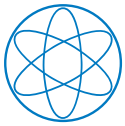
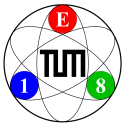


Institute for Hadronic Structure and Fundamental Symmetries
School of Natural Sciences
Technical University of Munich

Development of FPGA frontend electronics of the scintillating fiber hodoscope of AMBER at CERN

Tim Maehrholz

Bachelor's Thesis

Supervisor:
Prof. Dr.
Chair of

Second Examiner:
PD Dr.

January 2025

Abstract

The proton radius measurement (PRM) experiment at AMBER at CERN aims to measure the proton radius by scattering muons on protons. The scintillating fiber hodoscope (SFH) is a key component of the PRM experiment, providing essential time measurements of the incoming and scattered muons. In this thesis, the development of the FPGA-driven frontend electronics of the SFH is presented, focusing on the development of the FPGA firmware required for the control and readout of the Citiroc1A ASIC, a crucial part of the readout and trigger electronics. We concluded that the developed firmware is capable of controlling and reading out the signals from the Citiroc1A ASIC, but that the performance of the frontend electronics is limited by the noise of the input signal. We propose further reasearch to reduce this noise and improve the performance of the frontend electronics of the scintillating fiber hodoscope.
INSERT: this has to be improved

Contents

CHAPTER 1

Introduction

"What we observe is not nature itself, but nature exposed to our method of questioning."[Werner Heisenberg][9]
Progress in particle physics has always been driven by the desire to understand the fundamental building blocks of our universe.
Our current best theory for the innnerworkings of our world, the standart model of particle physics tells us, that the matter we see around us is mostly made up of down and up qurks and electrons. Combinations of these quarks, held together by the strong nuclear force form the proton and neutron, the nuclei of the atoms that make up the matter of the everyday world. Even though the proton was discovered over a century ago by Ernest Rutherford[13], it still holds several mysteries that continue to puzzle physicists. One of these mysteries is the size of the proton.
The semantic meaning of size in the realm of particle physics is not as straight forward as in the macroscopic world. An answer to the question, what is the size of the proton can be given by looking at the charge radius of the proton, first measured in 1956 by E. E. Chambers and R. Hofstadter.[4]
The radius of the proton has been measured several times more since then, with different methods, such as electron-proton scattering experiments and the lamb shift in muonic and ordinary hydrogen. The results of these measurements differ by five standard deviations, giving rise to the so called proton radius puzzle.[1]
The proton radius measurement (PRM) experiment at AMBER at CERN aims to resolve this puzzle by measuring the proton radius with a new method, the elastic scattering of muons on protons.

To achieve this, the PRM experiment will measure the cross section of this scattering process and from this extract the form factors of the proton, which in turn allows for the calculation of the proton radius.
The scintillating fiber hodoscope (SFH) is a key component of the PRM experiment, as it provides essential time measurments of the incoming and scattered mouns, needed for the measurment of the proton radius.[1]
This thesis will focus on the development of the FPGA driven frontend electronics of the scintillating fiber hodoscope (SFH), developed at the Technical University of Munich by the Physics Department E18 for the proton radius measurment at AMBER at CERN, especially on the development of the FPGA firmware required for the control and readout of the Citiroc1A ASIC, a crucial part of the readout and trigger electronics. (INESERT:developed used to often)
The framework of this thesis is structured in order to provide the neccessary background information for the development of the frontend electronics of the SFH.(INSERT: THIS HAS TO BE REWORKED)
The next chapter will give an general overview of the PRM experiment and the theoretical background of the proton radius measurment.
In chapter 3 the frontend electronics of the SFH, with a focus on the functionality and behaviour of Citiroc1A ASIC will be explained.
Chapter 4 will explain the developed FPGA firmware for the control and readout of the Citiroc1A ASIC, as well as give a short overview of the setup that was used for testing the developed firmware.
Chapter 5 will present the results of the testing of the developed firmware and the performance of the frontend electronics of the SFH.
In chapter 6 these results will be discussed and evaluated in the context of the PRM experiment.
The final chapter of this thesis will give a short summary of the results and an outlook on the future development of the frontend electronics of the SFH, neccessary for it to successfuly contribute to the resuliton of the proton radius puzzle.

CHAPTER 2

Theoretical concepts and overview of AMBER

2.1. Measurment of the charge radius of the proton (PRM)

The proton is a baryon, a composite particle made up of one down quark and two up quarks. From this follows that the proton is not a point particle, but has an internal sturucture.[15]
The internal structure can be discribed by the structure functions of the proton, the electric and magnetic form factors G_E and G_M.[1]

2.1.1. Previous measurements of the proton radius

The charge radius of the proton has been massured several times before with different methods. The two premier methods are electron proton scattering experiments and the Lamb shift in muonic and ordinary hydrogen. The results of these measurements differ by five standard deviations as shown in Figure 2.1, this has given rise to the so called proton radius puzzle.[1]

2.1.2. Elastic scattering of muons on protons

The AMBER PRM experiment at CERN aims to reslove the proton radius puzzle, by measuring the elastic scattering of muons on protons. The first order cross section, taking

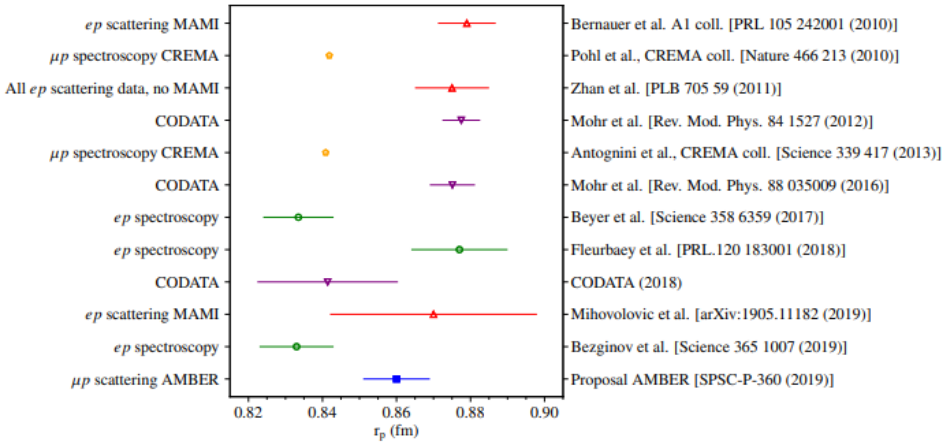

Figure 2.1.: Previous measurements of the proton radius from electron proton scattering experiments and the Lamb shift in muonic and ordinary hydrogen, the measurements differ from each other by five standard deviations.[1]

into account only interactions where one virtual photon was exchanged, for the elastic scattering of muons on a proton target is

$$\frac{d\sigma}{dQ^2} = \frac{\pi\alpha^2}{Q^4 m_p^2 p_\mu^2}\left[\left(G_E^2 + \tau G_M^2\right)\frac{4E_\mu^2 m_p^2 - Q^2(s-m_\mu^2)}{1+\tau} - G_M^2\frac{2m_\mu^2 Q^2 - Q^4}{2}\right] \tag{2.1}$$

with $Q^2 = -q^2$ the squared transferred four-momentum, $\tau = Q^2/4m_p^2$, $s = (p_\mu + p_p)^2$, G_E the electric form factor of the proton, G_M the magnetic form factor of the proton and α the fine structure constant.[2]
Through determining the form factor G_E for small Q^2 , the charge radius of the Proton can be claculated with the following equation[2]

$$r_p^2 = -6\frac{dG_E}{dQ^2}\bigg|_{Q^2=0} \tag{2.2}$$

2.2. General setup for PRM at AMBER.

2.2.1. Detectors for PRM

To determine the magentic G_M and electric form G_E factors of the proton and thus the charge radius of the Proton, the experimental cross section of the elastic scattering of muons on protons has to be measured.
The general setup of the PRM experiment, with focus on the new detectors needed for the proton radius measurment, is shown in Figure 2.2.

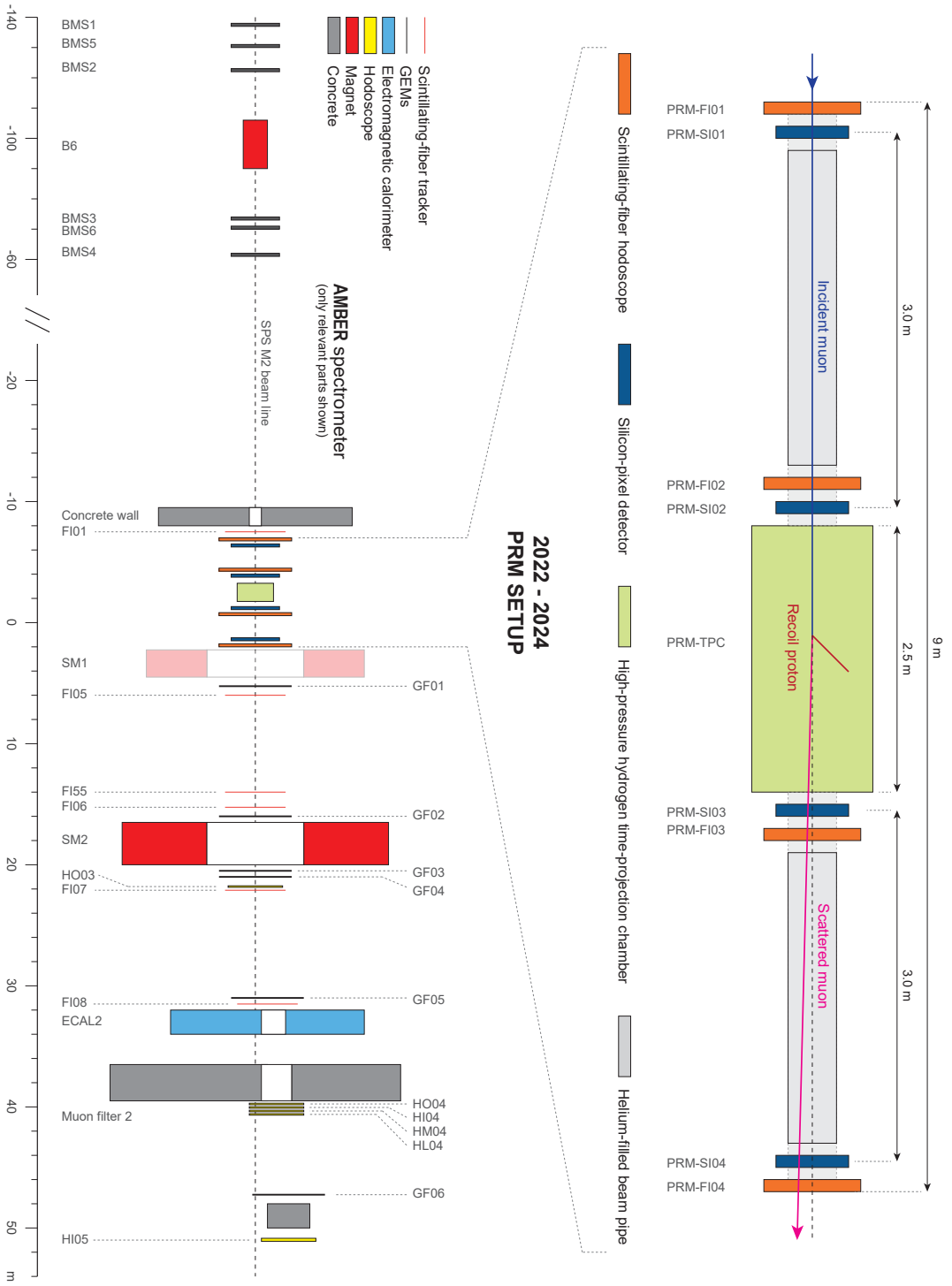

Figure 2.2.: General setup of the Amber experiment with new detectors for PRM.[7]

The incoming muon beam with an energy of 100 GeV[1] and an beam rate of 2×10^6[3] particles per second is scattered on a pressurized hydrogen gas target, located in the Time Projection Chamber (TPC), which also acts as the detector for the recoil path of

the scattered proton.
The reconstruction of the path of the muon is achieved through the usage of two detector types, combined into one unified tracking station (UTS) as shown in Figure 2.3.
Each UTS consists of three layers of pixilized silicon detectors (ALPIDEs), for precises positional measurments (spacial resolution of about 8 µm[8]) of the incoming and scattered muons, but lacking the time resolution(5 µs[8]) required for the PRM experiment. For this reason each UTS includes a scintillating fiber hodoscope (SFH), the detector of intrest for this thesis, which provides the time precision(300 ps[8]) for the measurment.
Four of these unified tracking stations, two before and two after the active target, are placed in the beamline as shown in 2.2. The measurment of the momentum of the scattered moun is done by existing COMPASS detectors located after the, for the PRM newly included, detectors[1].

