Karlsruhe Institut für Technologie
Fakultät für Elektrotechnik und Informationstechnik

DATA ACQUISITION FOR GERMANIUM-DETECTOR ARRAYS

Zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN (DR.-ING.)

von der KIT-Fakultät für Elektrotechnik und Informationstechnik

des Karlsruher Instituts für Technologie (KIT)

genehmigte

DISSERTATION

von

Mario Schütt, M.Sc.

geboren in Freiburg im Breisgau, Deutschland

Tag der mündlichen Prüfung:	26.10.2022
Erstgutachter:	Prof. Dr. Ivan Peric
	Institut für Prozessdatenverarbeitung und Elektronik
Zweitgutachter:	Prof. Dr. Werner Hofmann
	Max-Planck-Institut für Kernphysik, Heidelberg

Declaration of authorship

I declare that I have developed and written the enclosed thesis in hand completely by myself, and have not used sources or means without any declaration in the text.

Erklärung der Selbstständigkeit

Hiermit versichere ich, die vorliegende Arbeit selbständig angefertigt, alle dem Wortlaut oder Sinn nach entnommenen Inhalte anderer Werke an den entsprechen Stellen unter Angabe der Quellen kenntlich gemacht und keine weiteren Hilfsmittel verwendet zu haben.

Mario Schütt
Heidelberg, den 08.08.2022

Zusammenfassung

Die Wandlung von analogen zu digitalen Signalen und die anschließende online/offline Verarbeitung ist die technologische Voraussetzung zahlreicher Experimente. Für diese Aufgaben werden häufig sogenannte Analog-Digital-Wandler (ADC) und FPGAs („field-programmable gate array“) eingesetzt. Die vorliegende Arbeit beschreibt die Evaluierung der FPGA und ADC Komponenten für die geplante FlashCAM 2.0 DAQ (FC2.0 DAQ). Die Entwicklung der ersten FlashCAM (1.0) DAQ (FC1.0 DAQ) wurde unter Federführung des Max-Planck-Instituts für Kernphysik im Jahre 2012 begonnen und war ursprünglich eine exklusive Entwicklung für das Cherenkov Telescope Array (CTA) Experiment. In der Zwischenzeit wird FlashCAM in zahlreichen Experimenten (HESS, HAWK, LEGEND-200, etc.) eingesetzt, die sowohl Photomultiplier (PMTs) als auch High Purity Germanium (HPGe) Detektoren umfassen. Beide Detektorentypen unterscheiden sich massiv in ihren Anforderungen und können auch von der neuen DAQ abgedeckt werden.

Das Themengebiert der Arbeit umfasst den gesamten funktionellen Umfang einer modernen DAQ. Moderne DAQ Systeme benötigen eine möglichst hohe Read Out Performance zwischen dem DAQ Board und dem es kontrollierenden Server. Die Umsetzung eines leistungsfähigen Firmware Designs und das Design einer hierauf angepassten Hardware/Softwareschnittstelle wird am Beispiel der Zynq Familie vorgestellt. Die Zynq-Familie von Xilinx ist von besonderem Interesse, da der Hardwarehersteller Trenz Elektronik ein flexibles, einfach aufsteckbares Modulkonzept mit verschiedenen SoCs der Zynq-Serie anbietet. Neben der Read Out Performance einer DAQ ist ihre Auflösungsgrenze von entscheidender Bedeutung für das Gelingen des finalen Experiments. Die verwendete FADC Karte muss sich daher durch exzellente SNR und Linearitätseigenschaften auszeichnen. Die Evaluierung solcher FADC Karten setzt ein Testsetup voraus, dass in Signalreinheit und Stabilität die hohen Anforderungen der devices under test übertreffen muss. Praktisch sind diese Bedingungen nur unter hohem (Kosten) Aufwand erreichbar. Im Rahmen der Arbeit wurden daher auch alternative Testkonzepte entwickelt, die mit akzeptablen Abstrichen in der Genauigkeit eine Messung im experimentellen Umfeld ermöglichen können. Da sich die Themengebiete in ihrem Inhalt deutlich unterscheiden, wurde die vorliegende Arbeit in zwei Themenkomplexe aufgeteilt. Der erste Teil der Arbeit beschäftigt sich mit dem Einsatz der Zynq Familie in der geplanten „FlashCAM“ Nachfolger DAQ. Der zweite Teil widmet sich der ADC Nichtlinearitätsbestimmung.

Die wichtigsten Ergebnisse der Arbeit lassen sich folgt zusammenfassen:

- Die „High Performance“ (HP) Schnittstellen der Zynq-UltraScale+ haben eine aussetzerfreie Bandbreite von 2.4 GB/s in den externen Arbeitsspeicher der Trenz Module. Wird noch zusätzlich die standardmäßig vorhandene 1 Gb PS-Ethernet Verbindung betrieben, verbleibt der CPU noch eine Bandbreite von mindestens 0.5 GB/s in den Arbeitsspeicher. Im Fall der Zynq-7000 Serie ist eine effiziente Implementierung der HP Schnittstellen schwierig, da die CPU nur vergleichsweise

niedrige Arbeitsspeicherzugriffsraten erreicht. Die HP Schnittstellen sind eine wichtige Designalternative da ein durchgehender Datentransfer in den externen Arbeitsspeicher ein Design ermöglichen würde dass weniger stark durch den verfügbaren FPGA internen Speicher begrenzt ist. Dies wäre besonders für Anwendungen in der HPGe-Spektroskopie wünschenswert, da der praktische Nutzen des verwendeten Designs stark von der zur Verfügung stehende Puffergröße abhängt.

- Die “Accelerator Coherency” Schnittstelle (ACP) ermöglicht ein direkter Datentransfer aus der FPGA in den Cache der Zynq-CPU. Die entworfene ACP-CMA hat eine Bandweite von bis zu 2.4 GB/s und bietet für Cache-CPU Zugriffe noch ausreichend Reserve. Dass die Zynq-CPU die Cachedaten ohne ein Abwürgen der ACP-CMA verarbeiten kann, ist entscheidend. Wäre dies nicht der Fall könnte die CPU im Parallelbetrieb von Ethernet und ACP-CMA nicht die notwendigen Vorarbeiten zur Ethernet-Übertragung („Event Building“) bewältigen. In der Evaluierung wurde eine maximale Event Building Bandbreite von 0.7 GB/s festgestellt. Wahrscheinlich ist die reale maximale Bandbreite deutlich höher anzusiedeln. Einschränkend muss betont werden, dass in praktischen Applikationen zusätzliche Einschränkungen in Kraft treten, die de-facto einen kontinuierlichen Betrieb der ACP-CMA unmöglich machen. Diese Einschränkungen – die nicht prinzipieller Natur sind - wurden in der durchgeführten Ermittlung nicht berücksichtigt. Da weiterhin alle Zynq-FPGAs über einen Cache verfügen, ist die ACP-CMA eine Designlösung, die auf allen verfügbaren Zynq-FPGAs sinnvoll implementiert werden kann. Dies unterscheidet sie von der entwickelten HP-DMA, die häufig nur für Implementierungen in einer Zynq-UltraScale FPGA interessant ist.
- Der neuentwickelte FC2.0 Prototype wurde bereits in experimentellen Setups eingesetzt. Als Anwendungsbeispiel dient die Messung und Analyse eines γ-ray Spektrums eines HPGe-Detektors.
- Der Erfolg einer ADC Nichtlinearitätsbestimmungen ist stark von der Signalreinheit des verwendeten Eingangssignal abhängig. In Simulationen konnte gezeigt werden, dass die neu entwickelten Verfahren nur relativ schwach durch Pulsernichtlinearitäten verfälscht werden. Einen praktischen Vergleich zwischen den neuen und einer klassischen Methode konnte keinen signifikanten Unterschied feststellen. Die untersuchten Methoden können daher für eine zukünftige Implementation in FC2.0 empfohlen werden.

Abstract

Experiments often need to digitize analog signals that are then processed either online or offline. The data acquisition (DAQ) for such systems needs analog-to-digital converters (ADC) and the digital data are processed by field programmable gate arrays (FPGA). The thesis describes the evaluation of the FPGA and ADC devices of the designated FlashCAM 2.0 DAQ (FC2.0 DAQ).

2012 started the development of the first FlashCam DAQ (FC1.0). Initially FC was an exclusive development for the Cherenkov Telescope Array (CTA) Experiment and therefore fixed to photomultiplier tube detectors (PMTs). In the meantime, FC is implemented in several experiments (HESS, HAWK, LEGEND-200, etc.) and additionally has high-pure-germanium-detector (HPGe-detector) features. Both detector types significantly differ in their requirements and both detector types must be handled by the successor DAQ.

In the thesis the development of a DAQ firmware design is presented on instance of the Zynq System on the Chip device family. The Zynq-devices from Xilinx are of particular interest as Trenz Electronic has a hereupon basing flexible module system. Besides the performance of its implemented firmware design, the feasibility of a DAQ relies on its achievable resolution limits. As the resolution limit depends on the signal-noise-ration (SNR) and nonlinearity properties of the DAQ implemented FADC card, its accurate determination is especially important. The evaluation of such eligible FADC cards requires a test setup whose signal integrity and nonlinear behavior must exceed the already high requirements of the FADC under evaluation. In practice, eligible test setups require high spendings and efforts that mostly prevent an implementation of such a setup in an experimental environment. A focus of this thesis was the development of alternative nonlinearity determination procedures that facilitate a sufficient precision with default instrumentation. As the firmware and the ADC evaluation topics significantly differ, it was decided to split the thesis in two segregated parts. The first part is centered around the Zynq SoC devices as substitute of the current FlashCAM DAQ. The second part is dedicated to ADC nonlinearity determination.

Main results of the evaluation are

- The “High Performance” (HP) ports of the Zynq-UltraScale+ devices have a total interrupt free memory bandwidth of at least 2.4 GB/s. In enhanced designs with a parallel running 1 Gb/s ethernet unit, the CPU remains a concurrent memory read-write bandwidth of at least 0.5 GB/s. In Zynq-7000 devices an interrupt-free implementation of the HP ports can be thwarted by the poor memory access performance of the CPU. As the HP ports have direct access to the external DDR memory their interrupt-free bandwidth can be used to develop designs that do not require FPGA internal

RAM. This is especially useful in HPGe-spectroscopy, as where the digitized waveform length and hence the event data buffer capacity is decisive.

- The “Accelerator Coherency” Port (ACP) enables (interrupted) data writes into the CPU Cache of the Zynq devices. The developed ACP-CMA framework has a bandwidth of up to 2.4 GB/s. The CPU can access the data without stalling a concurrent ACP operation in all Zynq devices. This is important as the CPU must handle all data processing requirements (“event building”) of the ethernet transfer. A concurrent maximum event building performance of 0.7 GB/s was determined. The real maximum event building performance is likely higher. It should be noted that a real application can face limitations that prevent a continuous operation of the ACP-CMA. These limitations – that are not fundamental – were not considered in the event building performance investigation. As all Zynq devices have a performant CPU Cache, the ACP-CMA is a feasible solution for all available Zynq devices.
- The new designed FC2.0 prototype firmware design was already used in experimental setups. As application example a HPGe-detector measurement will be shown.
- ADC nonlinearity investigation setups strongly depend on the distortion level of their implemented signal source. In simulations was shown that INL determination procedures exist whose accuracy is only slightly degenerated by the implementation of a nonlinearity distorted pulser. A practical comparison of these new methods with a classic INL determination procedure roughly confirmed the determined INL trend of the new procedures. The methods therefore can be recommended for a practical implementation in future ADC evaluation campaigns.

Acknowledgment

At this spot I would like to thank all persons who supported me during my time at MPIK. Many persons have contributed to the success of my thesis, but unfortunately due to lack of space I am not able to mention them all.

First, I would like to thank to Ivan Peric and Werner Hofmann for their willingness to be the referees of my thesis. In my first suggestions the feasibility of the desired new DAQ was utterly unclear as a lot of information was missing and no similar publication was available. Regardless Professor Peric immediately offered his support. The adaption of the originally suggested DAQ design (cheap Zynq-7000 device whose firmware transfers a continuous data stream into memory and has an option to do online analysis) soon proved to be not feasible. Though this unexpected outcome, both referees were still eager to participate in the development process of a modified FlashCam 2.0 prototype. For this support I want to express my special gratitude.

Furthermore I would like to express my deepest sense of gratefulness to my supervisor Bernhard Schwingenheuer. His insightful comments, valuably advices and unremitting commitments are the pillar of this thesis.

I am thankful to my fellows Thomas Kihm, Michael Bantel, Christian Bauer and Christian Föhr. Their substantial knowledge of FlashCam (and many other loosely connected topics) was the basis requirement for the succeeded thesis.

Table of Content

List of abbreviations

ADC	Analog to Digital Converter
Amp	Amplifier
APU	Application Processing Unit
AXI4	Advanced eXtensible Interface 4 (protocol)
CM	Common Mode
CSA	Charge Sensitive Amplifier
DAC	Digital to Analog Converter
DAQ	Data acquisition (system)
DC	Direct current
DDR	Double Data Rate
DMA	Direct Memory Access
DMM	Digital multimeter
DNL	Differential Nonlinearity
DSP	Digital Signal Processing
EMI	Electromagnetic interference
FC	FlashCam
FPGA	Field Programmable Gate Array
FR	Full range
FWHM	Full width at half maximum
GERDA	Germanium Detector Array (experiment)
HPGe	High Pure Germanium (detector)
HV	High Voltage

INL	Integral Nonlinearity
IO	Input Output
IP	Intellectual Property
IPC	Inter Process Communication
ISA	Instruction Set Architecture
L1	Level 1
L2	Level 2
LSB	Least significant bit
LVDS	Low-voltage differential signaling
MGT	Multi Gigabit Transceiver
MPIK	Max-Planck-Institut für Kernphysik (Heidelberg-Germany)
OS	Operation System
PL	Programmable Logic
PMT	Photomultiplier Tube
PS	Processing System
RAM	Random Access Memory
RMS	Root meets square
SNR	Signal-Noise-Ration
SoC	System on the Chip
TUM	Technische Universität München (Munich, Germany)
Z7k	Zynq-7000
ZU+	Zynq-UltraScale+

List of symbols

$\Delta \boldsymbol{U}_{\boldsymbol{c}_{\boldsymbol{ideal}}}$	Ideal bin width of the ADC
$\boldsymbol{U}_{\boldsymbol{FR}}$	Full range input voltage of ADC
$\boldsymbol{U}_{\boldsymbol{c}}[\boldsymbol{i}]$	Transition code voltage of ADC Code i
$\boldsymbol{G}$	Gain
$\boldsymbol{F}_{\boldsymbol{S}}$	Sample frequency
$\boldsymbol{s}$	(ADC Code) Value of sample
$\boldsymbol{n}_{\boldsymbol{ideal}}(\boldsymbol{i})$	Expected Count Histogram Bin of ADC code *i*
$\boldsymbol{n}_{\boldsymbol{real}}(\boldsymbol{i})$	Real Count Histogram Bin of ADC code *i*
$\boldsymbol{M}$	Waveform size in samples
$\boldsymbol{k}$	Sample position in waveform {0...M-1}
$\boldsymbol{N}$	Bit width of ADC
$\boldsymbol{i}$	ADC code {0...2^N}

1 Introduction

In several experimental collaborations of MPIK, for instance in HESS, CTA and LEGEND-200, the FlashCam (FC) DAQ is a key component for the experiment’s successful operation. FlashCam is especially popular as it is versatile and cheap (in the cheapest setup 70€ per channel). The cheap price and its high scalability are an essential advantage together with the adaptation to experimet specific needs. For instance, in the CTA experiment the FlashCAM DAQ must comprise up to 2304 channels in one global trigger domain. A contrary application example is the use in HPGe-detector evaluation setups of the former GERDA collaboration that comprise merely a single channel. In this example FC is used to due to its high signal quality. Though both experiments differ strongly, their setups have the FC design philosophy in common. They use exactly the same boards and are being controlled by a central (ethernet) read-out server that has the complete read out responsibility. The experiments merely differ in the number of boards and server performance. FlashCam is popular, but due to the estimated end-of-life of the implemented Spartan-6 FPGA a successor development was started. In the meantime, Xilinx strongly promotes its available successors series, including its Zynq device family. The Zynq-device family is especially appealing to the successor DAQ “FlashCam2.0” (FC2.0) as its hard-wired ARM processors could enable an event building and analysis framework directly operating on the DAQ board. Another important point of a DAQ is its FADC card. Using a high-quality ADC is essential, especially in HPGe-spectroscopy experiments like LEGEND. A development of a FC successor is an opportunity to improve the FADC cards. As the progress pace in ADC technology is not as fast as in microprocessors (or FPGAs) the FADC evaluation setup must be capable to quantify reliably smallest distinctions in the signal quality. The demanded high resolution makes such setups particular challenging.

The target of this thesis mainly was mainly the inspection of the Zynq family as potential successor devices for the next generation FC2.0 DAQ. FC2.0 must preserve all supreme properties of FC1.0 and therefore just a limited selection of the Zynq devices is a potential nominee. Note that the device selection was mostly driven by the cost. Besides determining the Zynq device performance by implementation of highly specific test designs, a potential FC2.0 firmware was developed. This “successor” firmware misses merely some specific features of the designated FC2.0 trigger system and otherwise already fulfills all requirements of FC2.0. Furthermore a prototype DAQ with the most likely Zynq nominee was developed. The FC2.0 prototype DAQ uses the new developed firmware and was already used for first HPGe-spectroscopy measurements.

The nonlinearity characterization – especially the integrated nonlinearity (INL) – of an ADC developed into a second central topic of this thesis. In various publications the influence of the ADC nonlinearities and their compensation is elaborated. Also in various works it is described how to measure the INL with a signal pulser of sufficient resolution. But less frequently it is discussed what sufficient resolutions actually means and even less how to respond in cases when such a signal quality is not achievable by additional filtering. The latter will be decisive for INL determinations in experimental environments, at which it is not possible to input a pulser signal of the demanded quality. In this thesis two promising alternative INL determination methods are investigated.

2 General DAQ consideration

The new “FC2.0 DAQ” has the aspiration to be applicable in many experiments, especially those using PMTs or HPGe detectors. A real data acquisition (DAQ) system which can be implemented in a plenty of experiments is mostly not the DAQ system which offers the best price-performance ratio. So experimental designers always used to consider the DAQ (components) costs, the strongly needed experimental requisites and the wishful capabilities and then try to find a balance. Some important DAQ properties in this process are:

- Maximum channel amount (in one clock domain/per board)
- Maximum event rate
- Available Sample Rates and Sample Bit Width
- (Effective) Signal Noise Ratio (SNR)
- Nonlinearity (of signal chain)
- Signal stability (thermal influence)
- Available readout interfaces (Ethernet, VME Bus, PCIe, USB)
- Available trigger and synchronization links
- Temporal trigger resolution
- Available trigger schemas
- Maximum readout bandwidth (between Readout Server and Single Board)
- Software support

The final DAQ design is influenced by detector specific signal processing requirements. Before the advent of the digital signal processing with eligible DSPs and FPGAs the signal processing in pulse spectroscopy was done with analog electronic and just the last step included digitization. A proper modified digital signal processing chain in pulse spectroscopy adopts analog signal processing, but it should not be build up as an 1:1 replica in order to gain the advantages of digital processing. It is important to consider that digital signal processing chain comprises a FPGA with its main task to trigger and record an event and a second segregated signal processing chain in the analysis routine on a segregated analysis server that transforms the event data into an energy spectrum. How to arrange such an interplay is unfortunately poorly discussed in common recommended textbooks like [1] and [2]. At MPIK the DAQ development efforts iterated in the FlashCam DAQ, but FlashCam is for sure not the only possible solution. It is worth to note that the optimal signal processing of PMTs and HPGe-detectors is different, and these differences must be considered in FlashCam too.

Though in all nowadays experiments the resolution is getting improved by enhanced digital filtering methods, a well-designed analog setup of the DAQ is still important. A bad designed analog signal chain can charge the signal with hum and noise so far, that the final analysis conditions are out of reach despite usage of latest digital analysis approaches. Chapter 2.1 is therefore focused on the analog components of the DAQ signal chain which affect the filter settings of the final analysis.

The strong focus on bandwidth in the performance elaboration chapters of this thesis is caused by the linear relation between maximum event rate and readout bandwidth. This relation is important as experimental PMT/HPGe-detector setups need to require a minimum event rate that the DAQ must handle without event loss. Otherwise in this experiments coincidence analysis would not be possible. Figure 1 lists typical trace lengths and event rates for common PMT/HPGe-detector FC1.0 setups. Pay attention that experiments exist that violate the limits of Figure 1. For instance, LEGEND-200 would be a contrary HPGe-detector example where the majority of the measurement campaign is done with an event rate below 1Hz. An event rate in the kilohertz region it only has during calibration.

