Methods for FPGA pre-processing of data for the ALOFT readout system

A thesis by

Viljar Dahle

for a degree of

Master of Science in Physics

Department of Physics and Technology

University of Bergen

June 2022

Abstract

In 2017 the *Airborne Lightning Observatory for FEGS & TGFs* (ALOFT) campaign was completed with the goal of studying thundercloud related high energy phenomena, namely *Terrestrial Gamma-Ray Flashes* (TGFs) and Gamma-ray Glows, and their connections. This was done on a high-altitude airplane flying over the thunderclouds. The campaign observed two gamma-ray glows but no TGFs.

A new ALOFT campaign has been confirmed for 2023 and will contain several upgrades and improvements. Some of these improvements includes developing two new detectors that will be added to the instrument, as well as a new readout system based around a ZYNQ-7000 series *System on Chip* (SoC).

Through this thesis, my contribution to this development has been two-folded:

1. Verify that the integrated ZYNQ *Serial Peripheral Interface* (SPI) controller can be used to interface and configure the *Analog to Digital Converters* (ADCs) in the new detectors. This involves a) making a converter between the four wire SPI used by the ZYNQ and the three wire SPI used by the ADCs, b) modelling the behaviour of how the ADCs control registers communicate, and c) verify if the ZYNQ SPI controller can transmit the protocols required to configure the ADCs, and that it can access the FPGA part of the ZYNQ SoC.
2. During TGFs the data output of the new detectors far exceeds the ability of the system to send all the raw data to storage, requiring the data to be temporarily buffered. The buffer capacity needed to guarantee that no data is lost, would consume 73% of the total shared buffer capacity available to the FPGA part of the system. This leaves very little available capacity for the remaining detectors and any FPGA modules. In this thesis different approaches to reduce the required buffer requirements has been explored.

My work in this thesis shows that:

1. The ZYNQ SPI controller can be used to configure the ADCs by a) showing that the required converter can easily be made in the FPGA part of the ZYNQ, by b) modelling how the ADCs are configured can be created, and by c) testing that the ZYNQ SPI controller is compatible with the protocols used to interact and configure the ADCs.
2. After exploring both real-time analysis of the data on the SoC and compressing the raw data, the safest methods to use is compression. It is also shown that with the compression explored and considered, it is no problem reducing the buffer requirement from 73% to less than 30% of the total shared capacity.

Acknowledgments

Working on this thesis has at times been challenging, including changing objectives and covid.

Huge thanks to my supervisor, *Professor Johan Alme*, for guidance and giving me the inspiration needed to push through some of the challenges.

Thanks to *Professor Kjetil Ullaland* for all the provided help and patience.

Thanks to *Professor Martino Marisaldi* and peer student *Linn Amalie Kvaale Ramdal* for answering dumb questions and providing some of the underlying material this thesis is built on.

Thanks to everyone who has provided me feedback in the process of writing. Extra shout-out to *Anders Lerum Alme*, who spent some of his free days entirely to help.

Special thanks to my friends and family, who provided the support, love, and encouragements required to get through the ups and the downs.

Viljar Dahle

Table of Contents

Abbreviations

ADC Analogue to Digital Converter
AGILE Astro rivelatore Gamma a Immagini Leggero
ALOFT Airborne Lightning Observatory for FEGS & TGFs

BATSE Burst and Transient Source Experiment
BFM Bus Functional Model
BRAM Block Random Access Memory

CGRO Compton Gamma Ray Observatory
CPU Central Processing Unit
CSB Chip Select Bar

DUT Device under test

EMIO Extended Multiplexed I/O interface

FEGS Fly's Eye Global lightning mapper Simulator
FIFO First In First Out
FPGA Field-Programmable Gate Array

GOES Geostationary Operational Environmental Satellites

MIO Multiplexed I/O interface
MISO Master In Slave Out
MOSI Master Out Slave In

NOAA National Oceanic and Atmospheric Administration

PL Programming logic
PYNQ Python productivity for ZYNQ

RAM Random Access Memory
RHESSI Reuven Ramaty High Energy Solar Spectroscopic Imager
RWB Read/Write Bar
RxFIFO Reception FIFO

SCLK Signal Clock
SDENb Serial Data Enable bar
SDIO Serial Data In/Out
SoC System on Chip
SPI Serial Peripheral Interface
SPI3 3 wire SPI
SRAM Static Random-Access Memory
SS Slave Select

TGF Terrestrial Gamma-Ray Flash
TxFIFO Transmission FIFO

UVVM Universal VHDL Verification Methodology

VVC VHDL Verification Component

1 Introductions

1.1 Background

Over the years we have discovered that visible light from lighting strikes is far from the only type of electromagnetic radiation emitted from thunderstorms. The spectrum of these emissions varies greatly, from so-called *Whistlers* emitting Very Low Frequency radio signals [1], to *Terrestrial Gamma-Ray Flashes* (TGFs) with some of the most energetic gamma radiation seen on Earth [2]. Many of the higher energy phenomena have only been discovered the last few decades and are generally poorly understood. By studying them one can gain a greater understanding of the processes behind them, and a greater understanding of thunderclouds in general. Figure 1-1 shows an artist depiction of a TGF visualised from space.

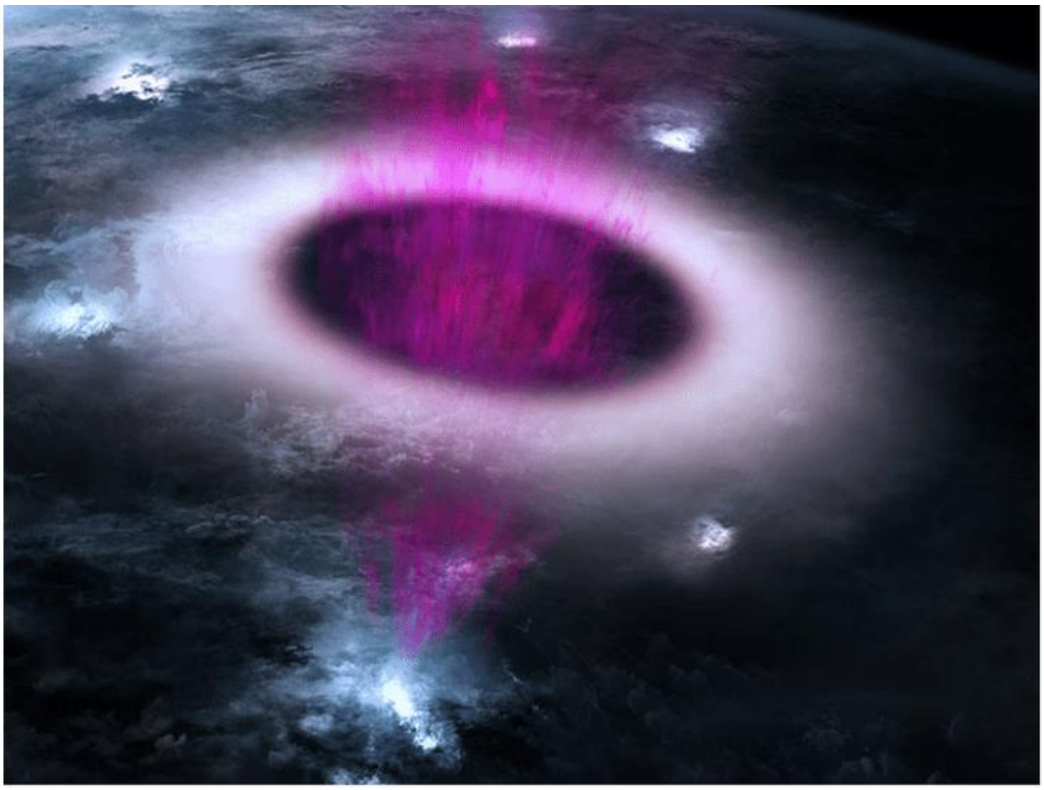


Figure 1-1: Artistic depiction of a terrestrial gamma ray flash. Credit: Birkeland Centre for Space Science and Mount Visual [3].

Airborne Lightning Observatory for FEGS & TGFs (ALOFT) is one such study and has two goals; to validate and test instruments going on a future satellite to observe lighting, and to study lightning phenomena generating gamma radiation. It does this by taking measurements from an airplane flying above the thunderclouds and was first done in 2017, with a planned repeat with some upgrades in 2023. ALOFT is a collaboration between the University of Bergen and the University of Alabama, Huntsville with support from NASA's Marshall Space Flight Center and Armstrong Flight Research Center.

1.2 ALOFT Instrument

As part of an upgrade between the 2017 and the upcoming 2023 ALOFT campaign two additional gamma detectors are being developed by the University of Bergen. Both new detectors can be divided into four steps, each part of converting a gamma photon to useable data. The first step is a sensor that converts any energy deposited in it by passing gamma photons into a proportionally strong electric signal. Next is some front-end electronics that amplifies, filters, and converts the signal to reduce noise and make it compatible with the third step, which consists of an Analogue to Digital Converter (ADC). The ADC samples the analogue electrical signal into a digital signal that it then sends to the last step, the capture card.

The capture card primarily consists of a Xilinx ZYNQ-7000 series SoC (System on Chip) and some mass storage medium (SD card). A single capture card is shared between both the two new detectors as well as two from the 2017 campaign.

The main function is to receive the raw data from the ADCs, do some pre-processing, and then send the data to the storage medium. It also has the task of communicating any configurations to the ADCs, as well as having an ethernet connection that can be used to control the system and read live status reports. Figure 1-2 shows a simplified overview of how the two new detectors are organised. Note that while the ADC controller lines and the ethernet are shown in the figure, they are treated as separate from the detector signal path in this thesis.

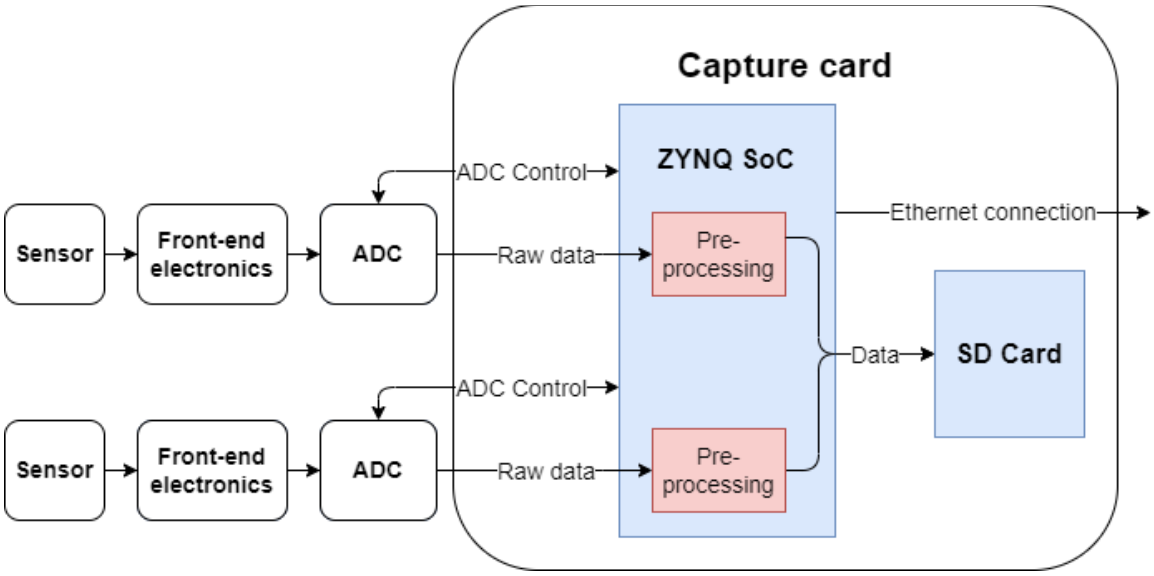


Figure 1-2 A simplified overview of how the detector system is set up, showing only the two new detectors. The old detectors are connected the same way.

1.3 Objective of Thesis

The scope of this thesis is divided in two parts, both related to how the ZYNQ handles data from and to the new detectors.

The first goal is to verify the ability of the built-in ZYNQ SPI (Serial Peripheral Interface) controller to interface and communicate with the control registers of the ADCs on the new detectors. This entails several steps. The first step is creating a converter that allows the four wire SPI that the ZYNQ uses to interface with the three wire SPI variant that both the ADCs use. Next is creating testbench models of the how the ADC control registers are accessed and behave. This can be used to simulate and verify that communication between the ZYNQ and the control registers will have the expected outcome in a

structured and reliable way. Lastly is verifying that ZYNQ SPI controller can be used and sent to the FPGA part of the ZYNQ SoC.

The background for the second goal is that during a TGF a large amount of data is generated and will require buffering. To guarantee no data is lost through buffer overflow an impractical large portion, 73%, of the available buffer capacity of the system would be required. The goal is therefore to find ways to reduce the size of the raw data to enable it to be safely buffered. It is of high importance that the system is reliable and that the reduction in data size happens without any loss of the actual event data. Finally, the quality of the data should not be impaired by the selected compression algorithms.

1.4 Thesis outline

Chapter 2: High energy emission from thunderclouds

Chapter 2 gives a short introduction to the history and physics of the two gamma radiation events that ALOFT is looking for, namely the terrestrial gamma-ray flash and the gamma-ray glow. This is relevant to understand the basis for the ALOFT campaign and its gamma detectors. The chapter also contains the background of the ALOFT campaign itself and how the upcoming 2023 ALOFT campaign differs from the previous 2017 campaign.

Chapter 3: ALOFT Systems

Chapter 3 provides a summary of the different components that makes up the two new detectors, how they work and how the interact with each other.

Chapter 4: ADC configuration

Chapter 4 documents the work done towards verifying the selected ADCs can be interfaced and controlled by the ZYNQ SPI controller.

Chapter 5: Signal recording methods

Chapter 5 documents the problem of handling the amounts of raw data processed the new detectors, as well as discussing two different approaches of how to solve this.

Chapter 6: Conclusion and outlook

Chapter 6 provides the overall conclusion of the thesis, as well as suggestions to future work for the project.

2 High energy emission from thunderclouds

This chapter discusses the historical and theoretical background for the ALOFT campaign and its goals to study Terrestrial Gammy-ray Flashes (TGFs) and Gamma-ray glows.

There are two different high energy phenomena from thunderclouds that ALOFT is looking for. The first is the Terrestrial Gamma-ray Flash (TGF), that has a sub-millisecond duration and a very high intensity. The other is known as Gamma-ray glows and is a phenomenon lasting several seconds to minutes where an area of thunderclouds starts to emit relatively low intensity gamma radiation.

2.1 Discovery of high energy phenomena

2.1.1 TGFs

What would become known as TGFs was first observed in 1991 by the *Burst and Transient Source Experiment* (BATSE) onboard the *Compton Gamma Ray Observatory* (CGRO) satellite [4]. While looking for celestial gamma-ray bursts such as supernovas it happened to record several short and intense gamma-ray events that originated from Earth. The position and timing of the events quickly linked it to thunderstorm.

In the decades since its discovery, a multitude of satellites have incidentally or intentionally observed TGFs. The short duration and high intensity make powerful TGFs reasonably easy to observe from satellites viewing large areas, but a significant portion of the details are lost to unknown source parameters such as location and production altitude.

Some of the missions that have observed TGFs include the *Reuven Ramaty High Energy Solar Spectroscopic Imager* (RHESSI) launched in 2002 [5], the *Astro rivelatore Gamma a Immagini Leggero* (AGILE) launched in 2007 [6], the *Fermi Gamma-Ray Space Telescope* launched in 2008 [7], and the *Atmosphere-Space Interactions Monitor* (ASIM) launched and connected to the International Space Station in 2018 [8]. Figure 21 shows a photo of ASIM on the International Space Station.

Figure 2-1 Photo of the ASIM module on the International Space Station. Credit asim.dk [9]

2.1.2 Gamma-ray glows

In 1981 a NASA F-106 jet flew through a thundercloud with a detector trying to establish whether high energy photons were generated in thunderstorms. The resulting increased detection rate compared to the background radiation marks the first confirmed observation of what is called gamma-ray glow. [10]

Gamma-ray glows cover a significant area and lasts up to multiple minutes at a time. This makes it possible to study them through different means, including planes, balloons [11], and ground-based instruments [12]. While there are quite good models for their production mechanisms, there is still uncertainty about their expansion in time and space, their frequency, and if they are linked to TGFs.

2.2 ALOFT Campaign

2.2.1 Background

The *Geostationary Operational Environmental Satellites* (GOES) is a family of geostationary satellites dedicated to monitor the atmosphere made and operated by United States' *National Oceanic and Atmospheric Administration* (NOAA)'s *National Environmental Satellite, Data, and Information Service* division. The program started in 1975 with the launch of GOES-1 and did its latest launch (as of writing) in 2022 with GOES-18. There is at least one more launch planned at the earliest 2024. [13]

One of GOES' recuring instruments is the *Global Lightning Mapper*, which as the name implies has as goal to monitor lightning strikes. To validate and calibrate the Global Lightning Mapper they created the *Fly's Eye Global lightning mapper Simulator* (FEGS), with the intent of putting FEGS on a high-altitude airplane for testing. [14].

In addition to the primary instrument, FEGS also has several additional instruments to improve the scientific value of the flight [14]. With a focus of the other instruments on high energy lightning phenomena, the complete platform got the designation Airborne Lightning Observatory for FEGS and TGFs (ALOFT).

While it is significantly easier to spot TGFs with satellites as they can view a large area at once, the resolution of exactly where the TGF happened is limited. For this reason, observing a TGF with a plane can give valuable data with a higher resolution. It is also makes it possible to look for TGFs too weak to be observed from space, as well as look for connections between gamma-ray glows and TGFs.

The airplane used by ALOFT is the NASA ER-2 high-altitude aircraft shown in Figure 2-2. ER-2 normally operates at altitudes from about 20k feet (6 km) to 70k feet (21 km) with a typical cruise speed of 410 knots (760 km/h or 211 m/s). [15]. ALOFT will be flown at 20 km altitude above the thunderclouds.

Figure 2-2 A NASA ER-2 in flight. During the 2017 ALOFT campaign the FEGS instrument was in the pod beneath the right wing. Picture by NASA/Carla Thomas [15].

2.2.2 ALOFT 2017 Campaign

The first ALOFT campaign was completed spring of 2017 and had two detector systems for detecting gamma rays. The large first detector, the *University of Bergen Bismuth-Germanium-Oxide* (UiB-BGO) detector, had a 27-ns time resolution and an energy range from about 300keV to about 40MeV. This detector is very similar to a detector on ASIM. The small second detector was a 1cm^2 *Lutetium–yttrium oxyorthosilicate* (LYSO) detector meant to handle higher fluxes of radiation. Due to its small size, it is unlikely that it can absorb all the energy of a gamma photon, making it more useful for counting the number of photons rather than also measuring its energy [16].

ALOFT was done aboard the NASA ER-2 High-Altitude Airborne Science Aircraft, which flew at 20km altitude above continental USA. The campaign consisted of 16 flights totalling about 70 flight hours of which about 45 were above thunderstorm regions. In the end, there were two gamma-ray glows observed and zero observed TGFs [16].

2.2.3 ALOFT 2023 Campaign

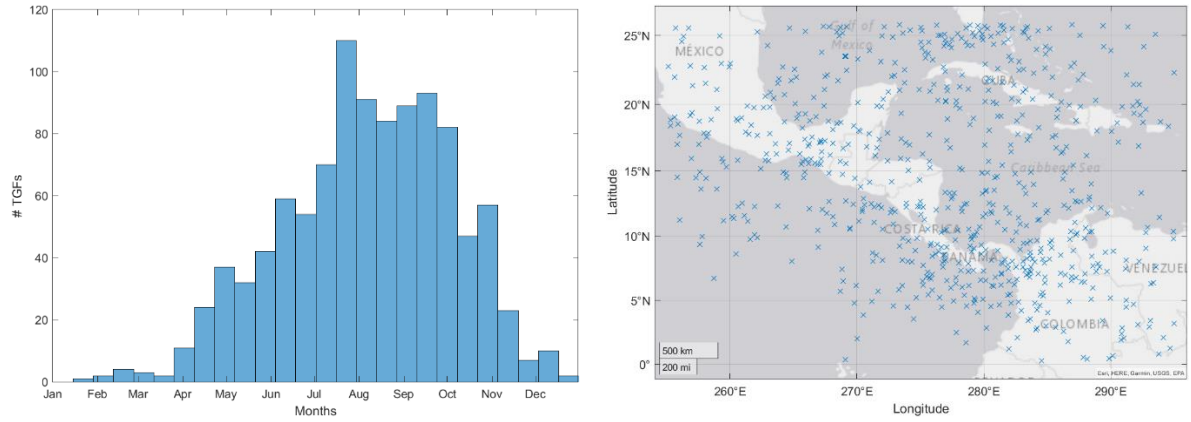
The following information is based on [17]

A new ALOFT campaign has been approved to be completed during midsummer 2023. Compared to the 2017 campaign it has a series of planned improvements to increase the odds of detecting TGFs and to extract more useful information should a TGF be observed.

The first improvement is where and when. Using data of historic TGFs detected by the FERMI satellite it has been determined that flying over the Central America and Caribbean area during mid-summer yields the best odds to see TGFs. Figure 2-3 shows 5 years of FERMI data, illustrating benefits of the time and area.

The second major improvement to the ALOFT 2023 is the instrument itself. There is planned an improved readout system as well as two new gamma detectors. The most significant new feature of the improved readout system is an ethernet connection that enables live communication between the instrument and an operator that is in contact with the pilot. This allows the plane to turn around and properly map an area of interest, such as a gamma-ray glow, should the plane pass it.

The new detectors consist of a tiny detector similar to the small detector, but capable of handling even higher fluxes, and a medium detector capable of measuring the energy of photons under higher fluxes than the large detector. Details about the new detectors can be found in Chapter 3.



(a) (b)

Figure 2-3 Data of TGFs detected by the Fermi satellite between January 1st 2010 and December 31st 2015 in the area latitude 0° to 30° N and 255° E to 295° E. Data taken from [18].

(a) A histogram of the time of year the TGFs were detected in 14 days bins.

(b) A map showing the TGF detections in the area during the months June to September

2.3 Characteristics of TGFs and gamma-ray glows

2.3.1 TGFs

TGFs are sub-millisecond long bursts of intense gamma radiation with individual photon energies of up to 100 MeV, although photons above 40 MeV are rare[19]. While the current model has problems explaining photons above 40 MeV, it is believed that the TGFs are caused by strong electric fields accelerating electrons to relativistic speeds causing gamma radiation through the process known as *bremsstrahlung* (see Chapter 2.4) [19].

It is estimated that TGFs generally are generated at altitudes between 12 km and 21 km[20], and that they radiate photons in a cone upwards with the greatest fluence near the centre, as show in Figure 2-4 [21]. Note that the fluence range from more than 10^4 photons/cm^2 in the centre to less than 10^{-1} photons/cm^2. This is why a wide range of detectors are used, each with different sensitivity to the photon flux.

Early it was believed TGFs where a relatively rare phenomena, but as better instruments have been developed their estimated frequency has increased. Some estimates go as high as 50000 TGFs every day[22], though the vast majority of these are never detected. While it was early believed that TGFs only coincided with lightning strikes, later observations have shown that some TGFs occur without visible lightning [23].

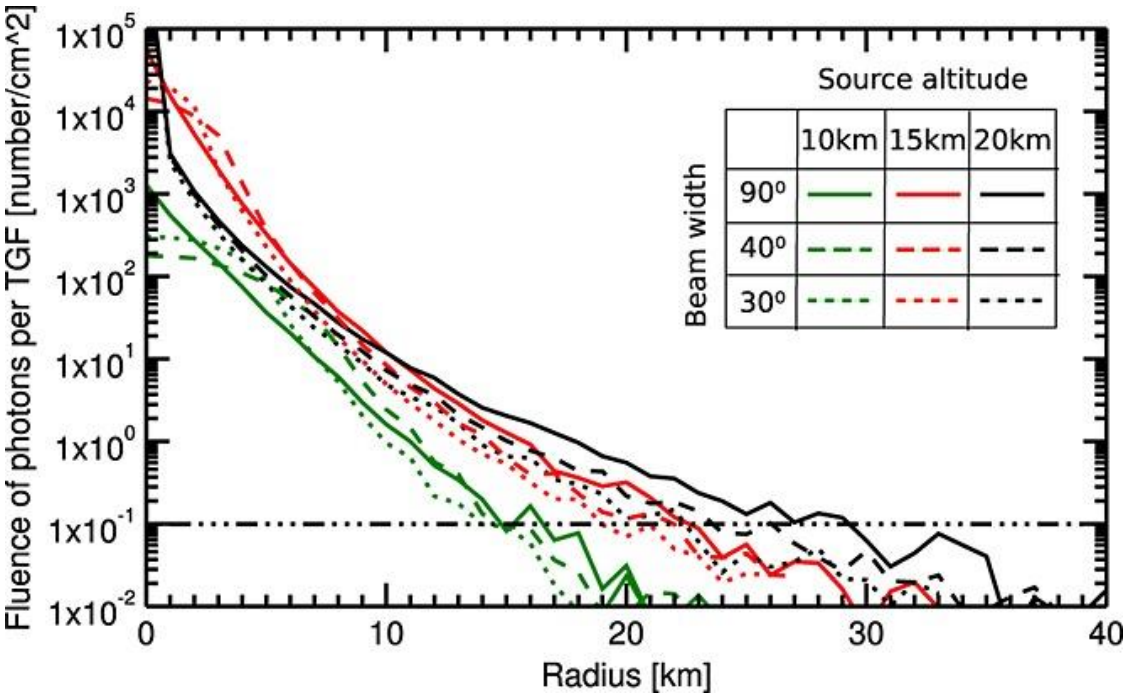


Figure 2-4 Simulated fluence of photons at 20 km altitude as a function of distance from the centre. Figure taken from [21].