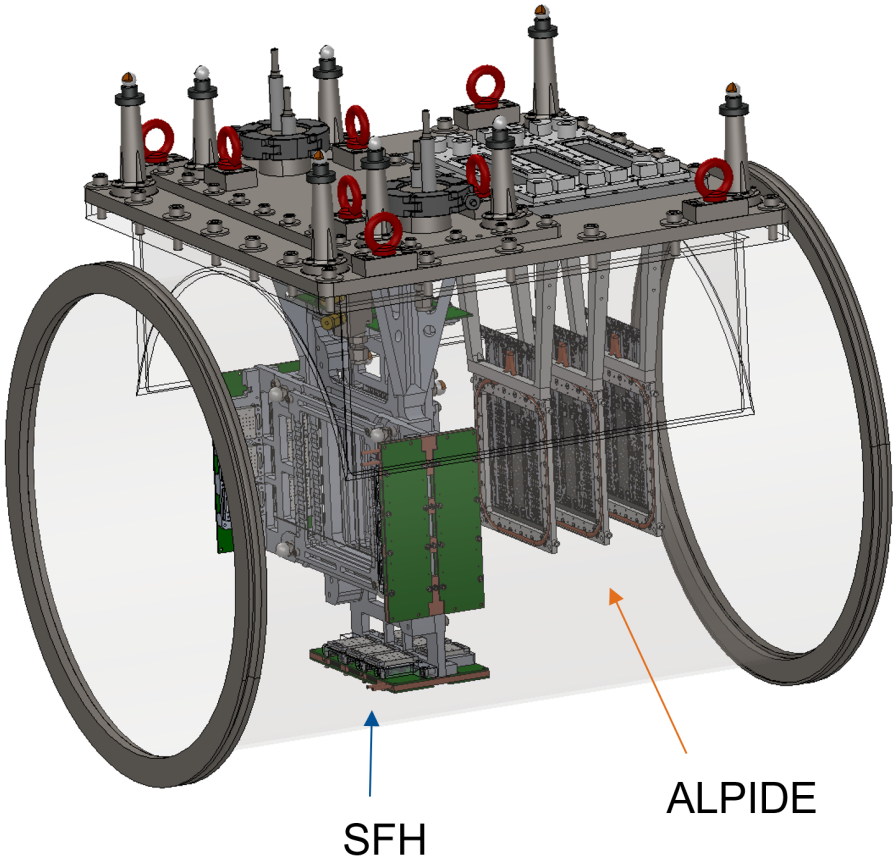

Figure 2.3.: Unified tracking station (UTS) with three layers of pixilized silicon detectors (ALPIDEs) and the scintillating fiber hodoscope (SFH).[7]

2.2.2. Scintillating fiber hodoscope(SFH)

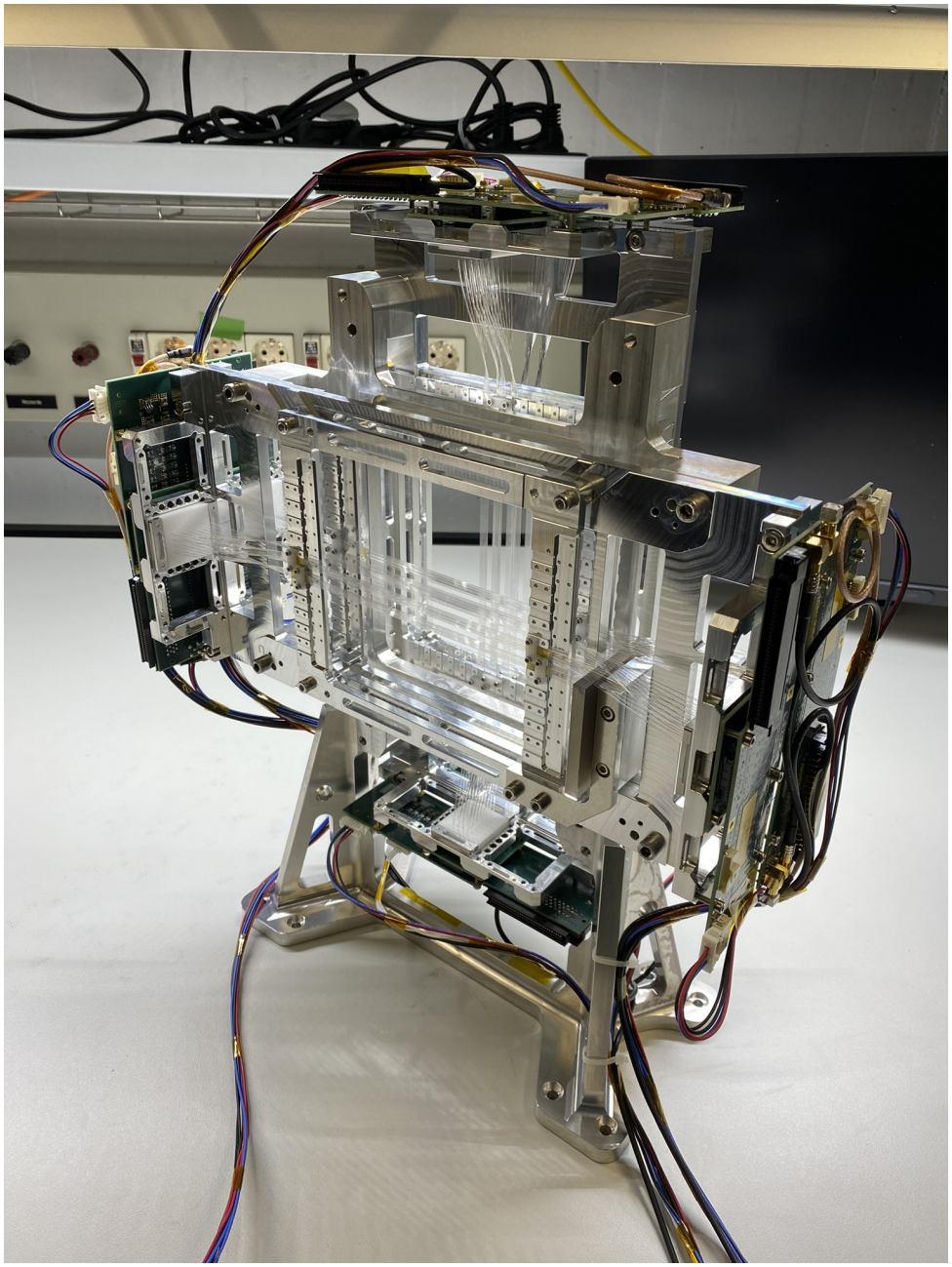

Figure 2.4.: Scintillating fiber hodoscope (SFH) with some of the scintillating fibers of the four layres installed.The frontend electronics are not attached.[7]

The scintillating fiber hodoscope shown in Figure 2.4, the detector for which the FPGA driven frontend electronics are developed in this thesis, is used to measure the precise timing(300 ps[8]) of the incoming and scattered muons. Every SFH contains four layers of scintillating fibers, two in x and two in y direction. Each layer is made up of 192[8], 500 μm thick[5] fibers, in total 768[8] fibers per SFH. When charged particels, muons in

this case, pass through a scintillating fiber they excite the scintillating material, which then emits photons. Both ends of every fiber are conected to a silicon photomultiplier (SiPM) which converts the photons into an electrical signal, that is then proccesed by the frontend electronics. To minimize cost and the amount of data that has to be processed, a mirrored readout is planned, where only one end is conected to the SiPM array and the other end is mirrored.[7]

2.3. Field Programmable Gate Arrays (FPGA)

Field Programmable Gate Arrays (FPGAs) are integrated circuits that can be programmed after manufacturing. Their flexibility, reconfigurability, and robust parallel processing capabilities make them ideal candidates for digital signal processing(DSP). The frontend electronics for the SFH reqiure large amounts of data to be processed in real time, wich makes FPGAs with their large data throughput and parallel processing capabilities ideal for this task.[6]
The FPGA used in the frontend electronics for the scintillating fiber hodoscope is part of the Xilinx Artix-7 family.[11] INSERT: maybe more information about the FPGA used in the frontend electronics(speed grade, number of LUT,number of BRAMs, number of IOs, etc.)

CHAPTER 3

Frontend electronic of the scintillating fiber hodoscope

3.1. Overview of the frontend electronics

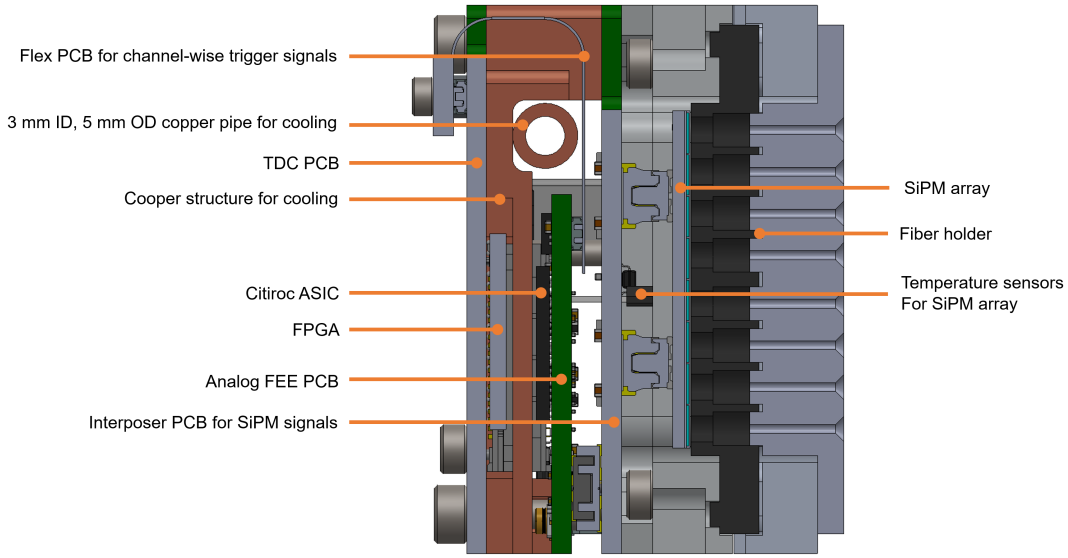

Figure 3.1.: Sideview of the frontend electronics that will be attached on the sides of the SFH, the fiber holders will be attached to the fibers. The SiPM arrays transform the incoming photons into electric signals, that are then transferred to the frontend electronics by the PCB interposer.[7] INSERT: old picture needs to be updated

3.1.1. Proccesing of the SFH signal

The frontend electronics of the scintillating fiber hodoscope process the signals from the scintillating fibers. They can be attached on all four sides of the SFH, as can be seen in Figure 2.4. The fibers are conected to the fiber holders on both ends as shown in Figure 3.1. There are in total 768[8] fibers per SFH. Since both ends produce an electric signal, a total of 1546 signals have to be proccesed. This would require a total of 48 Citiroc1A ASICs, spread over 8 frontend electronics units. To mittigate the cost of the frontend electronics, as well as limit the amount of data that has to be proccesed, a mirrored setup is planed. In this setup only one end of the fibers is conected to the SiPM array, while the other end is mirrored. This would reduce the amount of reqiured frontend electronics units to 4.[7]
The incoming photons are transformed into electric signals by the SiPM arrays. The SiPM signals are then transmitted to the analog frontend electronics (FEE) PCB by the interposer PCB also shown in Figure 3.1.[7]