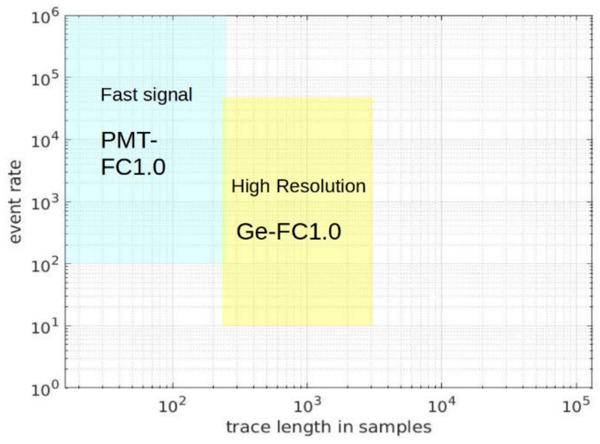


Figure 1 Overview on common FC experimental setups and their event rate/trace length requirements. It is important to note that the experimental setups also differ in sample rate. A typical sample rate for PMTs is at least 250MHz. HPGe-detector systems usually have a higher resolution, but also a lower sample rate. The trace length (and the sample rate) is also driven by the used preamplifier circuit and not only by the detector type.

The upper limit of the trace lengths in Figure 1 is motivated by feasible bandwidth considerations of a DAQ and not all detectors may be eligible for this rate. The upper limit is highly setup specific and therefore the event rate indications in Figure 1 must be observed with some caution. The bottom limit of the trace length is given by the analysis requirements. They differ a lot and therefore the specifications in Figure 1 are only a good guess trend. The (permanently) stored event data has to consider a given format which contains all required information in an inherent structure. The required information in PMT/HPGe-experiments is frequently the signal trend for a region of interest (already introduced under the term “trace”), the systems point of time and the channel identifiers. All tasks which belong to the management of this structure are part of the so called event building process. On command of the read-out server the FC event building is done locally on the FPGA boards without any further server support. In fact, in FC the server only verifies and reassembles the received ethernet packets. Hence the major workload of the event building process has to be done on the boards. This makes sense, as the reaction time of the server influences the systems performance and therefore all tasks which can be done in hardware should not be done by the server.

2.1 Analog signal chains for HPGe-detectors

In this chapter it is just assumed that the best possible resolution is given by the detector and the analog signal chain has to comply with this resolution. Remember that the possible resolution and the time range of signal strongly depends on the detector type. Figure 2 depicts a HPGe-detector single channel analog signal path with all main components.

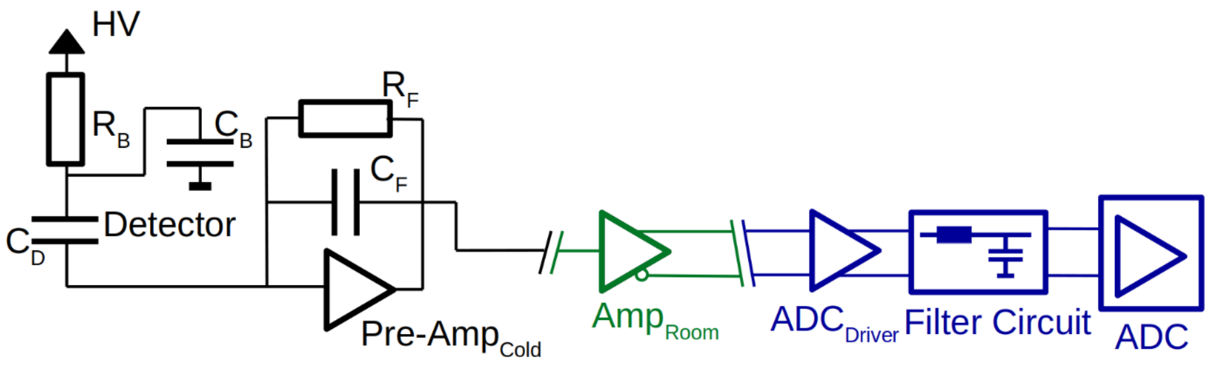

HV
R_B
C_B
R_F
C_F
C_D
Detector
$\text{Pre-Amp}_{\text{Cold}}$
Amp_{Room}
$\text{ADC}_{\text{Driver}}$
Filter Circuit
ADC

Figure 2 Analog signal chain of a HPGe-detector experiment. The analog signal chain has a front-end electronics part (black), buffers (green) and digitization electronics (dark blue).

The analog signal chain consists out of the front electronics (black components in Figure 2), buffers (green) and the digitization electronics (dark blue). The front-end electronics is centered around the detector. The HPGe-detectors need a high voltage (HV) supply to bias the detector. The required voltage level and operation currents depend on the implemented detector. Another central component of the front end-electronics is the preamplifier (preamp). In this thesis the corresponding topics are only a side elaboration and therefore the discussion is omitted here. More information is available for instance in [1], [2] and [3]. Designing a proper FADC card is a challenge on its own. Its filter circuits must consider the famous “aliasing” caused by ADC sampling, but the FADC cards design also must tackle other effects. An instance of a less known issue are the charge spike injections caused by sample switching of the ADC. The charge injections can be attenuated by so called “absorption” filters [4]. In the most DAQ systems with a proper designed FADC card the major noise contributor is the charge sensitive preamplifier and not the FADC card. But a proper designed FADC card is the error source of other deterministic signal issues that are more elaborately discussed in chapter 2.2.

2.2 Negative digitization effects on resolution

The resolution is one of the key specifications of a DAQ. In the analog signal chain of the HPGe-spectroscopy DAQ the noise behavior of the Front-End electronics and the resolution of the FADC card are decisive for the DAQ resolution. The Front-End electronics has a minor part in this thesis. Therefore, the discussion of Front-End electronics influence on the final DAQ resolution is omitted. The resolution of a HPGe-spectroscopy DAQ that uses a properly designed FADC card is commonly dominated by the influence of the Front-End electronics and not be the FADC card. This is especially true for its noise characteristics that mainly determine the needed signal shaping approach.

2.2.1 Noise and Jitter in digitization

The analog signal inputs of the ADCs also act as noise source and therefore influence the final resolution of the DAQ system. As the ADC input noise influences the ADC code transition voltages (Figure 3), it is also labeled as code transition or as code edge noise. Many different ADC types with significant design (differences in their noise filtering properties) exist and consequently the effective noise behavior varies much. Data sheets contain the code edge noise of the ADC. Besides the code edge noise, ADCs have quantization noise. An ADC maps a continuous signal into a limited discrete signal range. This quantization means information loss. The quantization information loss correlates with the signal trend and therefore it is not a real noise, as noise is an uncorrelated signal. With real varying signals (with just loose patterns), the effects of the rounding loss look like a random error and therefore the term quantization noise was impressed. The portion between the (ideal) quantization noise and the analog ADC input signal is the signal quantization ratio SNR$_{\text{Quant}}$ which follows as

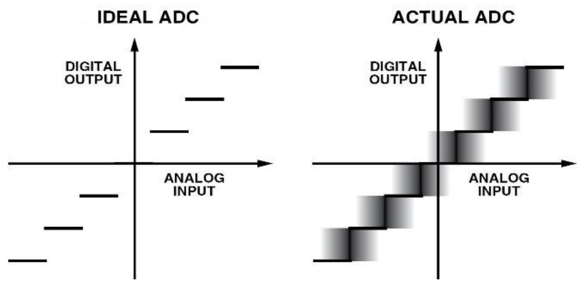


Figure 3 Input noise influence on code transition [5]. The ideal ADC of this example has no input noise and therefore its code transition voltages are constant.

$$SNR_{Quant} \approx 6.02 \cdot N + 1.76dB \tag{1}$$

The above equation is an approximation which is valid for an ADC with 7bits or higher. For instance, in [6 S. 100] an elaboration of SNR$_{\text{Quant}}$ is done.

2.2.2 Temperature drift effects of the reference voltage and ADC gain-offset amplifiers

In ADCs the reference voltage serves as comparison voltage in the ADCs comparator operations and consequently has a direct influence on the ADC output codes. The internal composition of the ADC determines the temperature drift effects on the ADCs gain and offset. Simply said, with increasing ADC/DAC resolution the stability of reference voltage gets more critical. Designing a reference voltage of sufficient stability for FADC cards in HPGe spectroscopy is a non-trivial task. ADC manufactures provide several designer guides to combat the connected issues. Note that a stable reference voltage does not guarantee a stable ADC gain and offset as these ADC properties not only rely on the voltage reference. The operation of such a FADC card is therefore only permitted in an environment with stable temperature or in application tasks whose time duration is sufficiently small.

2.2.3 Crosstalk in ADCs

Typical sources of crosstalk in ADCs include [7]

- Power supply coupling: In this case digital currents pulling on a power supply which is shared with an analog supply
- Capacitive crosstalk between digital and analog signals: Digital signals which are transmitted close to the analog signal path can couple into the analog signal path.
- Capacitive crosstalk between analog signals: If an ADC has more than one input channel, also the analog input signal of input x can couple into the analog signal path of input y.
- Ground ("bouncing") currents: Return currents effect a local movement of the ground potential. As the ground potential is the reference potential, this behavior distorts the signal value.

Crosstalk is therefore strongly influenced by the chosen ADC design and the ADC implementation in the FADC card. In the thesis only a single channel FADC is investigated.

2.2.4 ADC Nonlinearity

The derivation of the quantization noise (equation (1)) is valid for ideal, memory-less ADCs whose nominal transfer function is a linear staircase function with intervals of the same size (Figure 3). Manufacturing such an ideal ADC is not possible. The staircase function of a real ADCs has intervals of deviating size and is also dependent on the input signal itself. Commonly ADC data sheets poorly refer to input signal influence on the ADCs nonlinear behavior and surprisingly it is rarely discussed in literature. It is possible to show that the ADCs nonlinear behavior influences the typical performance metrics of a γ-ray spectrum. Some of its major consequences in HPGe spectroscopy are discussed for instance in [3]. In HPGe spectroscopy a precise knowledge of the nonlinear ADC behavior is a critical design key parameter. As not always an ADC with a sufficient nonlinearity characteristic is available, post-measurement correction strategies were developed. An important compensation technique, that will be elaborated in chapter 7.4, is for instance given by the MAJORANA collaboration [8]. The concepts of differential nonlinearity (DNL) error and integral nonlinearity (INL) error will be introduced as feasible description of the real ADC transfer function behavior. Chapter 7.1 is dedicated to them.

3 Zynq Device Family

In 2000 Xilinx announced its first “Virtex II Pro” hybrid devices with PowerPCs CPUs from IBM. Over the time the PowerPC architecture declined and the market position of ARM ecosystem steadily increased. In 2011 Xilinx unveiled its Zynq-7000 family as “the industry's first Extensible Processing Platform”. Soon after Xilinx also employed the System on the Chip abbreviation “SoC” in its Zynq publications. Figure 4 shows the hybrid structure and the Zynq particular components. Note that the FPGA area is labeled as “Programmable Logic” (PL) and the Embedded CPU as “Processing System” (PS). Zynq systems use a CPUs out of the ARM ecosystem. ARM is with its SoC infrastructure the leader of the mobile segment and supports the full hardware/software stack development. ARM CPUs commonly have their own Linux maintaining line.

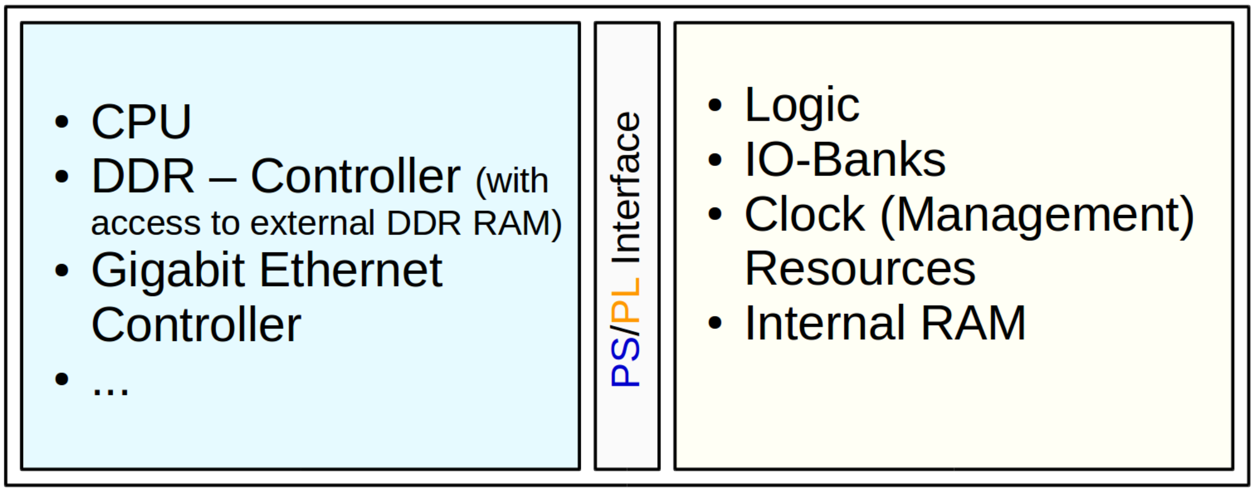

• CPU
• DDR – Controller (with access to external DDR RAM)
• Gigabit Ethernet Controller
• ...
PS/PL Interface
• Logic
• IO-Banks
• Clock (Management) Resources
• Internal RAM

Figure 4 Hybrid devices consist out of an Embedded Mircoprocessor with its periphals and a FPGA fabric area.

2015 Xilinx launched the enhanced Zynq-UltraScale+ family. Table 1 lists the main differences between the series.

Table 1 Main differences between Zynq-7k a. Zynq-Ultra+. Information is out of [9] and [10]

	Zynq-7k	**Zynq-Ultra+**
Market launch	2011	2015
CPU Architecture	ARM Cortex-A9	ARM Cortex-A53
Main Interconnect in PS	ARM CoreLink Network Interconnect “NIC-301”	ARM CoreLink Cache Coherent Interconnect “CCI-400”
ISA	ARMv7 (32bit)	ARMv8 (64bit)
FPGA Fabric Architecture	Artix-7, Kintex-7	Zynq-UltraScale+

The differences between Zynq-7k and Zynq-Ultra+ devices are significant. In this PhD thesis both series were scrutinized for their eligibility as FlashCam successor devices. More emphasis is given on the Zynq-UltraScale+ devices because of their better performance. The introduction to the Zynq devices is split in 2 parts. The first part discusses general FPGA architectures and special features of the Zynq devices which are relevant for the FlashCam 2.0 Prototype. The second part is dedicated to the processing system components that significantly influence the performance results of the setup.

3.1 General FPGA architecture and FPGA architecture in Zynq-Devices

The devices used in this PhD thesis base on 3 different FPGA architectures (Artix-7, Kintex-7, Zynq-UltraScale+). Roughly said, the Artix-7 devices offer the poorest performance, the Zynq-Ultra+ devices the best. The evaluation which was done in the scope of this thesis confirmed the superiority of the Zynq-UltraScale+ fabric. The firmware setups for the FC2.0 prototype do not have the intention to determine the performance/channel limit of the architecture. The main intention is to determine the feasible readout performance. But this performance is mainly given by the PS infrastructure of the device, not by the PL architecture which mainly influences the possible channel count per device and the operation frequency of the signal processing. The Zynq device implemented in the future FC2.0 card will depend on the application and its requirements. In some low channel count applications, the Artix-7 fabric architecture is sufficient. Requirements of other applications will have a better match with the Zynq-Ultrascale+ architecture. The precise FPGA architecture of the devices under test is a company secret of Xilinx. In order to support the customers in their development process, Xilinx published an extensive documentation. The published documents do not have the elaboration depth which is required to enable an utter reengineering of the FPGA architecture. Commonly they just give a glimpse to understand the high level of the architecture. As all FPGA have some common blocks, a tiny FPGA example could have an architecture like in Figure 5.

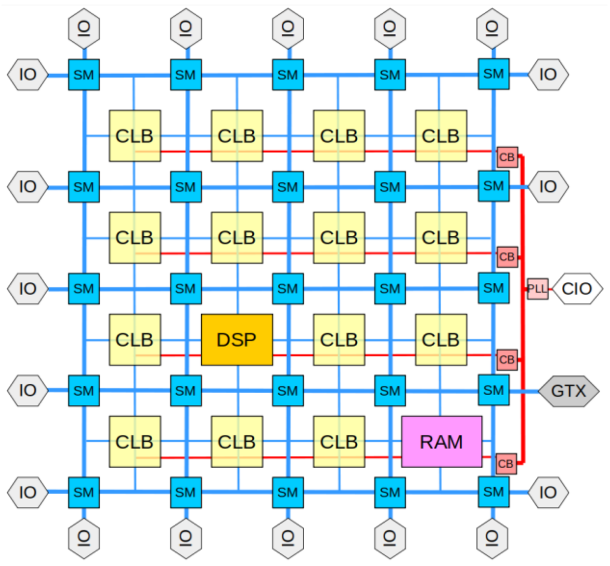


Figure 5 Model of a simple FPGA device Extended and revised version of original figure out of [11 S. 440]. The setup is simplified and far from a realistic implementation. For instance, already the first FPGA, the XC2064, was consisting out of 64 CLBs.

The simplified FPGA architecture in Figure 5 consists out of

- IO-Resources (IO (IO circuitry of pins), GTX (Multigigabit Transceiver/Receiver Pins)
- Clock Resources (CB (Clock Buffer), PLL (Phase Locked Loop), CIO (pins with clock optimized IO circuitry)
- (Configurable) Logic Resources (CLB)
- Routing Resources including Switch Matrix (SM)
- Memory Resources (RAM)
- Integrated Intellectual Property (IP) Resources (for instance “DSP” slices that include a multiplier)

Nowadays FPGA devices have all above listed resources in common, but the implemented resources design may differ significantly.

3.2 Processing system architecture in Zynq devices

In the Prototype the ARM CPU and its peripherals are responsible for read out control, the event building process and the ethernet transfer. The processing system is the bandwidth bottleneck of the design and thus main contributor to the final performance of the Prototype. The processing system of the Zynq-UltraScale+ devices is illustrated in Figure 6.

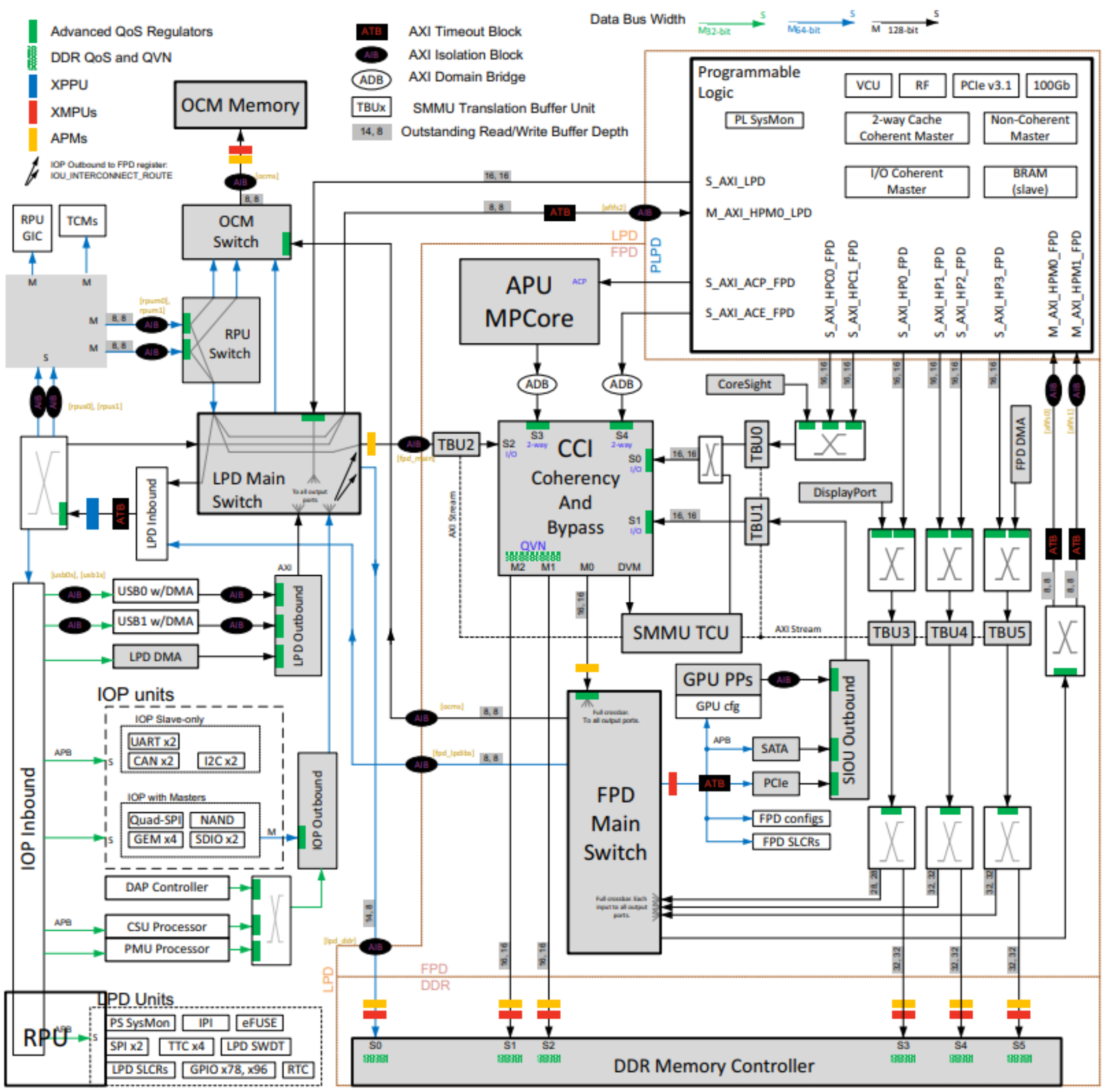


Figure 6 System Level View PS Zynq-Ultrascale+ [12 p. 1050]

Not all these components are available in the Zynq-7k PS. For instance, the Cache Coherent Interconnect (CCI) and the connected high performance coherent (HPC) port are a Zynq-Ultra+ exclusive component. Such exclusive components are not used in the FC2.0 prototype design. The main point behind this decision is the FC2.0 aspiration to run on Zynq-Ultra+ and Zynq-7k devices. Like already mentioned, the Zynq-

Ultra+ is a further development of the Zynq-7k and therefore both series are sharing a common PS structure (Figure 7).