2.3.2 Gamma-ray glows

Gamma-ray glows lasts between several seconds to several minutes and has photon energies of up to a couple of tens of MeV [16]. They are normally at an intensity of up to 2 orders of magnitude above the background radiation [16], but measurements of between 10 and 1000 times the normal background radiation have been reported [11]. Gamma-ray glows can normally be found at about 20 km altitude and below. Figure 2-5 shows the two gamma-ray glows observed in the ALOFT 2017 campaign.

It is believed that gamma-ray glows also are in large part generated by bremsstrahlung. Glows sometimes are recorded to abruptly stop, something generally thought to be linked with visible lightning strikes [16].

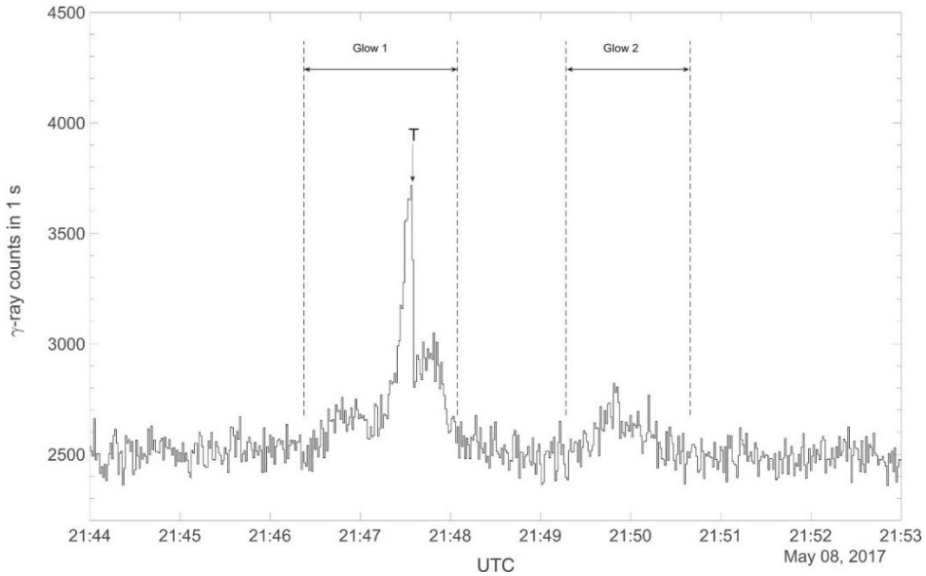


Figure 2-5 Graph showing the two gamma-ray glows observed by the ALOFT 2017 campaign.

2.4 Physics of high energy phenomena

2.4.1 Relativistic Runaway Electrons

The following section is based on [24]:

When an electron is subjected to an electric field, it will, like all charged particles, experience a force. If it is free to move this causes it to gain speed until the friction forces experienced by the electron equals the Coulomb force.

At lower kinetic energies, that is when the electron moves slowly, gaining speed causes the electron to come in proximity of more particles for it to be deflected by, slowing it down. This *inelastic scattering* translates into an increased speed causing increased experienced friction. At higher kinetic energies however the time it can interact with each particle decreases, allowing it to go straighter. This effect eventually overpowers the normal effect, making the experienced friction get reduced as the kinetic energy of the electron increases. If an electron in this situation experiences an electric field, it will accelerate freely. The electron has *run away*. As the electron gains speed eventually other effects will make it loose energy, such as *bremsstrahlung*.

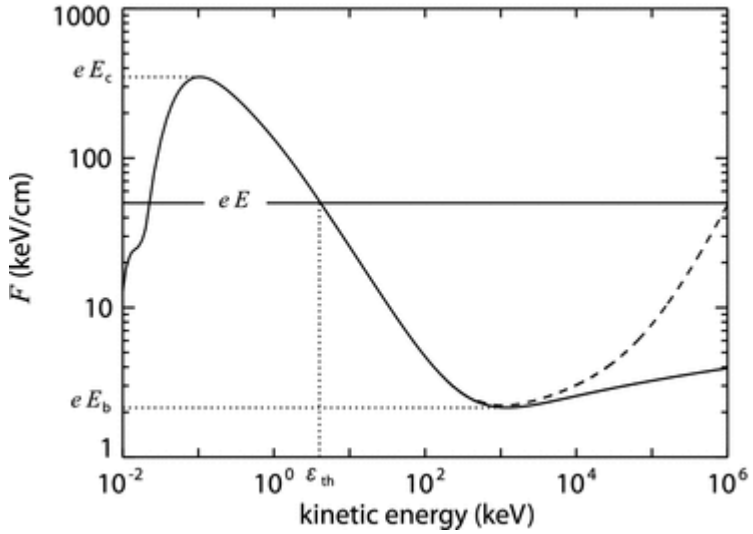


*Figure 2-6 The effective friction force experienced by a free electron moving through air as a function of kinetic energy. The solid curve shows only shows the effect from inelastic scattering, while the dotted line also includes the effects from bremsstrahlung emissions. The horizontal line shows the force experienced by an electron in a 5.0*10^6 V/M electric field. With this field electrons will run away if they gain kinetic energy $\varepsilon > \varepsilon_{th}$. E_c is the critical electric field strength required for any free electrons to run away while E_b is the so-called break-even field. [24]*

The curve in Figure 2-6 shows the effective frictional force experienced by a free electron moving through air as a function of kinetic energy. The solid curve shows only friction from inelastic scattering while the dashed curve is the sum of the inelastic scattering and radiative effects such as bremsstrahlung. The solid horizontal line shows the Coulomb force experienced by an electron in a strong electric field comparable to what found in thunderclouds. To run away the electron in this example needs an initial kinetic energy greater than the threshold energy, $\varepsilon > \varepsilon_{th}$.

External sources such as cosmic rays can *seed* an electron with the energy required to run away. An electron running away can also part some of its energy to a bound electron, knocking it free. If the newly freed electron has sufficient kinetic energy, it too can run away and potentially collide with another bound electron. This can cause an avalanche of runaway electrons, commonly called a *Relativistic Runaway Electron Avalanche*.

As the electric field cause all electrons to experience a force in the same direction (opposite direction of the field), the avalanche normally dies out as they eventually get outside the field. It is however possible to get secondary effects that go in the opposite direction. Radiation caused by the avalanche could move 'upstream' and knock an electron free with enough energy to reseed an avalanche. You could also get gamma radiation strong enough to cause pair production, making an electron-positron pair. The positron behaves identical to the electron but moves in the opposite direction due to its positive charge. This allows it to seed new avalanches the entire way back, in what is called *Relativistic Feedback Discharge*. Figure 2-7 shows- illustrations of all these effects.

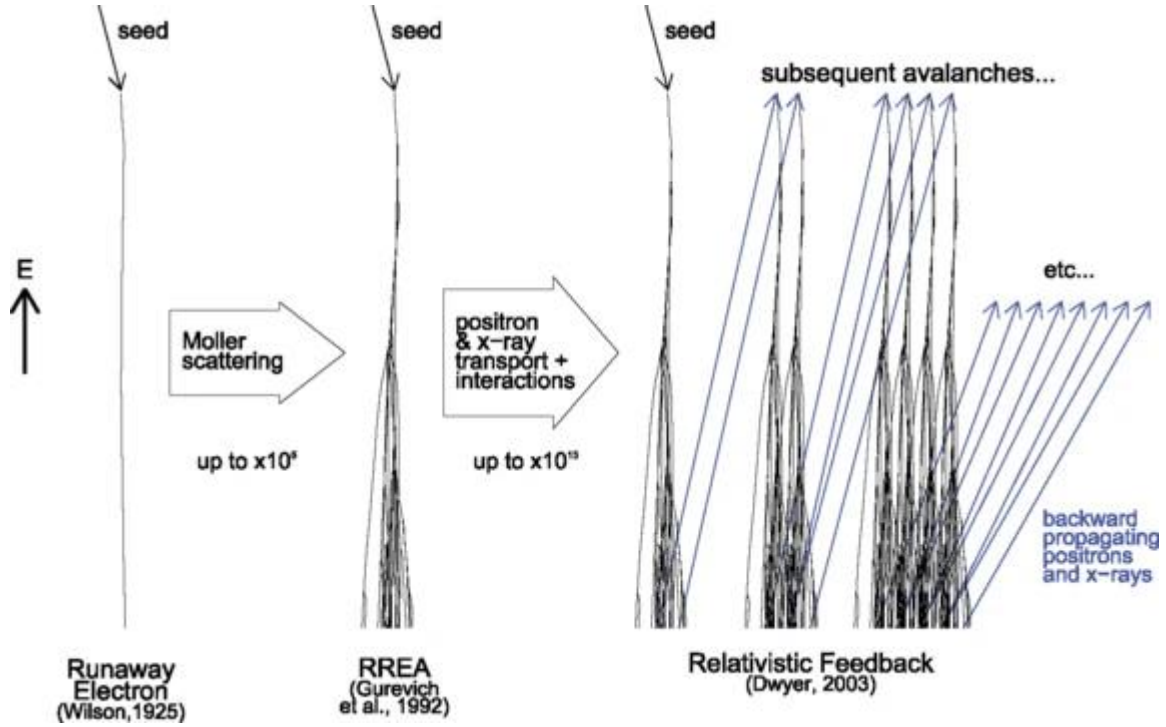


Figure 2-7 In an electric field a seed phenomenon can give enough kinetic energy to an electron to run away. A high energy runaway electron can through scattering cause secondary electrons to run away creating a Relativistic Runaway Electron Avalanche. With sufficient energy a RREA can create x-rays or positrons that travels 'upstream' potentially seeding new RREAs in a relativistic feedback loop. This can quickly drain the electric field of energy. [24]

2.4.2 Bremsstrahlung

The following section is based on [25]:

Bremsstrahlung is generally the name used for the radiation emitted when a charged particle deaccelerates due to moving close by a different charged particle. For electrons this can happen when it has enough energy to move through the electron shells of an atom and come in proximity of the nucleus. Its great mass and charge compared to the electron causing it to experience a significant acceleration, and subsequent lose some of its kinetic energy in the form of radiation as seen in Figure 2-8. How close to the nucleus determines both how much energy is radiated, and how likely it is to occur creating a continuous spectrum of variable intensity that can be recognised. A close 'flyby' causes a great columbo force in turn causing a great deflection of the electron and may cause a significant amount of its energy to radiate away as a gamma photon.

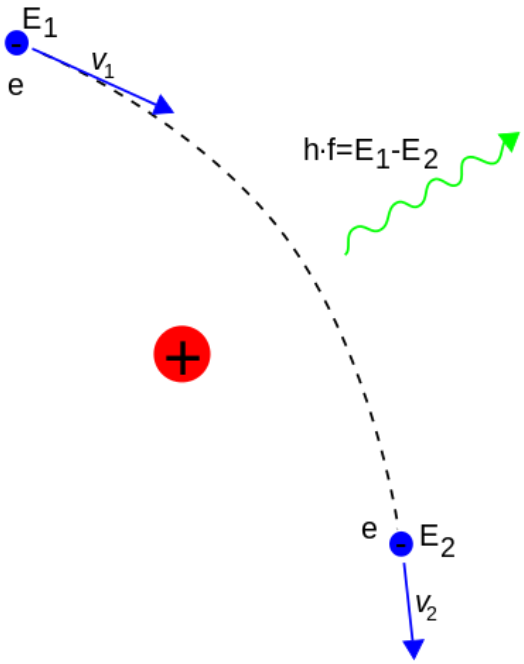


Figure 2-8 Bremsstrahlung emission caused by the electron accelerating, and its subsequent loss in kinetic energy. Illustration taken from Wikimedia Commons and is in the public domain.

3 ALOFT Systems

This chapter describes the updated ALOFT systems. ALOFT will from 2023 consist of the existing detector systems, as well as the new ones based on silicon PMs. As I only worked on the readout for the new detectors, it is the systems directly linked to my work that will be described here.

3.1 Overview

The 2023 ALOFT campaign system consists of four instruments that are built to detect photons and a shared physical capture card that individually pre-process the data coming from the detectors. The data is finally stored in a mass storage. Figure 3-1 shows a simple block diagram of the instrument. The old components used in the 2017 ALOFT campaign are coloured blue, and the new components in green. The mass storage is a general off-the-shelf storage.

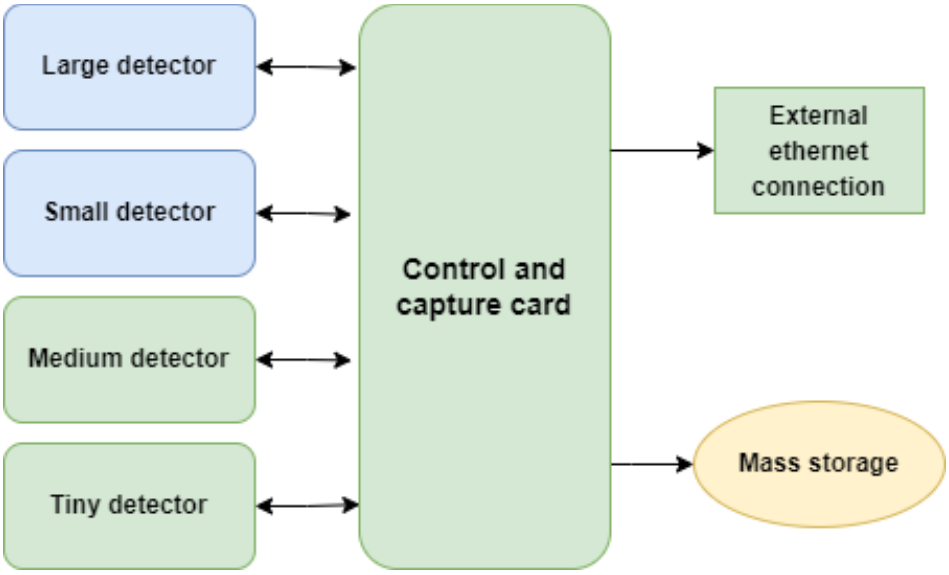


Figure 3-1 System level block diagram of the ALOFT instrument. The green boxes are changed or added between the 2017 and 2023 campaign.

Creating instruments for TGFs is a challenging endeavour, especially because of the range of incoming gamma photon fluxes. The estimates shown in Figure 2-4 tell us that the dynamic range in fluence, and as a direct result the flux, can vary by up to five orders of magnitude depending on the planes distance to the centre. By comparison gamma-ray glows are relatively easy to handle with their fluxes normally a few times above the background radiation. To handle the range of fluxes from the TGFs, ALOFT uses a range of detectors with different properties and sensitivities.

A detector becomes saturated when consecutive detections are indistinguishable from a single detection. Two of the factors determining how good a detector is at handling large fluxes are the duration of the signal generated by each detection and the physical size of the sensitive part of the instrument.

The duration is a factor in that a short signal pulse duration will allow multiple detections in the time it takes for a long signal pulse to be finished. How short a pulse can get is, however, limited by the properties of the sensors material, as well as requiring a higher sample rate to properly record the short pulse.

A smaller physical size causes the detector to be hit by fewer photons per unit of time, but has the disadvantage of having less volume for each photon to slow down in. Hence, small photons only

deposit parts of their energy in the sensor before continuing past it, which makes measurement of the photon energy effectively impossible.

For this reason, two new gamma-detectors are being designed to expand the range of measurable high fluences of the ALOFT campaign. The first is a tiny detector made to be able to count fluences in the order of 10k photons/cm^2, as opposed to 1 photon/cm^2 for the large detector and 1k photons/cm^2 for the medium detector. It is however way too small to reliably stop gamma photons, so only a photon count will be realistic. The medium detector can only handle about 50 photons/cm^2 and is large enough to read the energy. [17]

Comparing these numbers to the simulated fluence based on range found in Figure 2-4, tells us that the large detector can be expected to be saturated around 20 km radial from a TGF, the medium detector at about 7 km, and the tiny detector at about 1-2 km radial.

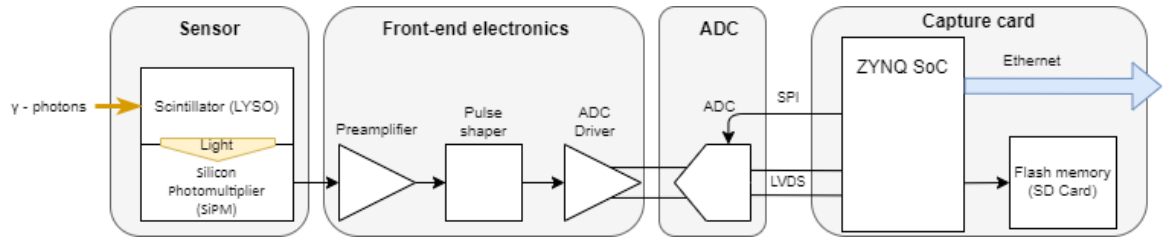


Figure 3-2 Signal path of the new detectors. The sensor converts a single gamma photon into a burst of light with is converted to an electrical pulse. The Front-end electronics amplifies, filters, and readies the electrical signal for the ADC. The ADC converts the electrical signal into a digital signal that is lastly pre-processed and stored in the Capture card.

The two new detectors can be roughly divided into four data-handling parts: The sensor, the front-end electronics, the *Analog to Digital Converter* (ADC) and the capture card. Figure 3-2 shows the signal path for one of the new detectors divided into the categories used here.

The sensor has as purpose to convert the phenomenon we are interested in, gamma photons in our case, into an electrical signal. The sensors chosen uses a two-step method of converting each high-energy gamma photon into a flash of visible light with a scintillator, and then convert that flash into a current using a Silicone Photon-Multiplier (SiPM). Details of this process and how the sensors of the two new detectors differ can be found in Chapter 3.2.

The term front-end electronics is in this context used for all the steps between the detector and the ADC. This includes converting the current signal the sensor outputs into the differential voltage signal the ADCs used use, amplifying the signal, and different filtering. In many cases front-end electronics include the ADC step, but I have chosen to distinguish them in this thesis, as part of the thesis is about controlling the ADCs. Further details are found in Chapter 3.3.

The Analog to Digital Converter does, as its name implies, convert analogue signals into a digital representation of the data. The two detectors use different ADCs, which both will be described further along with more on ADCs in general in Chapter 3.4.

The capture card includes everything that handles the data after the ADC and consists of some pre-processing before storing in mass storage. This all happens on a single physical card that is shared between the four gamma detectors, but each have its own path until it is stored. Further details are found in Chapter 3.5.

In addition to the data handling steps the final system also contains utility functionality such as power delivery. These systems are outside the scope of the thesis, and therefore mostly glossed over.

3.2 The Sensors

3.2.1 Scintillator

The information about the scintillator is based on [25].

When a gamma photon travels through matter it tends to interact with it and deposit some or all its energy. The main ways high energy photons can interact with matter includes:

- The photoelectric effect: When a photon interacts with a bound electron, it can give all its energy to that electron. This is called the photoelectric effect. At lower energies the photon may only have enough energy to cause excitation, but with sufficiently high energy it can instead free the electron, ionizing the atom. If the electron hit was in an inner shell, an X-ray emission can be created by the deexcitation of an outer shell electron. The free electron can also have enough energy to cause secondary (not as) high energy photons. See Figure 3-3 (a).
- Compton scattering: If the photon instead only gives part of its energy to the bound electron you get Compton scattering. In this case the electron will be knocked free (potentially causing secondary effects) while the photon is deflected at an angle with the rest of the energy. See Figure 3-3 (b).
- Pair production: If the electron is above 1.02 MeV it might instead interact with the nucleus and cause the spontaneous creation of an electron-positron pair. The pair split all the photon energy (except some minor recoil of the nuclei) and can cause secondary effects if they are fast. The positron will also cause secondary effects if slowed down enough to collide and annihilate with a nearby electron. See Figure 3-3 (c).

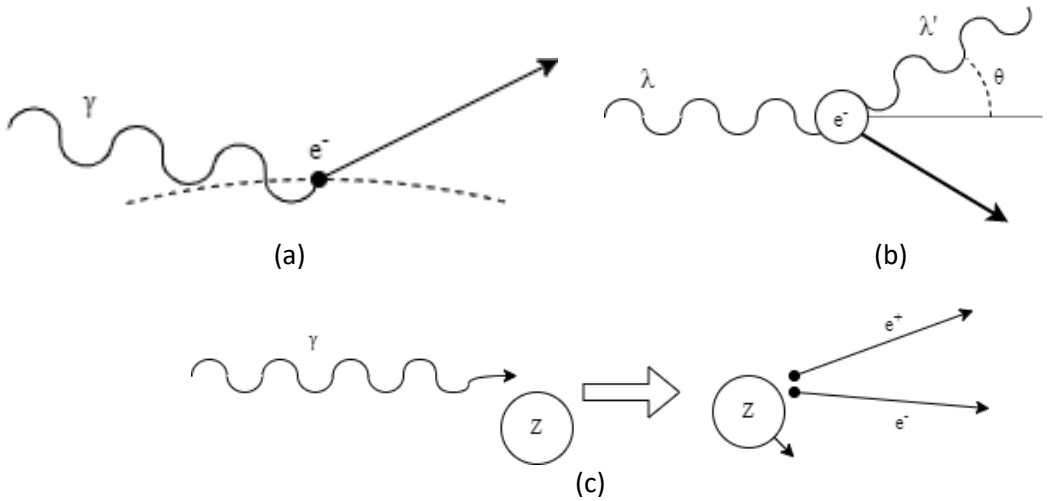


Figure 3-3 Illustrations of the three main ways high energy photons interact with matter. Not to scale. (a) The photoelectric effect. (b) Compton scattering. (c) Pair production.

Each of these creates new electrons or photons that may cause further interactions spreading the original photon energy. When the energy gets diluted enough it will start causing a number of electron excitations proportional to the original photon energy. In most materials these excitations will simply

end up as heat, but some materials may instead scintillate, meaning some of the energy instead escapes as light.

A scintillator is a material with the property of creating a flash of light of intensity proportional with the energy deposited in it. An inorganic scintillator, which is the type relevant here, is made by carefully doping a crystal with a desired bandgap between the conduction and valence band. This creates narrow intermediates bands just below the conduction band (activator excited state) and above the valence band (activator ground state). If an excited electron first decays from the valence band to the activator excited state, then decays to the valence band via the activator grounded state, a photon with a very specific and known energy/wavelength will be emitted. This photon will have insufficient energy to excite an electron directly from the valence band to the conducting band while the doping, allowing it to travel through and out of the crystal material mostly unhindered. Figure 3-4 illustrates this process. The number of photons of the specific energy that escape the crystal will be proportional with the number of excitations, which is proportional with energy deposited the crystal.

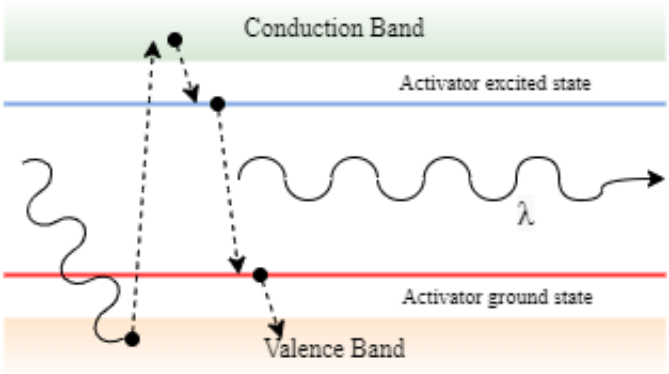


Figure 3-4 Simplified illustration of a scintillation. Something (here a photon) causes an electron to be excited and jump from the valence band to the conduction band. It then quickly decays to the thin activator excited state caused by controlled doping of the material. When the excited photon decays to the activator ground state it emits a photon with a very specific energy that is too low to excite any electrons in the valence band. It can therefore continue almost unimpeded out of the material.

