance has not changed between the Python and the C++ implementation. The RPU implementation might bring an advantage over the previous iterations, but it is not yet conclusive, as a component in the transmission chain is missing. Especially for data-heavy tasks like the equalisation, a performance gain is very likely, as the data does not have to be transmitted through the network. Only the results of the computation need to be published to the end user, which are usually not as large as the intermediate steps. This would require the driver to be implemented locally on the MPSoC.

In table 3, one can see the difference for the program’s run-time with and without the direct connection enabled. The tests were only performed with the Python implementation as it was

already concluded that no big time difference exists between C++ and Python. The time difference in all categories is not large. This could be explained by the procedure itself. The startup sequence requires a precise order of commands to be sent to the Timepix3. To ensure this order, the buffer is flushed before sending a new command. Therefore, no big advantage can be gained by aggregating the commands. In order to see a difference, a data taking task would probably be more suitable. The simulation of such a task would require additional implementations, which were outside of the scope of this thesis. As the IPbus software will most likely not be used in a future internal driver, the effort for doing this simulation might not be worth.

The responsiveness of the expert GUI has not significantly improved over the Panda GUI. In Panda, crashes or a stuck application were common when the RAM was filled with packets. The driver and the IPbus implementation would then try to empty the RAM, blocking the thread for any action. This problem could not be alleviated in the new expert GUI. It is likely caused by the underlying data transmissions still having to be transferred through the TN to the detector. If the IPbus is fully utilised by noisy data packets, it will still effectively freeze the GUI, as the computing resources of the process will be used up by it. A resetting of the RAM is not possible in this case, as the data transmission towards the detector is not usable anymore. This detail was not analysed in depth, as the IPbus will be replaced by a different network data-transmission technology in the MPSoC version of the readout.

5.2 EQUALISATION PERFORMANCE

The equalisation tests will be performed on the same Timepix3 BLM with all parameters kept the same except for the iterated one. The default run will consist of two fine-tuning iterations, 1000 cycles of shutter opening time, activated moving average, activated interpolation and a checkerboard level of 4. A checkerboard level lower than 4 usually results in an unstable chip and risks crashing the chip during the equalisation. The expert GUI is used to start and monitor the procedure. All power values and chip settings are kept the same throughout the procedures.

5.2.1 EQUALISATION PROCEDURE

The display of the equalisation result for the default run is shown in figure 35. It is described in section 4.2.2. The values in the bottom show the same point. The distribution of the applied DAC values is also Gaussian, which indicates that the result is reasonable. If the applied DAC values would be biased, it would indicate that the threshold values of the pixels are biased as well, which is highly unlikely.

In the lower part, one can then see the resulting mask and DAC values, which do not show a particular pattern on it. The delta-mu plot in the bottom is magnified in figure 37. It shows the cumulative distance of all pixels to the target value on a logarithmic scale against the different steps of the equalisation. The cumulative distance is calculated with

$$\text{delta} - \mu = \sum_{\text{column}=0}^{255} \sum_{\text{row}=0}^{255} |\text{crossing}[\text{column}, \text{row}] - \text{target}|. \quad (5)$$

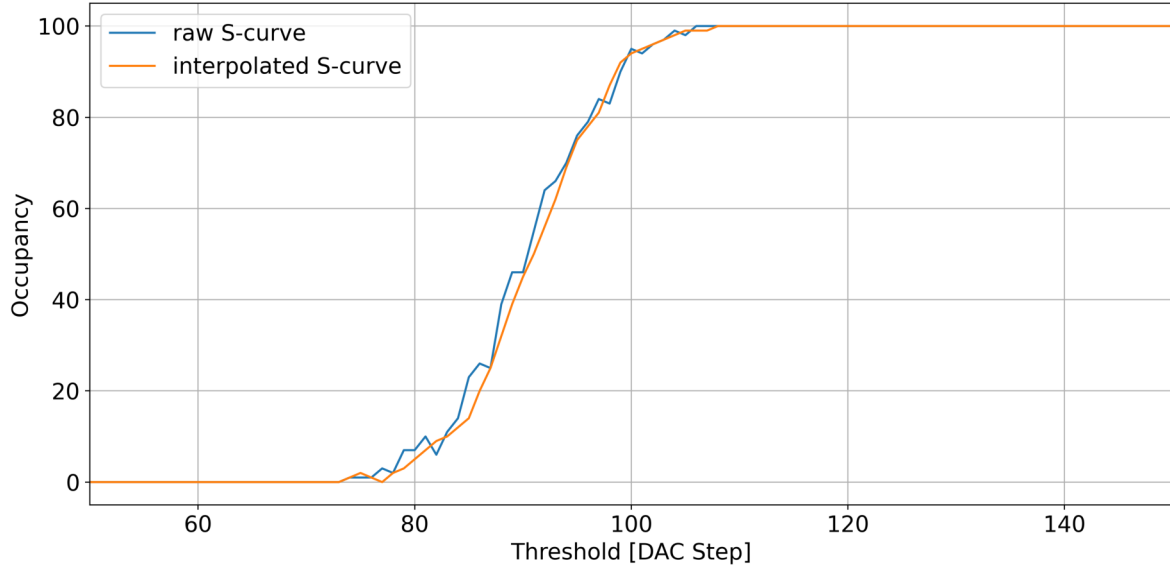


FIGURE 36: COMPARISON BETWEEN AN S-CURVE WITH INTERPOLATION AND MOVING AVERAGE APPLIED AND A RAW S-CURVE.

This distance correlates with the σ of the result distribution and therefore the quality of the equalisation result. As one can see, this value starts fairly high for the first two steps, which resemble the 0x0 and 0xF scans. Afterwards, the delta-mu decreases significantly in the first step, the interpolation. In the first fine-tuning step, the distance increases significantly. This effect happens due to the overshoot discussed in section 3.1.1. In the second fine-tuning step, one can see another decrease in the delta-mu value, but a very similar value to the interpolation step.

In order to increase the speed of the equalisation, an interpolation was implemented. Only every other value is measured and the values in between are then estimated. The S-curve is smoothed further with a moving average window. The result of this can be seen in figure 36. The difference between the two S-curves is minimal. The impact of the change is reduced further by the limited resolution of the local threshold DAC.

In the *ft eval* step, the minimum distance to the target is chosen for each pixel DAC. As one can clearly see in the plot, this leads to another decrease in the delta-mu value and thus increase in the equalisation quality. The difference between the *ft eval* and the *result* step is the masking of pixels that are outside of the 3σ environment of the target value. In this step, no significant change of the delta-mu value can be observed, as only a limited amount of pixels are masked. In the shown run, 70 of the 65536 pixels were masked in this step. The effect of this masking will be visible if the threshold gap is applied and the shutter is opened. When the pixels are not masked, it increases the probability that additional masking is necessary after the equalisation.