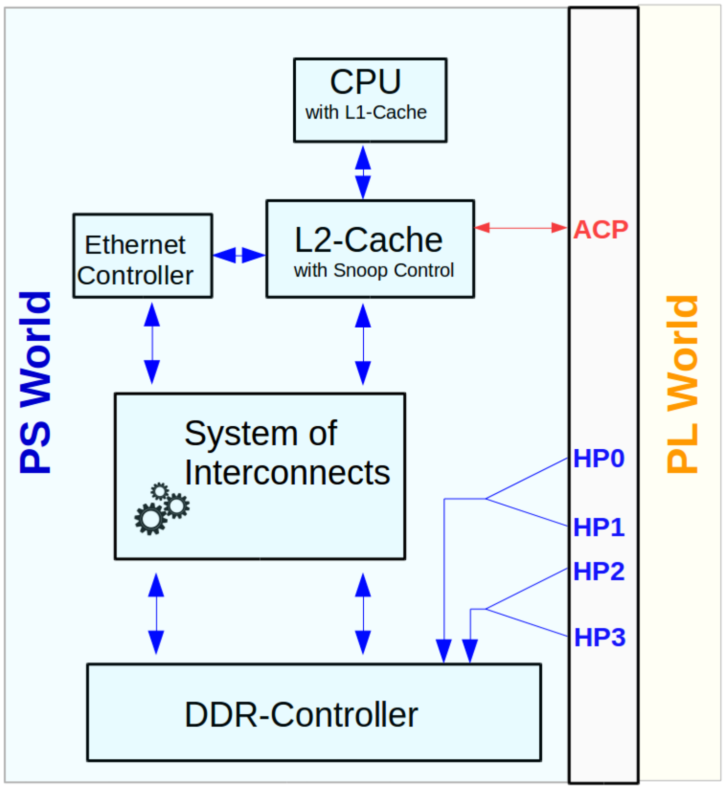


Figure 7 Common structure elements of Zynq-7k/Zynq-Ultra+ Architecture. The series have further units in common (like the Memory Management Unit (MMU), On Chip Memory (OCM), Real Time CPU, Interrupt Controller, etc). As in the performance evaluation the effect of this components is not discussed, they are not depicted in this figure. Note that the ACP Port (marked red) is directly connected with the Cache, the HP Ports (marked blue) relate to the DDR controller.

The processing system of both Series have a Level-2 Cache (with Snoop Control Unit), Ethernet Controller(s) and a PS/PL-Interface with four High Performance Ports (HP) and one Accelerator Coherent Port (ACP). In both architectures the HP ports are directly connected with the DDR-Controller of the external DDR3/DDR4 memory. The external memory is the main memory of the PS system. A main memory of sufficient size is inevitably required to operate default Linux kernels on the system. The ACP port always has a direct Cache System Access. Both Zynq device families contain a (less performant) Real Time Processing Unit (RPU) and a more performant Application Processing Unit (APU). The performance of the RPU is not adequate for event building tasks in a high performance DAQ system like FlashCam. Again, it should be underlined that event building is mainly an IO task and so the bandwidth of the memory system is the critical system point, the APUs arithmetic/logic operation performance is less significant in event building tasks. In event building tasks its (multi)store/load performance matters!

The motivation of caches is the relative high access time of (external) main memories. If a CPU has a value not in its registers and has no cache system, it must wait until the desired data arrives from its main memory.

In consequence the CPU store/load performance drops. A CPU with cache system has less average waiting time as cache access times are much lower. Building up a memory with the velocity of a cache demands many architectural trade-offs. Simplified, the access time grows with cache size. As with a dwindling cache size the chance of a cache miss increases, the final cache size is a settlement between many considerations. The best performance is commonly achieved with a hierarchical cache system where several caches with different sizes exist. The lowest level L-1 (tightest unit to the CPU) has the highest access rate, but also the smallest size.

The efficiency of the cache system strongly depends on the program sequence. Common code obeys to the “principles of temporal and spatial locality” [13 S. 388]. The cache controller utilizes this knowledge in order to increase the hit rate and preventing useless traffic between memory and cache. The final performance of the cache system strongly depends on the intelligence of the cache controller and its design matching on the cache architecture.

3.2.1 Application Processing Unit (APU)

The architecture difference between the ARM Cortex-A9 (part of the PS in Zynq-7k) and the ARM Cortex-A53 is significant. The ARM Cortex-A9 represent one of the first "high-end application processors" of ARM and was disclosed 2007. Several important ARM SoC concepts were unknown at that time – for example the Big.LITTLE approach (and the corresponding AMBA ACE protocol). The ARM Cortex-A53 ("LITTLE") was unveiled 2012 and could build up on the (multicore) SoC experience which was gained in the meantime. Table 2 lists all important key parameters of both processors.

Table 2 List the most important specifications of the ARM APUs used in the Zynq devices. The listed values are always related to the common value in the corresponding Zynq devices of the FC 2.0 prototype. As ARM sells IP which is adjusted by their customers the official ARM specification differs in a wider range. The information is, if not otherwise marked, out of [12] and. [14].

APU	ARM Cortex-A9	ARM Cortex-A53
Instruction Set Architecture (ISA)	ARM_v7	ARM_v8
Clock Frequency in MHz	667 [15 S. 15]	1200 [16 S. 26]
Number of cores	2	2/4
Interrupt Controller	Implemented	Implemented
Memory Management Unit	Implemented	Implemented
Floating Point Extension	Implemented	Implemented
NEON Extension Unit	Implemented	Implemented
DMA Controller	8 non-coherent 64bit channels (scatter-gather support)	8 non-coherent 128bit channels 8 IO coherent 64bit channels (scatter-gather support)
L1-D-Cache Size	32kB	32kB
L2-Cache Size	512kB	1MB
(L2) Cache Type	8-way set-associative	16-way set-associative
Cache Line Size	32B	64B
On Chip Memory (OCM)	256kB	256kB

An elaborated discussion of the Cortex A53 APU is done in [13]. As the arithmetic operation performance of the APU is decisive in analysis, but not in event building, their analysis is omitted.

3.2.2 Coherent Memory Access Management

Caching is for high performing computing a requirement, but it also has its perils. In systems without cache, the data just exists in the memory, but in systems with cache the data can concurrently exist in the memory and in the cache system. This has no effect as long as the data value is constant, but as soon the value is changed the challenge starts. The main memory is an independent slave unit, who is strictly controlled by the CPU or Cache Controller respectively. The parts actualize each other if it is demanded by the CPU (or the Cache Controller). When a routine changes a value of a cache element, the alteration is invisible for the memory. Hence the old value will persist until the memory is actualized by the CPU or the Cache Controller. In systems with one cache per memory the time of actualization does not matter. In systems with several (distributed) caches potential issues can arise as soon the caches work with inconsistent data elements. In systems with DMAs it is also possible that the memory elements are getting changed without any CPU intervention. In this case the content of the cache would be outdated and is therefore “invalid”. In order to have data coherency, Cache Controller must be informed that its cache contains invalid data. The invalidation of this outdated data must be done by software or by hardware. As hardware support demands corresponding (complex) logic and a suitable snooping protocol it is not supported by default in all SoC systems. Hardware support is granted by the HPC ports in the Zynq-UltraScale devices. The development of snooping protocols was driven by multi core systems. In multi core systems, the same element can be deposited in several caches (there each cache depends to its own CPU core). The data coherency management process needs to handle such cases too. 2011 ARM released its AMBA4 ACE protocol for its multicore ecosystem. In order to maintain cache coherency between its CPUs and the ACP Port, the Zynq Devices have a Snoop Control Unit.

In the previous discussion it was assumed that the CPU uses its Cache and does not bypass it. But this is not always beneficial (as not always the code obeys the “principles of temporal and spatial locality”). For instance, if the CPU data just needs once, buffering it in the Cache can be derogatory, as caching it, may expulse data (from the cache) which may be required in one of the subsequent steps again (keyword is “cache trashing”). Therefore, the CPU read access to memory has two feasible important options: “Write-Allocate” and “Non-Write-Allocate”. In “Write-Allocate” Mode the desired read data is buffered in the Cache, in “Non-Write-Allocate” Mode the Cache is bypassed and the demanded element is directly passed in one of the CPU registers. For results of CPU operations, the same thoughts can be applied. In same cases it is useful to write the results directly to memory (and bypassing the cache system), in other cases the buffering of the result(s) in cache system might be a better idea, as they soon might be invoked (by the CPU) again. In textbooks the latter approach is called “Write-Back”, the former “Write-Through”. As it is beneficial to manage this property dependent on the memory page, it is a task of the Memory Management Unit (MMU) to manipulate the Cache Controller behavior accordingly.

3.2.3 DDR Controller

Figure 8 depicts the setup of the DDR controller subsystem and its interconnections in Zynq-Ultra+ devices. The logical memory controller unit is responsible for the read/write operation prioritization and therefore must handle the given "Quality of Service" requests.

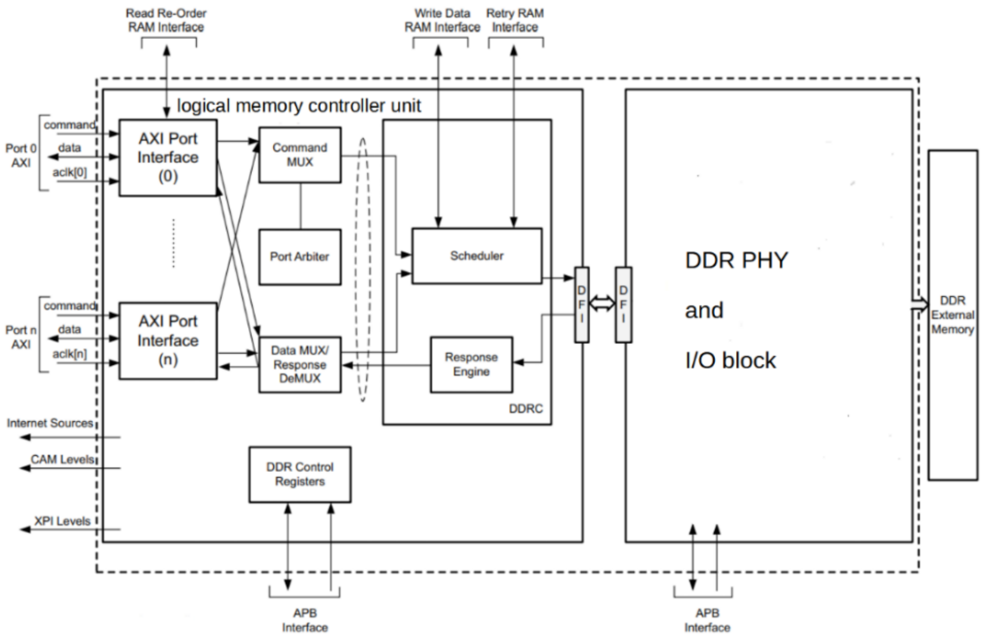


Figure 8 DDR Subsystem Block Diagram Zynq-Ultra+ [12 S. 436] (Modified by M. Schuett)

Besides the logical memory controller unit, the DDR subsystems has special DDR PHY and I/O block (Figure 8). This is a requirement as the DDR memory is an external chip on the module and is not a part of the Zynq devices. As several DDR memory designs with substantial performance differences exist, the final performance depends also on the implemented external DDR memory specifications. The logic part has besides the address adapter units (AXI Port Interface), a port arbiter, a content addressable memory (CAM) and a scheduler. The CAM supports the scheduling algorithms in traffic optimization management based on priority, bank/rank status and DDR timing constrains [12 S. 437].

The most important difference between the DDR controller in the Zynq-Ultra+ and Zynq-7k devices is the DDR4 compatibility of the Zynq-Ultra+ devices. Table 3 lists important DDR controller specifications of the Zynq-Devices.

Table 3 Compare of the DDR controller high level properties in Znyq-7k a. Zynq-Ultra+ . The table is a summary of the corresponding chapters in [12] a. [9]. Remember that in the case of DDR3/ DDR4 a width of 64bit (or less) is a "single rank", 128bit is a "dual rank".

	Zynq-7k	**Zynq-Ultra+**
DDR Type	DDR2, DDR3, DDR3L and LPDDR2	DDR3, DDR3L, LPDDR3, DDR4 and LPDDR4
Channel	Single channel	Dual channel
Total data width (bits)	16,32[1]	32,64
Component data width (bits)	8,16,32	8,16,32
Implemented external DDR memory on corresponding Trenz Modules (chapter 6.1)	2*512M*DDR3L-1600 SDRAM (listed Z7k modules of Table 9)	DDR4-2400 SDRAM (total size depends on ZU+ modul)

The effective performance of the DDR controller depends on the chosen DDR memory implementation and on the addressing style ("random address distribution" or "sequential incremental addressing"). Table 4 shows the theoretical bandwidth of common embedded DDR memories.

Table 4 Theoretical performance for common embedded DDR memory . Note that the Zynq-7k series just complies with 32-bit DDR3 modules.

	DDR3L-800 SDRAM	**DDR4-2400 SDRAM**
Theoretical bandwidth per 64-bit module	6.4 GB/s	19.2 GB/s
Theoretical bandwidth per 32-bit module	3.2 GB/s	9.6 GB/s

In operation a DDR controller never achieves its theoretical bandwidth. The effective bandwidth strongly depends on the addressing style, the complexity of the scheduler/arbiter logic and the chosen (Zynq device external) DDR memory implementation.

[1] In some Zynq-7k devices the total data width is limited to 16bit. None of these devices is used in the FC2.0 prototype evaluation.

3.3 The Processing System/Programmable Logic Interface in Zynq-Devices

The Zynq devices are hybrid devices out of an embedded microprocessor and FPGA fabrics. The FPGA portion is ideal in order to do all signal processing jobs in the field of pulse detection and triggering. Event building is a typical CPU task. CPU routines are working in addressed space. This means that unaddressed data elements of the FPGA area must be aligned to a corresponding destination address before they enter the PS area. In the simplest (not feasible) case, every data element has its own corresponding address element, the addressing just requires an address bus. In a SoC every unit has its own (overlap free) address range. In order to show that a data element is invalid, the corresponding address has to be out of the range of all units. In principle it is possible to build up a SoC system with this simple protocol. But its simplicity has many serious drawbacks that challenge its practical implementation. For instance, every unit has always to accept every input immediately as no ready/valid signals exist. Furthermore, error communication is not possible. Another fatal lack is the missing of bursts and the impossibility of data width adaption (“expansion/downsizing”). ARM therefore developed its AMBA4 protocol (AMBA is acronym for “Advanced Microcontroller Bus Architecture “). The AMBA protocol was developed under the focus of SoC development and OS considerations. In the PS area of the Zynq devices communication between different blocks is almost always handled on basis of the AMBA protocols. Xilinx decided that the data ports of the PS-PL interface are a subject of the AMBA AXI3/AXI4 protocol (Z7k/ZU+). The AXI3 protocol is a subset of the AXI4 protocol which Xilinx recommends to use for IP development in Vivado. The PS-PL interface signal and interfaces have besides the data ports amongst others following features:

- PS internal access to (some) special registers
- System error signals
- MIO-EMIO signals and interfaces
- Clock and Timer signals

The (Extended) Multiplex IO ((E)MIO) Interface intention is mainly to forward signals from units in the PS area into the PL area. This offers the advantage that these signals also can be routed to pins which are assigned to the PL area. A brief discussion of the data ports was already done in the introduction of chapter 3. The prescribed letters “S”/“M” are acronyms for Slave/Master. The S-HP and ACP data ports exist in both Zynq device series and significantly differ in their data transfer method. The M-HP ports exist like the S-HP ports in both Zynq devices series. Their design implementation is strongly PS centered and has optional PS traffic information which is the S-HP ports not accessible[2]. But their PS centered design impedes also continuous data transfers. The HPC and ACE ports depend on the CCI and are therefore Zynq-Ultra+ exclusive. The HPC port is an intermixture of the HP/ACP method and offers one-way coherent data

[2] This assumption depends on the Zynq device series.

transfers. The ACE port was established for “fully” two-way coherent data transfers. Its focus is on design extensions with cache coherence and not on IO-DMA tasks like the HP/HPC/ACP ports. The M-LPD port of the Zynq-Ultra+ is the functional equivalent in the M-GP port of the Zynq-7k. They offer low latency CPU accesses into the FPGA area and are therefore essential for PL register setting and control. In a SoC design not only the available amount of data ports, also their mutual dependence is important. All data ports have in all devices their own clock and reset signals, so that they can be driven independently. Furthermore every data port has its own asynchronization unit. A clock domain crossing via the PS-PL interface therefore needs no additional measures which the firmware designer has to care about. This is an enhancement in contrast to MicroBlaze designs, like FF1.0, where the firmware designer must consider measures against clock domain crossing (CDC) issues.

3.3.1 M-GP and M-LPD ports

Both ports have direct PL access and are therefore ideal for APU programmed IO with the FPGA area. The M-GP port obeys the AXI4_Lite protocol, the M-LPD port uses the unreduced AXI4 complement. Both ports have their low latency architecture in common – this means that they just have an asynchronization unit implemented in their path (in order to enable clock domain crossing), but no further FIFO buffering options.

3.3.2 S-HP ports

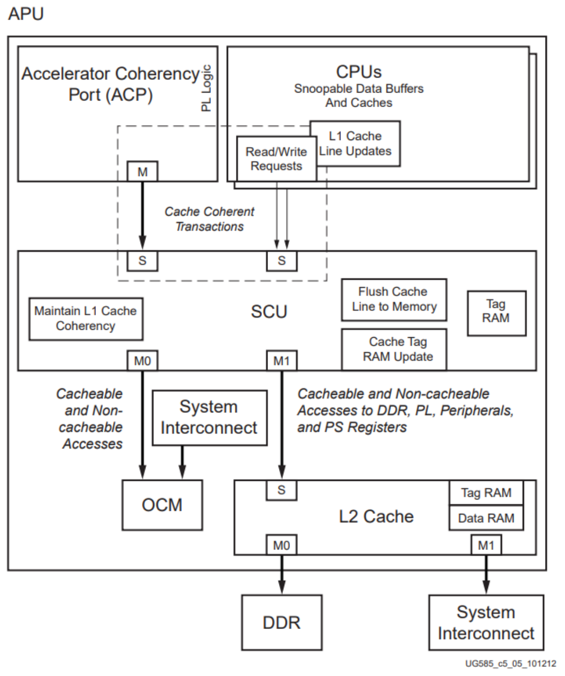
All S-HP ports have in common that they have a direct access to the external DDR memory (Figure 7). HP ports therefore could be a possibility to buffer a data stream directly on the external main memory. This enables firmware designs that do not need to buffer the accepted waveforms in Zynq internal RAM. In these (future) designs some restrictions due to FPGA RAM limitations could be surmounted. In order to set up a high performance data transfer in the HP port paths a FIFO buffer is implemented. A burden of this approach is an increased access latency. As the AXI FIFOs and their management logic are also responsible for address forwarding, the AXI FIFO performance partly depends on the implemented AXI protocol settings. Special care should be taken to data width and burst size settings. As all connected data links pass the DDR controller the effective S-HP performance is dependent on the DDR memory and controller performance. The design approach of the HP ports paths is roughly the same in the Zynq-7 and Zynq-Ultra+ series. In the Zynq-Ultra+ series the components were significantly enhanced and some additional units (regarding virtualization support) implemented. For all proposed setups in this thesis the functionality of the new units is not significant, as virtualization is not relevant. Table 5 lists all significant HP port properties for Zynq-7k a. Zynq-Ultra+ designs.