3.1.2. The analog frontend electronics (FEE) PCB

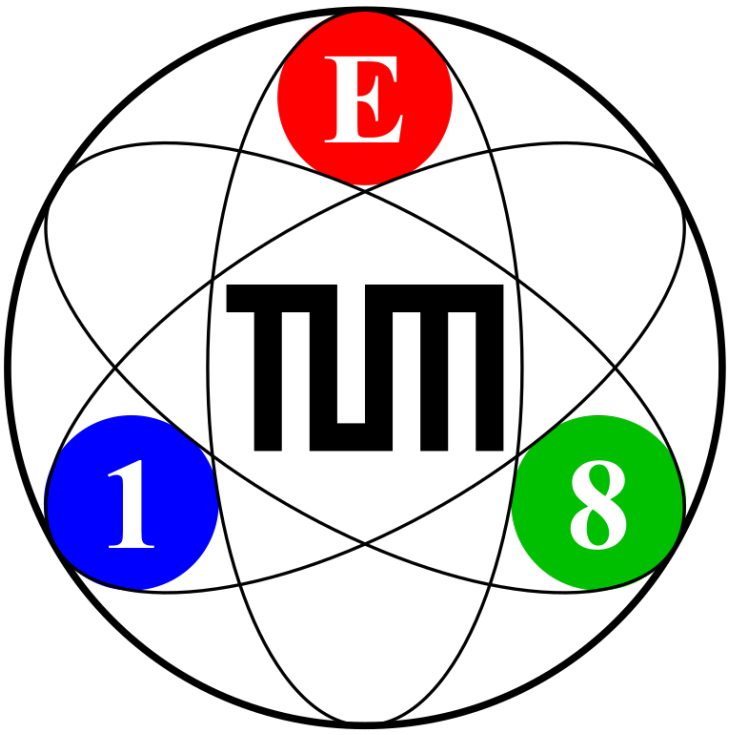

Figure 3.2.: The analog frontend electronics (FEE) PCB with the six Citiroc1A ASICs, on the left side the power supply is connected.The output of the Citiroc1A is transmitted to the iFTDC over three flex PCBs.[7]

The analog frontend electronics (FEE) PCB, shown in Figure 3.2, together with the iFTDC forms the heart of the frontend electronics. The FEE PCB incorporates six Citiroc1A ASICs, which are designed to amplify and process the signals from the SiPM arrays. Each Citiroc1A ASIC handles 32 signals. The output of the Citiroc1A is then transmitted to the iFTDC over three flex PCBs. The power supply is connected to the FEE PCB on the left side as shown in 3.2. Two Citirroc1A ASICs are each controlled by one Artix-7 FPGA located on the iFTDC.[11]

3.1.3. The iFTDC

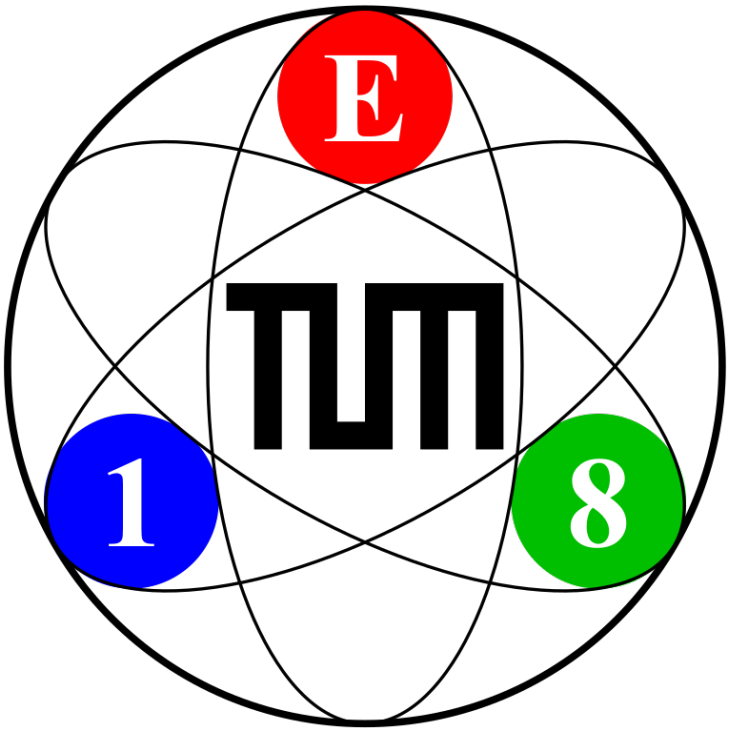

Figure 3.3.: The iFTDC with three Artix-7 FPGA, the three flex PCBs that connect the iFTDC with the FEE PCB and the power supply.[11]

The iFTDC, depicted in Figure 3.3 is a FPGA based time-to-digital converter. It consists of three Artix-7 FPGA, who each control two Citiroc1A ASICs. The FPGA handels the readout as well as the configuration of the Citiroc1A ASICs[11].
INSERT: here stil hast to be includes how ethernet works how ipbus works and how jtag is implemented ans stuff analong this line

3.2. The Citiroc1A ASIC

The Citiroc1A ASIC is a frontend application-specific integrated circuit developed by Weeroc for the readout of SiPM detectors. It allows for the readout of 32 channels and is sensitive to $\frac{1}{3}$ of a photoelectron.[14]
The Citiroc1A ASIC is controlled and readout by the Artix-7 FPGA on the iFTDC, each FPGA controlling two Citiroc1A ASICs.[11] The focus of this thesis is the development of the FPGA firmware for the control of the Citiroc1A ASICs, but a provesional readout firmware for testing the configuration of the Citiroc1A will also be developed.

3.2.1. Signal proccesing of the Citiroc1A

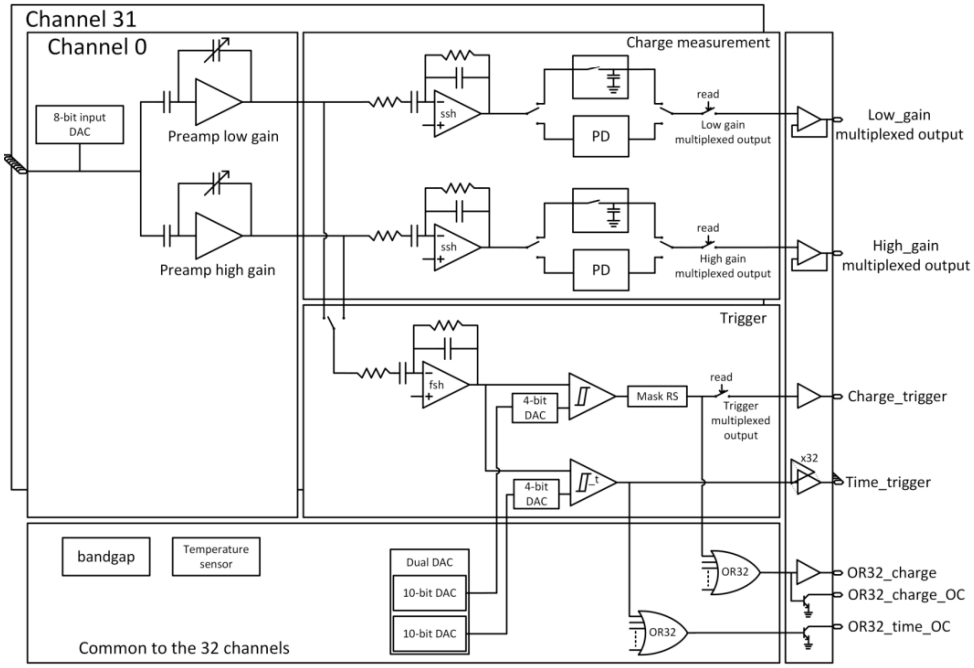

Figure 3.4.: General ASIC block scheme of the Citiroc1A.[14]

The general block scheme of the Citiroc1A is shown in Figure 3.4.
The Citiroc1A allows for the fine tuning of the SiPM bias voltage for each channel via the 8-bit input DAC.
The input signals are amplified with a variable high or low gain, configurable for every channel as depicted in Figure 3.5. The PRM experiment requires the maximal high gain of 62.[11]
The amplified signals are then shaped by either the slow (ssh) or fast shaper (fsh) as shown in Figure 3.4. The fast shaper is used for the PRM experiment, since it has a 15 ns peaking time and a better time resuliton, which is needed for the time precision of the

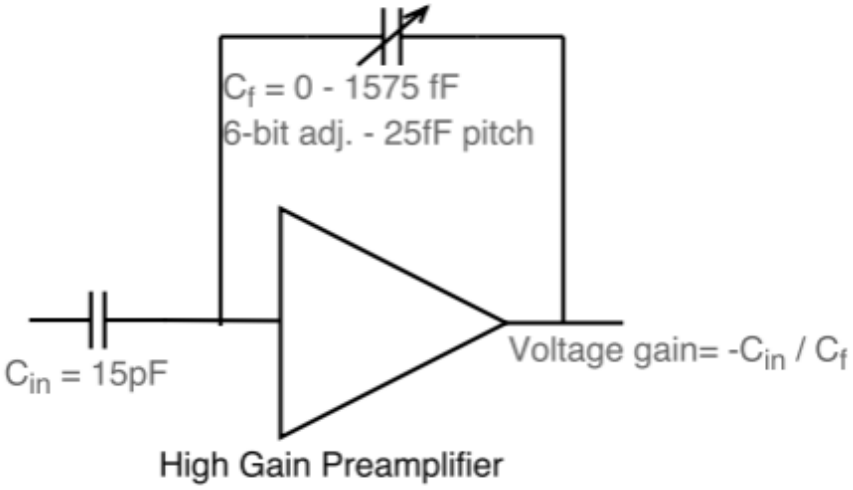

Figure 3.5.: High gain amplification of the Citiroc1A. The gain is adjustable from 0 to 1575 fF in 25 fF steps.[14]

SFH.[14]
The ASIC has two discriminators, the charge discriminator and the time discriminator. In this thesis we will only look at the time discriminator, since it provides the time information. The time discriminator threshold is adjustable via a 10 bit dac for all channels and an additional 4 bit dac for every individual channel as shown in Figure 3.4[14].

3.3. Configuration of the Citiroc1A

The configuration of the Citiroc1A is achievied by the FPGA via the five signals shown in Figure 3.6. The Select signal allowes the choice between configuring the slow control, for Select = 1 or the probe register, for Select = 0.[14]

3.3.1. The slow control register

The slow control register is used to set values for internal variables like the high gain for a channel or the time discriminator threshold. It also allowes for the FPGA to turn of spesific stages of the Citiroc1A, such as the slow shaper or the time discriminator. The register is 1144 bits long. A full list of all the register that can be set is shown in Table A.1 in Appendix A.
The process of writing the bitstream into the slow control register by the FPGA is illustrated in Figure 3.7.
The Rstb_sr signal is an asynchronous active-high reset for the serial register, applying to

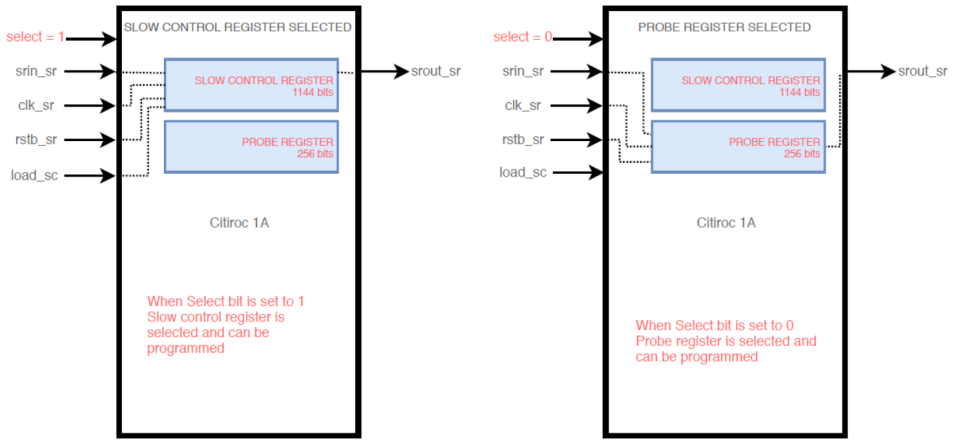

Figure 3.6.: The two configurable registers of the Citiroc1A are selected using the Select signal. The FPGA communicates with the Citiroc1A through the signals Clk_sr, Rstb_sr, Srin_sr, and Load_sr, while the Srout signal is sent back from the Citiroc1A to the FPGA for verification.[14]

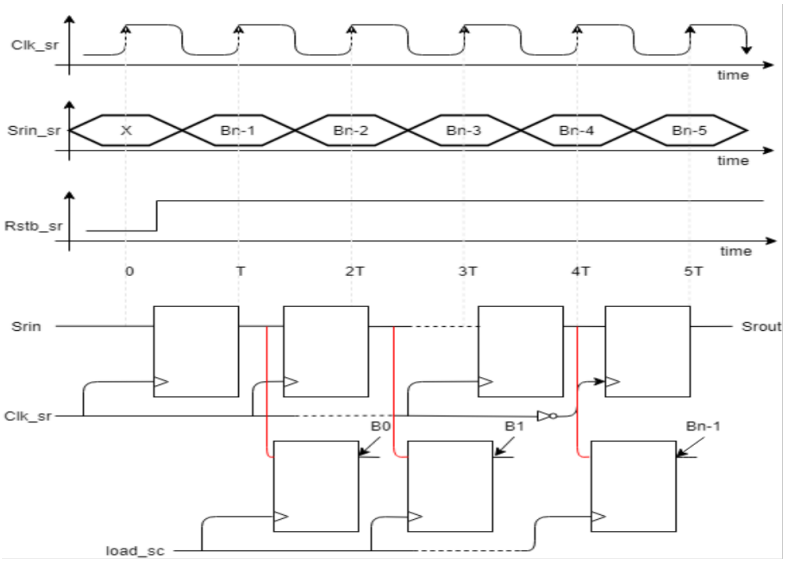

Figure 3.7.: The slow control chronogram, depicting the bitstream writing process controled by Clk_sr the clock signal and Srin_sr the data signal. A rising edge of Load_sr is required, after successful verification with the Srout signal to load the slow control register.[14]

both the slow control and probe register.