To check, if more fine-tuning iterations would improve the quality of the equalisation, another run was done with four fine-tuning steps. The resulting delta-mu plot can be seen in figure 38. As one can see in comparison to figure 37, the performance gain from the additional fine-tuning steps is not visible. This is most likely caused by the fact that the interpolated optimal DAC value is generally close to the optimal solution for most pixels. If a solution closer to the optimum exists, it most likely is located close to the interpolated value. If the optimal value would be

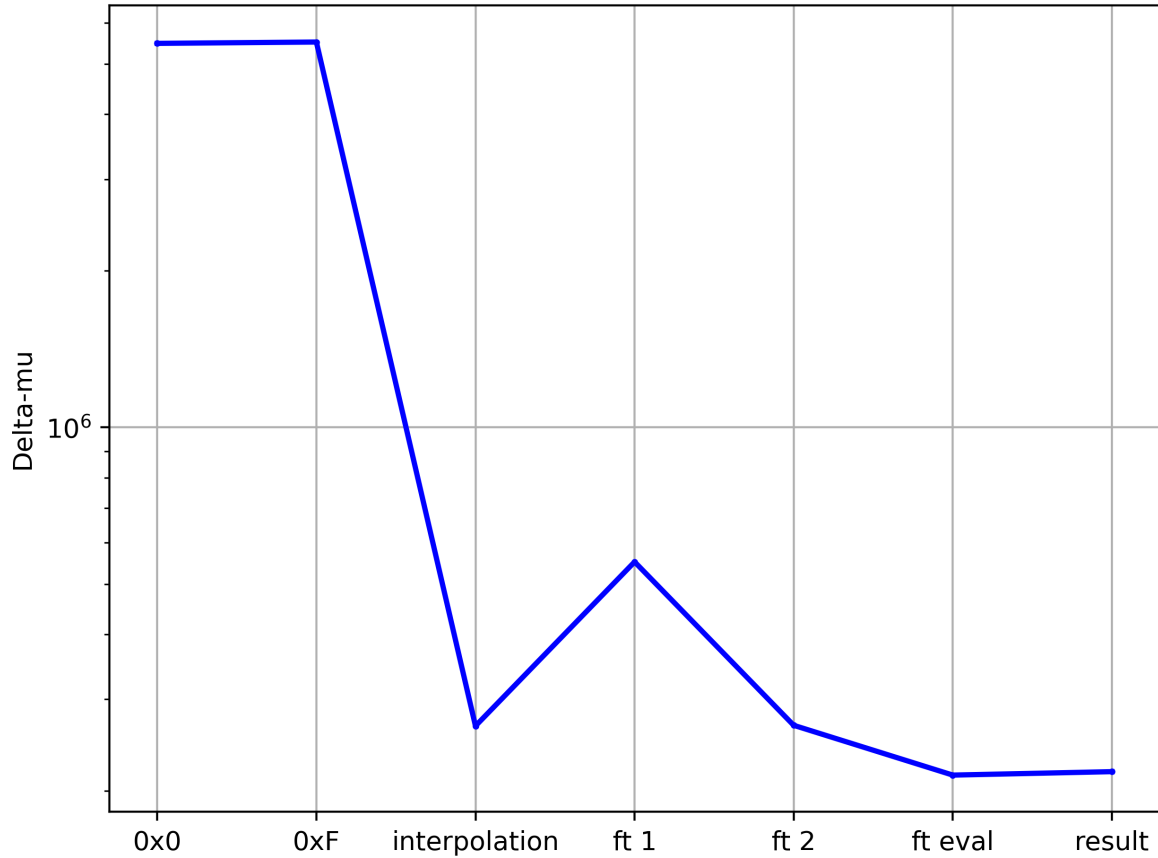


FIGURE 37: LOGARITHMIC CUMULATIVE DISTANCE OF THE 50% CROSSINGS FROM THE TARGET VALUE.

further away from the interpolated value, it would coincide with a very nonlinear distribution of the local threshold values over the global threshold value. The quantitative effects of the fine-tuning are discussed in section 5.2.3.

5.2.2 PERFORMANCE OF DIFFERENT REFERENCE SIGNALS

One newly implemented feature is the possibility to perform the equalisation with the TPs as reference signal. As stated in section 3.1.3, the TP can charged reliably only on a part of the matrix at once. This results in an additional constraint to the minimum duration of the equalisation with this reference.

In figure 39 and figure 40, one can see the results of an equalisation with the noise floor and the TPs as reference. For this comparison, both equalisations were done with a checkerboard level of 16. The resulting values are displayed in table 4. From the pictures in 39 and 40, one can see that no patterns are visible in the resulting DAC settings or masks. This indicates that the equalisation worked without parts of the chip being unpowered or noisy. The other parts of the results also seem to look similar.

For a further comparison of the results, the values in table 4 will be discussed. Added to the comparison is a default equalisation. For the comparison of the equalisations, the 0x0 scan is