Table 5 Listing of all noteworthy HP port settings.

	Zynq-7k	Zynq-Ultra+
Protocol Interface (PL side)	AXI3	AXI4
Data Width in bit	64	128
FIFO Size	128×64bit	128×128bit
Outstanding Transaction (read/write)	8/8	16/16

3.3.3 S-ACP Port

The Accelerator Coherency Port (ACP) Port has L2-Cache access via the Snoop Control Unit (SCU) (Figure 9). The ACP port data needs not to make a detour via the main memory – the event building process can be completely done in the cache without invalidation operations and memory accesses.



APU
Accelerator Coherency Port (ACP)
PL Logic
CPUs
Snoopable Data Buffers And Caches
L1 Cache Line Updates
Read/Write Requests
M
Cache Coherent Transactions
S
S
Maintain L1 Cache Coherency
SCU
Flush Cache Line to Memory
Tag RAM
Cache Tag RAM Update
M0
M1
Cacheable and Non-cacheable Accesses
System Interconnect
Cacheable and Non-cacheable Accesses to DDR, PL, Peripherals, and PS Registers
OCM
S
L2 Cache
Tag RAM
Data RAM
M0
M1
DDR
System Interconnect
UG585_c5_05_101212

Figure 9 ACP Connectivity Diagram in Zynq-7k devices [14 p. 137]. The setup in Zynq-Ultra+ devices is despite implementation of different interconnects in principle comparable. The acronym SCU means Snoop Control Unit.

The ACP port has the same rank as the APUs. This means that the ACP port competes with the APUs for the L2-Cache space and the APUs possibly expel ACP content out of Cache. Running jobs therefore might show influence on the ACP port performance. The job dependent behavior contrasts with the HP ports for which the performance is influenced by the total memory access rates which is not necessarily linked to the CPU job burden. The biggest burden of CPU centered transfers is their non-continuous data transfer. A DAQ which uses the ACP port therefore needs to prebuffer its data with FPGA-RAM. Furthermore, the ACP port does not offer prioritization. The ACP port implementation in the Zynq devices is not completely

AXI protocol conform. The discrepancies, which also depend on the Zynq series, are in [12] and [14]. A summary of all important ACP port properties in Zynq devices is in Table 6.

Table 6 ACP Port Properties in Zynq-Devices. The information is out of [14] and [12]. The listed "Cache Line Friendly Access Style" properties are incomplete, but sufficient for the understanding of the FC2.0 performance trends. The not listed limitations refer to AXI properties that are not used in FC2.0.

	Zynq-7k	**Zynq-Ultra+**
Protocol Interface (PL side)	AXI3	AXI4
ACP Port width	64b	128b
Cache Line Size	32B	64B
Cache Line Friendly Access Style	Transaction size must be a multiple of the cache line size, in particular cases the AXSTRB signal is limited [14 p. 105]	Only transaction size of 64/16B with incremental addressing (AXI4 signal AXBURST has a value of 2'b01) [12 S. 1064]

3.3.4 S-HPC Ports

The HPC ports are Zynq-Ultra+ exclusive. HPC ports are HP ports whose PS data path crosses the Cache Coherent Interconnect (CCI). As classic HP ports, HPC ports transfer the data into the memory (and not into the L2-Cache). If accordingly selected, the CCI invalidates the corresponding cache elements. In opposite to the classic HP ports, the CPU has not to care about cache invalidation. The CPU can immediately access the memory. Accessing the memory is still a requirement, as the Cache was just invalidated and not preloaded with the new HPC data. The HPC system implementation also has its strains. The CCI is passed by several units (Figure 6) and has a lower spare bandwidth than the classic HP port data path. So the chance of a non-continuous data transfer situation rises. A detailed comparison between the HP, ACP and HPC ports of the Zynq-Ultra+ devices is done in the Table 7.

Table 7 Compare between HP, HPC and ACP port in Zynq-Ultra+ devices. In the Zynq-7000 devices a HPC port is not available and all ports have a maximum data width of 64bit. Also the DDR controller port occupation is different.

	S-HP	**HPC**	**ACP**
Number of PS/PL interface ports	4	2	1
Number of DDR Controller input ports	3	2	2
Interference with L2-Cache Controller	No	Via CCI	Via SCU
Mutual units with CPU interference	DDR Controller	CCI and DDR Controller	Cache
Continuous transfers	Yes	Yes	No
Data width in bit	32, 64, 128	128	128
Burst/Data shaping via AXI FIFOs and APB in PS	Yes	No	No

4 FlashCam 2.0 Prototype Firmware Design

The FlashCam 2.0 Prototype Firmware design is strongly influenced by the firmware design of FC1.0. Whenever possible, it was tried to reuse the firmware blocks of FC1.0. The FC1.0 design could be described as “Microblaze” centric, whereas the FC2.0 design is centered around the ARM CPU and its PS periphals. In general the Microblaze softcore CPU is much less performant than the ARM CPUs of the Zynq devices. Hence one could expect that the data exchange between FPGA fabric and CPU in the Zynq devices is much simpler and outperforms the Microblaze bandwidth a lot. This assumption looks at the pure CPU performance and does not consider the CPU implementation in the final system. The Microblaze is a soft core CPU which was optimized for implementations in FPGA systems where the IO performance matters. As a result the Microblaze has a so called Fast-Simplex Link (FSL) Bus who can continuously transfer available data in one CPU clock cycle in a CPU register without fail. This enables an impressive effective system bandwidth. The ARM CPUs have no FSL comparable bus port. Even the fastest bus ports of the ARM CPUs have a significant latency. The ARM CPUs are just superior as they operate at a much higher frequencies and their latency is just one point in the total performance determination.

In this section the focus is on firmware design. As the firmware design strongly depends on the chosen software design and vice versa, a survey of the firmware/software interaction is a reasonable preface. Figure 10 illustrates the interaction. The PL and the PS World are both Zynq device exclusive. The firmware, that shapes the FPGA internal setup, roughly can be partitioned into three kinds of design resources. A part of the resources is responsible for control tasks, another part must provide the event data and another part must provide the essential status information. The client software routine, that is running on the Zynq device, can prompt the data via a DMA invocation.

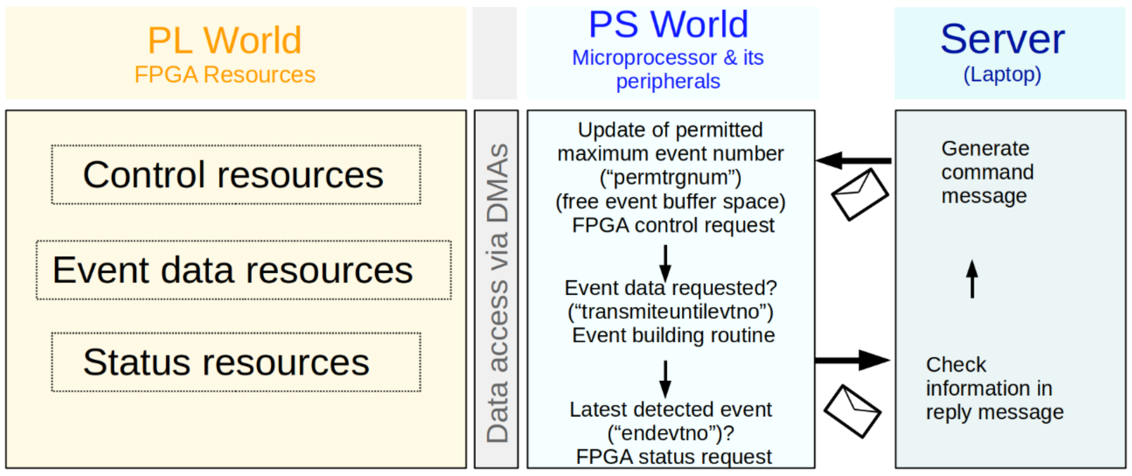

PL World
FPGA Resources
Control resources
Event data resources
Status resources
Data access via DMAs
PS World
Microprocessor & its peripherals
Update of permitted maximum event number (“permtrgnum”) (free event buffer space) FPGA control request
Event data requested? (“transmiteuntilevtno”) Event building routine
Latest detected event (“endevtno”)? FPGA status request
Server
(Laptop)
Generate command message
Check information in reply message

Figure 10 FPGA resources in FC2.0 prototype firmware and their implementation in the event read-out process

The interaction between the client(s) and the sever is hierarchical. The client messages are a rebound to the read-out server requests. These are

- requests of PL status information like number of received (accepted) triggers (in Figure 10 labeled as “endevtno”)
- requests of event data (trace data and corresponding header information (event time stamps, …))
- requests to change PL control registers like the “permitted maximum event number” (in Figure 10 labeled as “permtrgnum”) to enable an event update in the FPGAs event buffers. The instance control parameter “permtrgnum” is necessary as the event read-in (“update”) in the event buffers overwrites their current data. Hence an uncontrolled acceptance of new events could cause an event data loss. A detailed discussion of “permtrgnum” and the remaining status/control information is done in chapter 5.2.

To give reasonable commands, the server must know the client’s internal states and capabilities. A glimpse how this is done is given in chapter 5.2. Every instruction starts with the server’s build and transmission of the corresponding command message. Some commands demand an adjustment in the firmware’s control registers. If so, the client routine prompts the corresponding control functions that set the control resource(s). The majority of all control requests demands an incremental update of the introduced “permtrgnum” parameter. To access and set the FPGA resources a DMA prompt is done by the control functions. If the command requests the transfer of event data, the client prompts the event building routine. The event building routine receives the demanded event data from the event data resources in the FPGA. A detailed description of the event building is done in chapter 5.2 -5.3. To maintain the state of omniscience the server must be acquainted with the firmware’s significant internal states. In practice the information of the latest accepted event number (“endevtno”) is mostly sufficient.

4.1 Global FC2.0 prototype firmware design

Figure 11 presents the implemented blocks and their dependencies for a one channel FC2.0 prototype firmware example. The firmware partly resorts code snippets out of FC1.0. Table 8 lists the developer of illustrated blocks of Figure 11.

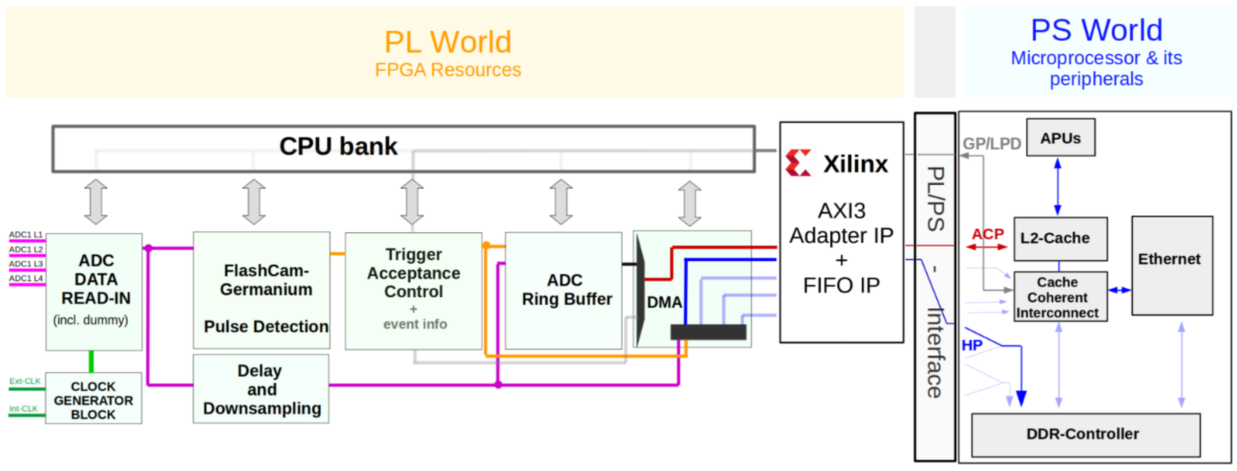


Figure 11 Firmware setup of FC2.0 prototype. The magenta lines indicate the ADC signal data flow as "unmapped" data stream. The address mapping is done by the DMAs. The flow of the trigger signals is marked by an orange line. The mapping of the event information (grey line (unmapped event information bursts)) depends on the settings and can either be done by the GP/LPD IO unit (which is also responsible for the CPU bank setting/read-out) or by the ACP-CMA.

The Zynq devices have the peculiarity of two functional zones (chapter 3). The firmware is entirely included in the FPGA resources ("Programmable logic (PL) World") (chapter 3.1). The A(C)PU is part of the processing systems (PS) zone (chapter 3.2). Both worlds are connected via a PS/PL Interface (chapter 3.3). The major functional firmware blocks are

- CPU bank unit: Does the read-out and setting of the status and control registers in the firmware blocks (depicted as "control/status resources" in Figure 10). Uses therefore the GP/LP IO unit (chapter 4.2.2). The design is partly adopted out of FC1.0.
- ADC data read in unit: Responsible for the read-in of the digitized analog signals. The ADC frame clock and the system clock that operates the firmware are not synchronous. Hence an unassisted clock domain crossing of the ADCs output data into the system clock domain is not possible. To complete the task, the ADC data read in block uses IO resources. As the ADC frame clock and the system clock commonly just differ in phase, the task is sometimes also known under the term "(ADC) clock delay compensation".
- ADC data read in dummy unit: For system performance checks and verification it is advantageous to have a known input data stream. This is the task of the "dummy" unit. The functionalities of the dummy unit are not required in real data measurements, therefore it is optional.
- Clock Generator unit: Responsible for the system clock generation. Derives it from an external PPS/10MHz synchronization ("GPS") signal or alternatively (less accurate and stable) from board internal clock sources.
- Pulse detection unit: FPGAs offer only a very limited amount of RAM to buffer signals. In a functional system, only a signal time range that covers the searched pulse shape should be accepted. The detection of a pulse coming from an energy deposition in a HPGe-detector in the signal is done

by the “pulse detection unit”. As soon as it detects a pulse, it transmits a trigger pulse to the trigger acceptance block. The trigger acceptance control does the final trigger decision. The implemented pulse detection is optimized for HPGe-detectors and adopted from FC 1.0. The pulse detection block uses recursive shaping filters to attenuate the noise and improve the resolution of the internally available energy estimate. On basis of the energy signal the pulse detection unit does the triggering. A special feature is the implemented baseline follower whose determined baseline value is essential to achieve high resolution spectrums with short trace lengths.

- Delay and downsampling unit: The delay unit has a continuously fed pipeline to buffer the input signal for a pre-selected time. Delaying the input signal is important as the “trigger acceptance control” unit requires time to determine if the buffer can accept an issued pulse signal.
- Trigger acceptance control: For several reasons it is not favorable to buffer every detected pulse in the ADC ring buffer. The “trigger acceptance control” unit must check if the incoming pulse triggers (of the pulse detection block) obey all rules. If all checks are passed, the “trigger acceptance control” unit forwards a final “frame” trigger to the ADC Ring buffer. As the trigger acceptance control block contains all trigger information, it has also the job to buffer the corresponding event header information. The event header information is buffered in so called Event Information Ring buffers. The event building routine can access the event header data by the GP/LPD ports or by the ACP port (delineated by the grey (output) lines of the trigger acceptance control block). In the event building the trigger acceptance control unit must prevail several essential management functions. For instance, it must enforce the actual maximum event policy of the read-out server (“permtrgnum”) that is essential for uncorrupted trace data buffering. Furthermore, it also can provide the latest event occupation status (“endevtno”) of the ring buffers to the Server Read Out Routine. The read-out process and the corresponding involvement of the trigger acceptance control is explained in chapter 5.2. The accurate derivation of all trigger acceptance rules and design interactions is done in appendix chapter A.5.
- ADC (Trace) Ring Buffer unit: Has to buffer (lossless) all traces which were accepted by the previous “trigger acceptance control” block. An incoming “frame” trigger signals the ADC Ring Buffer block that it must immediately read-in the issued data out of the delay unit (magenta line in Figure 11). The ACP-CMA (that is controlled by the event building routine of the ARM CPU) has direct access to the buffered samples of the ADC Ring buffer (this is implied by the black line that connects the ADC Ring Buffer Block and the DMA Block). The counterpart of the ADC (Trace) Ring Buffers are the Event(information) Ring Buffers. The Event Ring Buffer design differs, and they are implemented for a number of reasons in the “trigger acceptance control” block. Like the pipeline of the delay unit, the ring buffers are main consumers of the FPGA internal RAM. Amongst other design parameters their buffer capacity decides the maximum performance of the DAQ. A

ring buffer design therefore must efficiently use the available FPGA-RAM. This requires consideration of the device specific FPGA-RAM type and the device specific FPGA-RAM size. Furthermore, the interaction with the DMA and the trigger system must be considered. Elaborated information can be found in the appendix chapter A.4.

- DMA block: DMAs have the task to transfer data from the FPGA area into the PS area of the system. The possible functionality of the DMA depends on the connected PS/PL interface port (chapter 3.3).
- AXI Interconnect IP (as "AXI Adapter IP + FIFO IP"): The ports of the PS/PL interface have besides AXI conform signaling further (port) specific restrictions that require additional IP support. In all Z7k DMA ports (GP/HP/ACP) at least a signal adaption to the AXI3 standard must be done. An implementation of Xilinx's "AXI Interconnect IP" is the easiest approach to tackle the AXI3 adaption and the remaining design requirements

The FC2.0 prototype design is a multi-channel design. This means that a single Zynq device must be capable to handle several ADC channels. The upper channel limit in this design is 30 channels per Zynq device. All channels of the Zynq device are in the same clock and time domain. All channels can operate independent of each other, this means that every channel can have its own trigger domain with its own waveform and retrigger settings. As in some applications the merge of several channels is required the FC2.0 prototype firmware also has global trigger domain options to build up a common trigger domain of all implemented channels. If the detected pulses meet the trigger conditions of this sum trigger logic all implemented channel simultaneously start the corresponding trace buffering. Hence the firmware can be operated in three styles:

- Single channel mode: In this mode a selected single channel is used for trace read out. The remaining channels of the DAQ are not actively running and thus are not prompted by the server read out routine. This means that a read-out in this mode only can use the ADC and event information ring buffers of the selected channel: In this mode only the "Single Channel Trigger" Unit (appendix chapter A.5.1) of the selected channel must be operated.
- Independent Multi Channel: All implemented channels of the Zynq device are operated and thus are prompted by the server read out routine. As the event rate may be channel independent also the ADC ring buffer fill level can differ. The channels do not need to use the same waveform and retrigger settings. The server read-out routine must therefore manage the trace read-out transfer of all channels independent of each other. All buffers and "Single Channel Trigger" Units are participated in the trace read-out of this mode. This mode is currently not implemented in the server read-out routine.
- Global Multi Channel: All implemented channels are operated again, but the ADC Ring buffer read-of all channels is now controlled by the "Global Channel trigger" unit (appendix chapter A.5.1).

Like said, it can cause the simultaneously start of the trace buffer process whether the set trigger conditions were met.

Table 8 Firmware block designer. The table assigns the firmware blocks to their developer(s). The development efforts included the corresponding device drivers. Especially in case of the DMA block it is hard to split the design ideas between the authors. The not listed "Mizzis Vivado 2018.4 Development Toolchain" was mostly developed by Thomas Kihm (MPIK, Heidelberg/Germany) too. Its bash scripts uses the Vivado 2018.4 (and corresponding Petalinux) tools from Xilinx. The load-in of Trenz modul specific settings is implemented in "Mizzis" Toolchain and invokes board designs specifications that were generated and published by Trenz Electronic. The highly automated build and test commands of "Mizzis" Toolchain simplified development of the FC2.0 prototype design significantly, although in principle the toolchain was not be a must.