The FPGA processes the bitstream sequentially, starting with the least significant bit (LSB). The first bit of the bitstream to enter the serial register will be the last bit of the slow control register. Each bit is sent on the Srin_sr signal in coordination with a rising edge of the Clk_sr clock signal.

The Load_sr signal is used to load the bitstream into the slow control register. After all

bits have been sent to the Citiroc1A, a rising edge on Load_sr is required to load the slow control register.
The Srout signal is sent back from the Citiroc1A to the FPGA for bitstream verification. Only after the FPGA has sent the full bitstream twice, does the Srout signal take on the value of the bitstream, since the Srout signal is shifted by the length of the bitstream.[14] One should only set the rising edge of the Load_sr signal after verifying that the Srout signal takes on the correct values.

3.3.2. The probe register

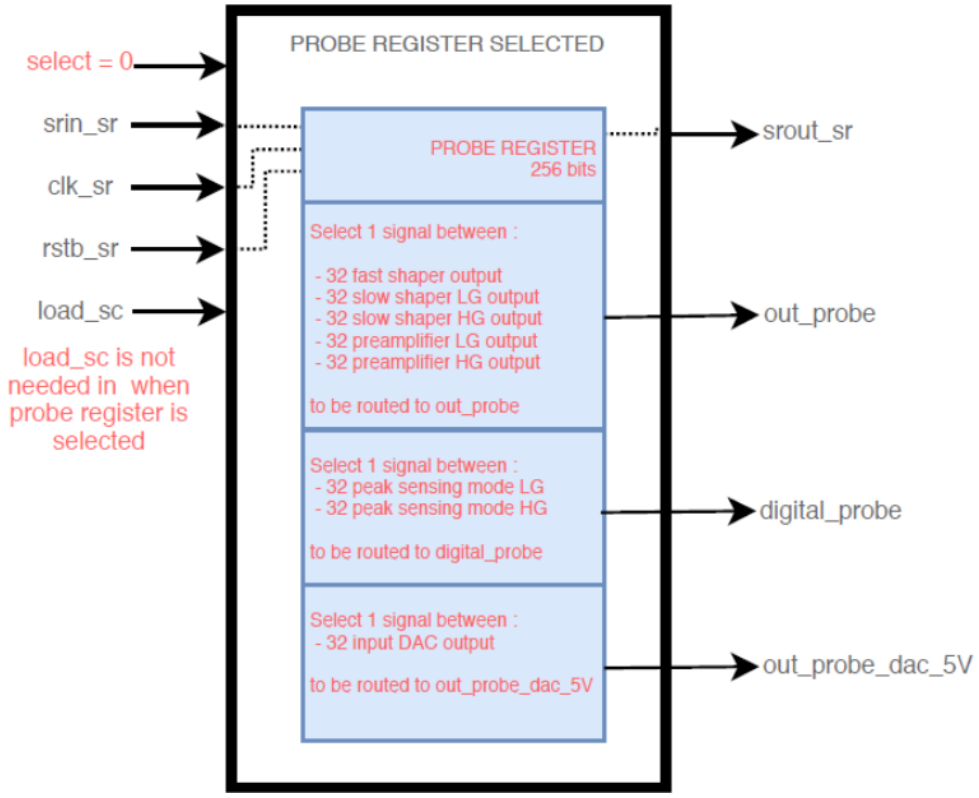

Figure 3.8.: Scheme block of internal probing system, allowing the routing of internal signals to probe pins for debugging purposes. It is configured via the probe register.[14]

The probe register is used for routing internal signals to several output pins for debugging purposes. It's functionality is ilustrated in Figure 3.8. The register consists of 256 bits and is written the same way as the slow control register, with the difference that the bits are directly written into the Citiroc1A without requiring a rising edge on Load_sc.[14]
The full list of all the register that can be set in the probe register is shown in Table A.2 in Appendix A.

The internal signals for each channel that can be routed to the output pins are shown in Table 3.1.

Signal Source	**Description**	**Output Pin**
High and low gain preamplifier, slow and fast shapers	Outputs of preamplifiers and shapers	`out_probe`
`PeakSensing_modeb_LG`	Internal peak-sensing signal for low gain	`digital_probe`
`PeakSensing_modeb_HG`	Internal peak-sensing signal for high gain	-
Output of input DAC	DAC output voltage (5 V)	`out_probe_dac_5_V`

Table 3.1.: Internal signal routing to output pins for each channel.

Only one signal can be routed to each output pin at a time, without potentially causing a short circuit.[14]

CHAPTER 4

Development of the FPGA firmware for the SFH

Insert: Explain diifrence between firmware for the three FPGAs

4.1. Overview of the firmware

The firmware developed in this thesis for the three Artix-7 FPGAs, located on the iFTDC as described in Section 3.1.3, must perform several functionalities. The main tasks of the firmware are to configure the Citiroc1A ASIC, explained in Section 3.3, communicaton with the controlling coputer via ethernet and the IPBUS protocol and a provisional read-out of the time triggerd data from the Citiroc1A ASIC.
The fimrmware is written in VHSIC Hardware Description Language (VHDL) and is synthesized and implemented using Xilinx Vivado. Several IP-cores, provided by Xilinx, are used to simplfy the development of the firmware for several tasks like the implementation of the configuration memory.

4.1.1. The IPBUS protocol

The IPBUS protocol used for the communication between the FPGA and the controlling computer is a simple protocol for controlling IP-aware hardware devices with a 32 bit read and write bus using UDP as the transport protocol.[12]
The IPBUS protocol defines a read and write command enabling successful write and read operations of a 32 bit register, with a 32 bit address in the FPGA.
The commands can be issued on the controling computer with the μHAL library, which allowes the user to issue read and write commands using a python script and an XML file

defining the addresses of the registers.[12]
The address space inside the FPGA is defined in the firmware. The address space is divided into seprate address spaces for each of the slaves by the leading bits of the address. For some slaves, the address space is further divided into subspaces for the different registers of the slave.

4.1.2. The firmware structure

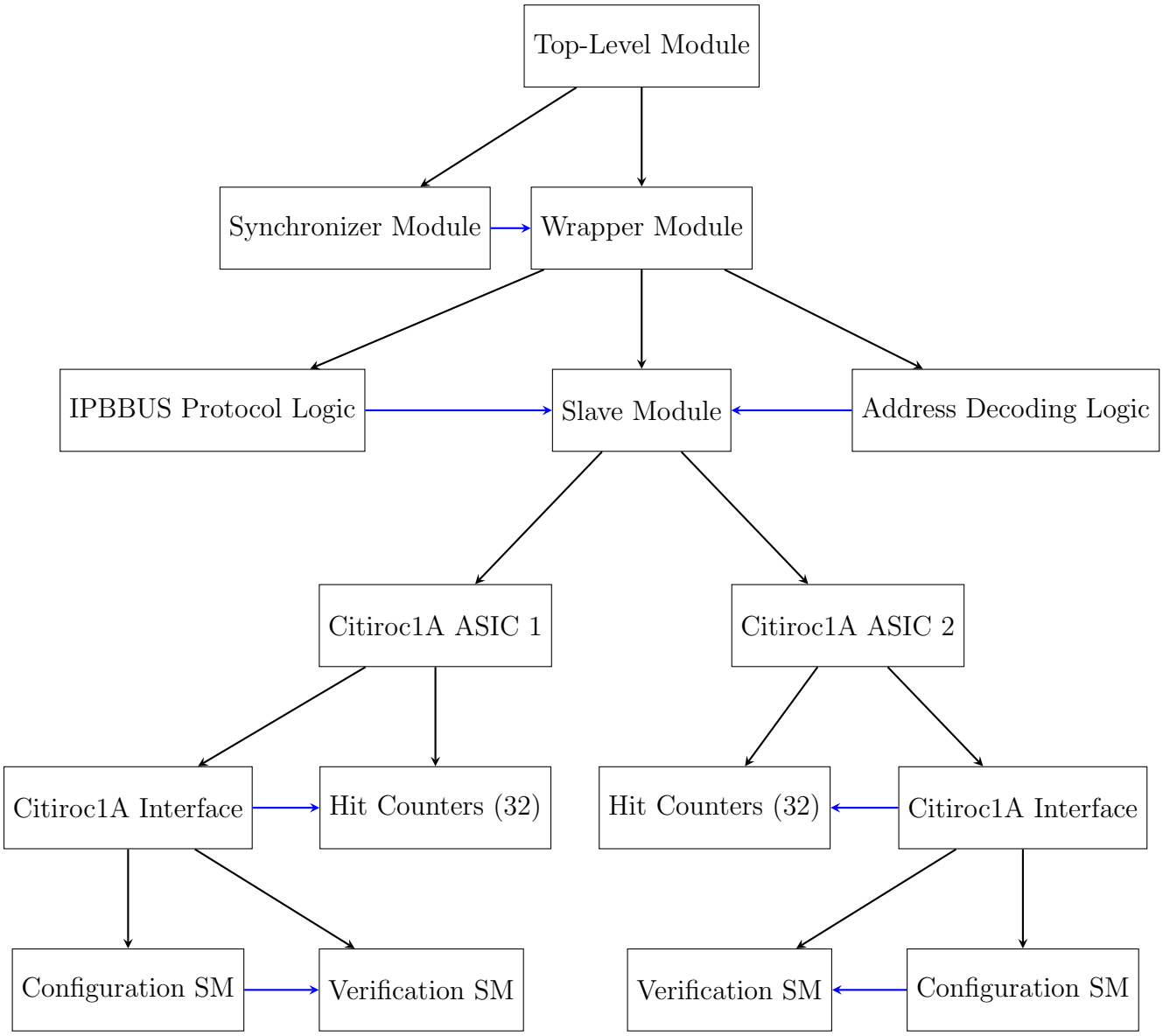

Figure 4.1.: Structure of the firmware developed for this thesis. The black arrows represent the true hierarchical conections between the moduls, while the blue arrows indicate connections that are semantically correct but are routed through higher-level modules.

The firmware has a hierarchical structure ilustrated in Figure 4.1.
The top level modul of the firmware contains the outgoing and incoming signals of the FPGA.
The top level modul instantiates a wrapper modul for the IPBUS protocol and the slave entities in addition to a synchronizer modul for the incoming time triggerd Citiroc1A signals.
The wrapper modul contains the IPBUS protocol logic and the address decoding logic as well as a modul containing the slaves.
The slave modul contains several slaves for the different tasks of the firmware. For each of the Citiroc1A ASICs, a Citiroc1A interface is instantiated for the configuration and verification of the ASIC, along with 32 hitcounters for the 32 channels of the Citiroc1A ASIC.