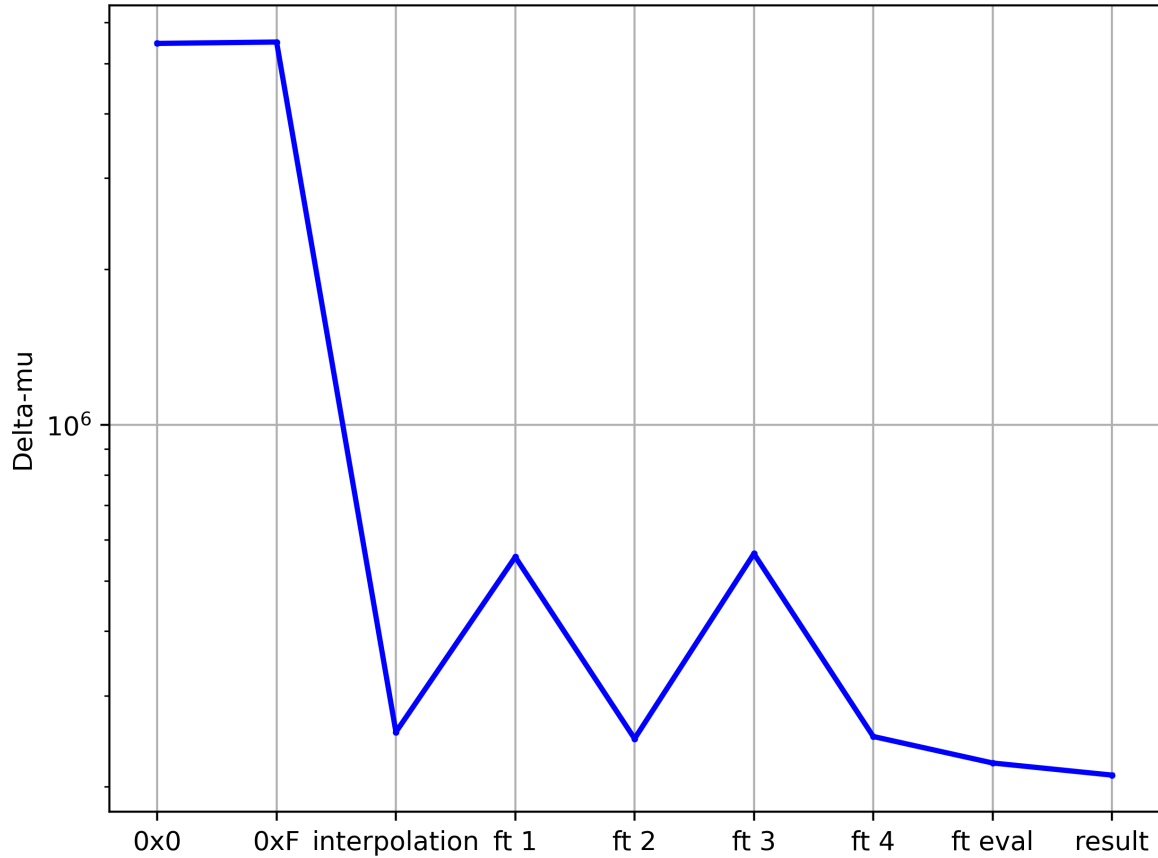


FIGURE 38: LOGARITHMIC CUMULATIVE DISTANCE OF THE 50% CROSSINGS FROM THE TARGET VALUE FOR 4 FINE-TUNING STEPS.

used as a reference width without the equalisation, as the 0xF scan should be characterised by a comparable value.

From the values, one can deduct that the difference of the performance between checkerboard level 4 and 16 with the noise floor as reference is not large. The mean values and σ are roughly the same with the exception of the σ of the equalised histogram. The value difference is equivalent to a result quality increase of roughly 10%. As the noise floor is used in both examples, the results will differ from run to run and the recorded difference is not large enough to prove a big advantage of checkerboard 16 above checkerboard 4. One major difference between checkerboard 4 and 16 is the duration, as the procedure has to be repeated 4 times. The duration of the shutter opening is kept the same between the checkerboard levels, resulting in the multiplication of the checkerboard 4 duration by roughly 4. The additional duration can be interpreted as the random duration variations between different runs.

Another difference in the values can be seen in the masked pixels. The value increases for the higher checkerboard level, both with the noise floor and TPs as reference. Even though the increase between checkerboard 4 and 16 is visible, it does not have an impact on the instrument performance. After the equalisation and masking, the RF shield will be masked in a separate step, which is not discussed in detail in this thesis [4]. During this procedure, additional pixels will need to be masked and it will be ensured that each column includes the same number of

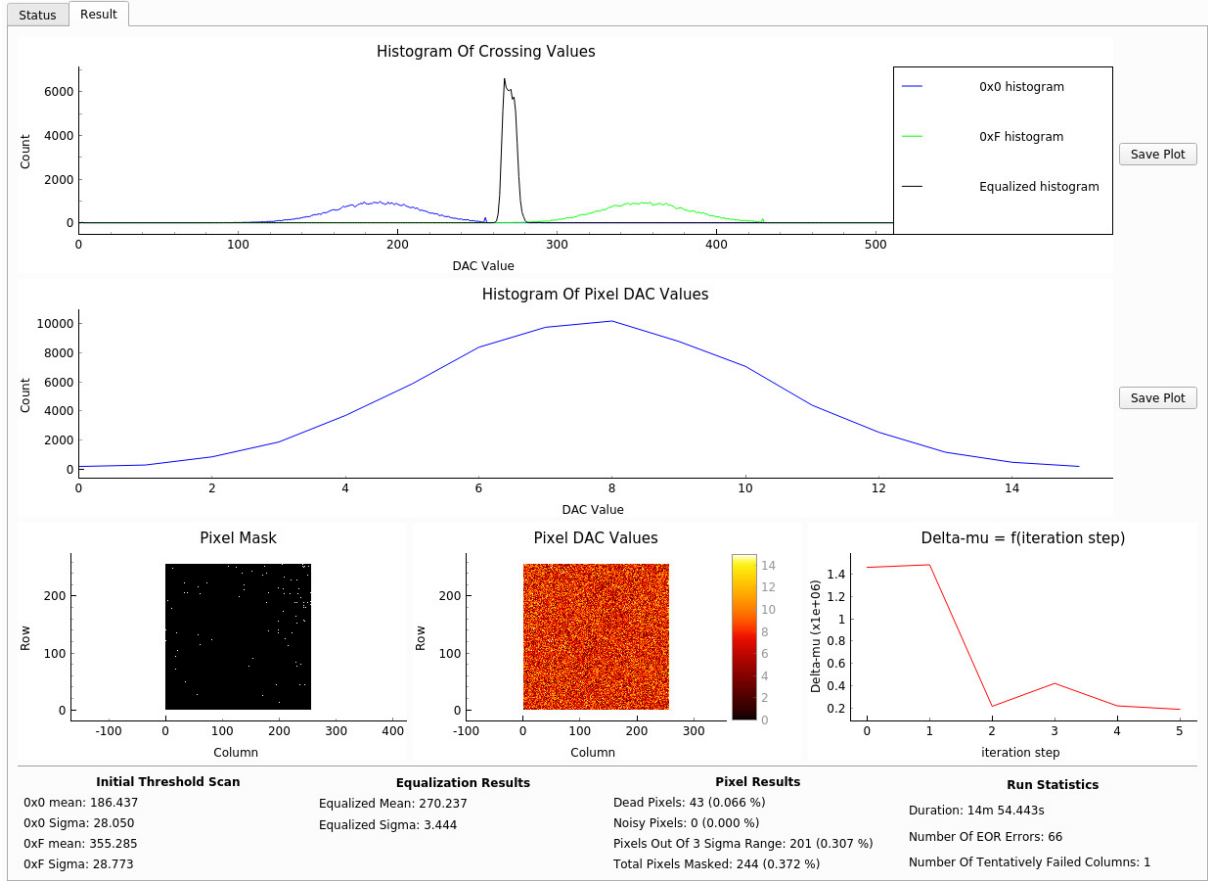


FIGURE 39: EQUALISATION WITH THE NOISE FLOOR AND A CHECKERBOARD LEVEL OF 16.

masked pixels. If this would not be the case, the creation of the profile would be biased, as it sums up the hits in a column. Small differences in the amount of masked pixels will therefore not make a large impact, as long as they are not clustered in single columns.

Between the noise floor as reference and the TP as reference, one can see an improvement with regards to the σ of the equalised distribution. One of the downsides can be seen in the duration: The noise floor equalisation can be run significantly faster than the TP equalisation. While the precise duration of the TP generation could be adapted, it still needs to be run with a checkerboard level 16, which will generally take a lot more time than checkerboard level 4. In the TP settings, the number of TPs as well as the charging voltage and period will have an impact on the duration of the equalisation. The duration of the noise floor equalisation could also be adapted by changing the shutter opening time in each data-taking segment. In table 4 as well as figure 39 and figure 40, one can see that the mean of the equalised as well as the 0x0 scans is different. This can be explained by the different injection energies. The energy of the TPs is chosen to be well above the noise floor at an energy equivalent of roughly 1500 electrons, while the noise floor is a lot lower than that value. This energy can be changed for the TPs by changing the charging voltage. The charge time and period of the TP generation might need to be altered when changing the voltage. A larger injection energy will generally require a larger period and charge time and consequently a increased run-time. The difference between parameters is explained in section 5.2.3.