The scintillator has a couple of key characteristics.

- Light output. The light output is a measurement on how many photons can be expected to be emitted per unit of energy deposited in the scintillator. It is usually measure in photons/keV or photons/MeV.
- Decay time. The duration and intensity can be modelled as a near instant rise time with an exponential decay as given by the specific scintillator type.
- Peak emission wavelength. What wavelength the output primarily consists of. The goal is to have a wavelength that the next step can efficiently convert to a signal.
- Density. The denser a material is, the higher likelihood it is that a high energy photon interacts with it. This increases the odds for each photon to deposit some or all its energy in the material for a given thickness. Most scintillators naturally have higher density than alternative methods of radiation detection.
- Effective atomic number. Most of the high energy interaction between atoms and both electrons and photons are dependent with the charge of the nucleus. A higher atomic number causes increased probability and efficiency of the interactions, making a high average (effective) atomic number in the scintillator desirable.

The scintillator used in the new ALOFT detectors are *Lutetium–yttrium oxyorthosilicate* (LYSO) crystals, the same as in the small detector. It has a comparably high light output of 25 photons/keV, a decay time of 40 ns and a peak emission wavelength of 410 nm. Its density is 7.15 g/cm^3 and its effective atomic number is 65. By comparison BSO has a decay time of 300 ns, making LYSO more suited for higher fluxes [26].

The tiny detector uses a LYSO cube of 3mm cubed, while the medium detector uses a LYSO cube of 5cm cubed. This enables the tiny detector to handle higher fluxes with its tiny cross section at the cost of not reliable being able to absorb all the photon energy. Meanwhile the medium detector is large enough to absorb the energy of the photons.

3.2.2 Silicon Photomultiplier

The following section is based on [27]:

A Silicon Photomultiplier (SiPM) is a component that converts a flash of light, such as from a scintillator, into an electrical current. It consists of an array of small, sensitive elements called microcells connected in parallel. Each microcell is essentially a *Single Photon Avalanche Diode* that creates an electrical pulse when triggered. This pulse is independent of what triggered it, but by summing up all the microcells in parallel you get an output proportional to any incoming bursts of light, as seen in Figure 3-5.

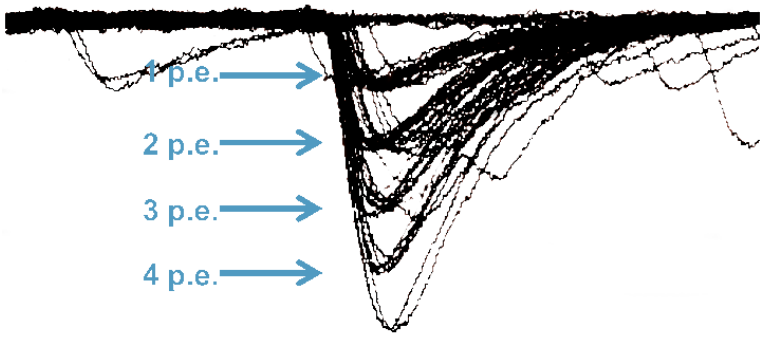
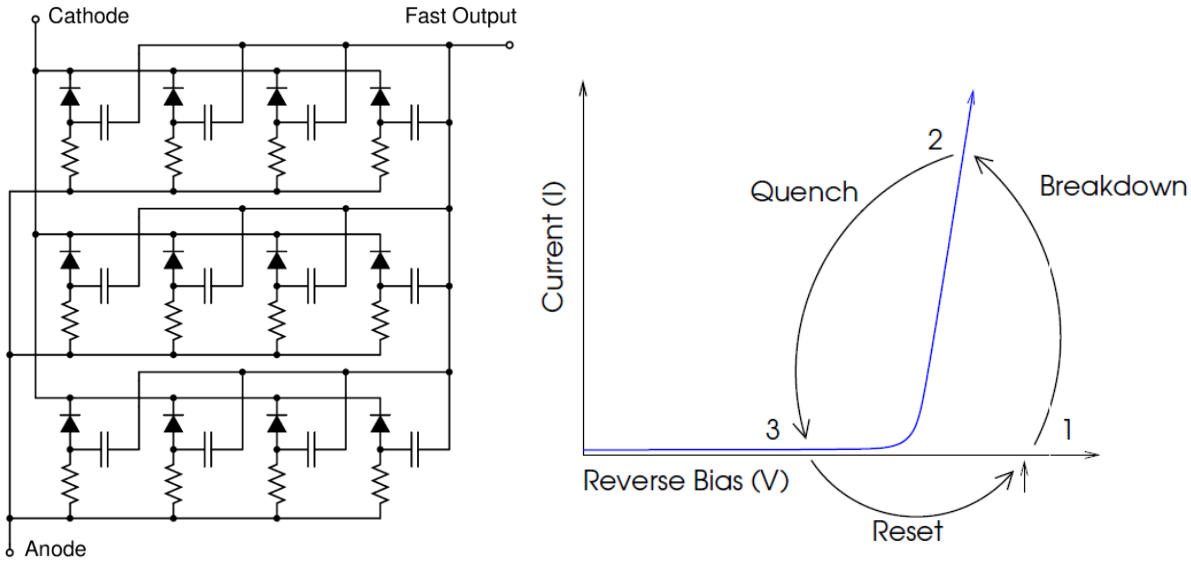


Figure 3-5 Each individual triggered microcell will add its pulse to the signal, making a single detection have an amplitude equal a multiple of the microcell pulse amplitude. [27]

Each microcell consists of a photodiode in a strong reverse bias. Thanks to the reverse bias you have a depletion area without any mobile charge carriers, but when an electron gets excited it creates an electron-hole pair. The strong electric field from the voltage causes the electrons to gain enough energy to excite new electrons, causing secondary excitations. The resulting avalanche of electrons causes the diode to be conductive, which can be detected. As this effect is self-perpetual, you also need a way to stop or 'quench' for it to be reused.

By putting a resistor in series with the diode an increasing current causes the voltage drop over the resistor to increase and the voltage drop over the diode to decrease. If the voltage across the diode is too small it is incapable of sustaining the avalanche, quenching the current. The microcell then uses some time to return to its initial state, depending on the effective capacity it sees and the resistor value. This recovery is visible on the output as an exponential decay. Figure 3-6 (a) shows a simplified example of how the microcells are built and stacked together, while Figure 3-6 (b) shows the activation cycle of the photodiode in the system.



(a) (b)

Figure 3-6 (a) Shows a simplified circuit schematic of a SiPM with only 12 microcells. Each cell consists of a photodiode in reverse bias (Single Photon Avalanche Diode), a quenching resistor and a capacitor connected to the Fast Output. When triggered the photodiode will start an avalanche and start conducting through the cell. As the current rise the voltage drop over resistor increases and the voltage over the diode is reduced eventually causing the avalanche to be quenched. The cell then goes through a recharge cycle as the charge trapped by capacitive effects are drained to the anode. The Fast Output uses the capacitor to effectively output the derivative of the normal output, giving a short spike at the initial avalanche followed by undershoot as the system recharges. There is as of writing no plans to use the Fast Output. [27]

Figure 3-6 (b) Shows the breakdown, quench and reset cycle of a Single Photon Avalanche Diode. [27]

The Photon Detection Efficiency (PDE) is a statical probability that an incoming photon induces an avalanche. It is dependent on the wavelength of the incoming photon, the voltage across the photodiode above the breakdown voltage (overvoltage) and physically how much of the area of the detector that is sensitive to light (the fill factor).

The likelihood of an incoming photon creating an electron-hole pair is dependent on the photon wavelength, as illustrated in Figure 3-7. Note that the wavelength of 410 nm emitted by the LYSO is very close to the peak responsiveness.

The breakdown voltage is the minimum voltage at which the photodiode has the capability of creating an avalanche, and any voltage above that is called overvoltage. By increasing the overvoltage, the chance that any electron-hole pairs create an avalanche increase and increases the number of electrons per avalanche. It does however also increase the odds of spontaneous thermal excitations and their chance of causing avalanches, something that creates noise.

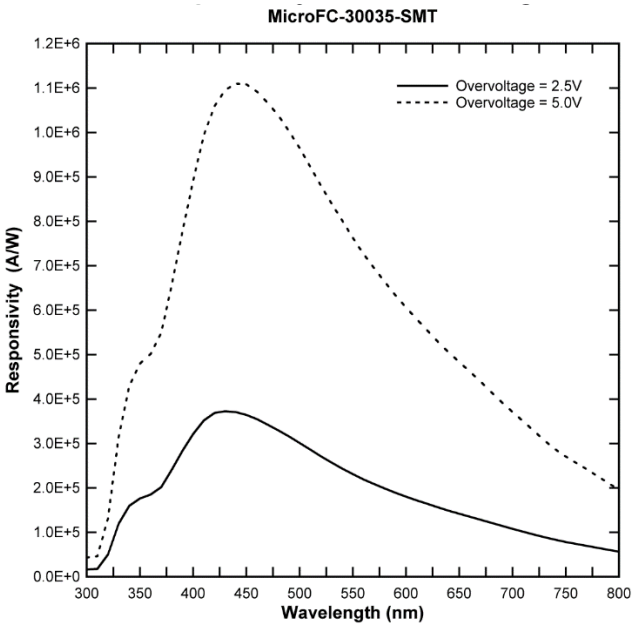


Figure 3-7 The responsiveness of the SiPM as a function of wavelength. The graph also shows how responsiveness and therefore PDE increases with overvoltage. [27]

The percentage of the total area sensitive to light is called the fill factor and is specific to each SiPM. The reason for this is that all the microcells require some spacing to isolate them optically and electrically. You can increase the fill factor by creating fewer, larger microcells, but this increases the capacitance of each microcell, making the recovery time longer.

Since there is a finite number of microcells on a detector, the higher the flux of incoming photons are the higher the odds that several photons hit a single microcell. This results in a limited dynamic range with growing nonlinearity depending at the top. The relationship between the number of incoming microcells fired can be expressed as

$$N_{fired} = M\left(1 - e^{\left(-\frac{PDE*N_{ph}*O}{M}\right)}\right) \quad (1)$$

where N_{fired} is the number of microcells fired, M is the total number of microcells and $N_{ph}$is the number of photons emitted by the scintillator. O is the optical coupling between the SiPM and the scintillator, and must be found experimentally on the physical component. Optical grease is commonly used to improve O.

The tiny detector uses a Onsemi J-series 30035, which has an active area of $(3.07\text{mm})^2$ with 5676 microcells. Its PDE is between 38 and 50 depending on overvoltage, and it has a recharge time of 45ns. [28]

The medium detector uses an Onsemi Array-J-30035-64P-PCB, which consists of 64 J-30035 sensors arranged in an 8 by 8 array[29]. Each sensor can be read individually as a pixel, but in our case the outputs will all be summed together. This is because light from the LYSO will spread among all the pixels and will not give any extra information.

3.3 Front-end electronics

The following section is in parts based on [30].

3.3.1 Preamplifier

The preamplifiers main task is to convert and amplify the weak current signal from the SiPM into a voltage signal strong enough to be further processed. In addition, the design that is used effectively functions as an integrator, ensuring that the contribution from all the microcells that are fired while the scintillator glows are combined.

An additional function of the preamplifier for the medium detector is to sum up the signal from all the pixels, as briefly mentioned above. This happens after each pixel is preamplified before joining into the next step.

3.3.2 Pulse shaping

After being amplified the signal is filtered with the goal of removing as much noise as possible while keeping the information contained in each pulse, or in other words improve the Signal to noise ratio (SNR).

A common approach to do this is to send the signal through a bandpass filter. You then not only filter out high frequency noise, but you also remove low frequency effects such as baseline drift. A candidate for the filters used in the detectors are based on a CR-$(RC)^n$ configuration, sometimes called a Gaussian shaper when n is greater than one [25]. It consists of a high-pass filter followed by one or more lowpass filters, commonly with the same time constant for all the stages[25]. Increasing the number of RC steps will cause the pulse shape to increasingly resemble a mathematical Gaussian curve, hence the name. One of the key benefits of doing this is that the duration of the peak lasts longer, allowing more time for an ADC to register the value. As of writing it seems that n=2 has the best properties for our use [30].

The second candidate for a filter stage is mostly similar but uses a $(CR)^2$-$(RC)^2$ configuration instead. The extra CR step essentially functions as a differentiator and makes the resulting peak shorter at the cost of having an undershoot, something that could be useful for the tiny detector.

Figure 3-8 shows the pulse shapes of the two candidates, normalized to have a max positive amplitude of 1. Here you clearly see that duration of the pulses, the time it takes for them to return to the baseline, is about equal while width of the peak of the $(CR)^2$-$(RC)^2$ pulse happens faster and is narrower. The narrow peak can be an advantage for uniquely identifying peaks that happen close in time but has the disadvantage of requiring higher sample rate to sample it correctly. Further details on this are found in Chapter 5.

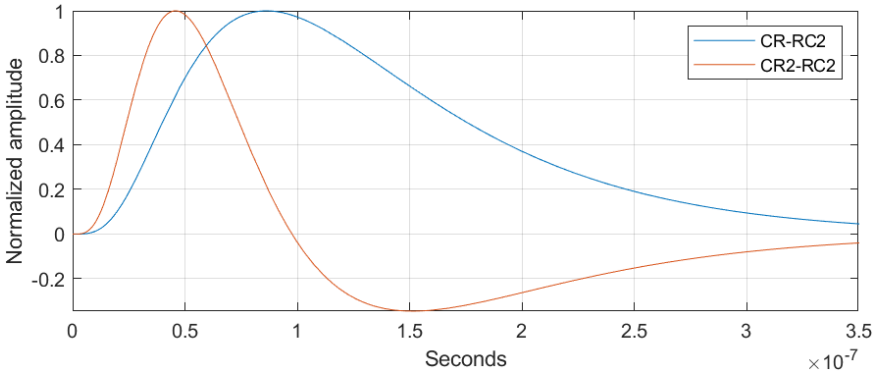


Figure 3-8 The pulse shape candidates normalised to have a max value of 1. Note that the peak-to-peak amplitude of the $(CR)^2$-$(RC)^2$ pulse on the figure is about 1.345.

3.3.3 ADC driver

The ADC driver has the task of readying the signal for the ADCs and consists of converting the signal into a differential signal and scaling it so that the signal has the same range as the ADC input.

In addition, the design used also functions as a lowpass filter, something that is important to do before sampling a signal with an ADC. This is because when sampling with a finite sampling rate, such as in an ADC, any frequency above the Nyquist frequency (half the sampling rate) is indistinguishable from lower frequencies causing aliasing. Sending a signal through a lowpass filter before an ADC is therefore essentially necessary. It is however not that relevant for our signal, as the filtering stage already have removed the higher frequencies.

As a last step, the differential signal also goes through diode clippers to ensure it does not go outside the wanted voltage limits. This is to protect the ADC sensors from damage. While in theory not needed in a designed system, it can negate unforeseen surges.

3.4 Analogue to Digital Converters

An ADC works by determining where along its input range the input voltage is, then assigning a digital value based on that. How many samples it takes every second is called its sampling frequency and is commonly measured in SPS (*Samples Per Second*) or MSPS (Mega-SPS). Its resolution tells how many distinct levels it can distinguish between and is commonly measured in how many bits each individual sample is.

3.4.1 Detector specific ADCs

The tiny detector uses an ADS5404 12bit 500MSPS ADC from Texas Instruments [31]. It takes an input of 1.0 V peak-to-peak, and outputs a digital *Double Data Rate* (DDR) *Low Voltage Differential Signal* (LVDS). Its high sample rate gives it a great temporal resolution needed to distinguish between photons when exposed to a high flux. Figure 3-9 shows a simple block diagram of the ADS5404. While it has two independent signal channels, only one will be used.

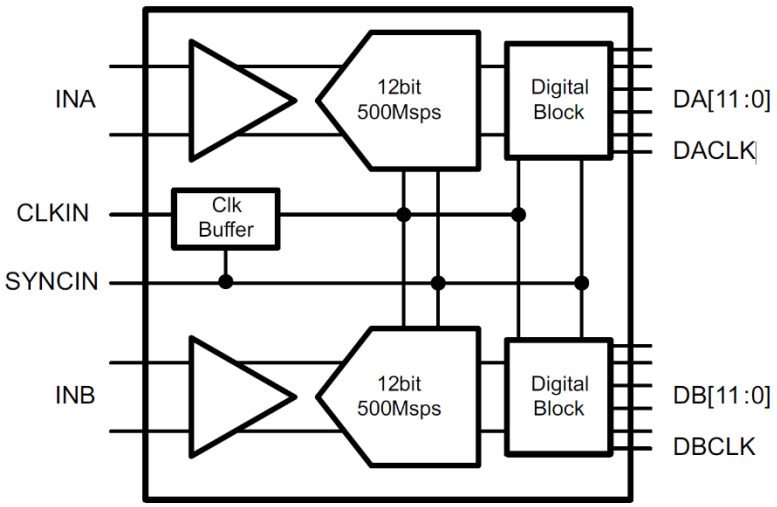


Figure 3-9 Block diagram of the ADS5404. [31]

The medium detector uses an AD9257 14bit 65MSPS ADC from Analog Devices [32]. It takes an input of 2.0 V peak-to-peak, and outputs a digital *Double Data Rate* (DDR) *Low Voltage Differential Signal*

(LVDS). Here a higher bit resolution makes it more capable of determining the energy of each individual photon. Figure 3-10 shows a functional block diagram of the AD9257. Again, only one signal channel will be used.

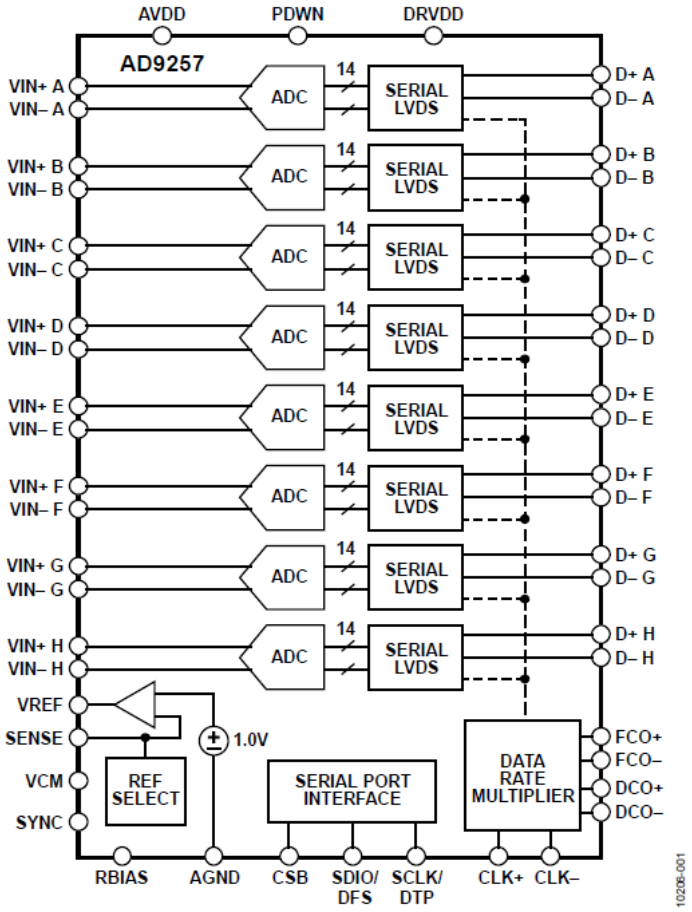


Figure 3-10 Functional block diagram of the AD9257. [32]

Both ADCs has control registers that can be accessed through a *Serial Peripheral Interface* (SPI). The details of these will be further discussed in Chapter 4.2.

3.5 Capture card

3.5.1 System on a Chip

Prior to giving a design description of the capture card some background terminology is useful.

While improved manufacturing technics allow increasingly denser transistors in integrated circuits, the pins out connecting each chip stay mostly the same. This is in part due to it being harder to solder or otherwise connect individual small pins to a wire, without breaking or shorting them.

One way to circumvent this problem is to have multiple different components on the same chip. Then you can connect them together in the factory using the same methods used to create each individual component. When multiple different components are connected into a system on a single chip is called a *System on a Chip* or an SoC. While you lose the customisability to mix and match individual components, the short physical distance between components and subsequent short wires allows for faster communication rates, reduced chance of external interference and reduced power consumption.

The investment costs of creating a custom SoC can be very high. As such there commonly needs to be a minimum volume or critical performance requirement for designing a new SoC. A good trade-off can be to create a less specialised SoC capable of handling a wide variety of tasks. One way to achieve this is to use *Field Programmable Gate Array* (FPGA) as components, essentially using the same arguments comparing FPGAs to application-specific integrated circuit but with the added SoC benefits.

A SoC with an integrated FPGA is a good option for use in the ALOFT detectors as the FPGA should be fast enough to handle any raw data processing, you can run software with all its benefits on the CPU and its quantity (one) makes creating a custom SoC an unsound action. Its flexibility also allows rapid design changes with the hardware in hand, and easy reuse of the SoC to other projects in the future.

3.5.2 ZYNQ-7000 SoC

The following section is based on [33].

The core of the capture card consists of a ZYNQ-7z030-FFG676, which is part of the Zynq-7000 SoC family from Xilinx. The ZYNQ SoC consists of a dual-core ARM Cortex based processing system (PS) and a FPGA programming layer (PL). As a general-purpose CPU, the ARM Cortex requires an operation system to function, which can be chosen by the user.

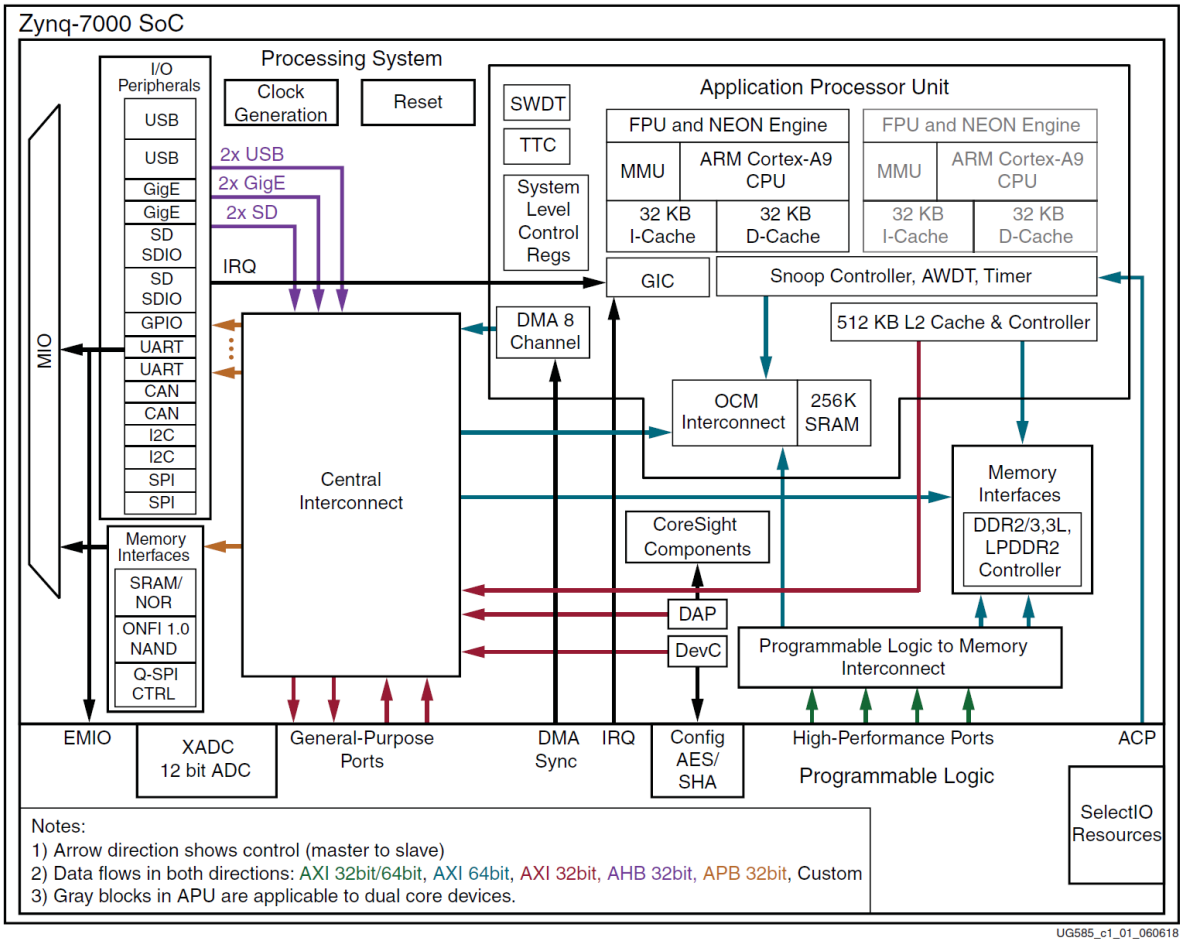


Figure 3-11 An overview of the functional blocks of the Zynq-7000 SoC [33]

Figure 3-11 shows an overview of how the PS is built up and how the PS and PL are connected. Some of the features that is worth noting are:

- The System-Level Control Registers. Much of the behaviour of the PS is controlled or interfaced by writing and reading these registers.
- The Multiplexed I/O Interface (MIO) and the Extended Multiplexed I/O Interface (EMIO). The PS contains a series of different I/O controllers that are multiplexed to a limited number of I/O pins (the MIO). The system also enables the controllers to be interfaced with the PL via the EMIO. What controller is connected to what is controlled via the System-Level Control Register.
- SD card controllers. The current plan is to use a SD Card as storage medium, and as such the data will be sent to it using this controller. The controller supports SD high-speed and SD high-capacity standards.
- Serial Peripheral Interface (SPI) Controller. The ADCs has control registers that are interfaced using SPI. More details will be given in Chapter 4.
- Gigabit Ethernet Controller. Ethernet is the planned protocol used to give live updates of the system.