4.2. Configuration of the Citiroc1A ASIC

The configuration of the slow control and probe registers of the Citiroc1A ASIC, along with the verification of this configuration, is handled by two finite state machines.
Each state machine each controls a random access memory (RAM) with a depth of 64 addresses, with each address storing 32 bits of data.
The state machiens in turn are controled by the status and control register of the corresponding Citiroc1A Interface, which can be written by the controlling computer via the IPBUS protocol.

4.2.1. Status and control register

Each Citiroc1A interface has a 32 bit status and control register, but only the first 7 bits are used. It is responsible for the control of the configuration and verification state machines as well as the hitcounters. The bits of the status and control register, with the exeption of bit 1 which is set to 0 by the FPGA after the configuration is loaded into the serial register, do not reset themselves after being set to 1 and have to be deaserted by the controlling computer.
It's structure is shown in Table 4.2.

Bit	Description
bit 0	Selects between slow control (1) and probe register (0)
bit 1	Loads the the configuration into the serial register
bit 2	Resets the Configuration and Verification SM and the Configuration RAM
bit 3	Resets the Citiroc1A serial register
bit 4	Loads the configuration into the slow control register
bit 5	Resets the hitcounters associated with the Citiroc1A
bit 6	Enables the hitcounters associated with the Citiroc1A

Figure 4.2.: Structure of the status and control register of the Citiroc1A interface. The specified commands are executed when the bits are set to 1.

4.2.2. Configuration state machine

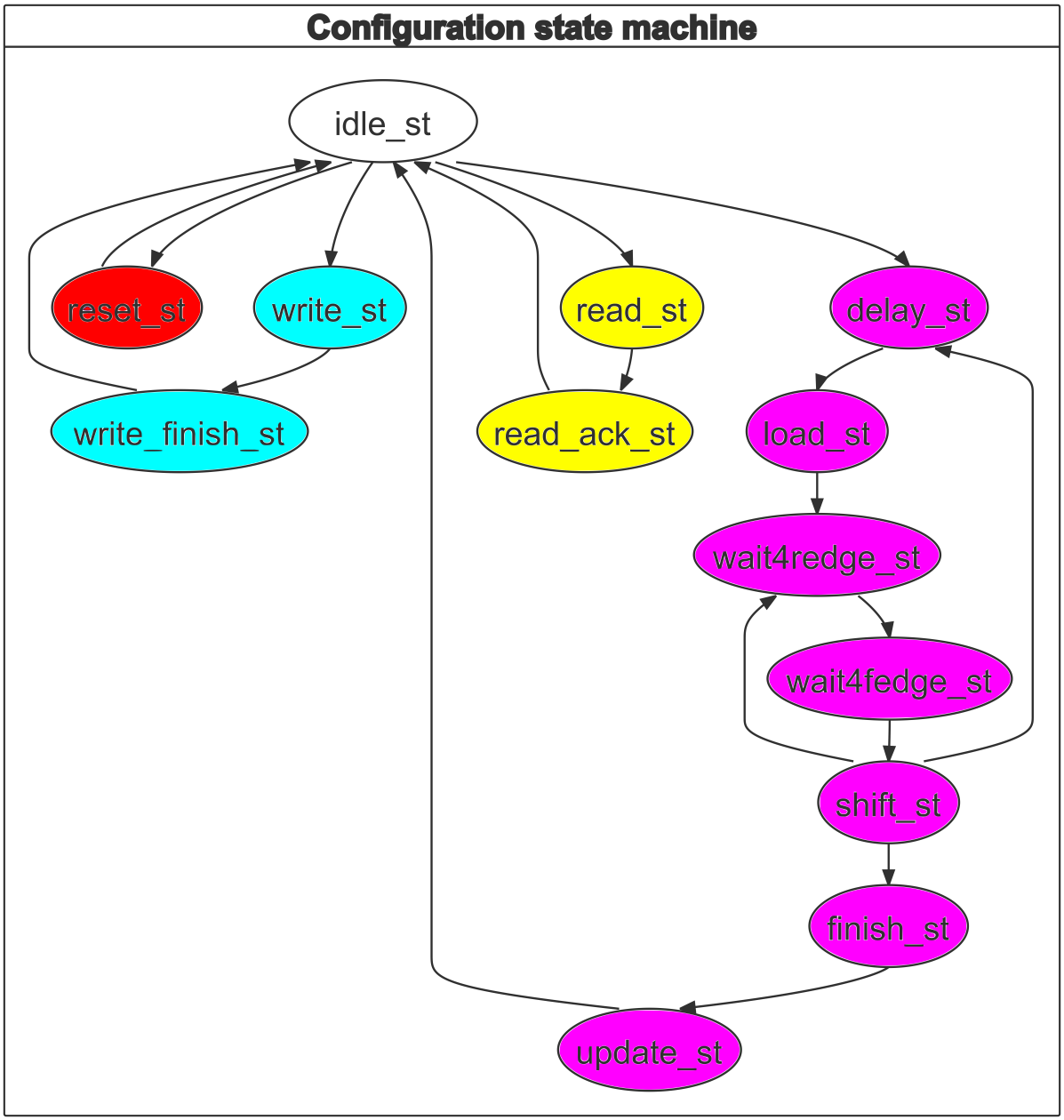

Figure 4.3.: Finite state machine for configuring the Citiroc1A ASIC. States associated with the same processes are highlighted using the same color.

The Configuration state machine is responsible for configuring the slow control and probe registers of the Citiroc1A ASIC. The state machine is controlled by the status and control register, which is written by the controlling computer via the IPBUS protocol. It has four processes, whose states are shown in different colors in Figure 4.3.
The write process is responsible for writing the configuration data to the configuration RAM. 32 bits can be written to a specified address in the configuration ram at a time by

the controling computer.
The read process allows the controlling computer to read the content of the configuration RAM at the specified address.
The reset process resets the configuration RAM and the Configuration state machine and is inniated by setting bit 2 of the status and control register to 1.
The load process loads the configuration data from the configuration RAM into the serial register of the Citiroc1A ASIC. The first 1144/256 bits of the configuration RAM are shifted into the serial register of the Citiroc1A ASIC, depending on the selected register. Each bit is shifted in by setting the data line Srin_sr and clock clk as disccribed in Section 3.3. Since as discribed in Section 3.3 the first bit written to the serial register is the last bit of the configuration, the controling computer has to write the bitstream in reverse order to the configuration RAM.
The clk clock signal for thr ASIC's serial register is generated by the FPGA with a frequency of 1 MHz. The load process modulates this clock signal to control the operation of the serial register.
The process is repeated until all the data is shifted in and the configuration is loaded. It can be inniated by the controling computer by setting bit 1 of the status and control register to 1 and is only interupteable by the reset process.
In order to load the configuration into the slow control register from the serial register, bit 4 of the status and control register must be set to 1, this is not neccessary for the probe register.

4.2.3. Verification state machine

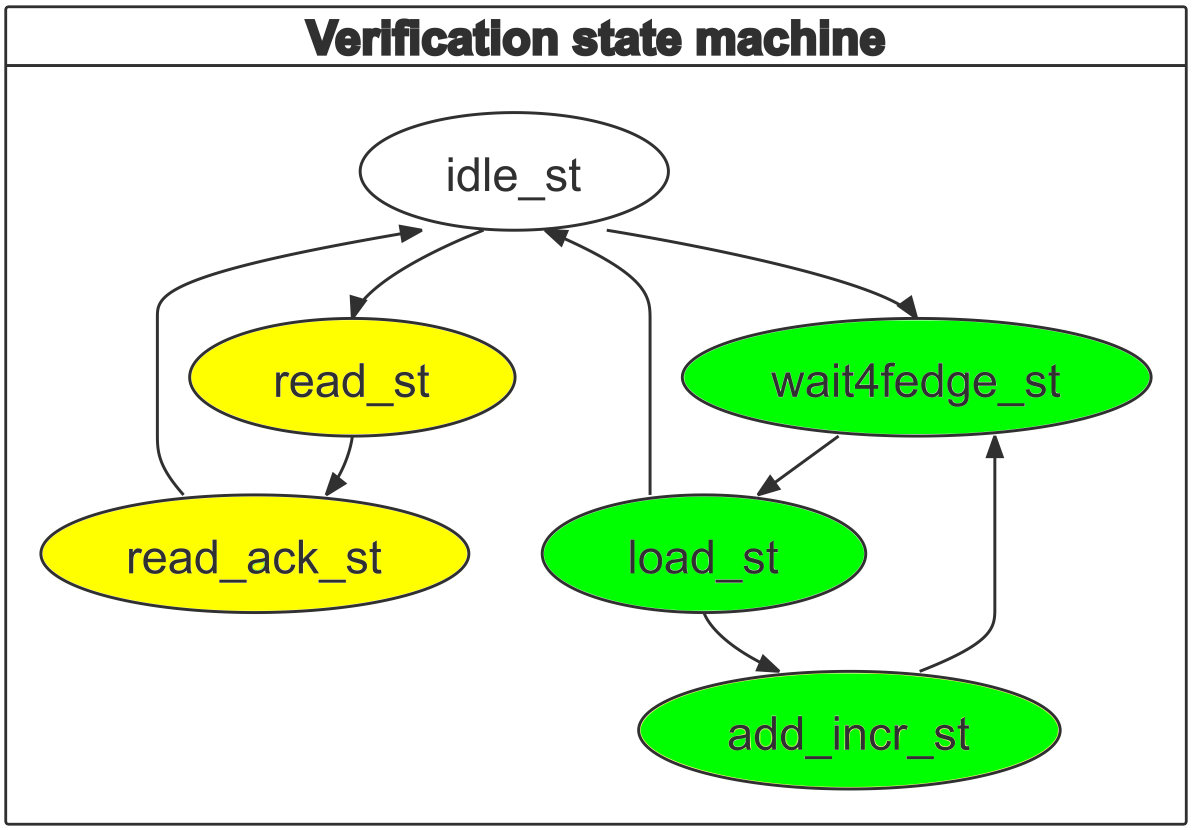

Figure 4.4.: Finite state machine for verifying the configuration of the Citiroc1A ASIC. States belonging to the same processes are represented with the same color.

The Verification state machine, ilustrated in Figure 4.4 is responsible for verifying the configuration of the Citiroc1A ASIC.
It can be reset by the controling computer by setting bit 2 of the status and control register to 1.
If bits are shifted into the serial register of the Citiroc1A ASIC, the verification process is automatically started. The srout signal coming back from the ASIC, explained in Section 3.3, is used to read back the shifted bits from the serial register of the Citiroc1A ASIC. The bits arrive at the FPGA in the same order as they were loaded in, but shifted by the length of the bitstream(1144/256). To compare the read back bits with the the loaded bits the bitstream has to be loaded twice.
The readback bits are stored in the verification RAM by the verification process and can be read by the controling computer via the read process. This allowes the controling computer to compare the loaded bits with the read back bits and ensure that the configuration was loaded correctly into the serial register of the Citiroc1A ASIC.

4.3. The provesional readout

In addition to the configuration and verification of the configuration of Citiroc1A ASIC, the firmware also provides a provisional readout in form of hitcounters and timecounters. The hitcounters are used to count the number of hits on each channel of the Citiroc1A ASIC.
They are limited to a minimal time between hits of two periods of the synchronization clock. In this case a 325 MHz clock is used, which corresponds to a minimal time between hits of 6.16 ns.
Each Citiroc1A has 32 hitcounters, one for each channel. The 32 bit hitcounters are enabled by setting bit 6 of the status and control register of the corresponding Citiroc1A Interface to 1. This also enables the timecounter, which counts the number of clock cycles since it was last reset.
The hitcounters and timecounters are reset by setting bit 5 of the status and control register of the corresponding Citiroc1A Interface to 1.
To read the hit and time counters, the controling computer first has to disable the hitcounters by setting bit 6 of the status and control register to 0, in order to avoid metastability issues arising from the clock domain crossing. The hitcounters can then be read by the controling computer via the IPBUS protocol.
The combination of the hitcounters and the timecounter allows the controling computer to determine the number of hits per second on each channel of the Citiroc1A ASIC.

4.4. Testing the firmware

4.4.1. Setup for the threshold scan

The firmware will be evaluated through a threshold scan of the frontend electronics of the scintillating fiber hodoscope. This threshold scan will be performed with the SiPMs disconnected from the frontend electronics. The experimental setup is illustrated in Figure 4.5.