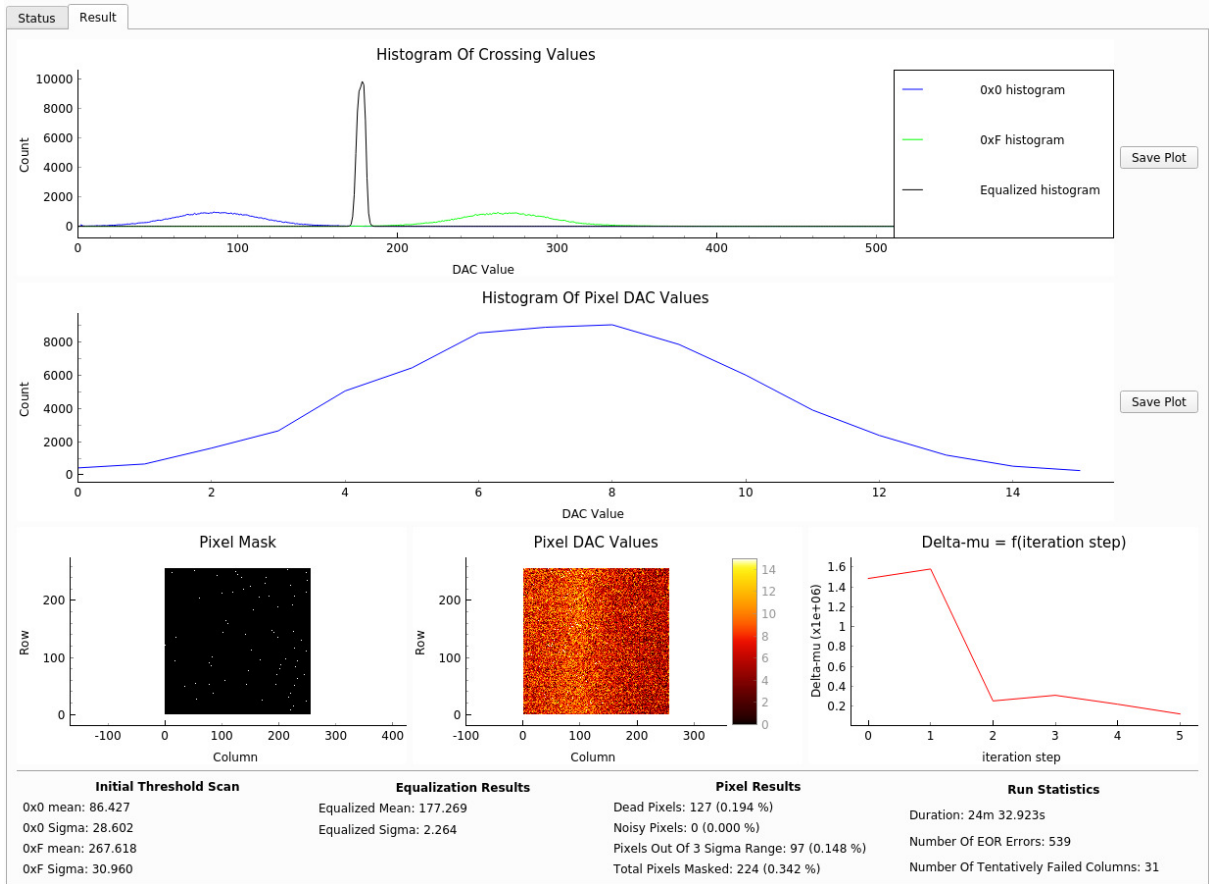


FIGURE 40: EQUALISATION WITH TEST PULSES AND A CHECKERBOARD LEVEL OF 16.

The equalisation performance generally has a lower boundary for the precision. As the local threshold DACs only have a limited resolution of 4 bits, no perfect equalisation can be achieved.

In figure 41, one can see three examples of bad equalisation results, which are easy to spot in the end of equalisation screen of the new GUI. In the first mask, one can clearly see that parts of the chip were almost entirely masked. This indicates a problem with the powering of that part of the chip and the equalisation as well as the chip itself should be restarted. The problem is easily identifiable in the mask, as too many pixels get masked.

In the second picture, one can see the result of a misconfiguration of the pixel mask during the equalisation. In the case of a pixel mask being wrongly transferred to the chip, such an error can occur. The pixel mask is applied by columns. In the third plot, one can see a bias of the DAC values. The values on the right side are a lot darker than in the middle or left of the matrix. This indicates a problem, as the values should be equally distributed over the matrix. Both of the last problems would not be easily identifiable by just looking at the statistics of the run, as no unusual amount of pixels were masked.

5.2.3 VERIFICATION OF EQUALISATION IMPROVEMENTS FOR DIFFERENT PARAMETERS

In this section, the impact on the quality of the equalisation that results from different parameters will be evaluated. The resulting values are displayed in table 5 and table 6. Generally, one

Value	NF CB 4	NF CB 16	TP CB 16
Mean of 0x0 DAC value distribution	185.942	186.437	86.427
σ of 0x0 DAC value distribution	29.004	28.050	28.602
Mean of equalised distribution	270.028	270.237	177.269
σ of equalised distribution	3.847	3.444	2.264
Duration	4m 35.956s	14m 54.443s	24m 32.923s
Masked pixels	82	244	224

TABLE 4: COMPARISON OF THE RESULTS OF DIFFERENT EQUALISATIONS, ALL VALUES AS GLOBAL DAC STEPS

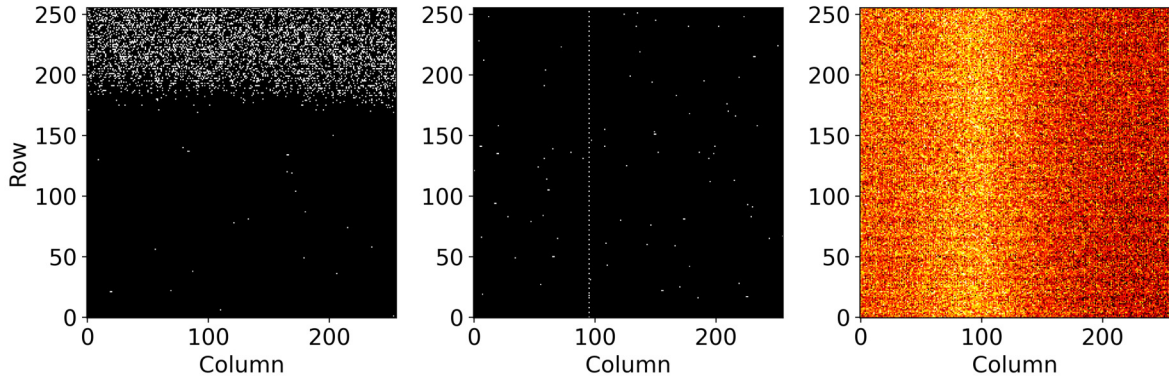


FIGURE 41: EXAMPLES FOR MASKS AND DAC VALUES THAT ARE NOT SUITABLE FOR OPERATIONAL DEVICES.

can say that the change of the parameters did not have a major influence on the end result of the equalisation. The mean and σ of the equalised distribution are similar for all tested configurations.