While not visible from the figure, the PL layer is compatible with the DDR LVDS that the ADCs output. Also relevant for the thesis is that the 7z030 has 265 BRAM (Block Random Access Memory) available as a resource to the PL, each 36Kb for a total of 9540Kb.

3.5.3 PYNQ

The following section is based on the documentation found in [34].

PYNQ is an open-source project from Xilinx that provides a Jupyter-based framework with Python APIs combined with a Linux operating system for Xilinx platforms.

Jupyter Notebook is a separate open-source project that consists of a web application where one can edit and see the output of code run on a Jupyter kernel. This enables one to create servers using the Jupyter kernel and then manipulate and run code on it from a browser. In the context of the capture card, this means that if a Jupyter kernel is running on the PS and it is connected to a network with a known IP address, you can easily run code from an external computer.

PYNQ expands on Jupyter in several ways. First it integrates Jupyter with a Linux operating system such that the Jupyter kernel is up and running when booting the Zynq. Next it contains Python libraries that contains premade functionality to configure and use the PS functionality. It also enables the use of hardware libraries, or *overlays*. Overlays are FPGA designs with a Python API that allows them to be dynamically loaded and interfaced via Python code running on the Jupyter kernel. Selecting a new overlay sends the new FPGA configuration to the Zynq, which then flashes the FPGA with the new configuration in the overlay. This allows for reconfiguration on the fly of the FPGA.

The choice to use PYNQ happened during the work on this thesis and was preceded by the operating system FreeRTOS (Free Realtime Operating System). FreeRTOS is an operating system that is primarily designed to run on microcontrollers and other small systems, and as such has very low memory-, computing- and storage overhead [35]. FreeRTOS uses the C program and has no official drivers, requiring more work to properly interface with the Zynq.

The combination of a ready to use OS, preinstalled drivers, effectively finished interface for real-time monitoring over ethernet and general ease of use, made PYNQ the OS of choice.

4 ADC configuration

This chapter documents the work done towards verifying that the ZYNQ SPI controller can be used to access and configure the ADC control registers of the two new detectors.

4.1 Objective

The ADCs in the system boots in a default configuration mode when powered. While in many cases this is sufficient, the ADCs has extra functionality that can be accessed by communicating with their control registers. The control registers for both ADCs are interfaced using a three-wire variant of *Serial Peripheral Interface* (SPI). The specific communication protocol differs between the two ADCs.

The objective of this part of the thesis is to verify that the built-in ZYNQ SPI controller can be used to interface with the ADC control registers.

The work could be separated into three discrete steps: (1) Creating a small converter capable of interfacing between the four wire SPI available from the ZYNQ to the three wire SPI used by the ADC controllers. (2) Create a testbench module that can be used to simulate and verify that communication between the ZYNQ and the control registers will have the expected outcome. This should be done in a structured way. Step (3) is verifying if the ZYNQ SPI peripheral in the processor system can be used for the communication with the ADC configuration interface.

To verify that the suitability of the PS SPI peripheral the ZYNQ FPGA was set up in barebone mode, i.e., without any operative system. This simplifies the stack and thereby reduces the number of potential sources of error. To use this, drivers that interface with the System-Level Control Registers on the ZYNQ are required. The drivers ensures that correctly formatted data to the PL via the PS SPI controller is sent. Testability and verification were prioritized throughout this work, so a fair amount of time was devoted to making a testbench for the ADC model and the interface.

4.2 Serial Peripheral Interface

The following section is based on information given in [36].

The Serial Peripheral Interface (SPI) is a synchronous serial communication interface specification that is commonly used for short distance communication in embedded systems. SPI has a master-slave architecture and is normally used with four wires allowing simultaneous two-way (full duplex) communication. Figure 4-1 (a) shows an example of normal SPI setup.

The four wires are a signal clock (SCLK), a data line transmitting from the master to the slave (master out slave in, MOSI), a data line transmitting from the slave to the master (master in slave out, MISO) and lastly an active low enable signal that needs to be active for the slave to receive or transmit (chip select bar, CSB). Many SPI master controllers have multiple chip-select signals, enabling multiple slave devices to be independently controlled while sharing the data and clock signals as shown in Figure 4-1 (b). The names of the different signals often vary between different vendors.

Two additional settings sometimes used with SPIs are signal clock polarity and signal clock phase. The polarity determines whether the clock is active high or active low, while the phase determines whether the bit should be read on the leading or trailing clock signal change. The most normal is that the clock is active high and reads on rising edge (leading edge for active high polarity), which is also the settings used in this thesis.

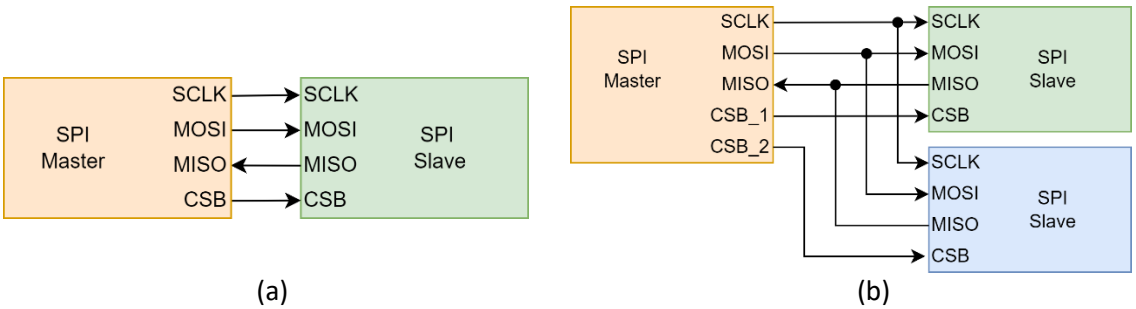


Figure 4-1 (a) Shows a normal four wire SPI setup with one master and one slave, as well as the signal directions of the wires. Figure 4-1 (b) shows how one master can control multiple independent slaves in parallel.

A transmission using SPI, assuming normal polarity and phase, starts by the master driving CSB low and the slave and master driving their first bit to the data lines. Then SCLK starts tick, and the value of the data lines is recorded every rising edge. A transmission is normally terminated by the CSB going high. Each device requires a minimum setup time to drive the correct data signal, as well as a minimum hold time required to read a data signal. As these timings tends to be symmetric it is common to express them as a maximum SCLK frequency.

There are several variants of SPI available. One common variant is to combine the two data lines into one that transfers data both ways, in this thesis referred to as SPI3. In this thesis this line is called Serial Data Input/Output (SDIO) and causes the transmission to be only half-duplex. Figure 4-2 shows a standard SPI3 setup. Both the ADCs discussed in this thesis (ADS5404 and AD9257) use SPI3 to interface the control registers and are in slave modus. To avoid floating signals on the control register both ADCs use pulldown resistors on the SCLK and SDIO wire and a pullup resistor on the CSB wire. [31] [32]

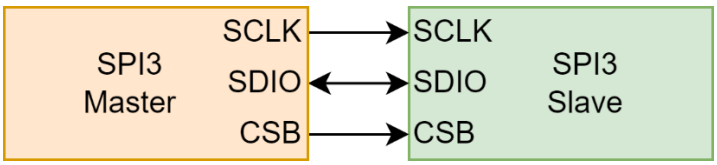


Figure 4-2 Normal SPI3 setup with signal clock (SCLK), Serial Data Input/Output (SDIO) and Chip Select Bar (CSB)

4.2.1 ZYNQ SPI Controller

The information is obtained from [33].

The ZYNQ has two independent SPI controllers, each with its own set of control and status registers. The I/O signals from the controller can be routed through either the MIO pins (pins on the chip) or through the EMIO pins (connection between the PS and PL). Figure 4-3 shows a block diagram of one such controller.

Some of the notable options are whether it should be in master or slave mode, and what phase and polarity it should have. When in master mode three different chip select signals can be controlled, shown as SS (Slave Select) in the figure. This enables the control of both ADCs using only one SPI controller.

Data to be sent and received is buffered in a transmission FIFO (TxFIFO) and a reception FIFO (RxFIFO) respectively, each 128 bytes deep. The signal clock from the master is derived from a 200MHz reference clock that is divided by 4, 8, 16, 32, 64, 128 or 256. When routing through the EMIO the clock can max 25MHz.

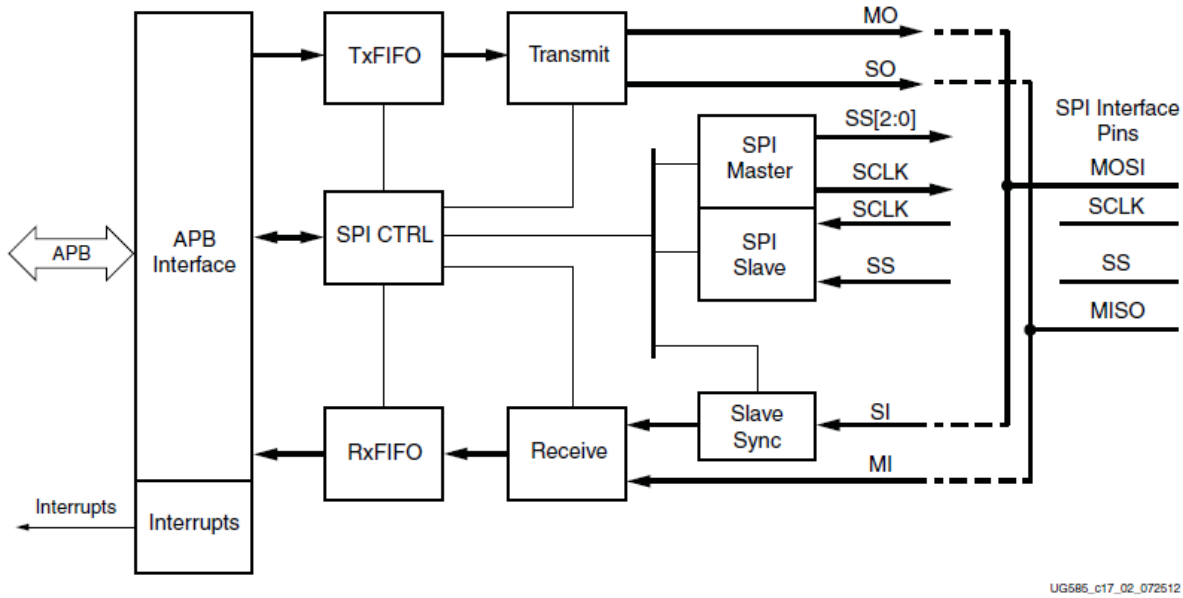


Figure 4-3 Block diagram of a Zynq SPI controller. [33]

4.2.2 SPI to SPI3 converter

Since the Zynq uses four wire SPI while the ADC control registers uses SPI3, a conversion is required between them. This can most easily be done by routing the signals through and converting them in the PL. As the SCLK and CSB is unchanged, the only conversion is to join the MISO and MOSI signals into the single SDIO signal. The problem of this conversion comes from the fact that the MOSI always drives a signal and SDIO will drive a signal when the slave is transmitting. Luckily this can be solved using a tristate buffer and a pulldown resistor. A block diagram of such an implementation is given in Figure 4-4.

Using this configuration, a master transmitting '1's will open the tristate buffer causing the SDIO to go high (overpowering the pulldown resistor), while '0's will cause the tristate buffer to go high impedance allowing the pulldown resistor to dominate. If the slave is transmitting, MOSI is set to '0' and MISO will read transmitted message.

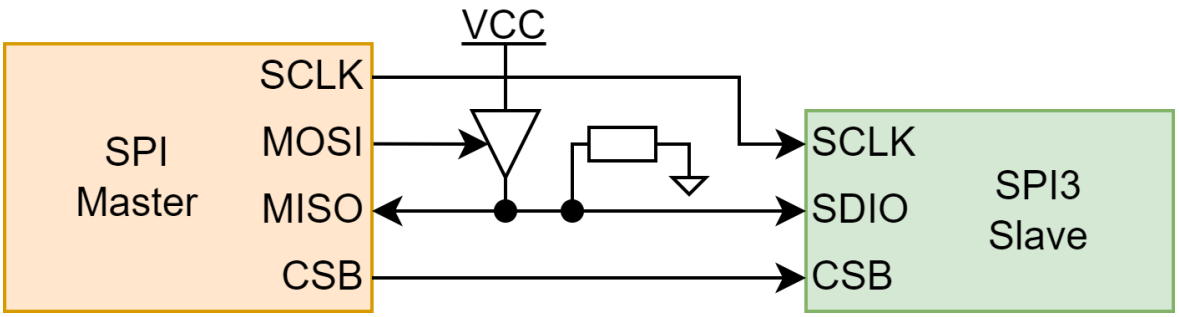


Figure 4-4 A connection diagram for the SPI to SPI3 converter. The triangle indicates a tristate resistor ('1' when MOSI is '1', 'Z' when MOSI is '0'). The rectangle between the MISO/SIDO line and ground symbolise a pulldown resistor.

Since the ADC already has a pulldown resistor on the SDIO pin, only the tristate buffer is required to do the conversion. This is implemented in VHDL, utilizing the tristate buffer in the IO cell of the FPGA.

4.3 The ADC controllers

The information about the ADS5404 and the AD9257 is taken from [31] and [37], respectively.

4.3.1 ADS5404 interface

The main structure of the ADS5404 controller is a register array with 7 bit addresses and 16-bit data. Only a limited number of the registers are noted to have a functionality in the datasheet. Different addresses have predefined functions, and by writing to it you can configure the ADC behaviour. There are also addresses that the ADC controls that can be read to collect information, such as an internal temperature reading. The registers controlled by the ADC tends to be read only.

Figure 4-5 shows the timing diagram and transmission syntax when communicating with the control register. Note that in the figure CSB is named SDENb (Serial Data Enable bar), but otherwise functions the same. To start the transmission the CSB is set low before starting to cycle the signal clock. Each transmission is 24 bits and CSB needs to return to high before a new transmission can start. The ADS5404 has a max SCLK frequency of 20 MHz, meaning that ZYNQ must run the SCLK at 12.5 MHz or lower.

The first bit (Read/Write-bar, RWB) states whether the transmission should write to a register or read from it, with a '0' being write and '1' being read. The next 7 bits is the register address, with the most significant bit first. The last 16 bits is the content of the register, most significant bit first. The sender of this data is dependent on the first bit, with the ADC returning the register content if the first bit indicated read ('1') and the master sending new content to register if the first bit indicated write ('0'). A highlight of a couple of registers and their function is given in section 4.3.2.

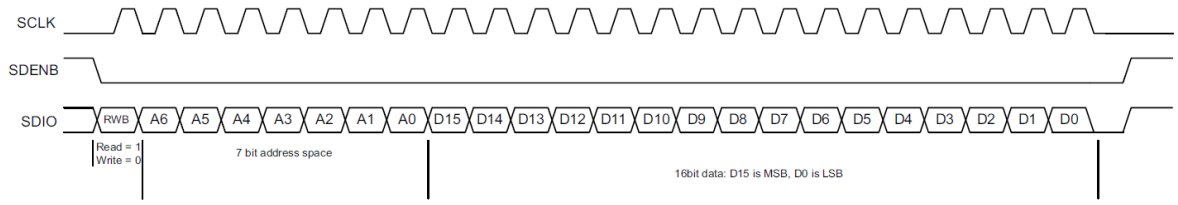


Figure 4-5 The ADS5404 SPI syntax and timing diagram. The first bit (RWB) informs the slave if it should receive data or return data while the next 7 bits (A6-A0) is the register address. If RWB was '0' the last 16 bits (D15-D0) is sent by the master to override the current content of the indicated register, while if RWB the ADC returns the current register content. [31]

The ADS5404 does have a couple of additional pins used for control. The first is the ENABLE pin, which is an active high chip-enable pin that when set high it will power down the entire chip It contains an internal pullup resistor, requiring it to either be controlled or grounded for the ADC to work. The ADC also has a SRESETb active low pin that resets the chip and returns all the control registers to the default vales on falling edge. It has a pullup resistor and is asynchronous.

Finally, the ADC has a SDO (Serial Data Out) pin that can used as a MISO enabling the interface to use four pin SPI if the correct control register address is set. As the AD9257 requires SPI3 and an SPI to SPI3 converter is easy to implement in the PL, this setting is not of interest in our use case.

4.3.2 ADS5404 control register features

One of the most relevant features of the ADS5404 control registers for the project is the ability to make the ADC output a selection of predetermined test patterns. This is very useful to verify that all connections and the signal handling is correct with the physical components.

Other noteworthy features include:

- The ability to half the sample rate to 250MSPS.
- Enabling four wire SPI. As mentioned not that relevant in this scenario.
- Read the internal temperature.
- Reset all the register back to the default value
- A series of options for exactly how the ADC output should be handled.
- Ability to disable the channels. As only one channel will be used, it is natural that the other will be turned off.

The datasheet [31] gives the full list of features and functionality available via the control registers.

4.3.3 AD9257 interface

Like the ADS5404 control registers, the AD9257 control registers also consists of an array of registers that control its behaviour, except it has 8-bit registers with a 13-bit address. As with the ADS5404, only a limited number of addresses are noted to have functionality connected to them. The AD9257 does however have a more complicated transmission protocol. A transmission with the AD9257 consists of a 16-bit instruction header as well as one or more 8-bit words consisting of register data. The AD9257 can also be in Most Significant Bit (MSB) first mode or Least Significant Bit (LSB) first mode, with MSB first mode being the default. In LSB mode the header and each word will be individually mirrored. The mode also determines whether the address counter should decrement (MSB first) or increment (LSB first) when reading multiple registers in a single transmission.

When in MSB mode the first bit (Read/Write Bar, RWB) in the instruction header communicates whether the register should be written to ('0') or read from ('1'). The next two bits (W1 and W0) communicates how many registers should be read, while the last 13 bits (A12:A0) is the initial register address. The numbers of accessed registers are 1, 2 and 3 when W1:W0 is '00', '01' and '10' respectively, while '11' indicates register streaming. When streaming it continues to access registers in sequence until CSB goes high. When not streaming it is possible to temporarily stall between words by letting CSB go high. Otherwise, the transmission is aborted when CSB and any partially transmitted data to the ADC is discarded. Figure 4-6 shows an example of the header and two words of data.

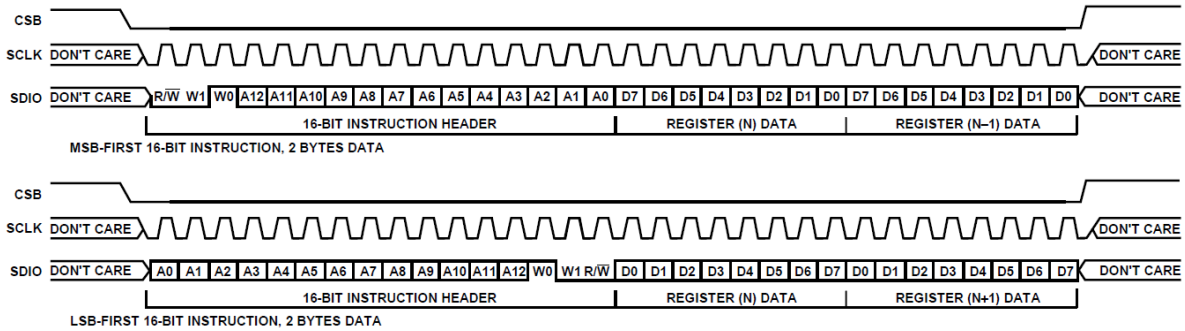


Figure 4-6 Sequence diagram when sending data to two registers in both MSB first and LSB first mode. [37]

The AD9257 SPI handles a maximum SCLK frequency of 25MHz. This means that if the two ADCs are connected to the same SPI controller the AD5404 SPI requirement dominates and forces us to use a SCLK frequency of 12.5MHz, while if each ADC has its own controller 25MHz is possible.

4.3.4 AD9257 control registers

Compared to the ADS5404, the AD9257 has a slightly more complicated setup due to its ability to be both MSB first and LSB first. It is therefore necessary to be able to unambiguously set the mode of the ADC regardless of what modes the ADC and the master are using. This is solved by controlling the mode by a mirror register at address 0. In addition to controlling the MSB/LSB mode, the mirrored register can also be used to reset the register values.

Some of the noteworthy features controlled by the control registers:

- It can be set to output a variety of test patterns.
- The ability to change the sample rate between 65, 50, 40, and 20 MSPS.
- The ability to turn off or on any of the channels. As such the unused ones probably will be turned off.

The datasheet [32] and SPI application note [37] give the full list of features and functionality available via the control registers.

4.4 Testbench model

4.4.1 Background

To debug, test and verify a VHDL module a testbench is commonly used. It does this by mimicking the expected surroundings of module through designed stimulation of its ports, and by monitoring the resulting behaviour. This is especially important in edge cases where it is easy to achieve unintended behaviour.

The testbench was created based on the *VHDL Verification Components* (VVC) framework of the *Universal VHDL Verification Methodology* (UVVM). This methodology is widely used by FPGA developers using the VHDL language, and it is the default testbench framework used internally at UiB as well. The methodology makes reusability of testbench parts easier, it ensures that corner cases are easier to detect due to allowing skewing of data on interfaces, and it allows for verbosity control to hide unwanted details. UVVM is developed and maintained by the companies Inventas (previously BITVIS AS) and EmLogic AS. [38]

4.4.2 UVVM

The information in this section is based on [38].

A testbench using UVVM mainly consist of the *Device Under Test* (DUT), a test sequencer, and modules (VVCs) that connect them together. It is also common to combine the DUT and the VVCs into a test harness, enabling the user to create different test sequences with only one test harness.

The DUT is whatever VHDL module or set of modules that are being tested. It will have one or more ports interfacing with the outside.

The test sequencer describes when and what input the DUT should receive, as well as monitoring the output and comparing it with the expected value/behaviour (like minimum/maximum hold time, etc). It does this by sending and receiving standardised function/procedure calls to the relevant VVC which functions as a translator between the sequencer and the port.

The VVC consists of two main components: a *Bus Functional Model* (BFM) and a command executer. The BFM consists of a bunch of procedure calls that describe some behaviour of the relevant interface and is used to determine the behaviour of the sequencer commands.

The command executer is the special part of the VVC framework, and essentially works as a shared scheduler between all the VVCs in the system ensuring they all occur at the correct time compared to each other. It is possible to use BFMs directly without using VVC, but this requires more manual work to get the timings right, especially if the DUT has more than one interface port. UVVM comes with a selection of premade BFMs and VVCs for common interface types, including SPI.

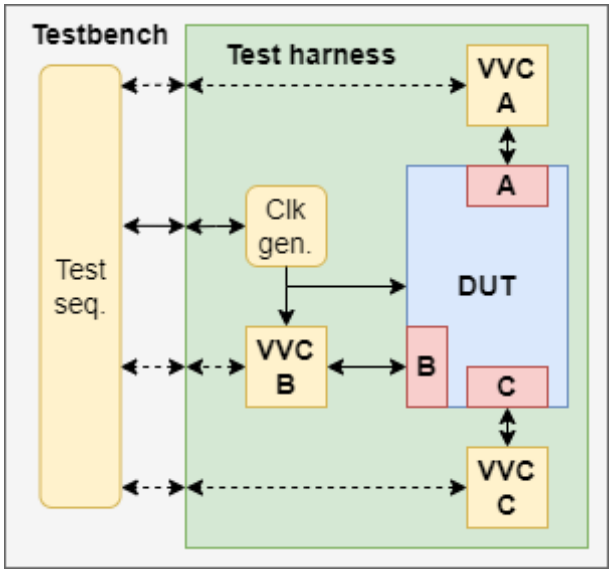


Figure 4-7 Block diagram of a generic testbench setup using UVVM.