Mario Schütt (Author of Thesis)	**Thomas Kihm (MPIK, Heidelberg/Germany)**
ADC-Data-Read-In (unified framework for several ADC types)	CPU bank (partly out of FC1.0, DMA with contr. by M. Schütt)
ADC-Data-Read-In Dummy	Clock Generator
Delay-and-Downsampling (delay unit resorted out of FC1.0 (Kihm), modified by Schuett)	Pulse detection (resorted out of FC1.0)
DMA (mainly HP-DMA and "headeracp" extension of the ACP-CMA)	DMA (mainly ACP-CMA "default" state machine)
Trigger-Acceptance-Control	
ADC Ring-Buffer (including "headerintrace" extension)	

In a pure dummy design of the current FC2.0 firmware, an (arbitrary) implementation maximum of 12 (dummy) channels was set. Such a dummy design, without any implementation of a real ADC, was successfully synthesized. The maximum number of possible ADC channels is given by the ACP-CMA design. The (current) ACP-CMA offers in total 32 channels whereas 2 channels were occupied for a special feature. The channel limit for the current design would therefore be 30 channels. Whether such a design in practice would be synthesizable with the current firmware code was never tried. Note that number of ADC channels is also limited by the available IO pins of the selected Zynq device (chapter 6.1). In the most designs the available IO pin number will limit the possible amount of ADC channels and not the ACP-CMAs channel maximum.

4.2 DMA design

DMA (originally an acronym for "direct memory access") design in Zynq devices was the main topic of the first development cycle. In this chapter a DMA is a (firmware) state machine that can move data from the FPGA area into the PS area of the Zynq-device. As the event building is done by the APU of the Zynq devices this data transfer is an essential part of the FC2.0 prototype. For the corresponding data transfers between the PS/PL area the Zynq devices offer special ARM AMBA4 compatible ports (chapter 3.3). Depending on port style, the event building approaches, and the DMA design varies. Useful (DMA) design differences and their interaction with the event building process are the main topic of the DMA design subchapter. Originally DMAs had the task to move data on the (main) memory without using CPU load/store instructions for this.

In chapter 3.3 it was already introduced that "Ports" (or DMAs) obey (bus) address protocols. In case of the ports of the PS/PL interface of the Zynq-devices this is the ARM AMBA AXI4 protocol and its subset AXI3. Though the name "ARM …" implies that the AMBA protocol may only be compatible with ARM CPUs und IPs, this is not the case. AMBA is a general addressing protocol without manufacturer restrictions.

The names of the DMAs in this thesis are derived of their connected input. The GP/LPD-IO unit is interconnected with the GP/LPD ports, the ACP-CMA with the ACP port and the HP-DMA(s) with the HP ports. As the design of the GP/LPD ports is not designated for high performance transfers, its performance is not evaluated in this thesis (performance investigation is done in appendix chapters A.13 - A.14). The HP ports offer direct (external DDR) memory access. If the HP-ports offer sufficient performance (and spare) for a continuous data transfer of the channels data stream(s), such designs could ignore ADC Ring Buffer implementation. The saved FPGA-RAM could be used for other options. Another advantage of this approach is a result of the (main) memory size. The main memory is commonly much bigger than all available FPGA-RAM resources. This could also enable options, especially trace lengths, which are not possible in the case of the strongly limited ADC Ring Buffer sizes. HP-DMAs that can operate continuous data streams are therefore a strongly desired design feature. The ACP port has direct access to the APUs cache system. Access to a cache area never can be continuously controlled by an interconnected DMA, as due to the cache design principles a cache area is assigned to several CPU space addresses. In a routine therefore the unpleasant situation could occur that two buffers share the same cache area, though this risk can be indirectly influence by the programmer of the read-out routine (including device driver). But even in the (theoretical) case of routines that use buffers without any cache overlaps, a cache area is never exclusively aligned to a single routine. The schedulers routine arbitration affects an eviction of cache data of the previous running routine. Consequently, an ACP data transfer will have stall times at which the designated destination cache area is occupied by another routine. A continuous write into the cache system via the ACP port is therefore not possible.

4.2.1 ARM AMBA protocol

The latest ARM AMBA protocol is defined as version 5 (Figure 12) and embraces several subsets without reference to the (firmware) DMA development in this thesis. As in this thesis solely the Zynq-devices ACP and HP ports are scrutinized, an understanding of the AXI4 and ACE Lite protocol (and the corresponding PS components) is in principle sufficient. As the corresponding AMBA specifications in [17] are rather abstract, Xilinx tries to support designers by its corresponding user guides manuals and its (strongly limited) design examples. Unfortunately, Xilinx distributes the significant information over several guides and sometimes important information is totally omitted.

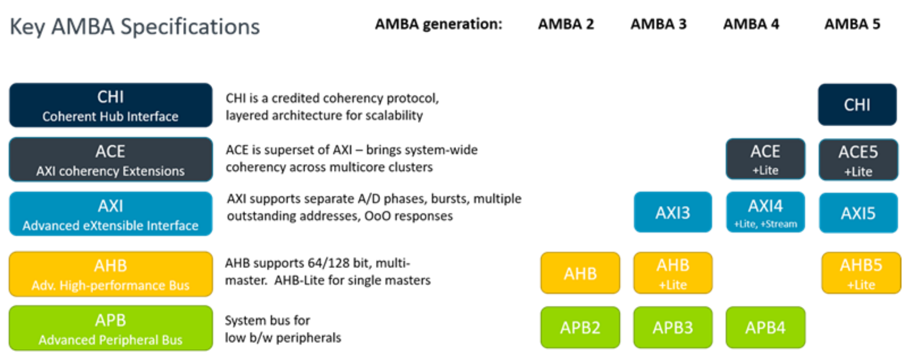

Key AMBA Specifications
AMBA generation:
AMBA 2
AMBA 3
AMBA 4
AMBA 5
CHI
Coherent Hub Interface
CHI is a credited coherency protocol, layered architecture for scalability
ACE
AXI coherency Extensions
ACE is superset of AXI – brings system-wide coherency across multicore clusters
AXI
Advanced eXtensible Interface
AXI supports separate A/D phases, bursts, multiple outstanding addresses, OoO responses
AHB
Adv. High-performance Bus
AHB supports 64/128 bit, multi-master. AHB-Lite for single masters
APB
Advanced Peripheral Bus
System bus for low b/w peripherals
CHI
ACE +Lite
ACE5 +Lite
AXI3
AXI4 +Lite, +Stream
AXI5
AHB
AHB +Lite
AHB5 +Lite
APB2
APB3
APB4

Figure 12 AMBA overview [18]

The AMBA4 protocol is a Slave/Master protocol where the read and write channels are concurrently and independently operatable. In the official specification the terms “read”/”write” depend on the direction (of the communication) and are fixed to the reference interface (Slave/Master) (Figure 13 and Figure 14).

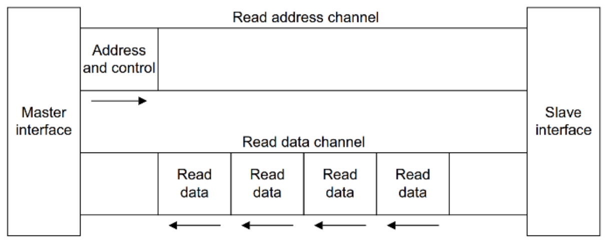


Figure 13 Channel architecture of reads [17 pp. A1-24]. A FC2.0 prototype example for an AXI4 read operation is the setting of the CPU banks control registers via the M-GP/LPD-IO unit.

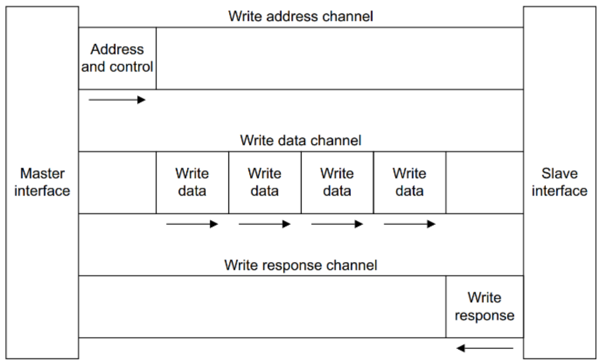


Figure 14 Channel architecture of writes [17 pp. A1-24]. FC2.0 prototype ADC Buffer-read-outs (via an AXI4 compatible S-HP-DMA or S-ACP-CMA) are AXI4 write operations. Another example for an AXI4- write operation in FC2.0 prototype is the prompting of the CPU banks status registers via the M-GP/LPD-IO unit.

The HP ports as well as the ACP port of the PS/PL interface are Slave interfaces. The connected output signals of the firmware DMAs therefore are Master interfaces. A better term as the used “ADC Ring Buffer read-out” might be “(user space) buffer write(-in)”. Both channel types consist out of an addressing and a data interface. The write channel additionally has a response interface. The read interface does not need a response interface, as the master in the case of a transfer fail can start a retry on its own. AMBA is a clock driven protocol. The read and write interface must be operated by the same clock and the same (global, low active) reset signal. For simplification, in the following only the most important signals of the interfaces are listed [17 pp. A2-31-A2-35]. “X” in signal names is a wildcard for “**R**ead” or “**W**rite” respectively. Such signals are therefore in the “write” as well as in the “read” channels available.

- AXADDR: Master gives the address of the first transfer in a burst transaction
- AXLEN: Master gives the number of transfers in a burst (number of active (operation) clocks that belong to the demanded burst). The transfer number (per burst) is equal to $AXLEN + 1$. The AXLEN value must be in the range of 0-255. In AXI3 the AXLEN value was limited to the range of 0-15. In some documents, ARM names the unit of all transfers of a burst as transaction. In the HP-DMA performance investigation chapter 6.7 the AXLEN influence on the performance is a major investigation parameter.
- AXSIZE: (Single) Data width in Bytes. The data width is equal to 2^{AXSIZE} Byte where the AXSIZE value is in the range of 0-7.
- AXVALID: Master (=HP-DMA and ACP-CMA implemented in firmware) indicates that the channel is signaling valid address and control information.
- AXREADY: Slave (= (HP or ACP) Port of PS/PL Interface) indicates its readiness to accept an address and associated control signals.

- XDATA: Data to write or read
- XLAST: Indicates the last transfer (“active (operation) clock”) in a write/read burst. Virtually the burst parameter AXLEN defined the bursts last sample and the XLAST parameter should not be necessary – but it is a useful assistant in response channel management (in S-AXI4 interfaces) that eases error detection. If the corresponding M-AXI4 interface (managed by the HP-DMA or ACP-CMA of this thesis) erratically manages the WLAST signal, the transfer must be asserted as erratic despite the fact, that the remaining signals indicate a successful transaction.
- XVALID: Master indicates that its part of the operation is “available”. In the case of a write interface this means that the master issues on its output valid data.
- XREADY: Slave indicates that its part of the operation is “ready”. In the case of a write process this means that the slave will accept issued input data. In case of a read interface its issued output data can be accepted by the master.

- BRESP: Slave indicates the status of the write transaction
- BVALID: Slave indicates that the channel is signaling a valid write response
- BREADY: Master can accept a write response signal

The addressing interface is used to define the burst transaction parameters, the data interface does the corresponding data transfer jobs and the response interface is used for success verification.

All DMA state machines have to consider the listed signal relations. Both ports (ACP and HP) use the same AMBA AXI signals. The AMBAs protocol signal dependencies are exhibited in Figure 15 and Figure 16. The implied similarity could lead to the misperception that both ports could be operated by an identical DMA that uses the same settings. As the ACP port is connected with the (APU corresponding) Cache System of the Zynq device, its permitted signal values have to comply with the requirements of the Cache System. For instance, data transfers in Cache Systems that violate the Cache Line Size (Table 2) can cause data corruption. Consequently, the AXLEN and AXSIZE values are (likely) more restricted in the case of the ACP port than the HP port. Caches accesses are commonly much more restricted and device specific as memory accesses whose behavior determines the best HP settings. In the previously listed most important "AXI4" signals only AXLEN and AXSIZE depends on the device. AXI4 has further signals whose "correct" value depends on the device and designated operation. These signals will be discussed in the corresponding DMA subchapters.

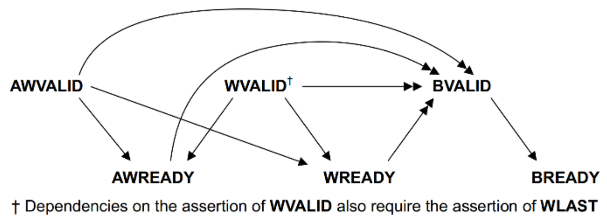

AWVALID
WVALID†
BVALID
AWREADY
WREADY
BREADY
† Dependencies on the assertion of WVALID also require the assertion of WLAST

Figure 15 Slave write response handshake dependencies [17 pp. A3-44]

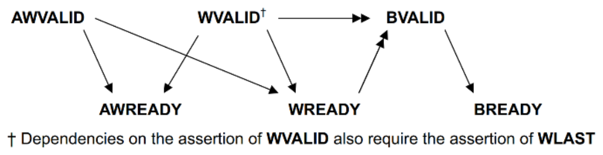

AWVALID
WVALID†
BVALID
AWREADY
WREADY
BREADY
† Dependencies on the assertion of WVALID also require the assertion of WLAST

Figure 16 Write transaction handshake dependencies (Master Interface) [17 pp. A3-43]. As a write transaction is considered, the wildcard "X" was replaced by a "W".

The handshake dependencies of the AMBA AXI4 write transactions are fairly loose and therefore counterintuitive. Note that in the dedicated handshake dependencies the "tightening" influence of AXSIZE and AXLEN is not considered. Furthermore, the handshake dependencies rules must obey all signal relations that were previously listed. As in the HP-DMA and ACP-CMA only the "Master write transaction" is implemented, their nine handshake rules (all out of [17 pp. A3-43]) must be considered. In the case of FC2.0 design write operations an accepted data input (that was confirmed via an asserted WREADY signal) must not be rejected. This has two consequences. Firstly, the developed DMA does not know all AXI4 write operations (for instance all atomic read-write operations are excluded). Secondly the response channel states in the HP/ACP FC2.0 DMA designs just assist verification, but the response channel signals have no direct influence on the assertion/desertion of the remaining channels signals. In order to simplify, the following rules that exclusively belong to the data and address channels will be listed.

- Rule 1: "the master must not wait for the slave to assert AWREADY or WREADY before asserting AWVALID or WVALID" is the most frequently misunderstood rule in the transaction procedure.

An intuitive transaction procedure would go like that: (1) the DMA issues the desired destination address and the desired burst settings (value of AWLEN and AWSIZE). (2) If (<u>and only if (!!!)</u>) the previous step was accomplished successfully (the designer must for now assume this is the case when the AWREADY signal was asserted by the slave interface), the DMA can start the transmission of the corresponding data. Informing the slave interface that the issued data is valid (=”transmission” of issued data), is quite easy. The DMA has just to asserted WVALID properly. Unfortunately, there is a high chance that the DMA would work – but the DMA procedure violates the AMBA4 specification in one critical point and therefore might (correctly) fail in another design (or in another situation). The condition “burst set must be successfully accomplished, before data can be transmitted” is false, as WVALID must be independent of AWREADY. If this is not true (WVALID is dependent on AWREADY), “rule 1” is violated.

- Rule 2: “the slave can wait for AWVALID or WVALID, or both before asserting AWREADY”
- Rule 3: “the slave can assert AWREADY before AWVALID or WVALID, or both, are asserted”
- Rule 4: “the slave can wait for AWVALID or WVALID, or both, before asserting WREADY”
- Rule 5: “the slave can assert WREADY before AWVALID or WVALID, or both, are asserted”

The loose rules have unintuitive indirect consequences. For instance, an interface can, but needs not, accept several burst transactions before the corresponding burst data must arrive. AMBA therefore can handle several outstanding addresses. The capability to manage several outstanding transactions is essential for burst merging capabilities like “Write-Combines” of cache controller. The summary of AXI4 specification is not complete with the listed handshaking dependencies. In case of the write transfers the following restrictions are additionally to consider [17 pp. A3-41]:

- Additional restriction 1: When asserted, WVALID/AWVALID must remain asserted until the rising clock edge after the slave asserts WREADY/AWREADY.
- Additional restriction 2: The default state of AWREADY/WREADY can be either HIGH or LOW, though ARM explicitly recommends HIGH. When AWREADY/WREADY is HIGH the slave must be able to accept in one clock cycle any valid address/data that is presented to it.
- The master must assert the WLAST signal while it is driving the final write transfer in the burst.

In the handshaking dependencies of the official ARM AMBA4 specification the interaction with the reset signal (as “ARESETn” in the official specification) is not implemented. Reset signals in AMBA4 have a single low active level [17 pp. A3-38]. Furthermore, the official specification demands that during a reset the signals AXVALID, XVALID and BVALID have a low level. In many designs it is a common approach that in an event of a rare DMA state machine stalling (=effectively a DMA crash) the read-out routine tries a revive of the read-out application by resetting and restarting the DMA. This approach is in AXI4 designs critical, as crashed DMAs have commonly issues with the “valid” signals and just deserting them (to low level) may have drawbacks on other components/state machines. A feasible DMA does not cause crashes – even if they are rare. Independent of their settings all burst transactions must be completely accomplished and never should be halted in the middle of the process (though the AXI4 specification does not prohibit that). In the AXI4 definition a burst transaction just can be accomplished, when the corresponding accepted amount of data transfers is equal to the AWLEN signal indicated number. Consequently, for every clock with an asserted AWREADY and AWVALID signal, $AWLEN + 1$ clocks with an asserted WREADY and WVLAID signal must exist. A (single) proper aligned WLAST assertion is a further must. If one of these rules is violated, the transfer process is in an undefined (error) state, which only can be left be extensive reset actions. In an undefined state corrupted data can show up.

This is important as continuous streaming DMAs (for instance like the continuous HP-DMA), commonly require a halt DMA feature that pauses the write/read process. In principle AXI4 enables an easy write halt via the AWVALID and WVALID signals. Desertion of them stops the write process, reassertion of them restarts the write process. In practice a data burst has to pass many buffers before it achieves its destination. Forwarding a burst to the next buffer demands less efforts when the complete burst is already available. An uncomplete burst therefore may not be forwarded to its final destination, though some samples of it already were accepted by the connected port interface and buffered somewhere in the chain. A buffer time out or a SoC/Interconnect issue can cause an error that causes a final rejection of the burst and the DMA consequently would receive an error response that belongs to the failed burst. An insufficient DMA state machine easily can be corrupted by such situations. Such (example) issues can simply be prevented by only deserting the WVALID signal when all accepted AWVALID jobs were completely accomplished. This delays the start of the "write halt" briefly, but avoids other potential conflicts.

4.2.2 GP/LPD IO unit

The GP/LPD port of the PS/PL-Interface is a Master-AXI4 interface. Therefore, the State Machine of the GP/LPD IO unit[3] must emulate a Slave-AXI4 interface whose handshaking must control the response channels. This is one of the most serious differences between the GP/LPD-IO and the ACP-CMA/HP-DMA. The implemented GP/LPD IO is mainly a modified GP IO unit instance of Xilinx (for Zynq-7k devices). A discussion of its internal design is omitted in this chapter as the read-write performance of the CPU bank is mainly given by the CDC and address condensation procedures of the CPU bank and not by the GP/LPD-IO unit.

[3] In case of the GP/LPD IO unit the term "DMA" would not be totally appropriate as the GP/LPD port accesses are directly managed by the CPU. In a typical DMA the data accesses are not managed by the CPU, the CPU just initiates (and sometimes polls) the DMA accordingly.

4.2.3 HP-DMA design

The in FC2.0 desired HP-DMA operation style is shown in the left subplot of Figure 17. The HP-DMA is fed with a continuous stream of input data, that it continuously forwards into the actual memory subbuffer (highlighted as light blue rectangle). If no trigger appears, the memory subbuffer elements are overwritten periodically. The current subbuffer has like all the remaining “trace subbuffers” exactly the size of the desired trace length. As a trigger pulse marks the time of the pulse edge arrival, the active write stream (blue arrow) skips in the next subbuffer after it buffered further post-pulse samples in the current buffer. If a trigger is perceived in the last subbuffer (“Buf n”), the (continuous) write process continues in the first buffer (“Buf 1”). In total the HP-DMA can buffer an amount of n subbuffers. The precise value of n depends on the trace length and the available total buffer size on memory.