In table 5, different values for the shutter opening time and the fine-tuning are used. After the shutter opening time was increased from 1000 cycles to 10000 cycles of the 40 MHz clock, the result of the equalisation did not improve, while at the same time the run-time of the equalisation increased significantly. It is noteworthy that the run-time value did not increase by the factor of the shutter opening time increase. This is most likely caused by the data transmission with the chip. The total run-time is seemingly dominated by the readout of the pixel matrix, which stays roughly the same between the different shutter opening times.

Another value, which was changed in table 5, is the number of fine-tuning steps. The evolution of the distribution during the first fine-tuning steps was discussed in section 5.2.1. The quality gain from more steps seems to be very small. This can be explained by the limited resolution of the local DACs. Once the optimal value is found for a certain pixel, more fine-tuning steps will not be able to improve the result. The only aspect that can be improved during additional fine-tuning steps is the minimisation of the amount of dead or noisy pixels. It is possible that a pixel is only temporarily dead or noisy, and therefore the 0x0 or 0xF value is wrong, resulting in

Value	Reference	4 fine-tuning steps	10000 cycles of shutter opening
Mean of 0x0 DAC value distribution	185.641	185.553	185.448
σ of 0x0 DAC value distribution	29.085	29.078	29.438
Mean of equalised distribution	268.823	269.158	269.347
σ of equalised distribution	3.931	3.878	3.972
Duration	4m 42.874s	5m 38.664s	8m 43.824s
Masked pixels	156	90	168

TABLE 5: COMPARISON OF THE RESULTS OF EQUALISATION RUNS WITH DIFFERENT PARAMETERS, ALL VALUES AS GLOBAL DAC STEPS

Value	No interpolation and no moving average	No interpolation	No moving average
Mean of 0x0 DAC value distribution	185.610	185.559	185.402
σ of 0x0 DAC value distribution	28.899	28.892	28.901
Mean of equalised distribution	268.889	269.103	268.965
σ of equalised distribution	3.947	3.912	3.908
Duration	8m 22.702s	8m 27.631s	4m 44.391s
Masked pixels	82	91	98

TABLE 6: COMPARISON OF THE RESULTS OF EQUALISATIONS RUN WITH INTERPOLATION AND MOVING AVERAGE SETTINGS, ALL VALUES AS GLOBAL DAC STEPS

a skewed value after the interpolation. As can be seen in the table, the number of these pixels is generally very low, resulting in only a minor quality gain in the equalisation result from the additional steps. It is therefore generally not worth the extra time each step takes.

As it can be seen in table 6, the interpolation of half the values decreases the time spent on the equalisation significantly without interfering with the result. The moving average on the other hand does not have a big impact on the output of the procedure, neither in time nor in impact on the final result. One advantage of the moving average is that it can reduce the impact of radiation or other particles that hit the pixel during the equalisation. The tests with the equalisation were done in the lab, where this could only happen due to background radiation, which only rarely impacts the sensor compared to the radiation level in the machine. As the moving average does not increase the time of the equalisation and has a potential gain, it can be seen as a good feature to keep.

6 SUMMARY AND OUTLOOK

6.1 SUMMARY

In the work on the calibration algorithms, the equalisation was modified significantly in order to improve the quality of the result as well as the run-time of the procedure. The performed adaptations include a more targeted approach to the threshold scans. From the result of the first two runs of the threshold scans, the target value is interpolated. With an educated guess, one can predict the rough width of the resulting distribution, enabling the following scans to only run over parts of the global threshold range and therefore be a lot quicker. To improve the speed of the scan itself, previously unused features of the chip were integrated into the procedure.

Previously, the equalisation relied solely on the noise floor as a reference signal. This approach comes with the problem that this reference can change due to a temperature variation. As the readout of the data during the equalisation will change the temperature, the signal is not stable during the whole procedure. To alleviate this issue, the equalisation with TPs was implemented. The TPs are available from capacitances, which can be charged and then unloaded into the pixel frontend. The charging voltage can be modified to allow the injection of an energy, which lies outside of the noise floor. The results of the equalisation with this approach were better than with the noise floor as reference signal. A downside of the approach with TPs is the additional time needed for the equalisation. The capacitances need to be fully charged, which takes a considerable current from the charging network. This requires the equalisation to only work on parts of the chip at once, which in turn increases the run-time of the equalisation significantly. The equalisation with the TPs therefore adds the option to run a longer procedure with an additional quality.

In order to ease the commissioning and initialisation of the instrument, several operations were automatised, which previously had to be done manually. Some DAC values of the chip need to provide a certain voltage in order for the chip to function properly. The threshold DAC values need to be adapted depending on the needs of the data-acquisition. At the same time, the data-acquisition thresholds need to be at the same value for all 4 chips in the operational BGI. For these tasks, an automated calibration was implemented for the setting of a DAC to a specified output voltage.

Another step of the calibration of the instrument, which was automatised, is the identification of noisy communication channels. Due to ageing and radiation damage, the properties of the communication channels will change. Eventually, it is possible that channels will fail entirely. In the backend the clock of the data-transmission needs to be picked up from the data. If different transmission lines introduce a delay because of ageing, it will not be possible for the readout chain to pick the clock up. An algorithm was implemented to detect bad channels based on the received data-rate. The algorithm can then try to alter delay modules in the backend to compensate for the clock skew on a certain communication channel.

At CERN, different GUIs are used for different tasks when controlling the instruments. On one hand, an operational GUI is used in the CCC, to allow the operators to take data and start the instrument. In addition, an expert GUI is available, which is used to calibrate and set up

the instrument and requires a full feature set. The previously used Qt based expert GUI needed to be run inside a Docker container or needed to be run on a machine connected directly to the backend of the BGI. In order to make the GUI available in the TN, a transition from Qt to PyQt was performed within this thesis. During this transition, many elements needed to be redesigned and were able to be moved in the GUI. The new expert GUI has a more user and application centred approach, making the initialisation of the instrument easier. The previous expert GUI was more hardware centred. Monitoring elements for the power infrastructure were implemented, which were not available to the previous GUI as it was not in the TN. The monitoring of the equalisation procedure was also implemented. Together with the added option for the cancellation of the procedure, it is now possible to inspect the intermediate results while the equalisation is still running and then restart the equalisation if necessary.