Figure 4-7 shows the block diagram of a generic example of how a testbench using UVVM can look. In this example the DUT has three different ports as well as a clock. Each port is connected to VVC blocks, while the clock is controlled by a clock generator. The test harness enables the user to create multiple test sequences while only creating the test harness once.

The test sequencer is set up to generate/print a rapport of how it went after finishing a test, with an adjustable amount of detail. The idea is that you can use high degree of detail when actively debugging but keep it to a minimum in tests that only needs confirmation that still work as intended.

4.4.3 Testbench for the ADC simulation models

The core of the testbench is custom made VVCs (and the required BFMs) simulating the behaviour of the control registers of the ADS5404 and the AD9257. The idea of the testbench is to be able to add layers of VHDL modules to test the behaviour in every step.

When creating the custom VVCs of the control registers they were directly tested against the SPI VVC that comes with UVVM. The SPI VVC only inherently supports four wire SPI. After the behaviour was confirmed the testbench was expanded to include the SPI four- to three-wire converter described in subchapter 4.2.2 to confirm that both the convert works as intended and that the control register VVCs works as intended. Figure 4-8 shows a block diagram over the testbenches as described.

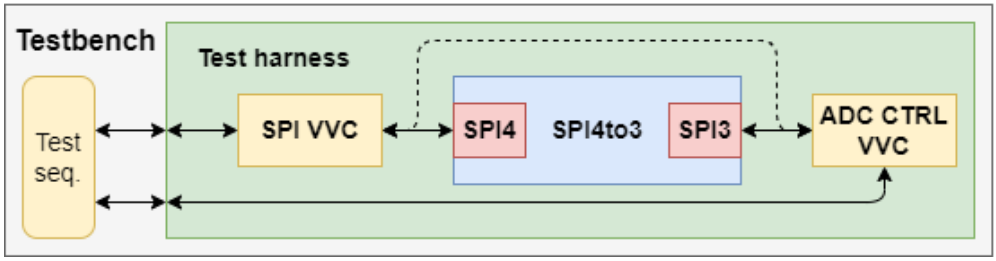


Figure 4-8 Block diagram of the testbench setup using the UVVM SPI VVC, an ADC control register VVC and the SPI to SPI3 converter. Each ADC has its own custom VVC, and its own testbench.

4.4.4 ADS5404 control register VVC

As with any VVC, the basis for the ADS5404 control register VVC is the BFM, which contains one or more procedures based on potential behaviour. UVVM has templates for creating BFMs as well as templates and guides on how to expand a BFM to a VVC, all of which was followed.

As a reactive component the ADS5404 BFM only requires one procedure and react based on whatever it receives from the master. The port itself consisted of the six wires discussed in 4.3.1, each with the function described. While the BFM itself can't properly handle pullup/pulldown, this is fixed in the VVC.

When a BFM is initialised in a testbench one of the things that can be manipulated is some internal configurations. One of the custom configurations of this BFM is whether it always should use 3 or 4 wire SPI or check the relevant register to determine. This was done to be able to seamlessly change when using the converter. It also has a list of default values for the ADC control register. The internal register values are accessible by the testbench to both read and change directly.

When simulating an exchange, the procedure is quite straight forwards. It first tries to read the instruction word from the master, and then read or write from the prompted address. Special behaviour, such as read only registers and soft reset of the registers are determined on an address-by-address basis when writing to a register. When the exchange is complete or somehow disrupted a report of the result is sent back to the testbench. When successful, the address and value sent are included in the report.

There are a couple of differences between the real thing and the simulation. The most obvious is that only the registers with functionality that influenced the ADC controller have any effect past changing a value in the simulated register array. The idea is that by giving the testbench access to the registers any further checks can be handled there.

Another limitation of the simulation is that it doesn't check the minimum timings of the SPI signals. Lastly the hardware reset and enable signals are only checked at the start of the exchange, meaning that if they change during the exchange nothing happens. This is not the case for the real component.

4.4.5 AD9257 control register VVC

The AD9257 control register BFM is very similar to the ADS5404 BFM, but with the differences mainly in how to handle the more complicated protocol.

When an exchange is started the AD9257 BFM first checks whether it is in MSB mode or LSB mode. This is normally done by checking the relevant value of the first register address, but the BFM internal configuration has an option to always or never use LSB mode instead. The code always works with MSB internally, so in LSB mode it mirrors any data after it receive it or before sending it.

It then waits to receive the instruction/header from the master determining the address, if it should read or write, and how many words it should expect. It then loops a read or write action similar to the ADS5404 BFM as many times as required. Again, any special action like read-only and soft reset is handled on a per register address level. The ability to stall between words when 1, 2 or 3 words are expected is implemented. When done or interrupted it sends a report to the testbench.

The AD9257 BFM has essentially the same limitations as the ADS5404 BFM. It has the same handling of registers not directly affecting the controller itself, and it lacks any checks of minimum timings. A noteworthy feature of the BFM is that if a non-mirrored word is written to address 0 no change will occur, and a warning will be sent to the testbench.

4.5 Hardware Tests

The third part of this objective was to verify if the ZYNQ SPI peripheral could be used for SPI communication with the ADC configuration. To achieve this, I would have to prove that it is possible to access and use the SPI controller and that the SPI signals can access the PL layer through the EMIO.

When testing this it is very helpful that the ZYNQ has two independent SPI controllers. This enables us to configure one as a master, one as a slave and then connect them together in the PL via the EMIO. It is then possible to confirm its functionality by comparing data sent and received by the master to that of the slave.

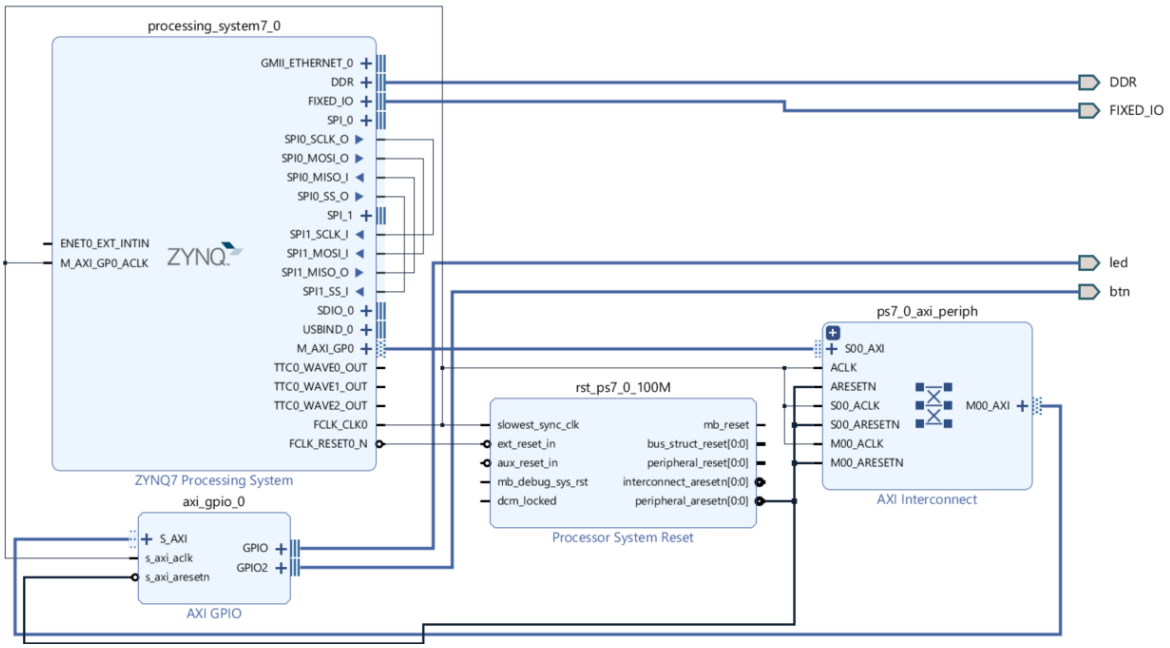


Figure 4-9 The block diagram of the SPI loop as set up in Vivado. The point of interest is the connections between SPI0 and SPI1 on the ZYNQ block. The rest is mainly there to enable the allow external (computer) access to the SPI control registers.

To do this, the SPI loop was created together with the modules required to enable external access to the SPI registers, seen in Figure 4-9. The SPI loop part of the build is connections between SPI0 and SPI1 seen by the ZYNQ block. The design was made in Xilinx Vivado Design Suite and tested on a ZYBO test card based around a 7z010 SoC. The 7z010 is for this purpose equivalent to the 7z030.

The external connection enables access to the System-Level Control Register via USB, which again can be used to configure the SPI control registers. This, combined with a library of drivers accessible from

Xilinx, allowed the creation of a C program that can transfer data between each of the SPIs and compare the result.

A test was created to verify that the ZYNQ SPI controller could send and receive data in the format of the protocols used by both the ADCs. By succeeding all the different combinations of transmitting and receiving, as well as handling multiple words and streaming when using the AD9257 control register protocol, it was verified that the ZYNQ SPI controller can be used to communicate with the ADC control registers.

4.6 Summary

The objective of this chapter was to verify if the SPI controllers in the ZYNQ could be used to interface and control the ADC control registers. This was split into three steps.

The first step was to create a four wire SPI to three wire SPI converter, which is necessary to allow two-way communication between the four wire ZYNQ SPI controller and the three wire ADCs. Thanks to the ADCs having integrated pull-down resistors, this is possible with the help of a single tristate resistor which is easy to implement in VHDL.

The next step was to create a testbench module capable of simulating the behaviour of the individual ADC control registers. This can then be used to simulate and verify that the expected communication. To do this in a structured way UVVM was used.

The last step was to confirm that it was possible to use the SPI controllers on the ZYNQ, and that they had access to the PL. This was confirmed by connecting the two SPI controllers together in the PL and confirming that it is capable of exchanging data using the protocols used by the ADC control registers.

5 Signal recording methods

This chapter discuss the approaches explored with trying to reduce the buffer capacity required to guarantee no loss of data during TGFs.

5.1 Background

The raw data output from a detector is determined by the sample rate and sample size of its ADC, which is 14-bit samples at 65 MSPS for the medium detector and 12-bit samples at 500 MSPS for the tiny detector. This results in a raw data output of 910 Mb/s or almost 114 MB/s for the medium detector and 6 Gb/s or 750 MB/s for the tiny detector. Combined this is more than 3.1 TB of raw data every hour of continuous usage.

The transfer rate between the PL and mass storage have previously been measured to allow a sustained transfer rate of 5 MB/s or up to 50 MB/s in short bursts [39]. This transfer rate is shared between all detectors. Clearly storing all the raw data isn't a feasible strategy.

A more sensible strategy is to only record the raw data when there are events happening. An event in this context is either a single detection or a series of detections in short succession, such as during a TGF.

The number of events on average a detector experiences each second, is dependent on the scintillator cross-section and the current flux. For simplicity, the average projection of the scintillator cubes is rounded down to be the square of its side length, giving the detectors a cross-section of 5cm cubed and 3mm cubed. So, on average, the medium detector can expect 25 detections each second for a flux of 1 photons/s while the tiny detector can expect 0.09.

The next valid assumption is that about 400 ns of each event pulse shape contains useful information. This gives 364 bits of data per event for the medium detector and 2400 bits of data per event for the tiny detector.

Combined this means that with a fluence of 100 photons/cm^2, which is far above the expected fluence during a gamma-ray glow, the total raw data generated on average each second would be become 100*(364*25+2400*0.09) = 912160 bits or about 114 KB. Even with large margins for the other detectors as well as the framework around this is clearly quite sustainable, assuming the system has good enough buffers to average the incoming data-rate.

If the plane carrying the detectors experiences a TGF in close vicinity, one can imagine up to one millisecond of continuous events. These events generate 910Kb and 6Mb of data from the medium and tiny detector respectively, which emphasizes the need of buffers to avoid losing valuable data.

The 7z030 FPGA has 265 BRAM modules available, each 36Kb, for a total of 9540Kb shared between all FPGA modules on the PL layer. This means that just buffering the raw data from the two detectors would require more than 72% of the available BRAM. This gives very little margin for the rest of design.

It is therefore important to explore methods that could reduce buffer requirements of the data. To achieve this, two main approaches was explored.

The first approach tries to reduce the buffer requirement by analysing the incoming raw data in real time and only saving a few select samples/values and timestamps. These samples should then provide information about the event counts, times and potentially energy.

The second approach instead tries to keep all the relevant data but ease its buffer requirement through different techniques of compressing or otherwise reducing the size of the raw data. A vital part of the approach is about successfully detecting when an event is occurring.

An alternative approach not explored in this thesis would be to buffer the data outside the PL. The easiest candidate would be the system Static Random-Access Memory (SRAM) that is accessible and shared by both the PL and PS [33]. It is also possible to attach external RAM to the system accessible to the PL [33]. I was informed it was preferable to contain the buffer to the PL, so this approach was not further explored.

The plots used in this chapter are generated using MATLAB.

5.2 Signal characteristics

Prior to discussing the two approaches, it is useful to provide more details on the signal and pulse characteristics.

For clarity the terms *online* refer to things that occur on the ZYNQ while the detectors are active, *offline* refers to things that occur after the plane has landed. Furthermore, a *pulse* refers to single, isolated pulse generated by a detection event, while the *signal* refers to the sum of all the pulses and is what the ADC observes.

5.2.1 Temporal resolution

When the continuous analogue pulse is sampled by the ADC, its temporal resolution is determined by the sample rate. For the tiny detector with the ADS5404 ADC this is 500 MSPS while for the medium detector with the AD9257 ADC this is 65 MSPS.

Figure 5-1 shows the sampled pulse shapes for both ADCs, although with unlimited amplitude resolution. An important observation is that 65 MSPS is potentially not fast enough to accurately measure the CR^2-RC^2 shape. While in the figure the peak sample happens to be very close to the true value, in a worst-case scenario the maximum recorded amplitude will only reach about 94% of the true maximum as opposed to about 99% for the CR-RC^2 shape (numerically calculated in MATLAB). This makes the CR^2-RC^2 ill-suited for use in the medium detector.

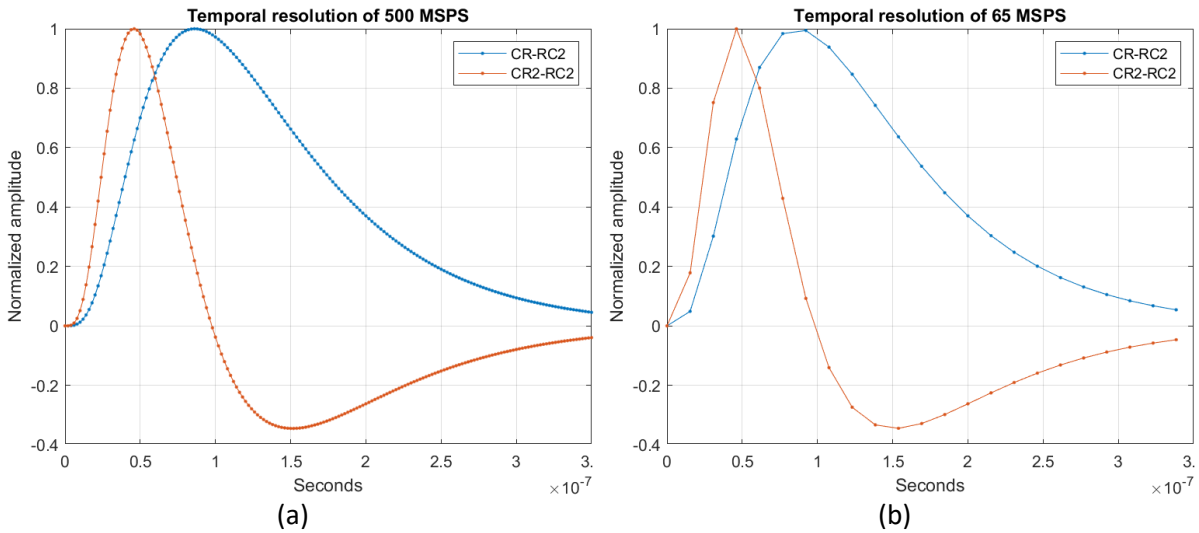


Figure 5-1 The temporal resolution of (a) the ADS5404 with 500 MSPS and (b) the AD9257 with 65 MSPS.

5.2.2 Amplitude resolution

The amplitude resolution of an ADC tells something about how well it can differentiate different amplitudes of a received signal. While primarily determined by the ADC sample size, it is also slightly dependant on how the expected signal range is mapped to the available ADC levels.

The simplest mapping is to set the maximums and minimums of the expected signal and the ADC equal. This ensures that no clipping of the recorded signal occurs even if all the microcells are triggered simultaneously. Figure 5-2 shows an example of how the two pulse shapes would appear at max amplitude with full range mapping. Note how the CR^2-RC^2 pulse shape has shifted baseline to include the undershoot.

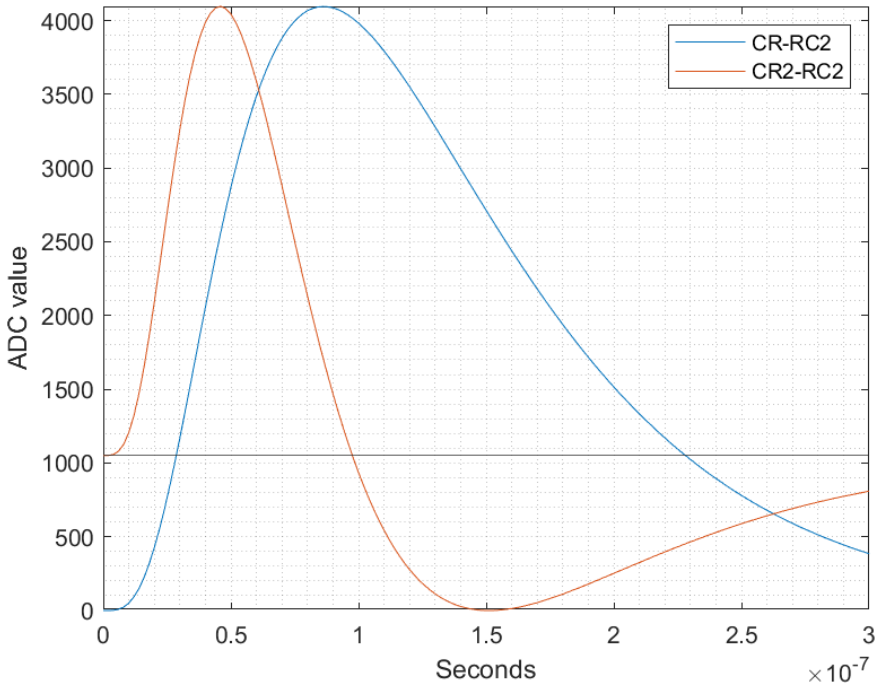


Figure 5-2 Maximum amplitude sampling using 12-bit samples and 500 MSPS. The black line is the baseline of the CR^2-RC^2 pulse.

One possible approach to improve the amplitude resolution would be to map the maximum values of the ADC to only say 90% of the maximum possible signal value. This would be physically achieved through increasing the amplification of the ADC driver while keeping the cut-off diodes the same.

The reason one might want to do this, is that a max amplitude would mean all microcells have fired simultaneously which can only happen if the detector is hit by one very energetic gamma photon or bombarded with a myriad of smaller ones. The first case is unlikely because it would, as was seen from Equation (1), require an exceptionally high energy photon, which again has a smaller chance of depositing its energy into the sensor. This is particularly true for the tiny detector. The second case would require multiple photons to hit the sensor in very short succession, which indicates that the sensor is saturated. This would limit the amount of useful data anyway.

A different consideration is that it is almost certainly ideal to have some clearing between the baseline and the minimum saturation point. This makes it possible to detect things like baseline drift or other effects that make the signal reliable. This is automatically the case for the CR^2-RC^2 pulse shape but must be included when using the CR-RC^2 pulse shape.

For the tiny detector, with 12-bit samples and 5676 microcells, a full range scaling will translate to each ADC level representing about 1.39 microcells fired. For the medium detector (14-bit sample, 5676*64 microcells) each ADC level represent about 22.17 fired microcells on full scaling.

5.2.3 Pileup

While the goal of the experiment is to analyse each individual detection, the signal the ADC observes consists of the sum of the pulses from all detections. This means that multiple detections close in time will create pulses that can add up to a signal that it is difficult or impossible to split into the underlying pulses. Figure 5-3 shows an example where the interference causes the signal (blue line) to have differences in amplitude and even timing compared to the underlaying pulses (dashed lines). Note that in this example all the pulses have a corresponding peak in the signal, something that is not a guarantee.

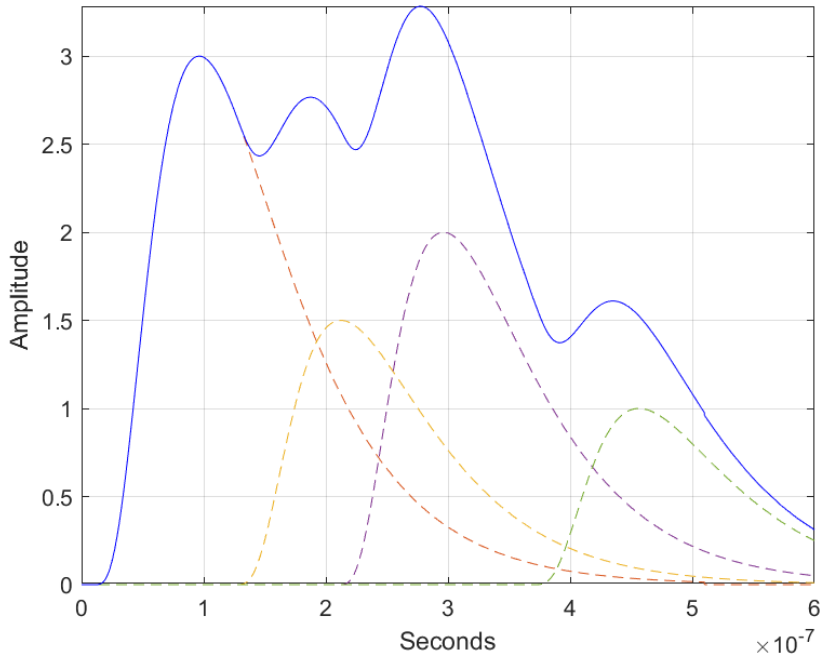


Figure 5-3 Example of how different pulses (dashed lines) with the CR-RC² pulse shape adds together to create a signal (blue line) with different peak amplitudes and peak times. This is not an example of pileup as the term is used in this thesis.

A *pileup* in the context of this thesis occurs when the underlying pulses interfere enough for it to be fewer peak in the signal than there are underlying pulses. This makes it significantly harder or impossible to extract information like how many and with what amplitudes does the underlying pulses have. Figure 5-4 shows an example of this where four pulses (dashed lines) has merged to create only a single peak in the signal (blue line).

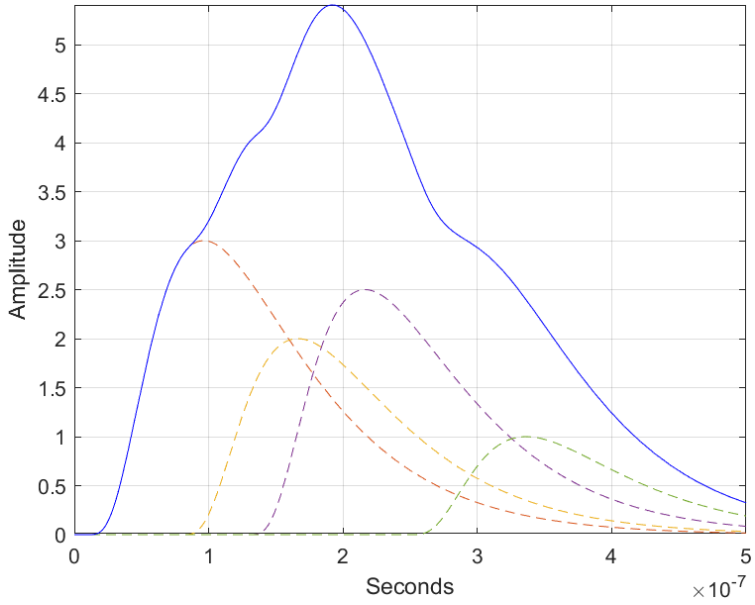


Figure 5-4 An example of a pileup with the CR-RC² pulse shape where the signal (blue line) has only a single identifiable peak while consisting of four underlying pulses (dashed lines).

The border between a group of pulses causing a pileup or not, is determined by a variety of factors. For the analogue signal seen by the ADC, important factors include the timing between the pulses, the relative amplitude the pulses, the shape of the pulses themselves, and what noise the signal experiences. The digital signal sampled by the ADC is also limited by the temporal and amplitude resolution. In some cases, a pileup may even be indistinguishable from a single pulse with a higher amplitude if the factors are sufficiently aligned.