Another HP-DMA operation approach is shown in the right subplot of Figure 17. In this operation style the HP-DMA is used to read-out an ADC Ring Buffer. The mode is therefore similar to the designated ACP-CMA operation style. A brief interrupt in the read-out transfer is not an issue, as all the trace samples are still buffered in the ADC Ring buffer and therefore still available. In this design proposal the HP-DMA needs not to write continuously in the external DDR memory – similar like the ACP that cannot write continuously into the cache. In contrast to the ACP-CMA the HP-DMA is not capable to organize cache synchronous transfers. In HP-DMA mode a CPU access of the trace data therefore always requires a cache invalidation of the corresponding area.

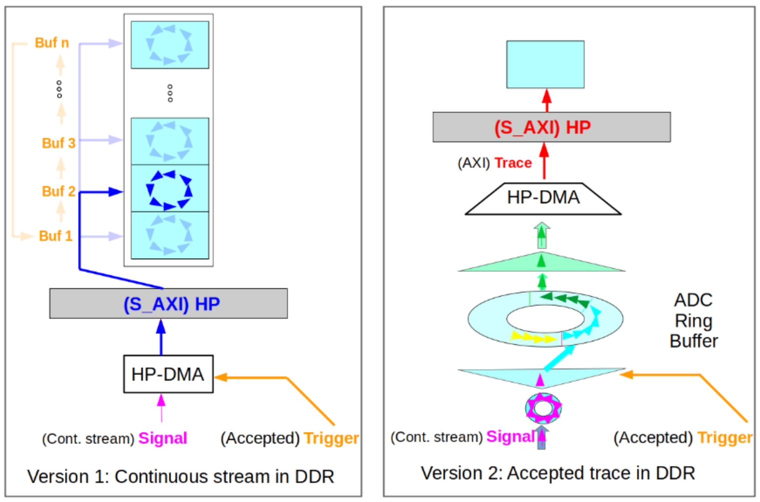


Figure 17 Possible HP-DMA implementations (simplified). The blue arrows in the blue highlighted rectangles symbolize a buffered sample burst. If it is assumed that a burst size of 8 is selected, a trace in the left subplot would contain 64 samples (as every subbuffer contains 8 bursts (64=8×8)). Recap that HP ports have FIFOs and a corresponding control/synchronization logic (with accessible status registers) that were optimized for high-performance data transfers (specification partly done in Table 5).

The AXI4 protocol defines so called data bursts as lowest viable data transfer unit. In the previous chapter 4.2.1 the burst size was introduced as signal “AXLEN”. In the continuous HP-DMA mode (of the left subplot) the trace length (“subbuffer size”) needs to be a multiple of the used burst size. As the number of bursts per subbuffer is practically only limited by memory size, a designer could assume that the lowest possible burst size enables the most trace lengths and consequently is a good design decision. Unfortunately the burst size also influences how effectively the HP port internal FIFOs can be operated. The determined best burst size is consequently for every application different. In AXI4 merely the indication of the destination address of the first "sample" of every burst is required. The remaining (incremental only) addressing is the task of the PS internal logic and need not be done by the DMA.

The implementation of the HP-DMA must be done as state machine. The state machine is never allowed to "lose" any sample of the continuous input stream in its active running state. In other states, like initialization and hold, where the applied input data is not used, a sample loss has no negative effects. Not all DMAs have such phases. For instance, at DMAs which are used for ADC Ring Buffer read out, every forwarding sample loss has to be avoided. A strongly simplified state machine setup of the HP-DMA state machine is shown in Figure 18. This state machine assumes that the HP port is always ready (this means wready and wvalid are always asserted). This premise simplifies the design of the initialization and the hold state of the state machine. Note that in the real state machine this premise is not suited. The internal states management used in Figure 18 is similar to the chosen state management in the HP-DMA which was used for performance evaluation. The signal "wen" must be asserted to start the run. In order to hold the DMA "wen" must be deserted. As in the simplified state machine "wready" is always asserted, the data transfer confirmation via "wvalid" assertion is quite simple. Its state machine just needs the "awvalid" signal as count basis. In order synchronize the other signals the counter "burst_pos" was introduced. In the evaluation phase of the HP-DMA development it was determined that the address should be issued four cycles before the corresponding data. Otherwise, a continuous data write is not possible. Conclusively only bursts with a minimum length of 8 transfers are eligible in continuous HP-DMA settings. Therefore, the "awvalid" register is asserted when the value of the "burst_pos" counter is equal to AXLEN-4.

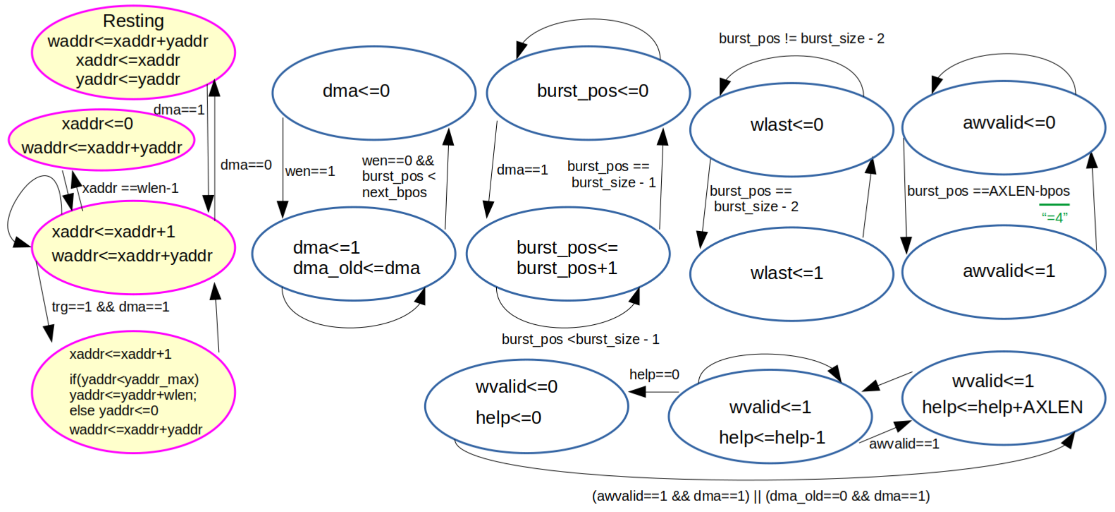


Figure 18 State machines example for HP-DMA (simplified). The illustrated state machine implicitly assumes that the HP-DMA is connected to a HP port that is always (instantly) ready to accept a given burst start address (via signal AWADDR) and data samples (via signal WDATA) (the port readiness is given by the AWREADY and WREADY signals). In real application this needs not to be the case – therefore this simplified proposal is not eligible for a real implementation. The real state machines of the HP-DMA must consider the port readiness. In the implemented real DMAs the surrounding logic is not enhanced enough to exclude the mentioned data corruption (due to forwarding sample loss) in the init/halt states. The degenerating effects of this behavior can be mitigated/precluded by software control.

The example addressing state machine (yellow circles in Figure 18) is simplified. The simplified state machine completely describes the desired periodical overwriting (via "xaddr") and the subbuffer jump behavior (via "yaddr") of the described continuous HP-DMA model of Figure 17. Like described, every subbuffer has a size of wlen samples and therefore exactly agrees with the demanded trace length. Between the subbuffers is no free memory space. In agreement with the desired amount of subtraces, the (software) initialization routine has to set the corresponding value of yaddr_max. An important difference to the real implementation partly affects the subbuffer jump behavior. The simplified state machine immediately jumps into the next subbuffer when it receives a trigger pulse and hence in general would not buffer the intended waveform parts. As this misbehavior is not significant in the system performance chapters of this thesis, the issue and the possible remedy actions are only discussed in appendix chapter A.6. The continuous HP-DMA which is used to evaluate the system performance (chapter 6.6) does not require several subbuffers. Therefore all "yaddr" subbuffer logic parts are not implemented in its addressing state machine logic.

The evaluation of the continuous write mode must be done by firmware. The implemented software checks are just an unsafe second opinion that affirms the firmware results. To determine the failed non-continuous transfers, it is necessary to count the number of clocks in that either the WREADY or the WVALID signal is deserted. To probe the plausibility of this result, counting of the last subsequent continuous write must be done. Synthesis (of firmware designs) with operation frequencies higher than 100MHz (a quite rough rule of thumb) are more likely successful when their internal counters have a restricted width below 32bit. A 32bit counter has a value range of approximately 4 billion figures. If this counter is incremented by a

200MHz clock, it will overflow after roughly 20s. This is much less than the demanded monitoring time, which at least should cover an hour. To implement a sufficient time duration, a long-time monitoring state machine like in Figure 19 was implemented. In the state machine the 32bit counter "synch_10us_cnt" is incremented in a period of 10µs. If in this 10µs period a write transfer interruption shows up, the counter value is set to zero at the end of the period.

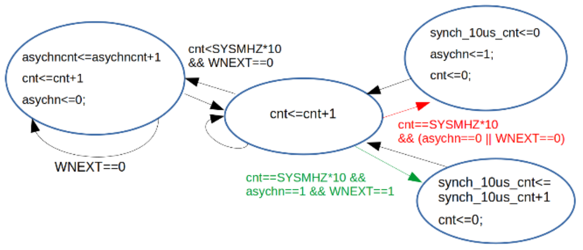


Figure 19 Suggested state machine to verify continuous HP-DMA writes (simplified). The signal WNEXT has the (logical) relation WVALID & WREADY. The real state machine also has a monitoring exclusively for WVALID (implemented as counter "wvalidcnt") and further check values that enable a conclusion of monitoring logic behavior.

Figure 20 delineates the chosen implementation of the HP-DMA in the performance investigations of chapter 6.6. An elaborated version of the proposed monitor state machine (Figure 19) is included. Its check parameters "lastdatain" and "lastawaddr" enable a superficial software check of the HP-DMA. The software check is a pure backup measure (in order to detect possible issues of the monitoring logic) and cannot affirm an error-free DMA run. It is just possible to check the error-free transfer of the last "wlen" samples.

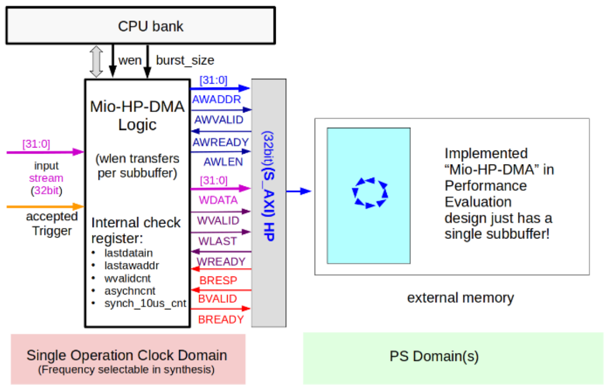


Figure 20 HP-DMA setup for performance evaluation

4.2.4 ACP-CMA design

The acronym CMA means coherent memory access and should not be mixed with the "contiguous memory allocator" of the Linux DMA device driver framework which uses the same abbreviation. The implied data coherency is achieved by using the ACP port ("Accelerator Coherency Port") whose system interconnection was introduced in chapter 3.3.3. As the Cache of the Zynq-devices is not eligible for strictly continuous data transfers, the continuous HP-DMA concept is not applicable at the ACP port. The ACP-CMA was developed for designs that use ADC Ring Buffer blocks to buffer accepted events and is completely optimized for this kind of implementation. The main developer of the basic ACP-CMA framework was Thomas Kihm (MPIK, Heidelberg). The following sections tries to describe the main differences between the HP-DMA and the ACP-CMA. One key difference between the ACP port and the HP ports is their availability. As only one

ACP port is existing, a multiplexer input is required for multichannel operation. The term “ACP-CMA channel” refers to the corresponding multiplexer input. The most severe advantages of the ACP-CMA are the direct data flow into the Cache and the given hardware support for data coherency. The CPU needs not to invalidate the cache anymore. As the trace data is buffered in the ADC Ring Buffer and hence not volatile anymore, the ACP-CMA can interrupt the transfer of trace data and insert “accelerator” data. This approach is used to enable an efficient polling whose basic idea is delineated in Figure 21. The ACP port transfers must consider the cache limitations. A burst transaction must have in total 64 B “alignment” (specification is out of Table 6) – this is equal to 32 16 bit sample or 16 32 bit samples. If the demanded trace length has less than 64B or is not an exact multiple of it, the last corresponding burst must be padded accordingly. If the polling routine knows the value of this padding transfers, they can be used to determine if the DMA operation was successfully accomplished. If the trace size is a multiple of the burst size, no polling information will be available in this approach. In such cases the ACP-CMA must attach a subsequent (additional) burst which exclusively consists out of known poll check values.

The overhead of the suggested transfer principle decreases with trace length. In practice this means that the ACP-CMA must be capable to read-out several traces in one transfer. The ACP-CMA interface parameter “wlen” therefore can comprise the multiple of the user defined trace length. The ACP-CMA interface parameter “sradr” conveys the ADC Ring Buffer block the desired ring buffer start address of the transfer (mostly the ADC Ring buffer start address of a trace that corresponds to the (read-out server) demanded event). It does not influence the destination address in the “cache”. Pay attention to the quotation marks around the term cache. For a CPU a cache access is just a (hopefully) faster main memory access strategy. In a cache access the CPU still invokes/prompts memory addresses (even though it gets the results from the cache).

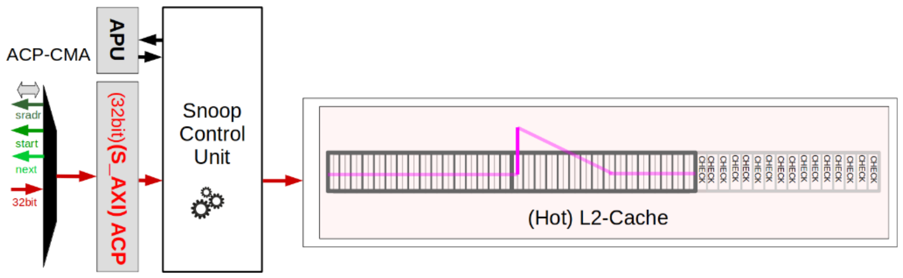


Figure 21 Polling in ACP-CMA (Simplified system overview). In this draft it was assumed that every transfer (indicated by the small rectangles) has 4B and consequently consists out of two 16bit samples (indicated by the vertical grey line that splits the rectangles). As the user selected a desired trace length of 64 samples, a third (additional) burst for the “check” poll information was required.

The ACP-CMA must obey the same AXI4 handshake regulations like the HP-DMA. Their “wvalid”, ”awvalid” and ”wlast” state machines are still a useful reference (Figure 18). The continuous HP-DMA state machine has the burden that even single clock interruption in the (trace) write stream must be asserted as trace corruption. But the continuous HP-DMA state machines design also has the advantage that a sample loss in their initialization and hold phases does not influence the future read-out traces (this can be excluded by the temporal exclusions in the read-out process). In the ACP-CMA trace corruption via sample loss (or different misbehavior) in all its states matters. Furthermore, the ACP-CMA has to organize the desired polling behavior. The good news is that the demanded cache coherency requires slightest adjustments in the presented AXI4 signal values. Out of the presented signals just AXSIZE, AXLEN and WSTRB are subject of further restrictions (Table 6). In the current FC2.0 prototype the ACP-CMA and the HP-DMAs use a data width of 32 bit (AXSIZE=3).

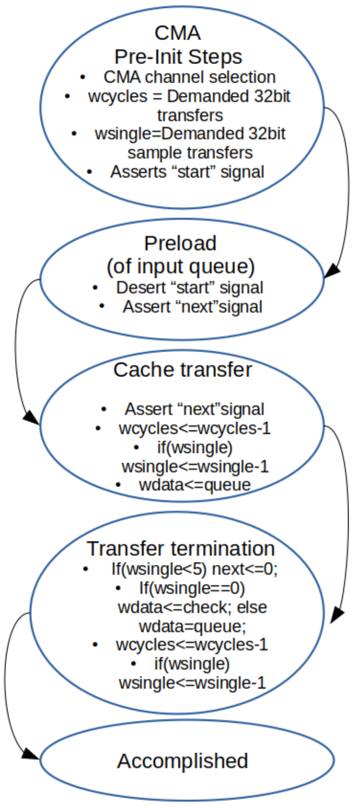


Figure 22 Internal procedure ACP-CMA (Developer: Thomas Kihm (MPIK, Heidelberg-Germany)) (Simplified). In order to simplify, it was assumed that the ACP-CMA has to transfer more than four 32bit "samples" – the real ACP-CMA can handle transfers between 1-32768 32bit samples. The labeled states names in this graph are not used in the firmware code.

The sequence of the ACP-CMA states is shown in Figure 22. In the first state (labeled "CMA Pre-Init Steps") the device driver must set the control register values and activate the state machine. Setting control registers (via CPU) includes setting of the desired channel, "sradr" value and subsequently the "wsingle" value. The ACP-CMA does transfers of 32 bit words. The ACP-CMA interface now asserts the "start" signal and forwards the designated "sradr" value. The input data of the state machine has to pass pipe and switch registers. It is the task of the "Preload" state to (pre) load these registers accordingly. The third state (labeled as "cache transfer") is the main state of the ACP-CMA. The AXI4 handshaking state machines now need to run. At some point, the last sample will enter the first pipe register of the pipe register chain. This is the case in the "transfer termination" state. In the remaining operation clocks of the ACP-CMA transaction the spill out of the pipe registers and the issuing of the poll check values must be done. In the last state ("Accomplished") the ACP-CMA must not transfer any further data.

In Table 2 (page 14) it was listed that the Zynq-7k/Zynq-Ultra+ has a cache size of 512 kB/1 MB (=256/512k(16 bit) samples). In operation it is not useful to try to occupy the complete cache space with a trace – recap that there must be sufficient cache space for the event building and other tasks. If no sufficient space is available these forces "cache trashing" and hence a performance loss is the consequence. Thomas Kihm determined in his scrutiny a feasible ACP-CMA maximum transfer size of 128 kB (=64k(16 bit) samples).

In the previous section the ACP-CMA processing was describe for a single channel. FC is a multi-channel design and therefore the ACP-CMA operation was optimized for multi-channel read-outs. The interaction of the single ACP-CMA channels in multi-channel mode is depicted in Figure 23. The interaction is mostly driven by the fact that every ACP-CMA channel has its own fixed ("cache") destination address after initialization. The allocated channel spaces have no overlaps. In fact, every allocated channel space offers a lot of spare and could occupy much more samples than the given maximum of 64k.

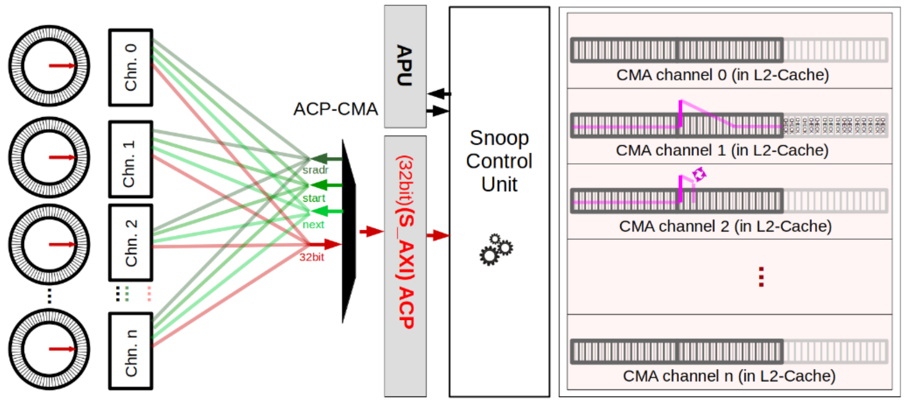

Chn. 0
Chn. 1
Chn. 2
Chn. n
ACP-CMA
sradr
start
next
32bit
APU
(32bit)(S_AXI) ACP
Snoop Control Unit
CMA channel 0 (in L2-Cache)
CMA channel 1 (in L2-Cache)
CMA channel 2 (in L2-Cache)
CMA channel n (in L2-Cache)

Figure 23 Optimal operation ACP-CMA : As just one ACP port is available all channels must be linked with a corresponding multiplexer input. In this instance the cache contains (besides the poll-check-values) just trace data.

The fixed destination address has serious consequences. Recap again, that every ADC Ring Buffer is just connected with a single, exclusive ACP-CMA channel. If only a single ADC channel is operated, the software read-out routine must wait as long as the returned trace(s) were processed (and forwarded) by the event builder "jobs" before the routine can start another (subsequent) read-out of new traces. In a run that operates more than one channel the ACP-CMA transfer break can be circumvented. During event building execution of channel X, the ACP-CMA data transfer of channel Y already can be prompted.