In the process of creating the expert GUI, it was necessary to write Python bindings for the driver. The BGI driver is written in C++, for which Pybind11 bindings were written. Before this step, the driver was refactored with the BGI team in order to make it more suitable for hardware changes in the future. An abstraction layer was added for most functionalities, hiding the hardware implementation from the user of the driver. This allows for a more adaptable software solution if the hardware should change.

In the last months, the BGI team began work on the implementation of a new readout chain. Instead of splitting the computations between frontend and backend FPGAs, now all computations will be done on a Xilinx MPSoC located in the backend. This opens the possibility to rethink the software stack, as the tasks can now be done closer to the actual hardware than previously. The APU on the MPSoC can be given direct access to gateway registers. Additionally, RT cores are available to be used. In the context of this thesis, firmware was written for the RT cores, which will be used in the new readout chain to incorporate monitoring and control tasks. The implemented firmware includes temperature monitoring as well as running the startup sequence to reset and initially configure the chips. An interface was also implemented to allow the APU to initiate the startup sequence as well as to get the temperature of the detectors. The interface was implemented in a way, which is easily adaptable to include more functionality in the future.

For the software projects in use, CI/CD pipelines were implemented together with the BGI team. These pipelines check the code for linting errors and compile the binaries. Additionally, tests are included for certain parts of the software to ensure compatibility. The binaries can then be deployed to operational devices from these repositories. Due to the structure of the MPSoC it is also possible to include the gateway binary and compiled FPGA code directly into the PetaLinux running on the APU. The implemented pipelines save effort when rebuilding the PetaLinux and make the process of releasing a new software version more reliable and repeatable.

6.2 OUTLOOK

The outlook section will focus on the possible future improvements regarding the software and readout.

One of the performance limitations is the current implementation of the network communication

from the driver to the detector using IPbus. Originally, this was necessary due to the fact that direct communication with the FPGA in the backend was needed. After the readout chain upgrade, the MPSoC is now connected to the FEC and the driver could be run on the MPSoC, making all communication with the detector a lot faster and more reliable. For this approach, a communication protocol between the application and the MPSoC would become necessary. This would also help to separate the driver tasks from the application tasks. A solution would also be necessary for the case, when different operators want to access the device at once.

The calibration of DAC voltages is currently only implemented for the Vfbk as well as the VThreshold voltages. During the commissioning and calibration of the instrument, it becomes often apparent, that the thresholds have to be adjusted after the procedure. If this is not done, the amount of background radiation, that each chip detects, can be vastly different. Theoretically, the detection should be mainly dependant on the threshold gap, which is configured to have the same voltage on all chips. This value is then adjusted to equalise the detection between the chips. One issue for the difference might lie in the output voltages on DACs used in the pixel front-end. These DACs are configured with the same digital value but will probably put out a different voltage due to production differences. Many of these DACs have a direct impact on the detection efficiency and ToA and ToT values that the chip will publish [63]. The implemented heuristic could also be used with minimal changes to change adjust the output voltages of all chips towards each other.

The usage of the RT cores is currently limited to the startup sequence and the temperature monitoring. As the startup sequence requires the setting of all configurations, it would only require a minor effort to adapt the current implementation for this task. This would mainly require additions of the communication interface between the APU and RPU, as the translation of the data for the gateway is already done. A possible addition to the feature set of the RPU would be the automatic monitoring of the communication channels. If a communication channel starts to get out of sync, it can produce lots of data, which might fill up the RAM in the backend quickly. This channel could automatically be turned off by the RPU or other measures could be taken.

To improve the reliability of the monitoring and controls of the chip, more monitoring, configuration and control tasks could be moved to the RPU. The RPU on the Zynq MPSoC supports a multi-processing mode. In this mode, the scheduler is able to select a task for execution on one of the two cores. It is also possible to specify, which task is allowed to be run on a specific core. This would allow the monitoring tasks to be run on one core and the control tasks to be run on another core. The inter-task communication could still be done with notifications. This would allow the monitoring task to notify the control tasks to reconfigure the chip automatically in case a problem is detected.

Another possible improvement would be the separation of the data-flows for data packets and other packets. On the gateway level in the MPSoC, one would need to implement a filter for the incoming data-stream. The filter would then need to write the control and monitoring packets into a separate FIFO. The FIFO could then be read out by the RPU, which would update the status of the chip internally. The APU would then be able to request specific status information

from the RPU, without the data-path being interrupted. This approach would increase the reliability of the control and monitoring infrastructure. Currently, it can happen, that due to the large amount of incoming data, monitoring packets are dropped in the backend.

In the framework of the HL-LHC, it is foreseen to implement an additional beam profile monitor for the usage in the LHC. One of the promising candidates for this would be a modified version of the PS-BGI. On the readout side, the main difference would be the readout bandwidth. The chip, which will most likely be used for this detector, would be the Timepix4. Compared to the Timepix3 it has a larger chip area as well as better timing resolution and bigger readout bandwidth. Because of the measurement requirements in the LHC, it would be necessary to significantly upgrade the performance of the readout chain. One possible point to do this would be the decoding of the data-packets in gateway. This is currently done in software, which is probably slower. Another approach to increase the performance of the readout chain would be to multi-thread parts of the driver. If the decoding of the packets would be done in a multi-threaded way, a performance gain could probably be achieved.

One value that will generally limit the performance of the equalisation in a data-taking application is the energy gain. With the equalisation, the matrix is equalised at one specific energy level. If the energy is larger or smaller, the equalisation theoretically needs to be redone. While the average energy of electrons hitting the sensor will be 10 keV, the energy of individual electrons will vastly differ. If the electron is close to the Anode, it will receive less energy from the electrical field than a particle, which is close to the Cathode. In the beam profiling application this will often happen, as the beam is not always in its ideal orbit and the bunches have a certain spatial expansion. As the energy changes, the matrix would theoretically need to be reequalised for each energy. One possibility to get around this issue is to perform an energy calibration [35]. During the energy calibration, multiple S-curves will be taken for each pixel at different known injection energies. These values will then be used to determine a function, which can be used to correct for the sensitivity differences of the pixels at different energies.

In the implementation of the expert GUI, the current, voltage and power delivery for all relevant power supplies can be monitored. To change the value, it is currently necessary to open a separate GUI. As all settings and monitoring values are available in the TN, one could implement a separate power tab into the expert GUI in order to manipulate the settings of the power supplies.

I herewith declare that I have composed the present thesis myself and without use of any other than the cited sources and aids. Sentences or parts of sentences quoted literally are marked as such; other references with regard to the statement and scope are indicated by full details of the publications concerned. The thesis in the same or similar form has not been submitted to any examination body and has not been published. This thesis was not yet, even in part, used in another examination or as a course performance.

A handwritten signature in blue ink, reading "Leonard Thiele". The signature is written in a cursive style with a light blue background.