Figure 5-5, which shows a case of pileup and almost a pileup using the CR^2-RC^2 pulse shape, illustrates some of these effects. Simply by swapping the order of the amplitudes the two first pulses create a pileup while the two last ones, with the exact same time difference, creates two individual peaks in the signal. It is also easy to imagine how signal noise, insufficient temporal resolution or amplitude resolution may effectively blur out the tiny first peak in the signal from the two last pulses.

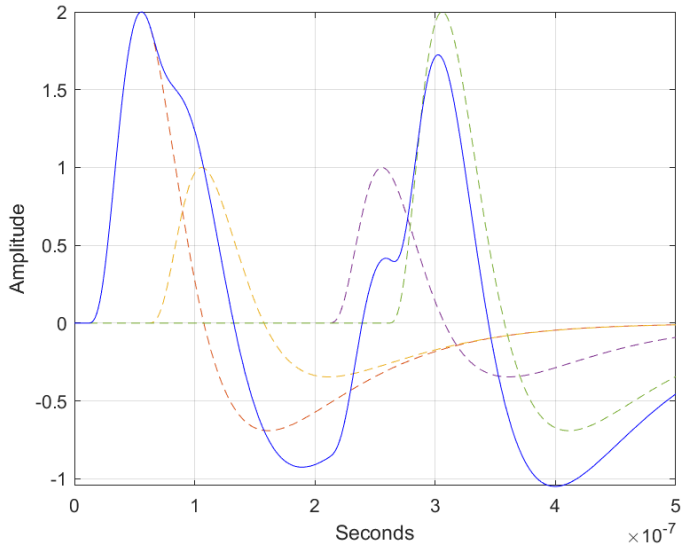


Figure 5-5 Example of a pileup and an almost pileup with the CR^2-RC^2 pulse shape. The blue line is the signal while the dashed lines are the underlying pulses isolated.

5.2.4 Pulse recovery

If a signal is not affected by pileup, it is very easy to isolate the underlying pulses. The first signal peak during a series of pulses will have the same amplitude as the underlying peak. By knowing the peak of a pulse means knowing when it starts and how it affects subsequent pulses. You can then subtract its effect on the rest of the signal, allowing you to find the correct peak of the next pulse. You can then repeat for all the pulses in the series.

If all the data of the signal is available this is as easy as calculating and subtracting the influence of the pulse from the signal. With less data it gets slightly harder, but even only knowing the time and amplitude of the signal peaks, it is possible to calculate what amplitude and start time the next pulse would need to have to cause the peaks in the signal.

While pileup can cause it to be significantly harder or impossible to isolate the underlying pulses, it is still possible to gain useful information from them. An important property of the decay of the signals is that when past the peak of all underlying pulses, the signal will decay at the same rate as a pure pulse of the right correct amplitude and timing. This can then be used to calculate its influence of subsequent pulses.

Figure 5-6 (a) shows how in some cases the peak of the pileup itself can estimate this equivalent pulse, although with some error. Figure 5-6 (b) shows the same example but where an additional pulse has occurred after the peak of pileup but without enough amplitude to create its own peak. This causes a decay very different from what the signal peak estimates. Also note how by subtracting the estimate from the signal the additional pulse appears in the result. This works if you know the decay of the previous pulse(s) and have access to the entire signal.

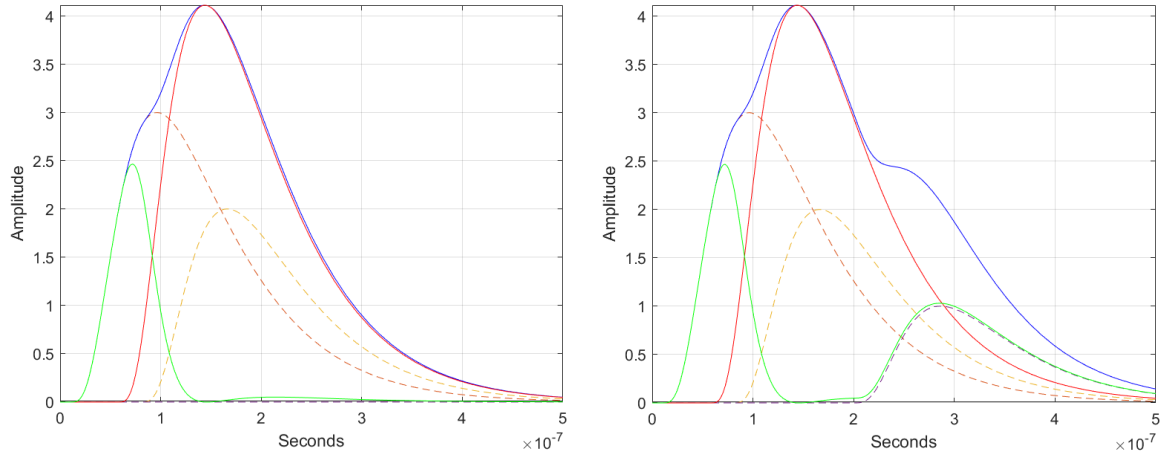


Figure 5-6 (a) An example comparing a pileup (blue line) comprised of two pulses (dotted lines), with a single pulse estimate (red line) having the same peak amplitude and peak height as the pileup. The green line shows the difference between the blue and the red line.

(b) Same as (a), but with an additional pulse occurring after the peak of the signal. Note how the green line almost mirrors new pulse.

A useful tool when analysing the signal is to use its derivative. One possible use is to find out how many underlying pulses there are. Figure 5-7 shows how the delta (derivative) of the signal has an equal number of peaks as there are underlying pulses.

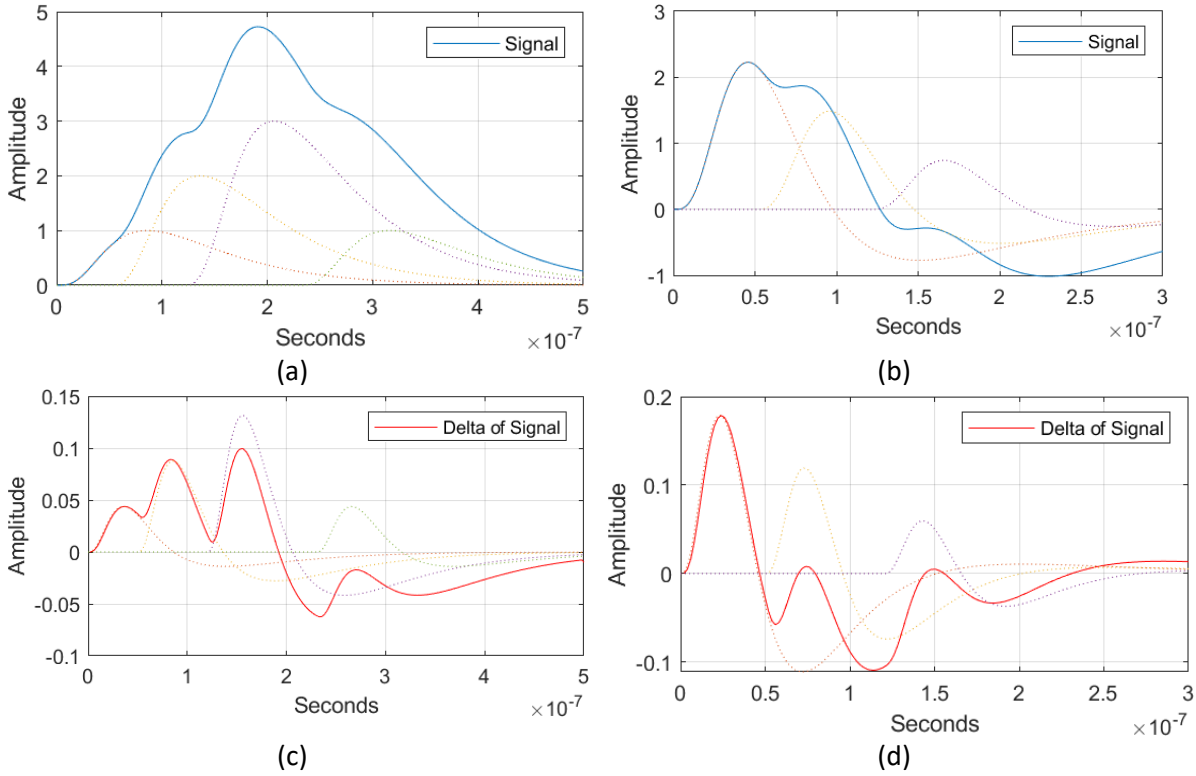


Figure 5-7 A pileup with (a) the CR-RC² pulse shape and (b) the CR²-RC² pulse shape. (c) and (d) shows the deltas (derivative) of (a) and (b) respectably. The dotted lines are the underlying pulses (or their delta) isolated.

For this method to work well the signals needs to be smooth, as any noise will be amplified in the delta of the signal and may obfuscate the peaks.

5.2.5 Timestamping

An important part of the campaign is to accurately record when an event occurs, as this is required to properly compare detections from different sources. For the detectors this consists of linking each sample with a timestamp.

By using the known sample rate of the ADC, it is possible to reduce the number timestamps required, as only the one of a continuous series of recorded samples requires a timestamp. The timestamp could then be added as a header when the ZYNQ is finished with any pre-processing. When using the data in post be sure to know if the timestamp shows when the sample was received or when the timestamp was added. In the last case compensate for the time the pre-processing would require.

While only one timestamp for a continuous series is required, it might be beneficial to add extra timestamps at fixed intervals, or at the very least the start and stop. This ensures that should some data be lost or somehow corrupted, the odds of losing track of when any remaining data is from is reduced. It can also help identify hidden data losses or inconsistent sample times through discrepancies between the timestamps.

5.2.6 Noise floor

While signal noise has not been a focus of this thesis, it is still relevant when discussing various aspects of manipulating the raw signal. The *noise floor* of a signal is the measured sum of all noise sources in the system and will in the following sections be used to describe the limit below which it is difficult or impossible to differentiate between a change in signal or just noise. The noise floor is assumed to be symmetric on both sides of the signal.

5.3 First approach: Online analysis

The idea behind the first approach is to do online analysis of the signal and only store a select few samples and values. This drastically reduces the amount of data produced, removing the need for large buffers.

The basis for this approach consists of finding and recording the peaks of the signal and its delta signal. This has the benefit of being very computationally light, subtraction to find the delta and value comparison to identify the peaks, as well as giving a decent amount of information as described in Chapter 5.2.4.

There are a couple of challenges and limitations of this approach. The first and most severe is the presence of noise. Any oscillations of the signal will cause significant variations in the delta of the signal. Figure 5-8 shows an example of how sufficient signal noise can completely eclipse any peaks of the delta signal. For this reason, significant filtering, averaging or otherwise smoothing of the signal is required. This does have the downside of removing details, reducing the accuracy of signal. The more you smooth the signal the less likely you are to get false peaks but the more likely it is that some of the true peaks are lost to the smoothing. This is less of a problem with the medium detector, as its smaller sample rate makes each individual step greater.

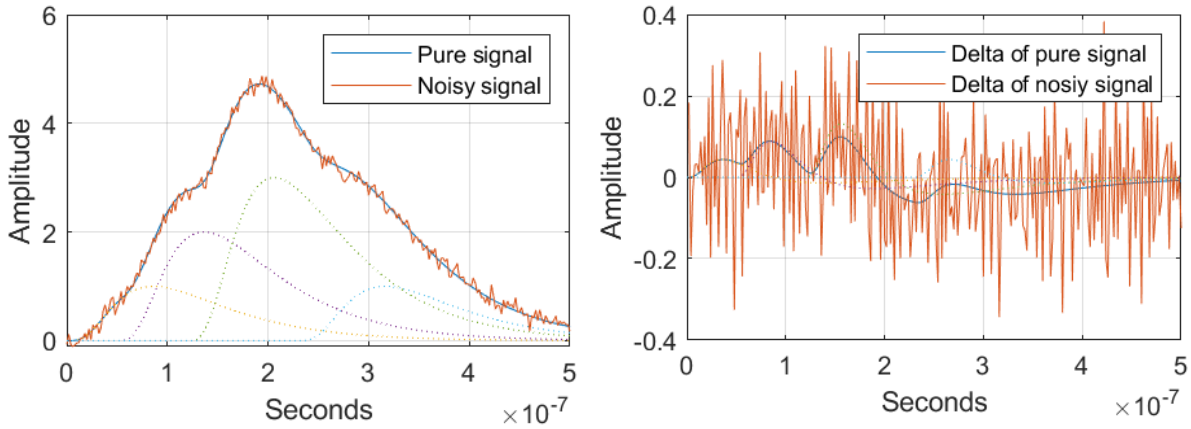


Figure 5-8 Example of how sufficient signal noise can eclipse the peaks of the delta signal. Shown with unlimited amplitude resolution and 500 MSPS.

A different potential problem is present when trying to calculate the pulse amplitudes. So far in the thesis the pulse shapes have been thought on to be perfect. In reality they are first order approximations based on numbers from the datasheets and some laboratory testing [30]. This means that there will be some differences from the model shape and the calculated one, especially when considering how the temperature of the components might change during flight. The result is that the effect the pulses have on each other can easily be erroneously modelled. This is less of a problem with the second approach as all the data is available offline and one can try to fit a pulse shape to the signal.

5.4 Second approach: Event detection

The basis for the second approach is to store all raw data when an actual event has occurred. I.e., the challenge of this method is to detect when there is a valid event, be it a single detection or a full TGF.

The base idea of the event detection is to define some minimum threshold the signal must reach to count as interesting pulse. The threshold needs to be sufficiently above the signal noise floor. It is however insufficient to purely use the threshold to select data, as you lose relevant data that happens before or after the part of the pulse above the threshold. One easy way to handle this is to also record a certain number of samples before and after breaching the threshold.

Figure 5-9 shows a somewhat exaggerated view how this might look, with a baseline set of 50 and the threshold set at 100. The red dots depict the data above the threshold, meaning that quite a lot of the samples needed to fully analyse the signal are missing. By also including 50 samples before and 100 samples after the threshold is surpassed (green dots), you get a much more comprehensive view. The blue dots are data that is discarded. A real system will naturally be calibrated so that an optimal threshold is selected. This will probably be closer to 80, all things being equal with the figure. The noise in the figure is added to better illustrate the point and is not properly scaled to the expected noise floor.

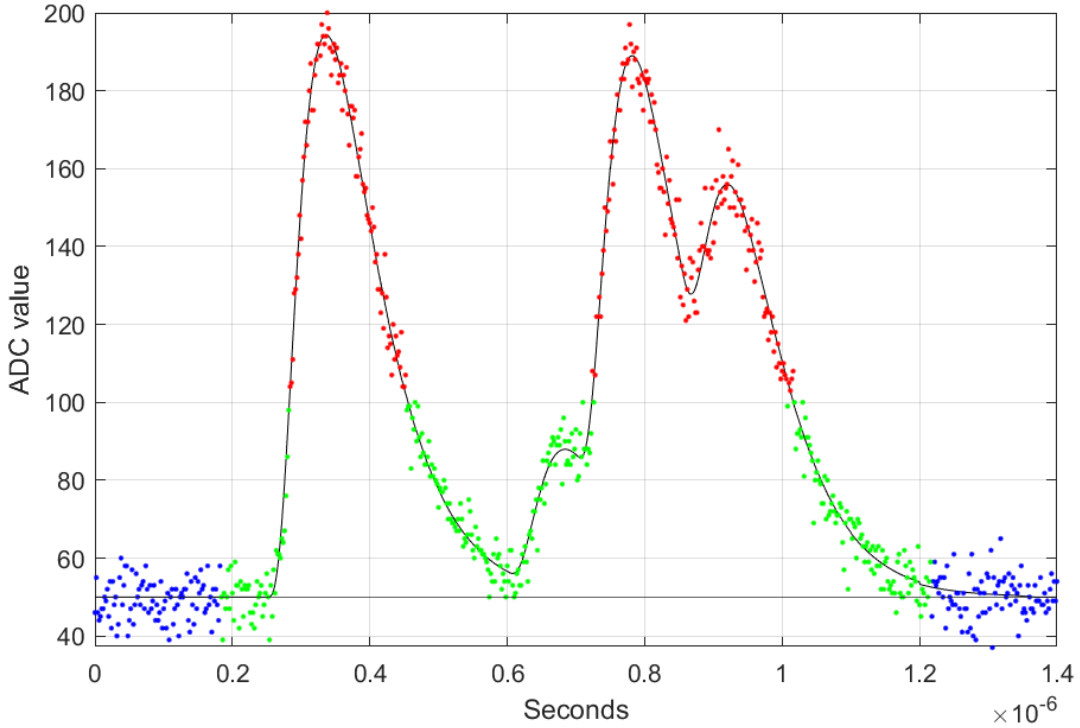


Figure 5-9 An example of how an event detection could appear for the tiny detector. The red dots are samples above the threshold, here set at 100. The green dots show samples that could be very beneficial to include that a pure threshold miss. The blue dots show data that can be safely discarded. Some noise is added to samples to show a more realistic signal.

There are additional reasons to add the extra samples. The first one is that it is required it is necessary to use the CR^2RC^2 pulse shape due to its undershoot. Even with a negative threshold the amount of information that can be found around the baseline is significant. A different reason is that this ensures a continuous series of samples which is beneficial to reduce the number of timestamps required, as well as for several of the compression methods that will be discussed in Chapter 5.5.

If the threshold is close to the noise level it might be beneficial to require a minimum number of samples above the threshold before it is flagged as an event. This will reduce the number of false positives from noise.

A different aspect is that if the signal baseline experience drifts, a static threshold will change its sensitivity during the drift. While it shouldn't be a significant problem in our case, one way to circumvent it is to use a dynamic baseline based on an estimate created by the received data. This could be done by sending the data through a low pass filter or make a simple moving average filter. The pulses from the actual events should be excluded to ensure that the baseline estimate is not set to an erroneous value.

5.4.1 Hardware implementation ideas

To ensure that the leading samples are available, all the received samples can be sent through a buffer while comparing them to the threshold. If the threshold is surpassed, you can then start clocking the oldest buffered value to the next stage. Each time the samples goes below the threshold you can start a counter that keeps track of how many more samples should be recorded. Reset the counter should the signal again go above the threshold. When using only one buffer for this purpose, there should be no danger of recording samples more than once should a trailing and a leading signal of interest overlap.

If baseline adjustment is required, it is possible to use the same buffers to create the filter, and simply pause that function while a 'there is an event' flag is active.

It is important to pair any event data with a timestamp. A benefit of storing continuous streams of event data is that inherently only one timestamp is required. It is, however, not necessarily practical to group an arbitrary amount of event data as a single file. A different approach would be to group the samples into smaller, independent chunks, each with their own timestamp. An added benefit would be that should a catastrophe occur, and some data gets lost or corrupted, only the chunks directly affected will be lost, as opposed to potentially affect a large part of the single file.

Figure 5-10 shows a block diagram of an example of how this could be implemented. The raw data is first baseline corrected, if needed, followed by an event detector. When an event is triggered, the event data is sent to the compression algorithm and notice that an event is in progress is sent to the timestamp logic and the baseline correction. The generated timestamp and any compressed data are then grouped as a chunk. When an appropriate number of samples are grouped, they are sent together with some metadata to the buffer waiting to be stored. A new timestamp is then generated for the now empty chunk buffer. By metadata I mean things like chunk ID, info of what detector the data is from etc, as well as any needed headers related to the storage. It could also contain what the calculated baseline was if baseline correction is required. In this example the main buffer the is shared between all the detectors but depending on context it might be preferable to have large detector specific buffers.

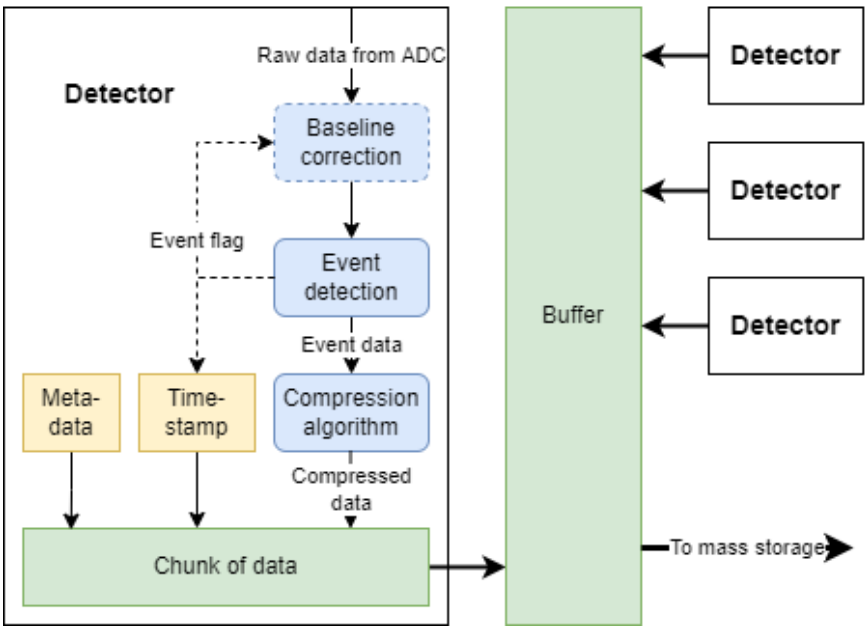


Figure 5-10 Block diagram of how the data from the detectors could be structured. The raw data from the ADC would first be pre-processed (blue boxes), which consists of a baseline correction (if needed), the event detector and one or more algorithms to compress the event data. Some appropriate amount of data could then be grouped into a chunk together with a timestamp and any other meta data required. The chunk is then sent to a buffer shared between the detectors.

5.5 Second approach: Data compression

The idea of data compression is to encode information using fewer bits than the raw data. In the context of the detector this will represent a reduction in the number of bits required to store each sample. If any header is required to add to the stored data, it should also be included in the consideration.

Compression can be either *lossy* or *lossless*. Lossless compression conceptually works by finding and removing redundant bits while ensuring no information is lost, enabling perfect reconstruction of the original data. Lossy compression meanwhile allows the removal of less important information, enabling better compression at the cost of losing some information. There is a myriad of different compression schemes available, but the fact that they need to encode a stream of data on an FPGA with limited time per sample drastically reduce the ones available. Some of the methods can be used in series for improved results.

I have had extra focus on trying to compress the signal from the tiny detector, as it alone would require 63% of all the available BRAMs to buffer a TGF. With 12-bit samples each bit compressed frees up over 5% of the total number of available BRAMs. Comparatively, the medium detector would require 7 bits compressed per sample to achieve similar effects.

A different relevant factor is whether the compressed data has a fixed or variable sample length. As the names implies, a fixed sample length means that every sample has a known, fixed size. Meanwhile, a variable sample length means that it can vary how long each sample is. Fixed length has the benefit of being predicable and easy to design around, such as each chunk having the same number of samples and the same size. It does however often require more bits per sample on average to cover all possibilities.

A variable length allows the algorithm to optimise its compression for each individual sample or series of samples. The variable length makes it impossible to predict exactly how many samples can fit into a chunk of maximum size, something that will increasing its protocol complexity or requiring trailing zeros to fill it to size.

5.5.1 Down-sampling

A very simple and effective lossy compression method is to reduce the sampling rate. Halving the sampling rate halves the buffer requirements but does also half the temporal resolution potentially making it harder differentiate handle pileup.

There are several ways to implement the down-sampling. The arguably easiest method is to use the ADC control registers via SPI to directly configure the sample rate of the ADCs. It is however limited by whatever sample rates the ADC has available. The sample rates available are 500 MSPS (default) and 250 MSPS for the ADS5404, and 65 (default), 50, 40 or 20 MSPS for the AD9257. Figure 5-11 illustrates how those sample rates handle a pure pulse with perfect amplitude resolution.