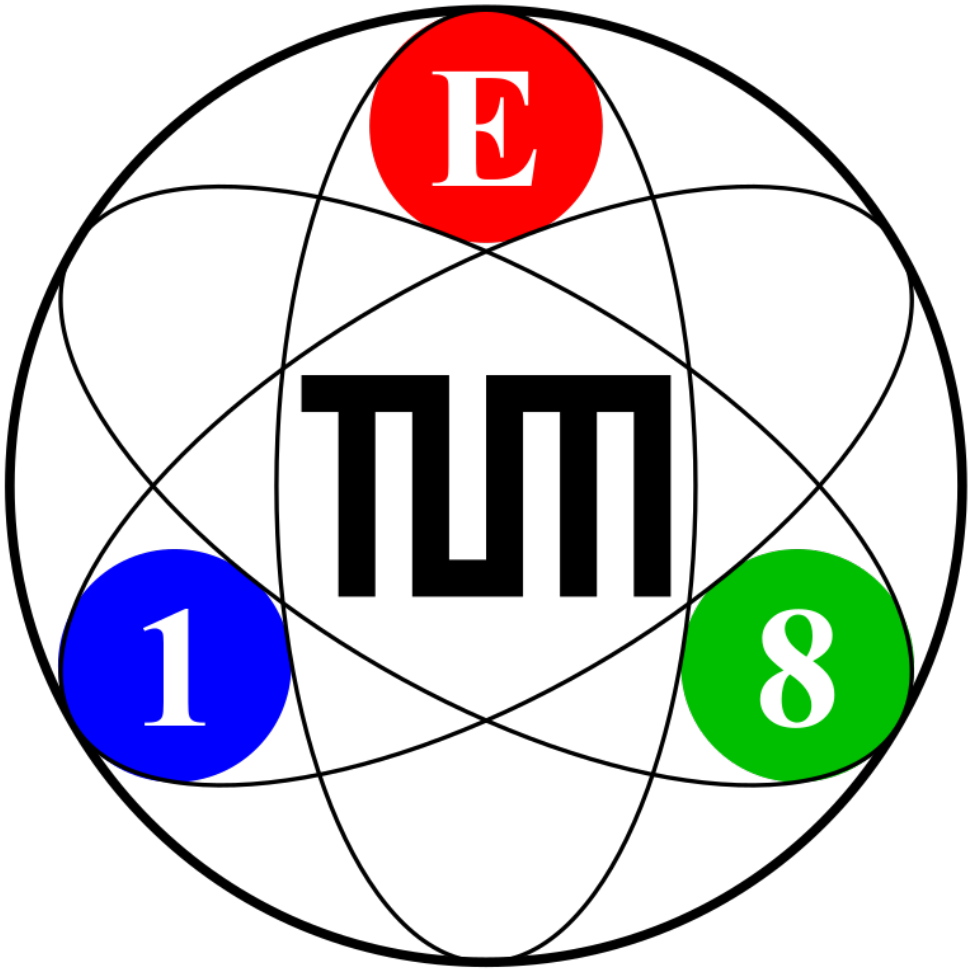

Figure 4.5.: Experimental setup for the noise analysis of the frontend electronics of the scintillating fiber hodoscope.

The scan will be performed by the controlling computer, which will scan through a range of thresholds of the time discriminator of the Citiroc1A ASIC and record the number of hits per second on each channel with an intergration time of 300 ms. This scan will be preformed for several different high gains near the maximum value of 62.

4.4.2. Theoretical backround for the threshold scan

The noise on the unconected frontend electronics is assumed to be of gaussian distribution,

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4.1}$$

where μ is the mean and σ the standard deviation of the noise.[10]
The number of hits per second on each channel of the Citiroc1A ASIC is assumed to be proportional to the number of noise events above the threshold of the time discriminator.

and thus to the integral of the noise distribution above the threshold.
The number of hits per second vs the threshold of the time discriminator is expected to follow an S-curve, which can be described by the error function,

$$N = \frac{A}{2}\left(1 - \mathrm{erf}\left(\frac{x-\mu}{\sqrt{2}\sigma}\right)\right) \tag{4.2}$$

where N is the number of hits per second and x the threshold of the time discriminator and A an renormalization factor.[10]

CHAPTER 5

Results

5.1. Threshold scan

For the threshold scan, the threshold was varied from NAN to NAN in steps of NAN for a range of high gains, starting with gain NAN and ending with the maxmial gain of 62. The number of hits was measured for each of the 32 channels of the Citiroc1A with an integration time of 300 ms.
The threshold scan was performed for both the Citiroc1A ASICs of FPGA 1 and FPGA 2.
The results of the threshold scan are shown in Figure 5.1.

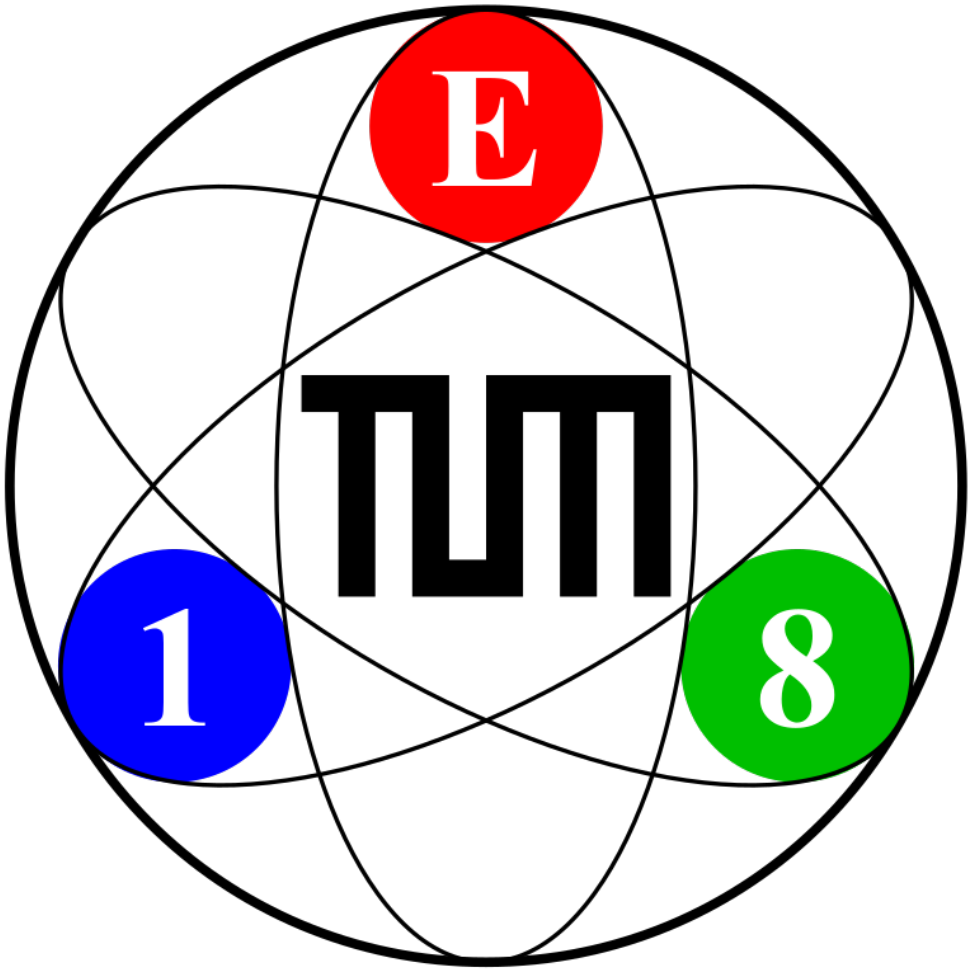

Figure 5.1.: Results of the threshold scan for Citiroc1A of FPGA 1 and FPGA 2.

The differantiated data of the threshold scan is shown in Figure **??**.

5.2. S-curve analysis

In order to determine the optimal threshold and caracterize the noise of the Citiroc1A ASICs of FPGA 1 and FPGA 2, an S-curve analysis was performed.
The falling edge of the pedestal was fitted with the S-curve described in Section 4.4.2
The results of the S-curve analysis are shown in Figure 5.2.

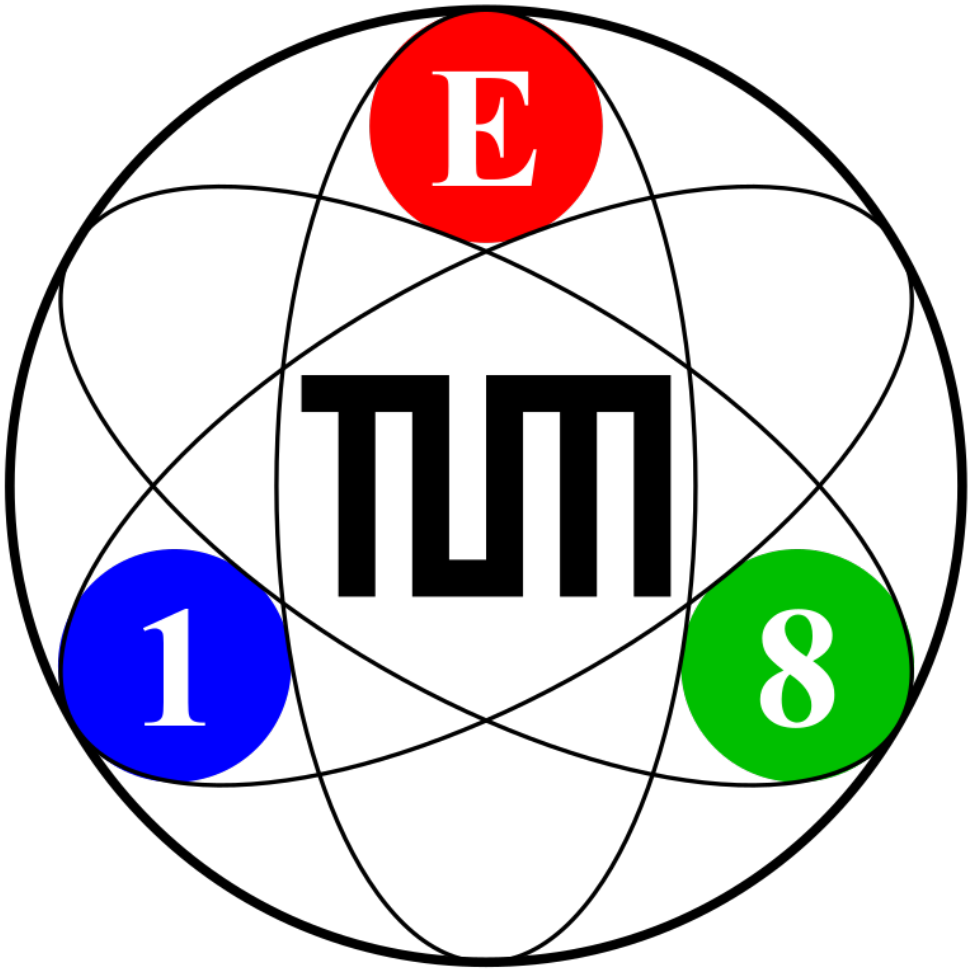

Figure 5.2.: Results of the S-curve analysis for Citiroc1A of FPGA 1 and FPGA 2.

The mean μ and the standard deviation σ of the noise distribution that were determined from the S-curve fits and are shown in Appendix **??**.

CHAPTER 6

Discussion

6.1. Evolation of the firmware

The noise analysis shows that the firmware allowes for the configuration of the Citiroc1A ASICs.
Furthermore, several test were performed to ensure that the Citiroc1A ASICs can be successfuly controlled by the FPGA, for example turning of stages of the ASIC or routing probing signals with the probe register, as explained in Section 3.3.2.

6.2. Threshold scan

A comperison of the threshold scan to one performed with the same configuration of the Citiroc1A ASIC performed on a by Weeroc provieded testboard is shown in Figure 6.1.