Rostock, 29.01.2023

REFERENCES

- [1] *Physics / CERN*. 2022. URL: <https://home.cern/science/physics> (visited on 27/12/2022).
- [2] *The accelerator complex / CERN*. 2022. URL: <https://home.web.cern.ch/science/accelerators/accelerator-complex> (visited on 01/11/2022).
- [3] *BGI Web*. 2022. URL: <https://bgi.web.cern.ch/> (visited on 27/12/2022).
- [4] Hampus Sandberg. ‘Development of a Novel Transverse Beam Profile and Emittance Monitor for the CERN Proton Synchrotron’. PhD thesis. Manchester U, 2020. URL: <https://cds.cern.ch/record/2747870?ln=en>.
- [5] ‘20210921-KTForumMA-BGI’. In: (2022). URL: <https://indico.cern.ch/event/1064535/contributions/4475592/attachments/2313242/3937259/20210921-KTForumMA-BGI.pdf> (visited on 27/12/2022).
- [6] CERN. *ALICE*. 2022. URL: <https://home.cern/science/experiments/alice> (visited on 03/12/2022).
- [7] CERN. *The subterranean ballet of ALICE*. 2023. URL: <https://home.cern/news/news/experiments/subterranean-ballet-alice> (visited on 12/01/2023).
- [8] *Physics / CERN*. 2022. URL: <https://home.web.cern.ch/science/physics> (visited on 01/11/2022).
- [9] *Experiments / CERN*. 2022. URL: <https://home.web.cern.ch/science/experiments> (visited on 01/11/2022).
- [10] CERN. *The Antiproton Decelerator*. 2022. URL: <https://home.web.cern.ch/science/accelerators/antiproton-decelerator> (visited on 01/11/2022).
- [11] *The birth of the Web / CERN*. 2022. URL: <https://home.cern/science/computing/birth-web> (visited on 03/12/2022).
- [12] CERN. *Contribute to society*. 2022. URL: <https://home.web.cern.ch/about/what-we-do/our-impact> (visited on 01/11/2022).
- [13] S. Baird. *Accelerators for pedestrians*. 22nd Feb. 2007. URL: <http://cds.cern.ch/record/1017689>.
- [14] *Radioisotopes: What Are They and How Are They Made?* 2022. URL: https://ehss.energy.gov/ohre/roadmap/achre/intro_9_4.html (visited on 01/11/2022).
- [15] B. Dehning et al. ‘The LHC beam loss measurement system’. In: *2007 IEEE Particle Accelerator Conference (PAC)*. 2007, pp. 4192–4194. DOI: [10.1109/PAC.2007.4439980](https://doi.org/10.1109/PAC.2007.4439980).
- [16] CERN. *Superconductivity*. 2022. URL: <https://home.cern/science/engineering/superconductivity> (visited on 01/11/2022).
- [17] CERN. *Accelerating: Radiofrequency cavities*. 2022. URL: <https://home.cern/science/engineering/accelerating-radiofrequency-cavities> (visited on 24/11/2022).
- [18] *Taking a closer look at LHC - Buckets and bunches*. 2022. URL: https://www.lhc-closer.es/taking_a_closer_look_at_lhc/0.buckets_and_bunches (visited on 29/11/2022).

- [19] CERN. *High-Luminosity LHC*. 2022. URL: <https://home.web.cern.ch/science/accelerators/high-luminosity-lhc> (visited on 01/11/2022).
- [20] S. Redaelli et al. ‘Hollow electron lenses for beam collimation at the High-Luminosity Large Hadron Collider (HL-LHC)’. In: *Journal of Instrumentation* 16.03 (2021), P03042. ISSN: 1748-0221. DOI: [10.1088/1748-0221/16/03/P03042](https://doi.org/10.1088/1748-0221/16/03/P03042). URL: <https://iopscience.iop.org/article/10.1088/1748-0221/16/03/P03042>.
- [21] *BGIPXL System portal*. CERN Internal Document, 2022. URL: <https://wikis.cern.ch/display/BEBI/System+portal+%3A+BGIPXL> (visited on 29/11/2022).
- [22] Swann Levasseur. ‘Development of a Hybrid Pixel Detector Based Transverse Profile Monitor for the CERN Proton Synchrotron’. PhD thesis. Royal Holloway, U. of London, 2022. URL: <http://cds.cern.ch/record/2720090?ln=en>.
- [23] A. S. Oates. ‘7 - Reliability of silicon integrated circuits’. In: *Reliability characterisation of electrical and electronic systems*. Ed. by Jonathan Swingler. Woodhead Publishing series in electronic and optical materials. Cambridge, UK: Woodhead Publishing, 2015, pp. 115–141. ISBN: 978-1-78242-221-1. DOI: [10.1016/B978-1-78242-221-1.00007-1](https://doi.org/10.1016/B978-1-78242-221-1.00007-1). URL: <https://www.sciencedirect.com/science/article/pii/B9781782422211000071>.
- [24] *ProASIC 3 FPGAs / Microchip Technology*. 2022. URL: <https://www.microchip.com/en-us/products/fpgas-and-plds/fpgas/proasic-3-fpgas> (visited on 04/12/2022).
- [25] Paulo Moreira. ‘GBT Specifications’. In: CERN internal document (2022). URL: <https://espace.cern.ch/GBT-Project/GBTX/Manuals/GBTXManual.pdf> (visited on 24/11/2022).
- [26] T Hoffmann. *FESA—The front-End Software architecture at FAIR*. 2008. URL: https://www.researchgate.net/publication/228831574_FESA-The_front-End_Software_architecture_at_FAIR.
- [27] Carles Kishimoto (CERN IT/CS), Tony Cass. *CERN Technical network upgrade*. 2019. URL: <https://cds.cern.ch/record/2665715/files/TN%20network%20upgrade.pdf> (visited on 07/11/2022).
- [28] C. Ghabrous Larrea et al. ‘IPbus: a flexible Ethernet-based control system for xTCA hardware’. In: *Journal of Instrumentation* 10.02 (2015), pp. C02019–C02019. ISSN: 1748-0221. DOI: [10.1088/1748-0221/10/02/C02019](https://doi.org/10.1088/1748-0221/10/02/C02019). URL: <https://iopscience.iop.org/article/10.1088/1748-0221/10/02/C02019>.
- [29] *Overview - BGI Documentation*. CERN Internal Document, 2022. URL: <https://bgi.docs.cern.ch/software/overview/> (visited on 07/11/2022).
- [30] *Timepix3 / Knowledge Transfer*. CERN Knowledge Transfer internal document, 2022. URL: <https://kt.cern/technologies/timepix3> (visited on 02/11/2022).
- [31] T. Poikela et al. ‘Timepix3: a 65K channel hybrid pixel readout chip with simultaneous ToA/ToT and sparse readout’. In: *Journal of Instrumentation* 9.05 (2014), pp. C05013–C05013. ISSN: 1748-0221. DOI: [10.1088/1748-0221/9/05/C05013](https://doi.org/10.1088/1748-0221/9/05/C05013). URL: <https://iopscience.iop.org/article/10.1088/1748-0221/9/05/C05013>.
- [32] Karl W. Boer. *Semiconductor physics*. New York NY: Springer Berlin Heidelberg, 2018. ISBN: 9783319691480.