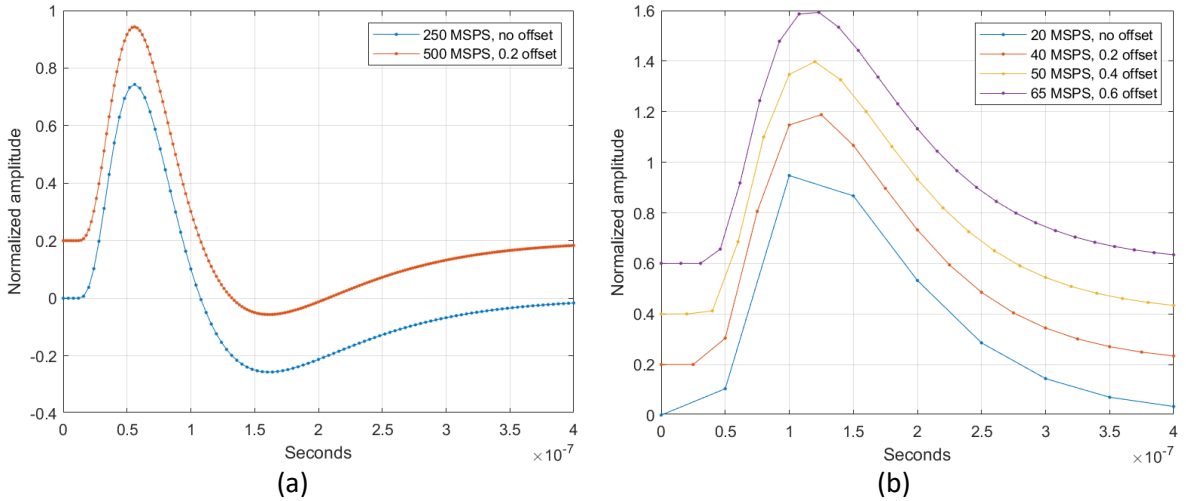


Figure 5-11 Illustration of the sampling rates (a) the ADS5404 and (b) the AD9257 can be configured to. The signals are offset to better visibility the differences.

The alternative is to down-sample in the PL, which offers a couple of alternatives. The most straight forward method is to only keep every N sample, reducing the sampling rate with a factor N. If noise is a concern, it is also possible to use some sort of weighted average when only sampling every N sample.

You can alternatively discard every M sample. Note that by doing so, the time between each sample will be uneven, which needs consideration when decoding the signal. The simplest way to do this is to have the discarding be determined by a counter that resets between each event. Figure 5-12 shows examples of both these approaches for the tiny detector with the CR^2-RC^2 pulse shape.

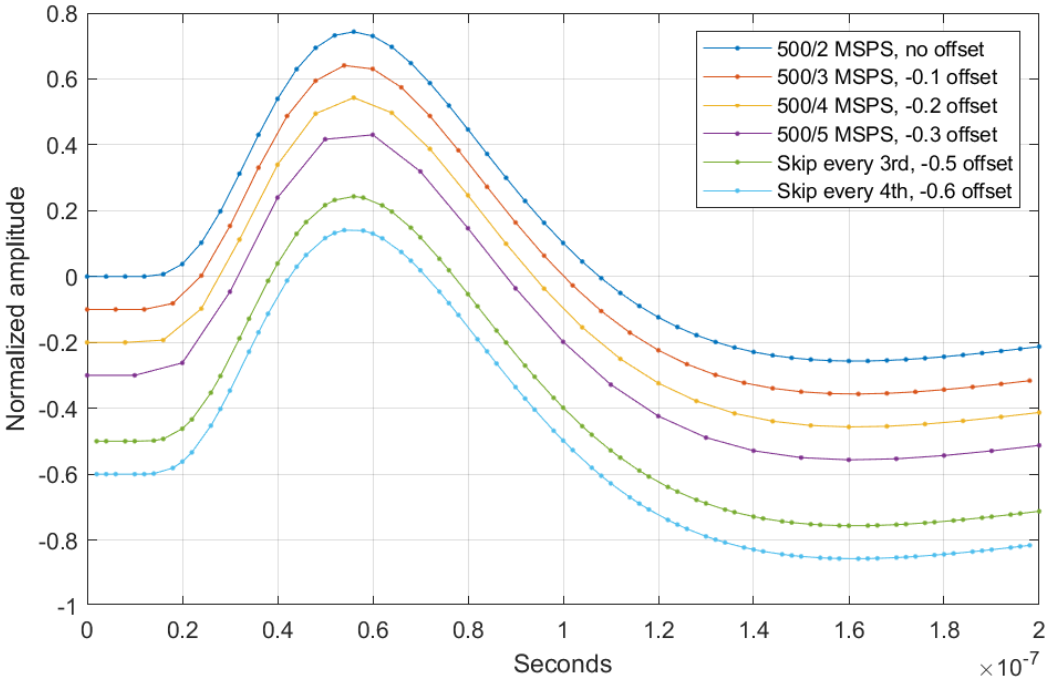


Figure 5-12 Examples of different reductions in sampling rate. Base 500 MSPS, perfect amplitude resolution, CR^2-RC^2 pulse shape.

Lastly it is worth mentioning the possibility of doing normal resampling with the goal of better control the resulting sample rate. It is however highly ill-suited for this purpose as it requires up-sampling an already high sample rate before down-sampling it. The added complexity probably negates any benefit it might bring.

5.5.2 Rounding

If the noise floor is too high, the information provided by the LSB may in large part become useless. This would allow the pre-processor to discard the LSB with very little loss of data, essentially lossy compressing each sample by one bit. When reconstructing the signal offline, one could then just set all the LSBs to '0'.

Figure 5-13 shows an example of how a rounded signal could compare with its unaltered counterpart and clearly shows how only the details, that are already unclear due to noise, are affected.

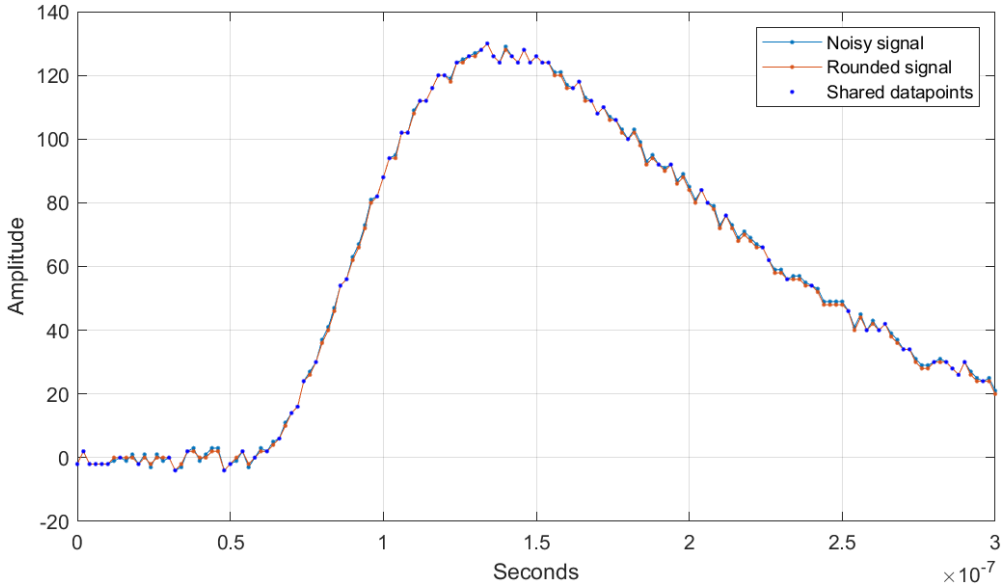


Figure 5-13 Comparison between an unaltered signal and its rounded counterpart. The dark blue spots are where the two are equal and, as expected, count about half of all the signals

This lossy compression method reduces the amplitude resolution but has the benefit of being essentially trivial to implement in VHDL. If executed first it also has the benefit of being compatible with essentially all other compression methods, functioning as if the sample sizes were one bit smaller.

If the signal has a very high noise floor, discarding more than one LSB is a viable strategy, but if that is the case smoothing the signal through a lowpass filter might be a better solution.

5.5.3 Delta compression

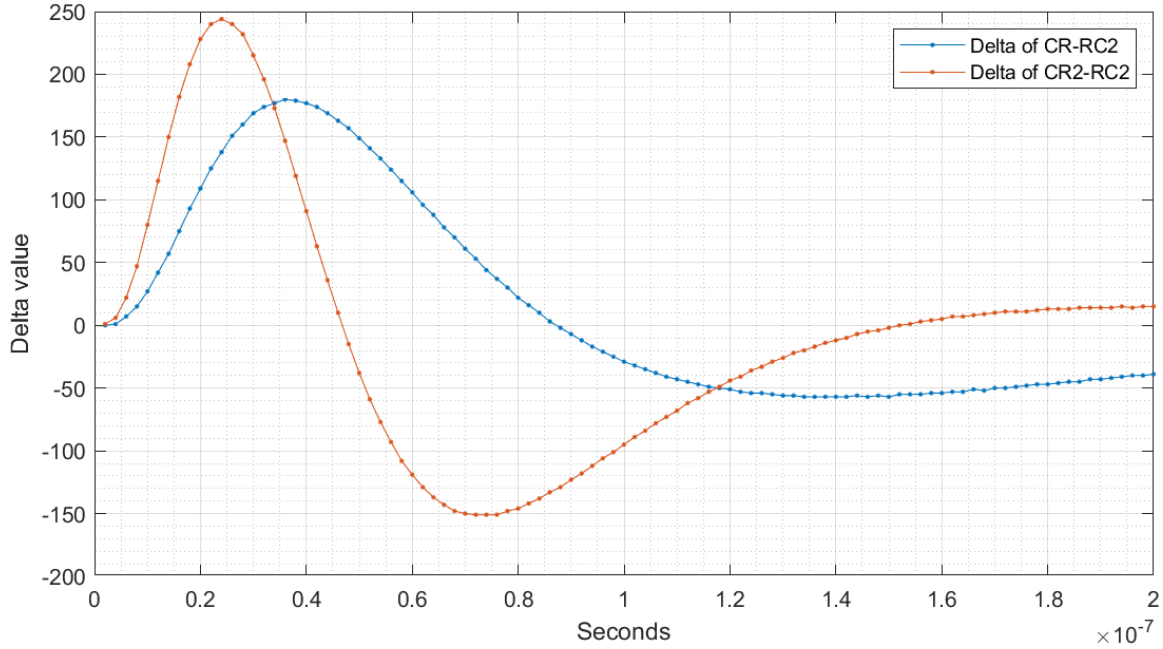


Figure 5-14 The derived noiseless signal of the two pulse shapes with maximum amplitude (4096) from the tiny detector.

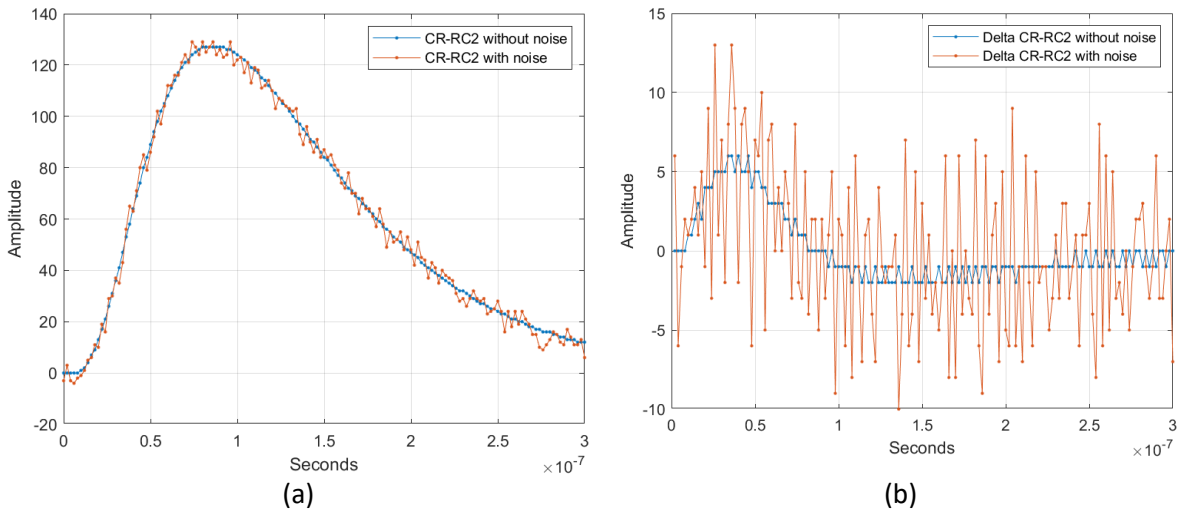
Delta compression background

An important property of the detector signal is that it is continuous. In practise this means that the *delta* of the signal (its derivation) will have a limited range, as illustrated by Figure 5-14. This in turn can be used to compress the signal.

To properly do this it is important to know what the maximum and minimum values of the delta can be. As the signal is the sum of all microcells fired, this is achieved when they all fire at once. There are also several other factors determining the values of the delta extremes.

One such factor is the maximum steepness of signal, which is determined by the pulse shapes. Closely related is the temporal resolution that determines how much of the slope is between each sample. Next is the amplitude resolution, as the more levels there are available to describe the pulse, the larger delta value is required.

Lastly there is the effect of the noise floor, which independently adds itself to the other effects. This is due to how noise essentially gets amplified by taking the derivative, as can be seen on Figure 5-15 comparing a noisy and noiseless signal. As such a low noise floor is highly beneficial to enable high delta compression.



(a) (b)

Figure 5-15 An example showing how a little noise in the signal (a) has large impact on the delta signal (b).

Table 1 shows what the approximately calculated delta extremes in the absent of noise of the different combinations of detectors and pulse shapes can expect. It is assumed that the delta extremes of the signal are mapped to the extremes of the ADC input. The amplitude shows the number of levels the extremes will jump, while the bits columns show the number of bits required to represent this value (via log2). The delta span shows the span from the maximum to the minimum.

	Maximum delta		**Minimum delta**		**Delta span**	
	Amplitude	**Bits**	**Amplitude**	**Bits**	**Amplitude**	**Bits**
Tiny detector, CR-RC^2	180	7.5	-57	5.8	237	7.9
Tiny detector, CR^2-RC^2	244	7.9	-152	7.3	396	8.6
Medium detector, CR-RC^2	5390	12.4	-1750	10.8	7140	12.8
Medium detector, CR^2-RC^2	6990	12.8	-4580	12.2	11560	13.5

Table 1 Calculated delta extremes of between the two detectors with the two pulse shapes. Noise is ignored and full mapping is used.

The interesting result is the delta span bits column, as it shows the minimum number (if rounded up) of bits required to accurately represent delta values. This can then be compared to the number of bits required to represent the original value, which is 12 for the tiny detector and 14 for the medium detector. The result shows that while the tiny detector might save a couple of bits per sample there is nothing to gain for the medium detector. For this reason, the majority of discussions about delta compression will focus on the tiny detector. The results also show how the CR^2-RC^2 pulse shape with its steeper slopes has, as predicted, less potential for compression than the CR^2-RC^2 pulse shape.

One strategy for using this knowledge to compress the data, is to store samples as the delta from the previous sample instead of the absolute amplitude. By knowing at least one absolute value from which a delta sample is based on, you can perfectly reconstruct the original signal. There are multiple ways to implement this strategy, with the main difference being fixed sample size or variable sample size.

Fixed sample size delta compression

The easiest way to implement this strategy is simply to store all the samples as a delta value. Each event could then have its own chain of delta values, with the first sample either stored as absolute value or showing the delta from a predetermined absolute value. This easily reduces the size of each sample from 12 bits to around 9 bits while still having some capacity for the noise floor. The delta sample values should be encoded to ensure both positive and negative deltas are covered.

It is possible to further improve the compression if potential lossy behaviour is allowed. The numbers shown in Table 1 are (ignoring noise) a worst-case scenario, and only occurs if all microcells are triggered as one. This is as previously discussed highly unlikely, especially in the tiny detector which lacks the volume to reliably absorb all the energy of the gamma photons. It would also require the detection to be isolated with no trailing decay from a previous pulse and no new detection until after minimum delta has occurred. As such the delta sample can be reduced by at least 1 bit with little risk.

It is however important to properly define how the encoder should handle if it were to encounter a delta larger than it can represent. This is worth adding even if delta sample should theoretically be able to represent any expected delta.

One way to do this, is to calculate the sum of all the previous deltas samples and comparing it to the real value. The difference should be zero if the delta never exceeds the cap of what the delta sample can represent. Any difference therefore means the cap is exceeded, which can then be summed with the next calculated delta sample. This ensures that the calculated sum, which is what the decoder will generate, will realign with the real value. In other words, it limits the slur rate of encoded signal.

Variable sample size delta compression

While the delta span in Table 1 shows the number of bits required to perfectly represent any possible signal (ignoring noise), most of the delta samples can be represented with fewer bit. This can be used to further compression the data through using variable size on the delta samples. It does however come at the cost of more complexity.

In hardware this could be structured as several fixed size delta compressions with different delta sample sizes in parallel. You could then choose the smallest size that can represent the signal. It is also possible to tailor the ranges the sample sizes represent, for example with some mapped to better fit the different starts of a pulse.

When reconstructing, it is vital for the decoder to known what the sizes the samples has and where the next sample starts. This can be handled by ordering the samples in headers and blocks, where the

header tells what type of delta compression is used in the next block, and the block consists of a fixed number of samples.

How well the compression works is dependent on several factors in addition to the specific delta sample sizes, namely the header size and block length. The more bits the header has, the more potential choices of samples sizes can be used. The longer a block length is, the fewer headers are required to add to the data, but the more likely it is for the block to contain a large delta.

An example on a setup would be a 3bit header followed by a 6 samples long block, giving up to 8 different tries to compress the data to as few bits as possible. It would give better compression if at least half the blocks are one or more bits better compressed than a fixed size variant.

With multiple potential paths it is easy to design delta ranges optimised for specific parts of a signal. Two delta samples with the same number of bits could cover different ranges simply by attributing a different constant to them. This means that especially when the signal represents the decay of a pulse, the range required to describe it only needs a very limited range. Note that the noise floor will always set a minimum delta sample size no matter how specific an option is.

It is also possible to use the header to combine different ways to manipulate and compress the data on. The simplest variant could be a 1bit header where '0' meant the raw 12bit sample while a '1' means a delta compressed sample of say 6 bits.

An added benefit of variable delta compression is that it can be used safely and effectively with the medium detector. Any period with great delta will be covered by a series of larger sample sizes, while the small delta of the decay requires much smaller sample sizes. Due to its worse temporal resolution, it can be beneficial to have shorter blocks of data to limit the span of its deltas.

5.5.4 Rollover compression

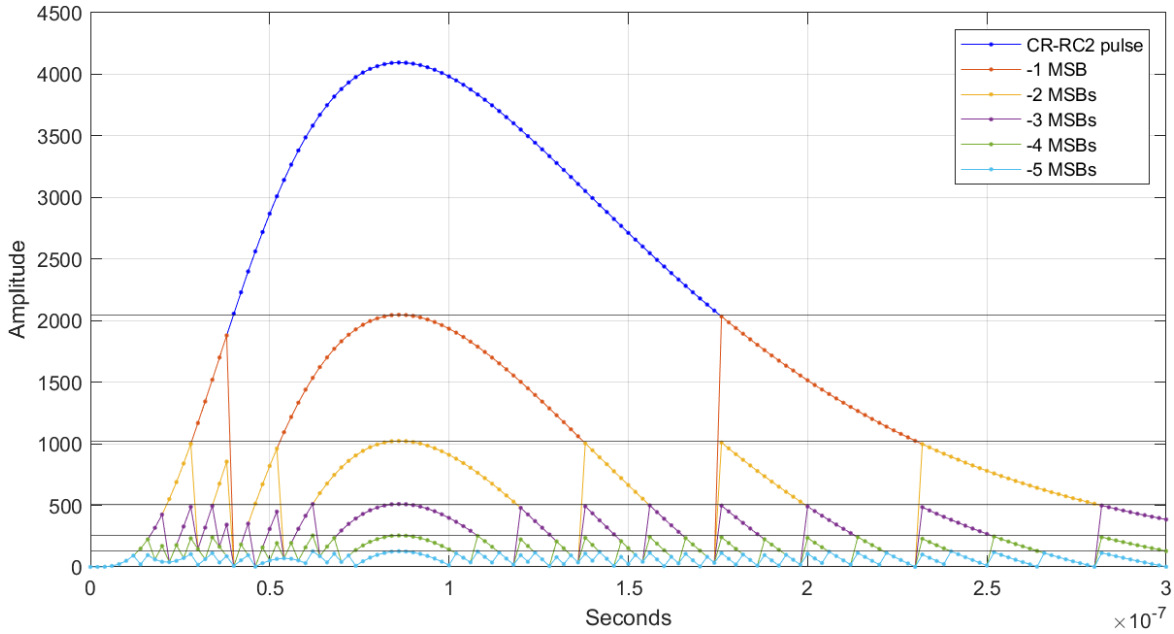


Figure 5-16 An example of increasing amounts of rollover with the CR-RC^2 pulse with maximum amplitude and without noise.

A different approach of using the continuous nature of the signal to reduce the sample size, is to allow controlled rollover by removing one or more MSB from each stored sample. This leads to signals that exceeds the now limited range of the sample wraps around and starts from the bottom. Figure 5-16 shows an example of how this affects a CR-RC^2 pulse with 0, 1, 2, 3, 4 and 5 MSBs removed, halving the available view height for each MSB removed.

While you can reduce the signal with any number of MSBs, it is only truly a compression strategy if it is possible to reliably reconstruct the encoded signal. The challenge of this compression method is therefore to find the limits of when reconstruction is possible. Because this approach is based on similar signal properties as the delta compression, the main focus is the tiny detector unless otherwise specified.

The easiest way to guarantee the reconstruction is to ensure that the view height is larger than twice the greatest delta between samples. This ensures that if the sample experienced rollover, its new position can only have been reached from the last sample through rollover. The individual greatest deltas expected can be found in the maximum and minimum columns back in Table 1. This means that for example the tiny detector with the CR-RC^2 pulse would require 9 bits to be able to guarantee describing any signal rise and 7 bits to guarantee describing any signal fall.

A relevant property of the pulse shapes is the rate at which the delta value can change between, here called *double delta*. Figure 5-17 shows the double deltas of the pulse shapes for the tiny detector. Note that it doesn't show the effect of noise, which is even more amplified than with delta sampling. The point is that the change in delta values from sample to sample is limited, which can be used to know what direction any rollover will take. Over a few samples this should be the case even with noise. The result is that you can reduce the original requirement for the window height from more than twice the height of the maximum delta, to simply greater than the maximum delta. This corresponds to one more bit compressed.

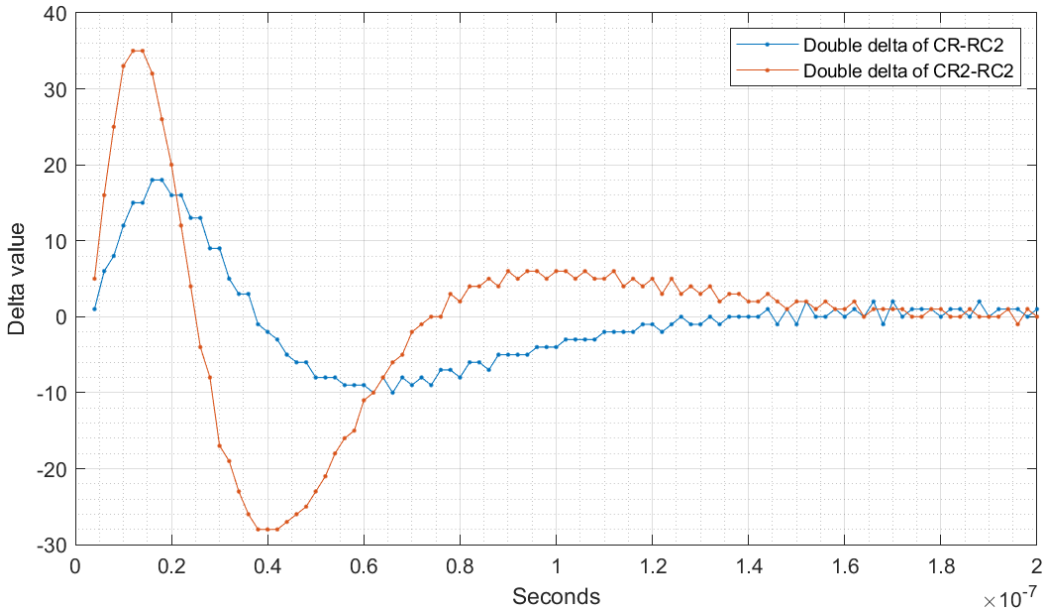


Figure 5-17 Noiseless comparison between the two double delta pulse shapes using the tiny detector. The base pulse shape has maximum amplitude (4096)

A slightly less absolute way to compress the signal is to ensure that both the first and last sample has a known absolute value, something easily done by requiring them to be within the first window. This allows the signal to be decoded from both directions. If the window height was limited by only the minimum delta from Table 1, a pure pulse with maximum amplitude would only have a small part ambiguous while the rest of the signal is known. Knowing the peak and start makes fitting the ambiguous samples to an estimate quite easy. This makes it possible to remove up to two extra bits from the CR-RC^2 pulse, depending on noise floor and margins. For the CR^2-RC^2 pulse it might give enough margin against noise to safely encode the signal to 8 bits.