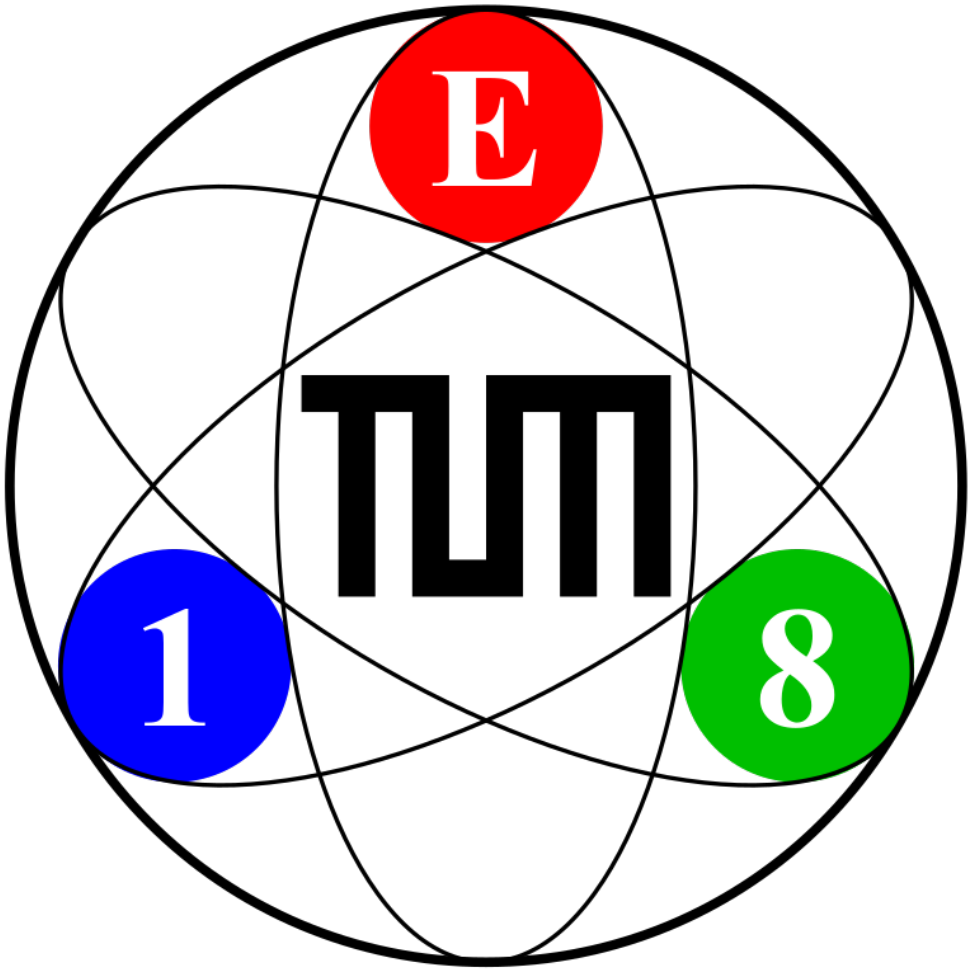

Figure 6.1.: Comparison of the threshold scan of the Citiroc1A ASICs of FPGA 1 and FPGA 2 to a threshold scan performed on a by Weeroc provided testboard.

The pedestal of the threshold scan performed with the frontend electronics is far larger than the pedestal of the threshold scan performed with the by testboard. This could be due to the fact that the input signals of the Citiroc1A ASICs pick up high frequency signals from the FPGA or the frontend electronics.
(INSERT:Place holder for actual results)
The s-curve analysis shows that the noise of the Citiroc1A ASICs is in the expected range.
(INSERT:Place holder for actual results)

CHAPTER 7

Conclusion and Outlook

7.1. Conclusion

Conclusion

7.2. Outlook

Outlook

APPENDIX A

Configurable registers of the Citiroc1A ASIC

Table A.1.: Configurable registers of the slow control register[7]

Field	Bits	Default	Position	Description
Register:channel_thr_time				
ch_0	4	0	0	Channel-dependent 4-bit threshold for time discriminator.
⋮				
ch_31	4	0	-	-
Register:channel_thr_charge				
ch_0	4	0	128	Channel-dependent 4-bit threshold for charge discriminator.
⋮				
ch_31	4	0	-	-
Register:discriminator_power				
discriminator_charge_en	1	0	256	Enable charge discriminator.
discriminator_charge_pp	1	0	-	Power pulse for charge discriminator.
discriminator_latched_output	1	0	-	1: latched, 0: direct output.

Configurable registers of the slow control register continued on next page

Field	Bits	Default	Position	Description
discriminator_time_en	1	1	-	Enable time discriminator.
discriminator_time_pp	1	1	-	Power pulse for time discriminator.
4bit_dac_charge_en	1	0	261	Enable 4-bit charge DAC.
4bit_dac_charge_pp	1	0	-	Power pulse for 4-bit charge DAC.
4bit_dac_time_en	1	1	-	Enable 4-bit time DAC.
4bit_dac_time_pp	1	1	-	Power pulse for 4-bit time DAC.
ch_0	1	1	265	0: masked, 1: unmasked.
⋮				
ch_31	1	1	-	-
Register:track_and_hold_power				
high_gain_pp	1	0	297	Enable high gain.
high_gain_en	1	0	-	-
low_gain_pp	1	0	-	Power pulse for low gain.
low_gain_en	1	0	-	Enable low gain.
weak_bias	1	0	-	1: weak bias (600kHz max), 0: high bias (5MHz max).
Register:peak_detector_power				
high_gain_pp	1	0	302	Enable high gain for peak detector.
high_gain_en	1	0	-	-
low_gain_pp	1	0	-	Power pulse for low gain.
low_gain_en	1	0	-	Enable low gain for peak detector.
Register:select_peak_sensing				
high_gain_th	1	0	306	0: peak detector, 1: track and hold.
low_gain_th	1	0	-	-
peak_sensing_cell_bypass	1	0	-	0: cell active, 1: bypass peak sensing cell.
peak_sensing_external_trigger	1	0	-	0: internal trigger, 1: external trigger.

Configurable registers of the slow control register continued on next page

Field	Bits	Default	Position	Description
Register:shaper				
fast_shaper_follower_pp	1	0	310	Power pulse for fast shaper follower.
fast_shaper_en	1	1	-	Enable fast shaper.
fast_shaper_pp	1	1	-	Power pulse for fast shaper.
low_gain_slow_shaper_pp	1	0	-	Power pulse for low gain slow shaper.
low_gain_slow_shaper_en	1	0	-	Enable low gain slow shaper.
low_gain_slow_shaper_time_const	3	0	-	See the table above for values.
high_gain_slow_shaper_pp	1	0	-	Power pulse for high gain slow shaper.
high_gain_slow_shaper_en	1	0	-	Enable high gain slow shaper.
high_gain_slow_shaper_time_const	3	0	-	See the table above for values.
Register:pre_amp_power				
low_gain_weak_bias	1	0	323	0: normal bias, 1: weak bias.
high_gain_pp	1	1	-	Power pulse for high gain preamp.
high_gain_en	1	1	-	Enable high gain preamp.
low_gain_pp	1	0	-	Power pulse for low gain preamp.
low_gain_en	1	0	-	Enable low gain preamp.
fast_shaper_low_gain	1	0	-	0: fast shaper on high gain.
Register:input_dac				
dac_en	1	1	329	Input DAC for bias correction.
dac_ref	1	1	-	Voltage ref: 1 = internal 4.5V, 0 = internal 2.5V, depends on vdd_dac.

Configurable registers of the slow control register continued on next page

Field	Bits	Default	Position	Description
ch_0	8	255	-	VSipm = V_HV - V_DAC (check what makes sense here).
ch_0_en	1	1	-	Enable channel 0 input DAC.
⋮				
ch_31	8	255	-	Same as ch_0 for channel 31.
ch_31_en	1	1	-	Enable channel 31 input DAC.
Register:channel_preamp				
ch_0_hg	6	62	619	High gain preamp setting.
ch_0_lg	6	0	-	Low gain preamp setting.
ch_0_ctest_hg	1	0	-	1: Connect injection capacitance for test signal.
ch_0_ctest_lg	1	0	-	1: Connect low gain injection capacitance.
ch_0_disable	1	0	-	1 disables preamp for channel 0.
⋮				
ch_31_hg	6	62	-	High gain preamp setting.
ch_31_lg	6	0	-	Low gain preamp setting.
ch_31_ctest_hg	1	0	-	1: Connect injection capacitance for test signal.
ch_31_ctest_lg	1	0	-	1: Connect low gain injection capacitance.
ch_31_disable	1	0	-	1 disables preamp for channel 0.
Register:service_blocks				
temp_pp	1	1	999	Enable power pulse for temperature monitoring.
temp_en	1	1	-	Enable temperature monitoring.
band_gap_pp	1	1	-	Enable power pulse for band gap reference.

Configurable registers of the slow control register continued on next page

Field	Bits	Default	Position	Description
band_gap_en	1	1	-	Enable band gap reference.
Register:threshold_dac				
charge_dac_en	1	0	1103	Enable charge threshold DAC.
charge_dac_pp	1	0	-	Power pulse for charge threshold DAC.
time_dac_en	1	1	-	Enable time threshold DAC.
time_dac_pp	1	1	-	Power pulse for time threshold DAC.
charge_threshold	10	0	-	Charge threshold value.
time_threshold	10	-	-	Time threshold value (e.g., 200 for 1 cell min, 250 for 2 cells).
Register:otaq_power				
high_gain_en	1	1	1127	Enable high gain for OTAQ.
high_gain_pp	1	1	-	Power pulse for high gain OTAQ.
low_gain_en	1	0	-	Enable low gain for OTAQ.
low_gain_pp	1	0	-	Power pulse for low gain OTAQ.
debug_probe_en	1	1	-	Enable debug probe.
debug_probe_pp	1	1	-	Power pulse for debug probe.
Register:input_output				
output_buffer_bias	1	0	1133	Output OTA buffer bias: 0 = auto bias, 1 = force on.
val_event_receiver_en	1	1	-	Enable validation event receiver.
val_event_receiver_pp	1	1	-	Power pulse for validation event receiver.
raz_chn_en	1	1	-	Enable RAZ channel.
raz_chn_pp	1	1	-	Power pulse for RAZ channel.

Configurable registers of the slow control register continued on next page

Field	Bits	Default	Position	Description
digital_output_en	1	1	-	Enable digital multiplexed output.
or32_output_en	1	1	-	Enable OR32 output.
or32_oc_output_en	1	1	-	Enable OR32 over-current output.
trigger_polarity	1	0	-	Trigger polarity: 0 = positive (rising edge), 1 = negative (falling edge).
or32_t_oc_en	1	1	-	Enable OR32 timeout over-current.
32_triggers_en	1	1	1143	Enable 32 triggers.

Table A.2.: Configurable registers of the probe register[7]

Register	**Field**	**Bits**	**Position**	**Type**
out_probe_fast_shaper	ch_0	1	0	Analog - Out_probe
	⋮		⋮	
	ch_31	1	31	
out_probe_slow_shaper_lg	ch_0	1	32	Analog - Out_probe
	⋮		⋮	
	ch_31	1	63	
digital_probe_peak_sense_lg	ch_0	1	64	Digital - Digital_probe
	⋮		⋮	
	ch_31	1	95	
out_probe_slow_shaper_hg	ch_0	1	96	Analog - Out_probe
	⋮		⋮	
	ch_31	1	127	
digital_probe_peak_sense_hg	ch_0	1	128	Digital - Digital_probe
	⋮		⋮	
	ch_31	1	159	
out_probe_preamp_hg	ch_0	1	160	Analog - Out_probe
	⋮		⋮	
	ch_31	1	191	
out_probe_preamp_lg	ch_0	1	192	Analog - Out_probe
	⋮		⋮	
	ch_31	1	223	
input_dac_probe	ch_0	1	224	Analog - Out_probe_dac_5V
	⋮		⋮	
	ch_31	1	255	