- [33] Leonardo Rossi. *Pixel detectors: From fundamentals to applications*. 1st ed. Particle acceleration and detection. Berlin and New York: Springer, 2006. ISBN: 9783540283331. URL: <https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=304428>.
- [34] Xavier Llopart and Tuomas Poikela. *Timepix3 Manual v2.0*. CERN Internal Document, 2016. (Visited on 02/11/2022).
- [35] Florian Michael Pitters et al. *Time and Energy Calibration of Timepix3 Assemblies with Thin Silicon Sensors*. 2018. URL: <https://cds.cern.ch/record/2649493?ln=en> (visited on 30/11/2018).
- [36] *ug1182-zcu102-eval-bd.pdf • Viewer • Documentation Portal*. 2022. URL: <https://docs.xilinx.com/v/u/en-US/ug1182-zcu102-eval-bd> (visited on 09/11/2022).
- [37] *PetaLinux - Xilinx Wiki - Confluence*. 2022. URL: <https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842250/PetaLinux> (visited on 15/11/2022).
- [38] *RPU CPU Configuration • Zynq UltraScale+ Device Technical Reference Manual (UG1085) • Reader • Documentation Portal*. 2022. URL: <https://docs.xilinx.com/r/en-US/ug1085-zynq-ultrascale-trm/RPU-CPU-Configuration> (visited on 09/11/2022).
- [39] *Memory Overview for APU and RPU Executables • Zynq UltraScale+ MPSoC Software Developer Guide (UG1137) • Reader • Documentation Portal*. 2022. URL: <https://docs.xilinx.com/r/2021.1-English/ug1137-zynq-ultrascale-mpsoc-swdev/Memory-Overview-for-APU-and-RPU-Executables> (visited on 09/11/2022).
- [40] *ug1182-zcu102-eval-bd.pdf • Viewer • Documentation Portal*. 2022. URL: <https://docs.xilinx.com/v/u/en-US/ug1182-zcu102-eval-bd> (visited on 24/11/2022).
- [41] Xilinx and Inc. ‘Zynq UltraScale+ MPSoC: Software Developers Guide’. In: (2022). URL: https://www.xilinx.com/content/dam/xilinx/support/documents/sw_manuals/xilinx2021_1/ug1137-zynq-ultrascale-mpsoc-swdev.pdf (visited on 24/11/2022).
- [42] FreeRTOS. *FreeRTOS - Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions*. 2022. URL: <https://www.freertos.org/> (visited on 09/11/2022).
- [43] FreeRTOS. *RTOS - Free professionally developed and robust real time operating system for small embedded systems development*. 2022. URL: <https://www.freertos.org/RTOS.html> (visited on 09/11/2022).
- [44] Xilinx. *Vitis Software Platform*. 2022. URL: <https://www.xilinx.com/products/design-tools/vitis/vitis-platform.html> (visited on 09/11/2022).
- [45] *ug1186-zynq-openamp-gsg-2021.2.pdf • Viewer • Documentation Portal*. 2022. URL: <https://docs.xilinx.com/v/u/en-US/ug1186-zynq-openamp-gsg> (visited on 09/11/2022).
- [46] *Qt / Cross-platform Software Design and Development Tools*. 2022. URL: <https://www.qt.io/> (visited on 08/12/2022).
- [47] *First steps - pybind11 documentation*. 2022. URL: <https://pybind11.readthedocs.io/en/stable/basics.html> (visited on 09/12/2022).
- [48] GitHub. *pybind/pybind11: Seamless operability between C++11 and Python*. 2022. URL: <https://github.com/pybind/pybind11> (visited on 10/11/2022).

- [49] 3. *Analog Front End — Timepix4 Manual 1.0 documentation*. CERN Internal Document, 2022. URL: <https://timepix4.web.cern.ch/timepix4/timepix4/ChipDescription/AnalogFrontEnd.html> (visited on 13/11/2022).
- [50] Xilinx and Inc. ‘UltraScale Architecture SelectIO Resources User Guide’. In: (2022). URL: <http://users.ece.utexas.edu/~mcdermot/arch/articles/Zynq/ug571-ultrascale-selectio.pdf> (visited on 14/11/2022).
- [51] *Riverbank Computing / Introduction*. 2022. URL: <https://riverbankcomputing.com/software/pyqt/> (visited on 07/12/2022).
- [52] GitHub. *libmetal/shmem_demo.c at main · OpenAMP/libmetal*. 2022. URL: https://github.com/OpenAMP/libmetal/blob/main/examples/system/linux/zynqmp/zynqmp_amp_demo/shmem_demo.c (visited on 19/12/2022).
- [53] GitHub. *nlohmann/json: JSON for Modern C++*. 2022. URL: <https://github.com/nlohmann/json> (visited on 14/11/2022).
- [54] Emanuel Eichhammer. *Qt Plotting Widget QCustomPlot - Introduction*. 2022. URL: <https://www.qcustomplot.com/> (visited on 12/11/2022).
- [55] *Matplotlib — Visualization with Python*. 2022. URL: <https://matplotlib.org/> (visited on 12/11/2022).
- [56] *PyQtGraph - Scientific Graphics and GUI Library for Python*. 2022. URL: <https://www.pyqtgraph.org/> (visited on 12/11/2022).
- [57] *GIL - pybind11 documentation*. 2022. URL: <https://pybind11.readthedocs.io/en/stable/advanced/misc.html> (visited on 08/12/2022).
- [58] *BIPXL Software Driver Specification / Document 2667909 (v.1)*. CERN Internal Document, 2022. URL: <https://edms.cern.ch/document/2667909/1> (visited on 29/11/2022).
- [59] *Arm Compiler for Embedded Reference Guide*. 2022. URL: <https://developer.arm.com/documentation/101754/0618/armclang-Reference/armclang-Command-line-Options/-fshort-enums---fno-short-enums> (visited on 14/11/2022).
- [60] *Develop • Vitis Unified Software Platform Documentation: Embedded Software Development (UG1400) • Reader • Documentation Portal*. 2022. URL: <https://docs.xilinx.com/r/en-US/ug1400-vitis-embedded/Develop> (visited on 29/11/2022).
- [61] Xilinx. *SmartLynq Data Cable*. 2022. URL: <https://www.xilinx.com/products/boards-and-kits/smartlynq-data-cable.html> (visited on 15/11/2022).
- [62] *time(1) - Linux manual page*. 2022. URL: <https://man7.org/linux/man-pages/man1/time.1.html> (visited on 21/11/2022).
- [63] P. Burian, P. Broulím and B. Bergmann. ‘Study of Power Consumption of Timepix3 Detector’. In: *Journal of Instrumentation* 14.01 (2019), pp. C01001–C01001. ISSN: 1748-0221. DOI: [10.1088/1748-0221/14/01/C01001](https://doi.org/10.1088/1748-0221/14/01/C01001). URL: <https://iopscience.iop.org/article/10.1088/1748-0221/14/01/C01001>.