The concept of the technique above is that ambiguity is manageable for as long as it doesn't cause any following errors (done by working both ways) and is limited to a few samples (that probably can be estimated by knowing the surrounding samples). The problem occurs if there are more than one such ambiguous section during an event. This effectively creates floating sections that are unambiguous relative to each other, but whose absolute value is unknown. In most cases it might be easy to estimate what values they must have based on context, the more such breaks there are, the more difficult it becomes.

One way to solve this problem is to vary how many MSBs that are removed from sample to sample, preferable in a fixed pattern to avoid requiring a pattern. This allows a series of highly compressed samples to effectively anchored by absolute or lightly compressed samples. This furthermore ensures there are a manageable number of possible combinations of rollover variants on the samples in between to achieve correct values.

By choosing an effective pattern and always starting (and possible stopping) the cycle in the same spot in the pattern, you can reliably reconstruct the original signal while keeping an excellent compression rate. If desired, you could also design the pattern such that each cycle takes exactly n bytes of data. It is even easy to implement in hardware by creating a state machine or a list that is repeatedly cycled through to determine how many MSBs that should be discarded before sending them off to storage. An example of a rollover cycle patterns for the CR-RC^2 pulse on the tiny detector could be [10,6] averaging 8 bits per sample.

An important observation is that maximum possible delta between every two samples must be equal or less than twice the maximum delta between adjacent samples. This means that if 8bit samples can be used to accurately recreate any possible path, every second sample with 9 bits can be used to do the same. A smaller sample size could then be used in the intermediate steps to save space. The same logic could be used for every third and fourth sample, but much more than that is not recommended before the double delta starts to get sufficiently large to introduce other types of uncertainty. On the other hand, the effects of noise should stay reasonably consistent independently of how many samples there are in between. This would allow the pattern in the example above to be expanded safely to [10,6,6,6].

The cyclic rollover compression method was discovered quite late in the thesis workflow, so the detail of what patterns could be well suited have not been properly explored. Because it is cyclical, it is effectively a fixed sample length compression method, assuming it always finish a begun cycle.

5.5.5 Huffman coding

Huffman coding is a well-known method for lossless compression. It works by finding the probabilities of each symbol in a string, and then creating a reference tree (Huffman tree) where the more frequent symbols have shorter addresses and the less frequent have longer addresses. You can then create a

string where each symbol is swapped by its reference, compressing its size if some symbols are more frequent than others. In this context a symbol is a single sample or a short collection of them, while a string would be either the entire event or fixed number of samples.

While being a good and computationally easy compression method, Huffman coding face several challenges to be a viable option in this use case.

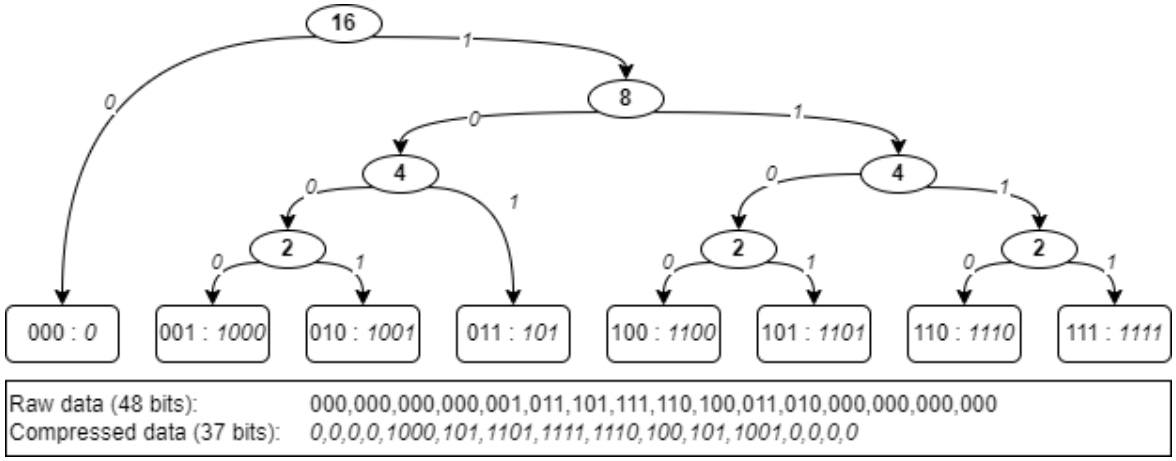


Figure 5-18 An example of a Huffman tree and how it can compress a string of data. The italic numbers are the symbol references. The compression improves with larger differences in symbol frequency.

The first and most obvious is that it requires a Huffman tree, seen in Figure 5-18, which normally is made by scanning through the string while counting the number of instances for each symbol and generating it from there. This is not viable to do on the capture card, especially if the string contains the entire event.

One solution is to generate a Huffman tree beforehand through simulation, removing the need for buffering and calculating the tree. This also has the added benefit that the tree will be known by the decoder removing the need to also store it. In operation each sample would be compared to a lookup register containing the tree, and its assigned reference would be stored in its place.

The next major problem is that Huffman coding is most effective when the majority of the string consists of a minority of the symbols, something not necessarily the case with a signal, especially with noise. Combined with a high amplitude resolution the odds of multiple samples having the same value is low. Some potential solutions to improve this will be mentioned below.

Another potential problem is the size of a Huffman tree required to cover all possibilities. One way to reduce the size of the tree is to only use Huffman encoding on the n MSBs while adding the remaining LSBs as a tail. This also in large parts solves the last problem, as most of the detail and noise will be contained in the ignored LSBs, making the likelihood of multiple samples of equal value much higher.

One potential strategy is to use Huffman encoding on the delta values rather than the absolute values. This has as benefit a reduced range of possible values as well as reducing the spread of the most likely values.

Even with all these options to choose from a bad fit between the data and the Huffman tree, the processed data will end up larger than original. For this reason, it can be beneficial to compare the size of a fixed number of raw samples and Huffman encoded samples. These can then be differentiated with a header like how variable delta compression is done. If a larger (2+ bits) header is used it is even possible to have multiple different Huffman trees 'compete' in parallel, each optimised for different situations.

5.5.6 Bitwise run-length encoding

Run-length encoding is a lossless compression method where the concept is to record how many times a symbol is recorded in succession and rather save each samples individually. For example, "wwwwwwwwwwww" could be saved as "w12". The challenge of the method when dealing with individual bits is to select a range of possible numbers that are large enough to be able to represent long strings of repeating symbols while being short enough to not bloat the final compression when there are few repeating symbols.

Run-length encoding works best when the data compressed consists of long stretches of repeating symbols, something not commonly present in the noisy signal that is expected. It is however possible compress the signal by using run-length encoding bitwise on a buffered series of samples.

To illustrate the point assume you have a signal consisting of 4-bit samples, here shown MSB to LSB as **abcd**, with lowercase letter meaning it has the value '0' and uppercase represents a '1'. So **aBCd** means the sample is '0110'.

Say the signal received is **abcd abcD aBcd aBcD aBCD abCD abCd abcd**. By using run-length encoding on each bit position individually you could encode the signal as **a**8+**b**2**B**3**b**3+**c**4**C**3**c**1+**d**1**D**1**d**1**D**3**d**2. This in itself is not particularly helpful, but say you defined the following rules:

- The bits **a**, **b**, **c**, and **d** each are followed by a number *n* that consists of 3, 2, 2, and 1 bit(s) respectively.
- The value of *n*+1 represent how many times the preceding bit value should be repeated
- If *n* is not its maximum possible value (or the block that is encoded has reached the end), assume the subsequent *n* describes the opposite bit value current one without specifying it

Following these rules, the above signal can be encoded to **a**111 **b**011010 **c**11**C**1000 **d**00**D**101 and would require 26 bits to be stored compared to the original 32.

Like the example, the actual sample will have (on average) the least variation of the MSB and the most variation of the LSB. This means that you can vary the number of bits used to describe *n* with the most for the MSB and few to non for the LSBs. Due to noise, it might be preferred to leave several of the LSBs unencoded.

It does have the weakness that if the signal oscillates around a value that is an exponent of 2, the encoded signal can be larger than the original. This makes adding a header as described in previous sections beneficial.

Bitwise run-length encoding is suitable for both the tiny and the medium detector, although the tiny detector can probably dedicate more bit to *n*, making it proportionally more effective.

5.6 Discussion

This chapter has looked at different approaches and methods to reduce the buffer capacity required should an instrument experience a TGF. There are multiple considerations to take when choosing an option.

On the one hand the more details you can preserve the more options you got when analysing the data offline. On the other hand, should the data obtained during an TGF exceed the buffers you have available large parts of the potential data will becomes lost. For this reason, a balance must be found

between keeping enough detail to allow a good analysis and compressing the data to allow it to be buffered if a long TGF should occur.

There is also the fact that not all information is equal. The minimum requirement is to gain enough information to reliably find the number of pulses, their times, and, for the medium detector, their energy. It is however not trivial to find these things when pileup occurs, and the more information that is available the more options there are to find them.

The first approach leans far in the 'make sure there is no buffer overflow' direction by trying to extract the relevant data needed directly. It does however have quite a few limitations and leaves little to further analysis. Even limited to the tiny detector that has limited value of the pulse amplitudes I would claim that it should not be the primary sampling method unless the buffer capacity is very low.

It does however have a potential use as a secondary sampling method with the purpose of counting the number of events detected. This is functionality the second approach lacks and is useful to know for the operator to monitor the situation and detector.

The second approach meanwhile tries to lean as far to the 'keep all the details' side as feasible to allow great freedom of analysis offline. The goal then is to reduce the 72% of available BRAM usage down to a more manageable level. Most of this effort is focused on the tiny detector, as it itself requires 63% of the available BRAM.

To achieve this, one or more compression algorithms needs to be chosen. Table 2 briefly summarises the compression methods previously discussed, as they would work for the tiny detector. Due to each sample being 12 bits, each bit compressed frees about 5% of the total available BRAMs.

Lossless / Fixed size **Method**			**Average compression potential per sample**	**Other comments**
Down-sampling	No	Yes	Reduce sample number to or by 1/M, where M is an integer	Will increase maximum and minimum delta
Rounding	No	Yes	1 bit per sample	Stacks with any other method here
Fixed delta compression	Yes*	Yes	About 3b for CR-RC^2 and about 2b for CR^2-RC^2	*: Better if potential loss of data via limited slew rate is ok
Variable delta compression	Yes	No	Maybe between 2b and 4b better than the fixed version	Can be combined with the other variable length to further optimise compression
Constant rollover compression	Yes	Yes	Up to ~5b for CR-RC^2 and up to ~4b for CR^2-RC^2	Noise dependent. Can cause uncertainty if unlucky, but it should be mostly recoverable
Cyclic rollover compression	Yes	Yes	Maybe 1 or 2 bits better than the constant version	Less than the constant version, but still noise dependent. Mostly negates the uncertainty
Huffman coding	Yes	No	Difficult to say without simulation. Better if of delta	Possible candidate for a complex variable length algorithm
Bitwise run-length encoding	Yes	No	Difficult to say without simulation	Possible candidate for a complex variable length algorithm

Table 2 Summary of how the much compression can be achieved for the tiny detector. The numbers are rough estimates that may change based on factors like noise.

Down-sampling is undoubtedly the most effective compression method of the bunch, particularly when factoring to the effort. Figure 5-12 showed that even the CR^2-RC^2 is more than sufficiently sampled at M = 2, 3 and potentially even 4. In BRAM usage this is 31%, 21% and 16%. There are however downsides. The primary goal of the tiny detector is to count the number of detections. As previously mentioned, a good way to do this is to find the delta peaks. The more samples you got, the better can you smooth the signal, and the easier it is to detect any delta signal peaks. How much this is worth it is questionable, as there are diminishing returns on oversampling. A sort of compromise could be to record the average of the samples rather simply discarding them. If down-sampling is done it will affect most of the other methods through decreasing the temporal resolution.

If there is noise in the signal and this can't for some reason be removed by other filters, rounding might be a potential action. It does however reduce the amplitude resolution, which make pulses with small amplitude harder to isolate. On the plus side it doesn't affect any of the other methods as they mostly try to compress the signal via the MSBs.

When it comes to the lossless methods there are two main paths, fixed sample size and variable sample size. Of the two, the best compression should be achieved by the variable size method, as it easily can be set up to choose whatever method works best by testing them in parallel. There is however a large problem with it, namely complexity. Complexity in the algorithms doing the compression, as well as complexity in the chunk assembly stage because the variable sample size. The more complex something is, the more parts there are to design, to test and that can go wrong. The tiny increase in compression is not worth it. Meanwhile the fixed size ones are simple and predicable. Of the presented methods cyclic rollover are slightly ahead in both potential compression as well as 'safety'.

With all of this in mind, for the tiny detector I would recommend as a starting point a M = 2 down-sampling combined with a cyclic rollover aiming for about 3-4 bits (depending on noise) saved per sample. Should the noise be sufficient, rounding could also be done freely. This gives a final buffer requirement of 0.5*8/12 = 1/3 the original, so about 20% of the BRAM capacity. This would mean 2 MB of compressed data would be produced during a 1ms TGF. There is however still more potential for compression by improving the cycle pattern.

For the medium detector compression is much less important. It also has more limited options among the methods explored in this thesis. With down-sampling it as a wider selection compared to the tiny detector due to the options available from the ADC, but it is already on the limit of what its temporal resolution can gain. The same limited temporal resolution also limits most of the methods using the limited delta available. The variable length methods may offer decent through situational optimisation, the problem of complexity is also present here.

As such I would recommend as a baseline to either store the data raw, or at most use a cyclic rollover pattern that compress a couple of bits.

6 Conclusion and outlook

A new ALOFT campaign is planned for 2023 and will contain improvements such as more optimal flight time and area, two additional new detectors, and an improved capture card. The new capture card is based around a Xilinx ZYNQ-7z030 SoC, and will receive data from all the detectors, pre-process it as required, and send it to mass storage.

The objectives of this thesis have been (1) to verify that the built in SPI controller of the ZYNQ can be used to configure the control registers of the new detectors, and (2) find methods of reducing the required buffer capacity preventing buffer overflow during TGFs.

6.1 ADC control

The verification that the ZYNQ SPI controller could be used configure the ADC control registers was divided into three steps. The first step was to show that the four wire SPI of the ZYNQ could be correctly communicate with the three wire SPI interface used by the ADCs. This was done by showing how a simple converter could be implemented in the FPGA.

The second step was to create testbench models of the two ADC control registers such that it is possible to verify that any communication from the ZYNQ to the ADC control registers has the intended behaviour. This was done by creating VVCs based on UVVM.

The third step was to ensure that the ZYNQ SPI controllers worked and could be routed to the PL. This was done by creating a SPI loop that connected the two SPI controllers together via the PL and testing that it could send data with the same protocols as the used by the ADC control registers.

Combined the three steps shows that ZYNQ SPI controller is capable of connecting to the ADCs via a SPI3 converter in the PL, as well as handle the protocols required. This makes it a viable choice.

6.1.1 Future work

The next natural step is to combine the three steps and simulate the ADC controllers in hardware on the PL. It was originally the plan do this step as part of the thesis, but due to changing priorities it was set aside for the second objective of this thesis.

A different avenue for future work is to create the user interface for easy configuration of the ADCs. This would most likely be in the form of software functions calls that automatically exchange the required data to the correct control register addresses to use the predetermined functionality, as well as some general 'send this data to this address' type calls for more specific used. With the switch to PYNQ this should now easily be possible in Python with minimum work required.

6.2 Signal recording

To buffer all the raw data generated by the new detectors during 1ms, which should be long enough to guarantee a TGF has finished, 6.9 Mb, or 73%, of the available of BRAM would be required. To reduce the required buffer capacity two main approaches was explored: online analysis and data compression.

Online analysis tried to a) give count the number of underlying pulses in an event; b) detect when there is a pileup; and c) enable the calculation of pulse energies when there is no pileup. The analysis does this all through recording the amplitude and time of the signal peak and the delta peaks, both of which are computationally light operations. The delta peaks give a direct measurement of a), b) is found by comparing the number of signal and delta and c) is found using the signal peaks. The method is however very sensitive to noise making it a bit too uncertain for use as the primary data-collection

approach. It does however have potential, if sufficiently filtered, to give live estimates of the number of detections

When exploring data compression multiple different compression algorithms was explored, as well suggestions on how to determine an event is happening and a potential way to structure the entire pre-processing part of the detectors. Because the tiny detector alone produced 6 Mb of raw data, it was the main focus of the compression effort.

The compressions explored was a mixture of lossless or lossy, and fixed or variable sample size. The lossless vs lossy says something about whether the compressed data could be perfectly recreated when decompressed or not. The variable vs fixed sample size says whether the compressed samples always have a predictable number of bits or if it unpredictable and varies between samples. None of lossy methods explored where variable sized.

It was argued that due to complexity in both implementing the methods themselves and in the surrounding infrastructure that a variable sized compression method was undesired despite having the potential to be slightly better than the fixed size counterparts.

The compression methods that stood out as best where down-sampling, cyclic rollover compression, and possibly rounding should the noise be sufficient. Of these, only the cyclic rollover is fitting for the medium detector.

The configuration recommended for the tiny detector is a down-sampling to 250 MSPS, which can be implemented via the ADC control registers, a cyclic rollover with up to 4 bits per sample compression and, should the noise floor be high enough, a 1 bit rounding. The medium detector can safely handle 1 maybe 2 bits of compression with the cyclic rollover compression. This reduces the buffer requirement down to 1/2 * 8/12 * 6 Mb + 12/14 * 0.91 Mb = 2.78 Mb, or 29% of the available BRAM.

6.2.1 Future work

While a good compression method was found, it is far from the only step required to finish a pre-processing module.

First is the cyclic rollover pattern itself. Due to being discovered late in the process there was no time to find an optimised pattern. For example, an optimised cyclic rollover pattern should allow 6 bits of compression on average per sample if down-sampling turns out to be undesired.

The next step would be to simulate a full TGF with an appropriate amount of noise to confirm that the chosen pattern can be decoded into the original signal. When the noise is appropriately estimated it is also possible to find a threshold for the event detector.

Next is the actual implementation in VHDL. A nice benefit of all the compression methods chosen is that they are rather simple.

There is also the larger structure of designing and developing the data-chunk assembly. Questions that need answering include how many samples there should be per chunk, what metadata is required, how should it be structured and how should it handle if an event is over before a chunk is filled.

References

1. Helliwell, R.A., *Whistlers and related ionospheric phenomena*. 1965, Stanford, Calif.,: Stanford University Press. viii, 349 p.
2. Briggs, M.S., et al., *First results on terrestrial gamma ray flashes from the Fermi Gamma-ray Burst Monitor.* Journal of Geophysical Research-Space Physics, 2010. **115**.
3. MountVisual, B.C.f.S.S.a. *Simultaneous detections of TGFs and Elves by ASIM*. 2022; Available from: https://birkeland.uib.no/simultaneous-detections-of-tgfs-and-elves-by-asim/.
4. Fishman, G.J., et al., *Discovery of Intense Gamma-Ray Flashes of Atmospheric Origin.* Science, 1994. **264**(5163): p. 1313-1316.
5. Smith, D.M., et al., *Terrestrial gamma-ray flashes observed up to 20 MeV.* Science, 2005. **307**(5712): p. 1085-1088.
6. Marisaldi, M., et al., *Detection of terrestrial gamma ray flashes up to 40 MeV by the AGILE satellite.* Journal of Geophysical Research-Space Physics, 2010. **115**.
7. Meegan, C., et al., *The Fermi Gamma-Ray Burst Monitor.* Astrophysical Journal, 2009. **702**(1): p. 791-804.
8. Ostgaard, N., et al., *First 10 Months of TGF Observations by ASIM.* Journal of Geophysical Research-Atmospheres, 2019. **124**(24): p. 14024-14036.
9. asim.dk, *iss066e123349*.
10. Parks, G.K., et al., *X-Ray Enhancements Detected during Thunderstorm and Lightning Activities.* Geophysical Research Letters, 1981. **8**(11): p. 1176-1179.
11. Eack, K.B. and W.H. Beasley, *Long-duration X-ray emissions observed in thunderstorms.* Journal of Geophysical Research-Atmospheres, 2015. **120**(14): p. 6887-6897.
12. Wada, Y., et al., *Meteorological Aspects of Gamma-Ray Glows in Winter Thunderstorms.* Geophysical Research Letters, 2021. **48**(7).
13. NASA. *GOES History*. Available from: https://www.goes-r.gov/mission/history.html.
14. Quick, M., et al., *Fly's Eye GLM Simulator (FEGS)*. 2016. p. AE23A-0405.
15. Gibbs, Y. *NASA Armstrong Fact Sheet: ER-2 High-Altitude Airborne Science Aircraft*. 2014 [cited 2022 04.05]; Available from: https://www.nasa.gov/centers/armstrong/news/FactSheets/FS-046-DFRC.html.
16. Ostgaard, N., et al., *Gamma Ray Glow Observations at 20-km Altitude.* Journal of Geophysical Research-Atmospheres, 2019. **124**(13): p. 7236-7254.
17. Collaboration, T.A., *Airborne Lighting Observatory for FEGS and TGFs Campaign (ALOFT)*, in *Internal white paper*. 2020.
18. NASA. *GBM Terrestrial Gamma-ray Flashes (TGF) Catalog*. 2018 [cited 2022 27/5]; Available from: https://fermi.gsfc.nasa.gov/ssc/data/access/gbm/tgf/.
19. Marisaldi, M., et al., *On the High-Energy Spectral Component and Fine Time Structure of Terrestrial Gamma Ray Flashes.* Journal of Geophysical Research-Atmospheres, 2019. **124**(14): p. 7484-7497.
20. Mailyan, B.G., et al., *The spectroscopy of individual terrestrial gamma-ray flashes: Constraining the source properties.* Journal of Geophysical Research-Space Physics, 2016. **121**(11): p. 11346-11363.
21. Hansen, R.S., et al., *How simulated fluence of photons from terrestrial gamma ray flashes at aircraft and balloon altitudes depends on initial parameters.* Journal of Geophysical Research-Space Physics, 2013. **118**(5): p. 2333-2339.
22. Ostgaard, N., et al., *The true fluence distribution of terrestrial gamma flashes at satellite altitude.* Journal of Geophysical Research-Space Physics, 2012. **117**.
23. Inan, U.S., et al., *Terrestrial gamma ray flashes and lightning discharges.* Geophysical Research Letters, 2006. **33**(18).

24. Dwyer, J.R., D.M. Smith, and S.A. Cummer, *High-Energy Atmospheric Physics: Terrestrial Gamma-Ray Flashes and Related Phenomena.* Space Science Reviews, 2012. **173**(1-4): p. 133-196.
25. Knoll, G.F., *Radiation detection and measurement*. 3rd ed. 2000, New York: Wiley. xiv, 802 p.
26. Advatech. [cited 2022 04.04]; Available from: https://www.advatech-uk.co.uk/scintillators-crystals_a_c.html.
27. ON-Semiconductor, *AND9770/D Introduction to the Silicon Photomultiplier (SiPM).* https://www.onsemi.com/pub/Collateral/AND9770-D.PDF, 2011.
28. ON-Semiconductor, *Silicon Photomultipliers (SiPM), High PDE and Timing Resolution Sensors in a TSV Package.* 2021.
29. ON-Semiconductor, *Silicon Photomultiplier (SiPM) High Fill-Factor Arrays.* 2021.
30. Ramdal, L.A.K., *Characterization of Silicon Photomultiplier and Design of Front-End Electronics for ALOFT*, in *Department of Physics and Technology*. 2022, University of Bergen.
31. Instruments, T., *ADS5404 Datasheet.* https://www.ti.com/lit/gpn/ads5404, 2014.
32. DEVICES, A., *AD9257 Datasheet.* https://www.analog.com/media/en/technical-documentation/data-sheets/AD9257.pdf.
33. XILINX, *UG585 Zynq Technical Reference Manual.* https://docs.xilinx.com/v/u/en-US/ug585-Zynq-7000-TRM, 2018.
34. XILINX. *PYNQ: Python productivity for Xilinx platforms*. 2022 [cited 2022 22/04]; Available from: https://pynq.readthedocs.io/en/v2.7.0/index.html.
35. FreeRTOS. *The FreeRTOS™ Kernel*. 2022 [cited 2022 25.04]; Available from: https://www.freertos.org/RTOS.html.
36. Motorola. *SPI Block Guide*. 2004; Available from: https://web.archive.org/web/20150413003534/http://www.ee.nmt.edu/~teare/ee308l/datasheets/S12SPIV3.pdf.
37. DEVICES, A., *AN-877 APPLICATION NOTE.* 2017.
38. bitvis. *UVVM VVC Framework Manual*. 2017 [cited 2022; Available from: https://github.com/UVVM/UVVM/blob/master/uvvm_vvc_framework/doc/VVC_Framework_Manual.pdf.
39. Asbjørn Magnus Midtbø, H.T.L., *Design of an embedded data storage system using PYNQ*. 2021, Western Norway University of Applied Sciences.