APPENDIX B

Fit parameters of the S-curve analysis

gain	FPGA 1	FPGA 2	Citiroc1A	Channel	μ	σ	FPGA 1	FPGA 2	Citiroc1A	Channel	μ	σ
57	ch_0	NAN	NAN	ch_0	NAN	NAN	ch_0	NAN	NAN	ch_0	NAN	NAN
	ch_1	NAN	NAN	ch_1	NAN	NAN	ch_1	NAN	NAN	ch_1	NAN	NAN
	ch_2	NAN	NAN	ch_2	NAN	NAN	ch_2	NAN	NAN	ch_2	NAN	NAN
	ch_3	NAN	NAN	ch_3	NAN	NAN	ch_3	NAN	NAN	ch_3	NAN	NAN
	ch_4	NAN	NAN	ch_4	NAN	NAN	ch_4	NAN	NAN	ch_4	NAN	NAN
	ch_5	NAN	NAN	ch_5	NAN	NAN	ch_5	NAN	NAN	ch_5	NAN	NAN
	ch_6	NAN	NAN	ch_6	NAN	NAN	ch_6	NAN	NAN	ch_6	NAN	NAN
	ch_7	NAN	NAN	ch_7	NAN	NAN	ch_7	NAN	NAN	ch_7	NAN	NAN
	ch_8	NAN	NAN	ch_8	NAN	NAN	ch_8	NAN	NAN	ch_8	NAN	NAN
	ch_9	NAN	NAN	ch_9	NAN	NAN	ch_9	NAN	NAN	ch_9	NAN	NAN
	ch_10	NAN	NAN	ch_10	NAN	NAN	ch_10	NAN	NAN	ch_10	NAN	NAN
	ch_11	NAN	NAN	ch_11	NAN	NAN	ch_11	NAN	NAN	ch_11	NAN	NAN
	ch_12	NAN	NAN	ch_12	NAN	NAN	ch_12	NAN	NAN	ch_12	NAN	NAN
	ch_13	NAN	NAN	ch_13	NAN	NAN	ch_13	NAN	NAN	ch_13	NAN	NAN
	ch_14	NAN	NAN	ch_14	NAN	NAN	ch_14	NAN	NAN	ch_14	NAN	NAN
	ch_15	NAN	NAN	ch_15	NAN	NAN	ch_15	NAN	NAN	ch_15	NAN	NAN
	ch_16	NAN	NAN	ch_16	NAN	NAN	ch_16	NAN	NAN	ch_16	NAN	NAN
	ch_17	NAN	NAN	ch_17	NAN	NAN	ch_17	NAN	NAN	ch_17	NAN	NAN
	ch_18	NAN	NAN	ch_18	NAN	NAN	ch_18	NAN	NAN	ch_18	NAN	NAN
	ch_19	NAN	NAN	ch_19	NAN	NAN	ch_19	NAN	NAN	ch_19	NAN	NAN
	ch_20	NAN	NAN	ch_20	NAN	NAN	ch_20	NAN	NAN	ch_20	NAN	NAN
	ch_21	NAN	NAN	ch_21	NAN	NAN	ch_21	NAN	NAN	ch_21	NAN	NAN
	ch_22	NAN	NAN	ch_22	NAN	NAN	ch_22	NAN	NAN	ch_22	NAN	NAN
	ch_23	NAN	NAN	ch_23	NAN	NAN	ch_23	NAN	NAN	ch_23	NAN	NAN
	ch_24	NAN	NAN	ch_24	NAN	NAN	ch_24	NAN	NAN	ch_24	NAN	NAN
	ch_25	NAN	NAN	ch_25	NAN	NAN	ch_25	NAN	NAN	ch_25	NAN	NAN
	ch_26	NAN	NAN	ch_26	NAN	NAN	ch_26	NAN	NAN	ch_26	NAN	NAN
	ch_27	NAN	NAN	ch_27	NAN	NAN	ch_27	NAN	NAN	ch_27	NAN	NAN
	ch_28	NAN	NAN	ch_28	NAN	NAN	ch_28	NAN	NAN	ch_28	NAN	NAN
	ch_29	NAN	NAN	ch_29	NAN	NAN	ch_29	NAN	NAN	ch_29	NAN	NAN
	ch_30	NAN	NAN	ch_30	NAN	NAN	ch_30	NAN	NAN	ch_30	NAN	NAN
	ch_31	NAN	NAN	ch_31	NAN	NAN	ch_31	NAN	NAN	ch_31	NAN	NAN

Table B.1.: Mean μ and standard deviation σ of the noise distribution gained from the fit of S-curve for gain 57. ASIC 1 and ASIC 2 are the Citiroc1A ASICs of FPGA 1 and ASIC 3 and ASIC 4 are the Citiroc1A ASICs of FPGA 2

gain	FPGA 1	FPGA 2	Citiroc1A	Channel	μ	σ	FPGA 1	FPGA 2	Citiroc1A	Channel	μ	σ
58	ch_0	NAN	NAN	ch_0	NAN	NAN	ch_0	NAN	NAN	ch_0	NAN	NAN
	ch_1	NAN	NAN	ch_1	NAN	NAN	ch_1	NAN	NAN	ch_1	NAN	NAN
	ch_2	NAN	NAN	ch_2	NAN	NAN	ch_2	NAN	NAN	ch_2	NAN	NAN
	ch_3	NAN	NAN	ch_3	NAN	NAN	ch_3	NAN	NAN	ch_3	NAN	NAN
	ch_4	NAN	NAN	ch_4	NAN	NAN	ch_4	NAN	NAN	ch_4	NAN	NAN
	ch_5	NAN	NAN	ch_5	NAN	NAN	ch_5	NAN	NAN	ch_5	NAN	NAN
	ch_6	NAN	NAN	ch_6	NAN	NAN	ch_6	NAN	NAN	ch_6	NAN	NAN
	ch_7	NAN	NAN	ch_7	NAN	NAN	ch_7	NAN	NAN	ch_7	NAN	NAN
	ch_8	NAN	NAN	ch_8	NAN	NAN	ch_8	NAN	NAN	ch_8	NAN	NAN
	ch_9	NAN	NAN	ch_9	NAN	NAN	ch_9	NAN	NAN	ch_9	NAN	NAN
	ch_10	NAN	NAN	ch_10	NAN	NAN	ch_10	NAN	NAN	ch_10	NAN	NAN
	ch_11	NAN	NAN	ch_11	NAN	NAN	ch_11	NAN	NAN	ch_11	NAN	NAN
	ch_12	NAN	NAN	ch_12	NAN	NAN	ch_12	NAN	NAN	ch_12	NAN	NAN
	ch_13	NAN	NAN	ch_13	NAN	NAN	ch_13	NAN	NAN	ch_13	NAN	NAN
	ch_14	NAN	NAN	ch_14	NAN	NAN	ch_14	NAN	NAN	ch_14	NAN	NAN
	ch_15	NAN	NAN	ch_15	NAN	NAN	ch_15	NAN	NAN	ch_15	NAN	NAN
	ch_16	NAN	NAN	ch_16	NAN	NAN	ch_16	NAN	NAN	ch_16	NAN	NAN
	ch_17	NAN	NAN	ch_17	NAN	NAN	ch_17	NAN	NAN	ch_17	NAN	NAN
	ch_18	NAN	NAN	ch_18	NAN	NAN	ch_18	NAN	NAN	ch_18	NAN	NAN
	ch_19	NAN	NAN	ch_19	NAN	NAN	ch_19	NAN	NAN	ch_19	NAN	NAN
	ch_20	NAN	NAN	ch_20	NAN	NAN	ch_20	NAN	NAN	ch_20	NAN	NAN
	ch_21	NAN	NAN	ch_21	NAN	NAN	ch_21	NAN	NAN	ch_21	NAN	NAN
	ch_22	NAN	NAN	ch_22	NAN	NAN	ch_22	NAN	NAN	ch_22	NAN	NAN
	ch_23	NAN	NAN	ch_23	NAN	NAN	ch_23	NAN	NAN	ch_23	NAN	NAN
	ch_24	NAN	NAN	ch_24	NAN	NAN	ch_24	NAN	NAN	ch_24	NAN	NAN
	ch_25	NAN	NAN	ch_25	NAN	NAN	ch_25	NAN	NAN	ch_25	NAN	NAN
	ch_26	NAN	NAN	ch_26	NAN	NAN	ch_26	NAN	NAN	ch_26	NAN	NAN
	ch_27	NAN	NAN	ch_27	NAN	NAN	ch_27	NAN	NAN	ch_27	NAN	NAN
	ch_28	NAN	NAN	ch_28	NAN	NAN	ch_28	NAN	NAN	ch_28	NAN	NAN
	ch_29	NAN	NAN	ch_29	NAN	NAN	ch_29	NAN	NAN	ch_29	NAN	NAN
	ch_30	NAN	NAN	ch_30	NAN	NAN	ch_30	NAN	NAN	ch_30	NAN	NAN
	ch_31	NAN	NAN	ch_31	NAN	NAN	ch_31	NAN	NAN	ch_31	NAN	NAN

Table B.2.: Mean μ and standard deviation σ of the noise distribution gained from the fit of S-curve for gain 58. ASIC 1 and ASIC 2 are the Citiroc1A ASICs of FPGA 1 and ASIC 3 and ASIC 4 are the Citiroc1A ASICs of FPGA 2

APPENDIX C

Code

```
1 this is code
```

List of Figures

Bibliography

[1] B Adams et al. *COMPASS++/AMBER: Proposal for Measurements at the M2 beam line of the CERN SPS Phase-1: 2022-2024*. Tech. rep. Geneva: CERN, 2019. URL: https://cds.cern.ch/record/2676885 (cit. on pp. 1–6).

[2] B. Adams et al. *Letter of Intent: A New QCD facility at the M2 beam line of the CERN SPS (COMPASS++/AMBER)*. 2019. arXiv: 1808.00848 [hep-ex]. URL: https://arxiv.org/abs/1808.00848 (cit. on p. 4).

[3] Maxim Alexeev et al. "Design and Testing of a new Tracking System for the Proton Charge Radius Measurement with the AMBER Experiment at CERN". In: *PoS* VERTEX2023 (2024), p. 049. DOI: 10.22323/1.448.0049 (cit. on p. 5).

[4] E. E. Chambers and R. Hofstadter. "Structure of the Proton". In: *Phys. Rev.* 103 (5 Sept. 1956), pp. 1454–1463. DOI: 10.1103/PhysRev.103.1454 (cit. on p. 1).

[5] The AMBER Collaboration. *AMBER Status Report 2024*. Tech. rep. Geneva: CERN, 2024. URL: https://cds.cern.ch/record/2907624 (cit. on p. 7).

[6] Procedia Cpi. "Classifying FPGA Technology in Digital Signal Processing: A review". In: *International Journal of Engineering & Technology Sciences* 2023 (July 2024), p. 10 (cit. on p. 8).

[7] Karl Eichhorn. Private Communication. Nov. 2024 (cit. on pp. 5–10, 35, 41).

[8] J M Friedrich and O Denisov. *AMBER Status Report 2022*. Tech. rep. Geneva: CERN, 2022. URL: https://cds.cern.ch/record/2810822 (cit. on pp. 6, 7, 10).

[9] Werner Heisenberg. *Physics and Philosophy: The Revolution in Modern Science*. Ed. by Ruth Nanda Anshen. Amherst, N.Y.: Prometheus Books, 1958 (cit. on p. 1).

[10] P. Kmon, P. Maj, P. Grybos, and R. Szczygiel. “An Effective Multilevel Offset Correction Technique for Single Photon Counting Pixel Detectors”. In: *IEEE Transactions on Nuclear Science* 63.2 (2016), pp. 1194–1201. DOI: 10.1109/TNS.2016.2527834 (cit. on pp. 25, 26).

[11] Igor Konorov. Private Communication. Oct. 2024 (cit. on pp. 8, 11, 12).

[12] C. Ghabrous Larrea et al. “IPbus: a flexible Ethernet-based control system for xTCA hardware”. In: *Journal of Instrumentation* 10.02 (Feb. 2015), p. C02019. DOI: 10.1088/1748-0221/10/02/C02019 (cit. on pp. 17, 18).

[13] Barrie M. Peake. “The discovery of the electron, proton, and neutron”. In: *Journal of Chemical Education* 66.9 (1989), p. 738. DOI: 10.1021/ed066p738 (cit. on p. 1).

[14] Weeroc. *citiroc1a-datasheet-v2-53*. Dec. 2019. URL: https://www.weeroc.com/~documents/products/citiroc-1a/citiroc1a-datasheet-v2-53/?layout=file (cit. on pp. 12–16).

[15] R.L. Workman et al. *Review of Particle Physics: 2022*. Vol. 2022. Oxford: Oxford University Press, 2022. DOI: 10.1093/ptep/ptac097 (cit. on p. 3).

Acknowledgement

I want to thank:

Tutor .

Professor .

Eidesstattliche Erklärung

Ich versichere hiermit an Eides statt, dass ich die von mir eingereichte Arbeit bzw. die von mir namentlich gekennzeichneten Teile selbständig verfasst und ausschließlich die angegebenen Hilfsmittel benutzt habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ort, Datum Unterschrift