In the previous examples of Figure 22 and Figure 23 every ACP-CMA invocation only transferred the trace data of a single event. This was done to simplify the illustrations and is not a real limitation. The implemented ACP-CMA can invoke a transfer of an arbitrary number of subsequent buffered traces as long as their total sample number does not exceed 65536 samples. In the first global firmware design draft it was intended to read out the event information ring buffers via the GP/LPD IO. During the development process it became apparent that the event information access procedure is not suitable for all application cases. Hence the "headeracp" mode was developed. This mode is additionally implemented and is an extension of the default "via GP/LPD" mode that is illustrated in Figure 22 and Figure 23. In "headeracp" mode the trace as well as the event information are read-out via the ACP-CMA. An instance is illustrated in Figure 24. In the illustrated instance the event building routine demands 3 events ("a-c"). After the completion of the transfer, the traces (combined magenta arrows in Figure 24) and their trace headers (grey arrows in Figure 24) are properly aligned available in the cache space of the corresponding ACP-CMA channel. For this result only a single prompting of the ACP-CMA in “headeracp” mode was necessary. That “properly aligned” means will be discussed in chapter 5.2. In “headeracp” mode the client routine only must provide the start address of the first trace in the ADC ring buffer, the sample number of a single trace and the total transfer size (can be obtained out of the total sample number and total header number).

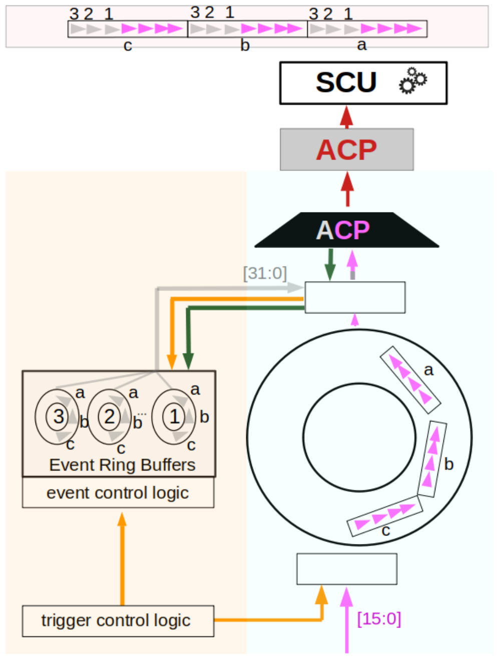


Figure 24 Combined trace and trace header read out in “headeracp” mode. The differences in the background color emphasize the different block affiliation of shown buffers. The event information ring buffers (orange background) are part of the “trigger acceptance block”, the ADC Ring Buffer (bright blue background) is part of the ADC Ring Buffer block. In the “headeracp” mode hence several different state machine must interact with each other.

From this information the control logic can determine its read-out procedure. The “headeracp” mode uses an enhanced control logic that is not discussed in this thesis. It should be noted that the ACP-CMA state machine only got minor extensions and all introduced interface signals are still participated in the transfer process. As the demanded information is distributed in several different block resources the challenge in the development process is mainly the proper synchronization of the participated components. In Figure 24 only a single channel and a reduced signal set is shown in to simplify the illustration. In the FC2.0 prototype all channels have the “headeracp” capability independent if they are operated in single channel or multi-channel mode. The ”headeracp” mode still requires the “event information ring” buffers and hence its buffer capacity is also limited by their size. If non-seamless retriggering is acceptable, the event information ring buffers would not be necessary. In this case the event information data besides the trace data could be buffered in the ADC Ring Buffer. In this mode, that will be introduced in the appendix chapter A.4.3 as “headerintrace” mode, only the ADC Ring Buffer sizes determines the event buffer capacity

5 FlashCam 2.0 Prototype Software Design

Software design is an essential part in DAQ development that significantly determines the final performance of the DAQ. The chosen software design approach depends on many variables (available memory size, operation system, available libraries, required reliability, …) and so said obviously not a single solution exists. Xilinx promotes various support libraries and operating systems (OS). The strongest support besides its “Bare Metal” solution (no OS is used) is focused on its Linux chain. In “Bare Metal” environments the device driver development is significantly simplified as in such systems a split into Kernel and User Space does not exist. In Linux the separation in Kernel/User Space is one of the essential design key points and consequently device driver development is getting more challenging. Linux also offers many advantages and consequently it was decided to build up a DAQ that uses Linux as OS. Linux is massively supported by all major tech companies and a huge free software community. Currently Linux supports various devices driver implementation styles. After an elaboration phase it was decided to use a “user space” device driver approach for intended DMAs. The functionality of the new developed DMA libraries is presented in chapter 5.3, A DAQ not only consists out of DMA device drivers. Already available MPIK software is also used in the FC2.0 prototype DAQ. Chapter 5.2 is dedicated to MPIK’s raw-ethernet library and contains a strongly condensed summary of the used read-out protocol interconnection. To understand the developed read-out protocol, it is necessary to understand a typical measurement sequence with the FC2.0 prototype. A discussion of it is done in chapter 5.1.

5.1 Typical sequence of a FC2.0 prototype measurement

In this subchapter the typical sequence of a FC2.0 prototype measurement is discussed. The interaction between the Read-Out “Server” and the FC2.0 prototype (“Client”) determines the possible measurement performance. In the power up process the board autonomously boots its Linux OS. After accomplishing the boot process, a ssh server is available for remote control accesses. If necessary, the read-out server therefore can transfer the firmware and software “binaries” and subsequently start them.

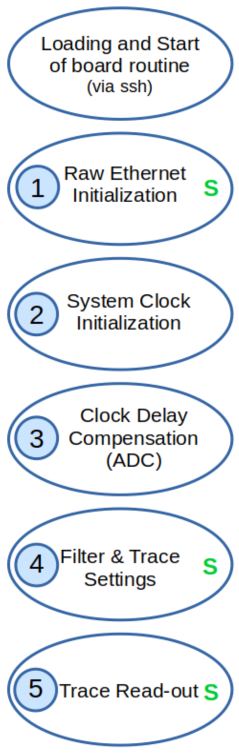

Loading and Start of board routine (via ssh)
1 Raw Ethernet Initialization S
2 System Clock Initialization
3 Clock Delay Compensation (ADC)
4 Filter & Trace Settings S
5 Trace Read-out S

Figure 25 Typical sequence of a measurement

(1) In the first phase the server and the client routines must initialize the ethernet transfer settings. Important specifications which have to be set are the (socket) timeout (time) and the ethernet packet size.

(2) In the second step the system clock initialization needs to be done. This assumes the availability of a "Clock Generator" block (Figure 11) and corresponding devices drivers.

(3) In the current setup the "clock delay compensation" step (between ADC Framing Clock and corresponding input clock of the system) is required in the initialization run of the ADC data read-in block.

(4) The trace length and the corresponding frame trigger settings must be done in the penultimate step. The pulse detection unit can be operated with different shaping parameters. Amongst other things the optimal shaping parameters depend on the detector/signal generator properties and the desired application. The server read-out routine must determine their accurate value in this step too.

(5) The "Trace Read-out" performance is decisive for the performance of the whole run. Figure 10, that illustrated the interaction of the different FC2.0 firmware resources and the read-out server's commands, was dedicated to the measurement phase. The clients' routine tasks in this "measure" state are summarized under the term "event building". Event building includes the trace transfer into the cache ("cache transfer"), reply message generation ("ethernet buffer building") and its ethernet transfer. Its performance critical points and their interleaving with the used ethernet protocol will be discussed in the next chapters.

5.2 The FC2.0 ethernet read-out protocol

The MPIK raw ethernet framework was developed by Thomas Kihm and Frank Köck (both MPIK, Heidelberg-Germany). In experiments that use FC, data is transferred inside a single network. Data is not passed between networks. The MPIK raw ethernet framework could be described as a smart and slightly extended version of the original IEEE 802.3 raw ethernet framework. It uses the BSD socket library and has its own control structure ("efbp"), but the framework is designed in a way that the framework implementor never directly accesses the code basis. The protocol assumes that the data exchange takes part between two hierarchical partners (Figure 26). A read-out that uses such a protocol is simple, but it also must consider two major issues:

(1) The board ("Zynq" in Figure 26) never can know when it receives its next command.

(2) A server that uses the MPIK raw ethernet protocol cannot know if the corresponding recipient successfully received the command. If after a well-defined time the server does not get the expected response, it must assume that the transfer failed and therefore must repeat the request.

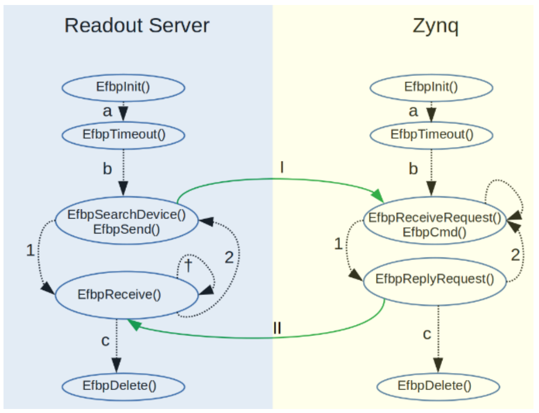
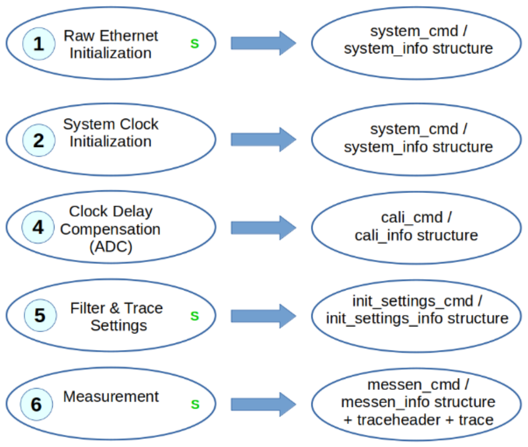


Figure 26 Raw Ethernet Framework Function Interaction (single partner mode). In this figure the read-out server routine just must control a single board (therefore "Single partner mode"). In real life experiments the read-out server must control several boards.

The default interaction of the MPIK raw ethernet framework partners can be seen in Figure 26. First the initialization of the ethernet transfer settings is done (state transition arrows a,b). Afterwards the read-out is done ("1,2,I,II"). State transitions 1 and I represent the transfer of the command messages. Command messages contain command structures that precisely instruct the addressed Zynq. Transitions 2 and II represent the transfer of the status messages. Status structures, that are content of the status message, must inform the server if its related command was processed successfully. The termination of the routines requires the deallocation of the implemented socket resources. This is done in Step c. Every state of the measurement (Figure 25) is controlled by a command structure. The designed command/status structure alliance for the FC2.0 read-out protocol is illustrated in Figure 27. The system_cmd/system_info structures are used to inform the server about the rough technical specifications of the board. The settings of the firmware blocks are modified by the cali_/init_settings_ structures. The command and the status structure of the virtual event read-out phase start with "messen_" ("messen" is German verb for "measure"). The in Figure 10 already introduced parameters "permtrgnum" and "endevtno" are part of the messen structures. In the event read-out phase every client message consists out of the messen_info structure, the demanded traces (of the prompted events)



1 Raw Ethernet Initialization S
system_cmd / system_info structure
2 System Clock Initialization
system_cmd / system_info structure
4 Clock Delay Compensation (ADC)
cali_cmd / cali_info structure
5 Filter & Trace Settings S
init_settings_cmd / init_settings_info structure
6 Measurement S
messen_cmd / messen_info structure + traceheader + trace

Figure 27 Measurement states and their corresponding command and status structures

```
struct traceheader{
unsigned int evtno;
unsigned int deadevents;
unsigned int pps;
unsigned int ticks;
unsigned int adc_num;
int          num_of_adcs;
unsigned int num_of_samples;
float        deadtime;
unsigned int cmd_num;
unsigned int blsum;
};
```

Figure 28 Trace header structure

and the corresponding trace header. The trace header content must enable an error-free read-out of the (file) stored data and furthermore provide all information which is required for the desired analysis. The FC2.0 prototype store files contain all "_info" structures (Figure 27) and all desired traces (including trace header) of the corresponding measurement run. The "_info" structure contain a significant amount of information. Status structure and trace header information can be used to check the validness of the permanently stored event data file on the server's hard drive. In the current design the trace header has the parameters evtno, pps, ticks, adc_num, num_of_samples and blsum. "pps" and "ticks" contain the accurate time information of the event. "blsum" is the baseline level that was determined by the baseline follower. "pps","ticks" and "blsum" have in common that they are automatically buffered in the Event Information Ring Buffers by the Trigger Acceptance Control unit. The event building routine typically accesses the Event Information Ring Buffers via the ACP port. This decision has primarly performance reasons. In principle also other ports could be used.

Before the board's ethernet transfer "job" can transmit the requested information to the server, the event building routine must update the information in the corresponding ethernet buffer. In the default procedure the EfbpReplyRequest() function transmits only data out of this buffer to the server. The structure in the user space "ethernet" buffer is shown in Figure 29.

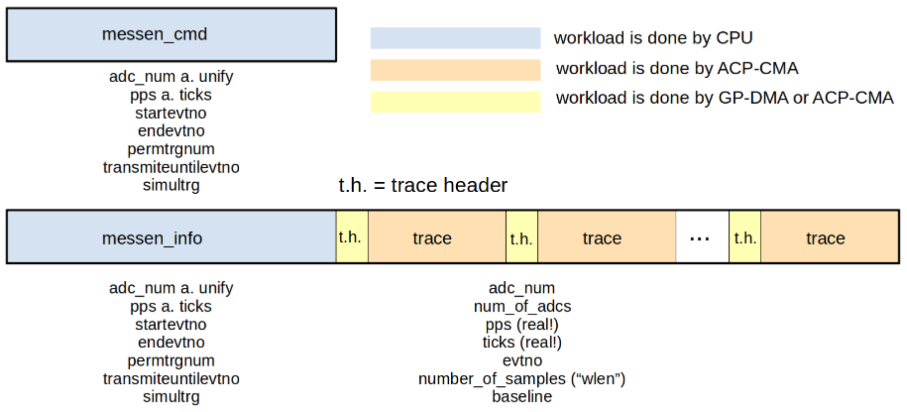


Figure 29 Ethernet Buffer setup : The messen_cmd structure is the only content of the request message (transmitted by the read-out server). The corresponding reply message (transmitted by the board ("client")) would require (run time) dynamic structures that are not available in C. The first part of the reply message is occupied by the messen_info structure. The messen_info structure size is independent of the measurement settings, therefore the read-out routine can always determine the position of the attached trace header of the first trace. It is important that the trace header is in front of the trace as otherwise the read-out routine cannot determine the length of the trace. The scheme stops with the last trace of the reply message. All reply messages must include the messen_info structure at the beginning (even if no trace is demanded), as the read-out server determines the clients state out of this information. The desired reply structure can be achieved by usage of memcpy(). An efficient usage of memcpy() requires that the demanded transfers have a size that is a multiple of 4B (in ARM and X86). The structures ("…_cmd"/"…_info"), trace and trace header size always consider this efficiency requirement.

Now all structures that are used in the event control & read-out process are known. Figure 30, a modified extraction out of Figure 10, shows the interaction between the read-out server and the Zynq. The read-out server is repeatedly polling the Znyq for its status. The client replies with the current value of the latest detected event ("endevtno"). As soon as the server notices a change, it initiates an event readout by incrementing "transmituntileventno" parameter accordingly. Once the events are received, the server must inform the Zynq that its now free buffer can accept new events. In the read-out protocol it's the "permtrgnum" parameter task to prompt the client to do so. A simplified instance state machine that does the update process of the cmd_messen structure can be found in appendix chapter A.9.2.

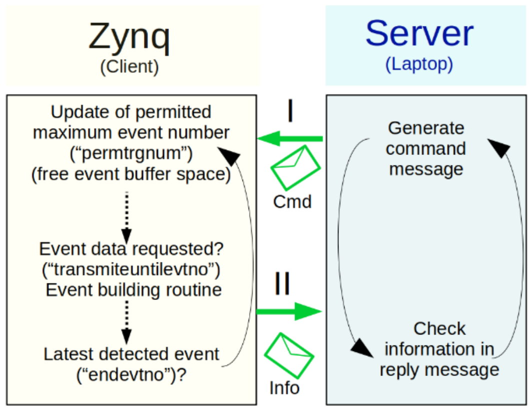

Zynq
(Client)
Server
(Laptop)
Update of permitted maximum event number ("permtrgnum") (free event buffer space)
Event data requested? ("transmiteuntilevtno") Event building routine
Latest detected event ("endevtno")?
I
Cmd
II
Info
Generate command message
Check information in reply message

Figure 30 Read-out server and Zynq interaction during the "measurement state" of the read-out process

5.3 DMA frameworks in FC2.0 prototype

The ACP-CMA framework (Figure 31) completely supports the ACP-CMA cycle during a measurement run. The InitCpuMio() function enables all (firmware) DMAs of the FC2.0 prototype. It is already invoked in the initialization step ("Load and Start of board routine" in Figure 25) and therefore has no interaction with the read-out server routine. The allocation of an ACP-CMA channel is done via function CpuCMA().

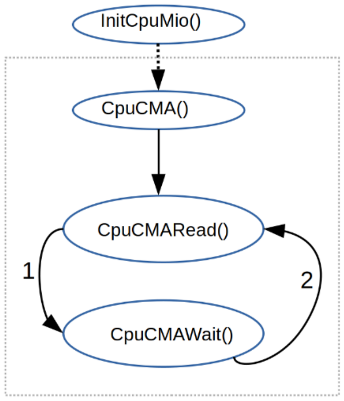


Figure 31 ACP-CMA Framework

Like malloc(), CpuCMA() returns a pointer to the start address of the reserved "Cache" space. The CpuCMA() function is therefore also responsible to determine the virtual address of the physical address of the ACP-CMA state machine. The ACP-CMA read-in process is started by the CpuCMARead() function. The CpuCMARead() functions sets the signal "sradr" ("Start address in ADC Ring Buffer") and the transcation length "wlen" of the ACP-CMA interface and afterwards starts the transaction. The operation of the ACP-CMA just needs the CPU for this initialization settings. After this brief period the ACP-CMA transaction does not need further run support by the CPU. The CpuCMARead() function returns immediately after the transaction initialization was accomplished. The ACP-CMA transaction is still running. During the running phase, a CPU access of reserved ACP-CMA channel buffer, will return corrupted buffer values. The function CpuCMAWait() frees the corresponding ACP-CMA channel as soon as the transaction is accomplished. The function detects accomplishment via the described “check value” polling strategy. A data corruption free CPU access of the ACP-CMA cache channel buffer is possible now.

The presented HP-DMA framework (Figure 32) of this section is the basis of all HP-DMA device drivers. For successful operation, the device drivers need firmware specific extensions. CpuDMA() is the HP-DMA equivalent of CpuCMA(). In contrast to the ACP-CMA framework the HP-DMA framework has a maximum of 4 channels (not implemented in the actual FC2.0 prototype firmware). Like the CpuDMAReset() function name implies, it is responsible for the reset of the corresponding HP-DMA channel. The CpuSyncDMA() function instructs the invalidation of the selected cache lines. A modified version of the default library memcpy() is function ReadCpuDMAWords32(). It is used to copy the trace data out of its reserved HP-DMA channel buffer to the (user space) ethernet reply buffer. In the FC2.0 prototype the HP-DMA has no direct polling features, therefore a HP-DMA function like CpuDMAWait() is not existing. Both DMA frameworks have in common that their buffers address space must be protected by unwanted accesses of other user routines or kernel threads. The setup of this reserved memory regions is done by the kernel. In the FC2.0 prototype, the development toolchain does the reservation of the addresses space by fixing the corresponding device tree properties. As the kernel build process must consider the device trees specifications, the running kernel knows the reserved address and consequently unwanted DMA buffer accesses won’t be an issue.

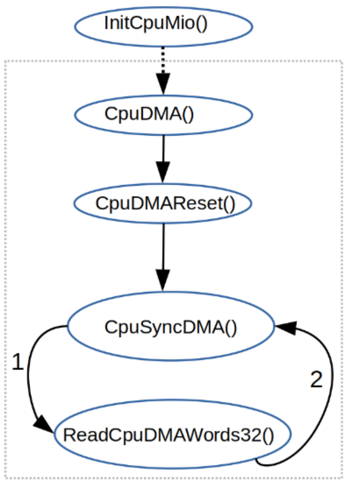


Figure 32 HP-DMA framework

6 FlashCam 2.0 Prototype evaluation

In the following subchapters the performance of the Zynq devices and their feasibility for a FC1.0 successor DAQ is investigated. Chapters 6.2 and 6.3 probe the memory access and arithmetic operation performance of the bare APU (“Zynq CPU”) and their feasibility for event building and signal processing. Chapter 6.4 and chapter 6.5 classify the bare bandwidth performance of the developed ACP-CMA framework. The ACP-CMA is a rich topic and due to lack of space the chapters only contain results that are required to understand the outcome of the subsequent performance elaborations. Non-essential results are omitted or are partly presented in the appendix. For instance, the performance comparison between the GP/LPD-IO, the ACP-CMA and the “Microblaze-IO-DMA” in FC1.0 is part of appendix chapter A.14. Chapter 6.6 and chapter 6.7 belong to the investigation of the continuous HP-DMA performance.

6.1 Evaluated Trenz Zynq modules

In this subchapter it is not possible to explain all the design ideas that influenced the development of the functional blocks and consequently it is not possible to derive their principal design limits and requirements. The focus here is on the FPGA resources requirements of a typical multi-channel HPGe-detector DAQ. The resource consumption of this instance DAQ is only roughly comparable with the FPGA resource consumption of the FC2.0 prototype DAQ as the latter is also intended for PMT experiments and additionally includes several trigger modes.

The FC2.0 prototype module basis, is like in all future FC2.0 DAQs, a Zynq SoC module from Trenz Elektronik (Hüllhorst-Germany). Trenz-Elektronik is a manufacturer of pluggable FPGA Modules (for several device series and devices) that all use the same (as far as possible) plug interfacing system. This empowers the development of DAQ boards whose selected FPGAs are tightly adjusted on the required performance of the application. Due to time and cost reasons not all FC2.0 compatible Trenz SoC modules could be elaborated. The elaboration nominees are listed in Table 9. All listed Trenz SoC modules have besides the Zynq devices an external DDR memory system implemented. The size and the used DDR memory type depend on the Trenz module. A later exchange or extension of the memory system is not supported. Please note that the Trenz Modul cannot be operated on their own. Their power supply and IO signals (including the FADCs data streams) must be provided by a so-called “Carrier board”. For the future FC2.0 DAQ it is intended to develop a customized carrier board, but the current FC2.0 prototype uses the Trenz developed and shipped “TEBF0808” Carrier board (separately emphasized in Figure 35). Besides the switching power supplies, it has an FMC plug that is used to interconnect the available FADC Evaluation boards, a RJ45 plug (for the 1Gb/s ethernet link), a SD-card plug and a clock generator chip whose clock is partly used to drive the SoC implemented clock tree.

Within a Zynq device series, the setup of the PS system is roughly comparable. The main difference is the available APU number. As in FC2.0 the focus is on event building and data movement, the APU amount is not the main concern of a device and consequently the differences in the APU specifications are not part of Table 9. The Zynq devices most severely differ in the FPGA fabric and in the IO circuits. Important FPGA metrics for FC2.0 are

- Available Pins with LVDS
- Available Pins with compliance to CMOS 3.3V technology (or similar)
- Available Multi Gigabit Transceiver (MGT) Pins (in Table 9 labeled by their Xilinx subclass term)
- Available FPGA-RAM in BRAM
- Available FPGA-RAM in UltraRAM (ZU+ only)

The pin amount of a device determines the implementable number of ADC channels and the total available IO bandwidth per device. The former is an important metric for the costs per channel, the last determines the readout-performance, synchronization and trigger possibilities of the final system. The available FPGA-RAM limits the buffering capabilities and consequently the trigger capabilities of the design.

Table 9 SoC device nominees for FC2.0. The "TE0808-05-9BE21-AK" is the most likely SoC nominee for the designated FC2.0 master card were costs are less important and the (buffer a. IO) performance is the main issue. The elaborated "FC2.0 prototype" design has a higher conformity with the designated FC2.0 boards that likely will use "TE0803-04-4AE11-A". The "TE0803-04-2AE11-A" could be an option for experiments with a high cost sensitivity and has not been investigated yet. The "TE0720-03-1CF" was originally intended for ultra-low-cost options, but a future design implementation is due to the changed pricing policy of Xilinx unlikely.

Trenz Module	**Bases on**	**Speed-grade**	**FPGA-RAM**	**Pins**		Price **(100pieces)**
				LVDS	MGT	Jul. 2020
TE0720-03-1CF / TE0720-03-61C33FA	XA7Z020-1CLG484Q	1	4.9Mb	200	-	182,07
TE0715-04-30-1C	XC7Z030-1SBG485C	1	9.3Mb	150	4 (GTX)	315,95
TE0803-04-2AE11-A	XCZU2CG-1SFVC784E	1	5.3Mb	156	4 (PS-GTR)	222,23
TE0803-04-4AE11-A	XCZU4CG-1SFVC784E	1	18.0Mb	156	4 (PS-GTR) + 4(GTH)	400,73
TE0808-05-9BE21-AK	XCZU9EG-1FFVC900E	1	32.1Mb	156	4 (PS-GTR) + 16 (GTH)	1.047,80

The experimental setups in the “INL determination” part of the thesis uses as FADC card the Ltc2107 evaluation board “DC2266A” (Analog Devices, Wilmington-Massachusetts). The 16bit ADC Ltc2107 can sample at up to 210 MHz and uses the DDR LVDS technique to serialize its digital output data. This means that the single channel board “DC2266A” requires in total requires 8 LVDS pin-pairs (16LVDS pins). The design therefore can be implemented in all SoC module nominees. In the final FC2.0 Germanium DAQ it was originally intended to operate 8 or more channels per device. This would require 128 available LVDS pins, if the ADC Ltc2107 is used. As in the Trenz Modules the interconnection of the LVDS is not freely selectable (in the FC2.0 prototype only LVDS pins that are routed to the FMC plug (on the module carrier board) of the used carrier board can be used to receive the ADCs output streams) and the LVDS pins are also required for other tasks (clock input, trigger links, …) the implementation of such a DAQ in Trenz Modules is challenging and not necessarily possible with every Trenz module. Note that the majority of the available ADCs uses enhanced serial data transmission techniques that are capable to accommodate more than two bits per lane and therefore in general the interconnection with 8 (or much more) ADCs is a feasible aim. An instance FADC card that was successfully implemented and uses such a denser transmission approach is the “fmcadc100m14b4cha” card (CERN, Geneva-Switzerland) (depicted FADC card in Figure 33).

To find the best Trenz SoC module for HPGe applications a more elaborated requirement specification is required. Before in future chapters a more versatile design is discussed let us consider a simple single channel DAQ for HPGe-spectroscopy. For default HPGe-spectroscopy a 16bit ADC with a sample rate of 50 MHZ is sufficient. This is important as the sample rate massively influences the required buffer capacity. The same FPGA RAM can buffer at a sample rate of 50 MHz a time range that is 4 times higher as a DAQ system with the same FPGA RAM size that uses a sample rate of 200 MHz. As in the experimental measurement campaign of this thesis the elaboration of the ADC and the determination of the possible maximum resolution is the main investigation object, the down-sampling options of the FC2.0 prototype firmware are not used. Thus, the corresponding measurements have a sample rate of 200 MHz. In FC1.0-Germanium with its sample rate of 62.5 MHz every ADC Ring Buffer can buffer up to 32768 samples. In case of the FC2.0 prototype adaption this means that an ADC Ring Buffer should be capable to buffer up to 131072 16bit samples (as $F_{s_{FC2.0}} \approx 4 \cdot F_{s_{FC1.0-Germanium}}$.). 131072 samples need 2 Mb FPGA-RAM. According to Table 9 all listed SoC devices have more FPGA-RAM and hence all listed devices could be used. The delay pipeline and the Event Ring Buffers are the remaining major FPGA-RAM consumer besides the ADC Ring Buffer. The size of the delay pipeline buffer must be sufficient to fulfil the analysis requirements (discussed in appendix chapter A.18). In the performance investigations measurements, a lot of required information was missing and therefore it was decided that the delay line must span at least a time range of ~ 40 µs. For this time span the delay line requires 64 kb FPGA-RAM. The event ring buffers

FPGA RAM consumption depends on the event information number, the chosen trigger domain coherency and the chosen buffer size. In the performance investigation setup only a single channel trigger domain is possible and as event information only the precise pulse detection time (2×32 bit) and the baseline level (1×32 bit) is required. A feasible event ring buffer in FC2.0 prototype applications can buffer up to 512 events. Such event ring buffer would require 48 kb FPGA-RAM. In total the suggested 1channel HPGe-detector setup of the general FC2.0 prototype firmware would only require 2.112 Mb FPGA-RAM and therefore all Trenz Modules could be implemented in all listed Zynq devices. The suggested buffer sizes of this section thus even could be described as "dissipative" as it does not use the majority of the SoC devices internal FPGA-RAM. This is painful as the FPGA-RAM is one of the SoC devices cost driver and unused FPGA-RAM is consequently a waste of money. A practical design therefore would extend the buffer sizes accordingly.

The developed "FC2.0 prototype" design must also consider that the available FPGA-RAM type depends on the SoC device. This is important as the "implementation efficiency" of its firmware code structure depends on the FPGA-RAM type. In the initial discussion of the FPGA-RAM consumption of the suggested ADC performance investigation design it was assumed that Vivado can transfer the code efficiently in a FPGA design that uses the internal FPGA-RAM blocks without any loss. The FC2.0 prototype firmware code cannot be implemented without any loss. Reducing loss is not trivial, as the available FPGA-RAM types strongly differ in their data width. For instance, the current ADC Ring Buffer firmware code implemented in UltraRAM only has a maximum efficiency of 88,9%[4]. For the instance values of our TUM setup this means that the ADC Ring Buffer would require at least 2.25 Mb UltraRAM (instead of 2 Mb).

All listed Trenz Modules could comply the needs of the "Single Channel HPGe-detector" FC2.0 prototype setup. In a real device selection process the developer therefore would use the cheapest Trenz Module ("TE0720-03-1CF") out of Table 9. In fact, device costs were the main factor in the preselection process of Table 9. Table 10 lists an elaboration of the Trenz Modules in focus of same specific function costs.

[4] Hint: The maximum efficiency indication relates to the default "16bit" signal width mode. All published results in this thesis are operated in this mode. The also available "18bit" signal mode, that can be feasible in down-sampling cases or in special debugging/verification procedures, has a maximum efficiency of 100%.

Table 10 Estimated costs in dependence on the design. Consider that the calculations are more a rough estimation than precise analysis. Especially the calculation of the total IO bandwidth was strongly simplified and is partly misleading. Despite that, the IO bandwidth values show a trend which is useful for feasible assumptions. In FC1.0-Germanium a default ADC Ring Buffer had a size of 32768 samples. Scrutinize that the feasibility of the Buffer size strongly depends on the sample rate of the DAQ. Therefore these estimations assumes a sample rate which is identical to FC1.0-Germanium (Fs=62.5 MHz).

Trenz Module	Ltc2107 channels maximum	Channel Price in €	Total IO Bandwidth in GHz	GHz IO Bandwidth Price in €	FC1.0-Ge Buffer Equivalent	FF1.0-Ge Buffer Price in €
TE0720-03-1CF / TE0720-03-61C33FA	12	15,17	200	0,91	9	20,23
TE0715-04-30-1C	9	35,11	190	1,66	18	17,55
TE0803-04-2AE11-A	9	24,69	156	1,42	10	22,22
TE0803-04-4AE11-A	9	44,53	260	1,54	36	11,13
TE0808-05-9BE21-AK	9	116,42	572	1,83	64	16,37

The pin consumption of the (16 bit) Ltc2107 is due to its DDR LVDS serialization quite high. ADC manufactures implement in their High Speed ADCs commonly stronger serialized output procedures. For instance, the ADC Ltc2174 (implemented in the already mentioned “FmcAdc100M14b4cha” FADC card) can output the data stream in “2-Lane 16 bit serialization” mode and hence only needs 4 LVDS pins per ADC channel. This increases the possible ADC channel number (per device) dramatically. Another popular

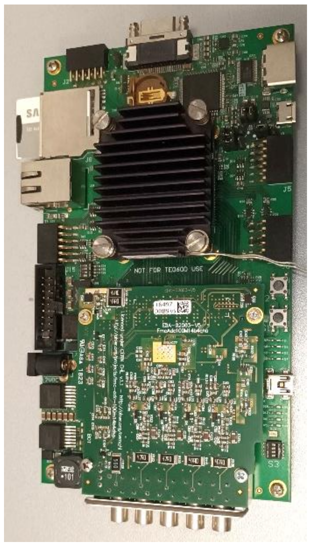


Figure 33 (Superseded) first FC2.0 prototype setup (with "FmcAdc100M14b4cha" as FADC card). The first setup could use the Trenz Modules "TE0720-03-1CF"/ "TE0715-04-30-1C". The Trenz Module is covered by its black heat sink.

procedure is JESD204. As the latter procedure requires MGT pins its usability strongly depends on the Zynq device. Since the intention of the prototype development was to check and verify complex design ideas, using the cheap, but performance limited Trenz module "TE0720-03-1CF" was not really an option for the final version of "FC2.0 prototype". The first measurements (with an utterly superseded firmware version) were done with a combination of the Trenz Modul "TE0715-04-30-1C" and its recommended Carrier Board "TEF1002" (both also from Trenz Elektronik) (Figure 33). After first performance checks and an aggressively changed pricing policies (by Xilinx) the switch to the Trenz Module "TE0803-04-4AE11-A" and the corresponding carrier board "TEBF0808" was decided (Figure 35). Currently the "TE0803-04-4AE11-A" is the only Module that can be operated by the latest FC2.0 prototype firmware/software stack in a real experimental setting[5]. It is also the setup that was used for a HPGe-detector measurement in the TUM physic facility (chapters 8.6-8.7). The modules and their implementation in the final housing of the FC2.0 prototype is shown in Figure 34.

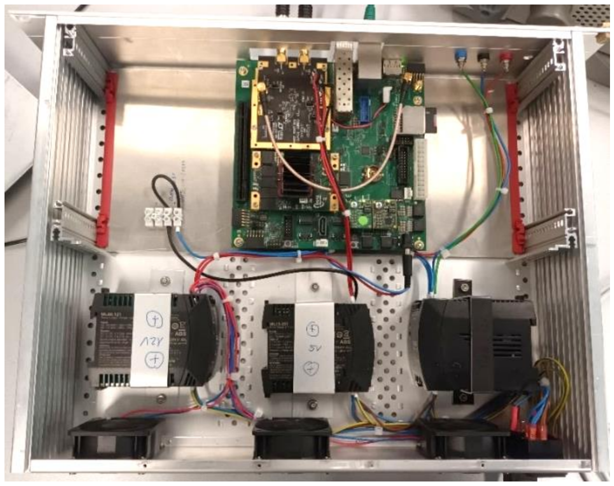


Figure 34 Equipped carrier board in final housing with its power supplies

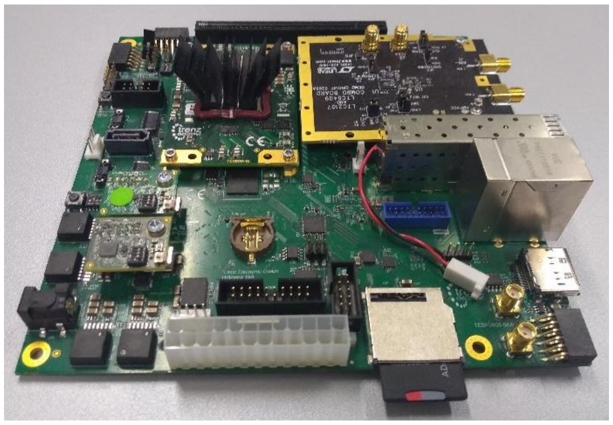


Figure 35Carrier board "TEBF0808" equipped with the Trenz Module TE0803-04-4AE11-A and the FADC card "DC2266A" (with ADC Ltc2017). The heat sink covers the ZU+ device (on the Trenz Module). The front side of the also implemented RJ45 plug is not visible in this figure.

[5] Exclusive ADC dummy operation – like in chapter 6.5.3 – should be possible with all listed Zynq-UltraScale+ modules after updating the Pin mapping of the clock IO Pins. The "ADC Read In" Block implementation in a Trenz Module "TE0808-05-9BE21-AK"/" TE0808-05-BBE21-AK" in principle only requires an update of the Pin mapping.

6.2 Setup Zynq-APU performance evaluation

The performance of the event building routine depends on the ethernet performance, the block (“FPGA fabric”) DMA performance, the bandwidth between CPU and memory system and the CPUs arithmetic/logic performance. Various CPU benchmarks exist, the most famous try to determine the most general CPU performance. The performance of a CPU depends strongly on the routine’s procedure and the compiler performance. The demands of pulse spectroscopy just require a special subset of the ALU capabilities and therefore a general CPU benchmark may be misleading. For this work the “speed2” benchmark (Developer: Thomas Kihm, MIZZI Computer Software (Mannheim-Germany)) is used for the first evaluation of a Server-CPU. “speed2” determines the bandwidth for various transfer sizes in dependence on the used data type. A read transfer in speed2 is data movement from the memory system to the CPU, a write transfer is a data movement from the CPU to the memory system. The selected transfer size determines the source/destination memory (Read/Write) in the memory system. The transfer is done via an unrolled for-loop, without memcpy() implementation. The low-level analysis tasks in pulse spectroscopy frequently use mean filters or recursive low/high pass DSP filters (with multiply-accumulate operations). The “speed2” routine’s arithmetic elaboration tries to imitate the common processing operation weights of pulse spectroscopy for various data types (char, int, float). The performance results of the “speed2” routine are still dependent on the optimization settings of the used g++/gcc compiler. Like in all synthesis, the g++ optimization level “O2” was selected. In the setup the speed2 routine is running as exclusive foreground process. No further (user started background) process is running on the machine.

6.3 Results Zynq-APU performance evaluation

The determined double bandwidth results of the speed2 routine are depicted in Figure 36.

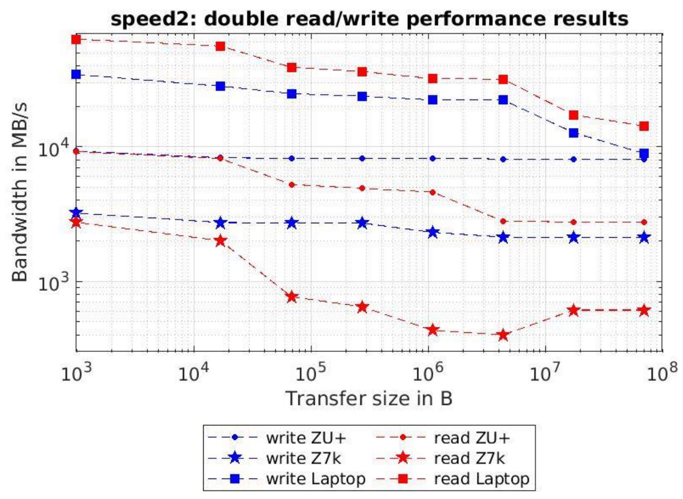


Figure 36 speed2: double read/write performance results. The speed routine does a bandwidth determination with several data types. In this discussion the "double" results are highlighted, as the remaining types (char, short int, int) have less than 64bits. A data type with less than 64bit is possibly not invoked efficiently by an ARMv8 CPU. The data width is also the explanation of the "Laptop" anomaly: The Laptop's write operations are slower than its read operations. In all Zynq devices its reverse. In contrast to the Zynq devices, the laptop's data bus between the CPU and memory has a data width of 128bit. A double read operation therefore gets two double values in a single invocation. In a double write operation still only a single double value on the memory can be set.

The speed2 routine can only indirectly influence which memory of the memory system it elaborates. Small transfer sizes just require the L1-cache (32kB size in Zynq devices). Greater transfer sizes that exceed the L2-cache size (512kB in Zynq-7k/1MB in Zynq-Ultra+) require the main memory. With increasing transfer size, the utilization of the main memory increases. Transfers that comprise at least the tenfold of the cache size enable a good approximation of the bandwidth between CPU and main memory. The read and write bandwidth trends of the Zynq-UltraScale and the used Laptop[6] ("Read-out server") are without any peculiarity. The write bandwidth of the Zynq-UltraScale slowly decreases with transfer size. In write mode the APU needs mostly not wait for the transaction response before it transmits the next store command. After the transaction of a load command, the APU must wait for the response ("desired data element"). The response time decreases with the memory hierarchy (L1-Cache has highest hierarchy). The Zynq-7k is a 32bit architecture that (very roughly) is operated with (more than) the half clock frequency. In both series the cache controllers have prefetch capabilities ((Xilinx, 2017, S. 101) / (ARM, 2018, S. 337)) and a Cache/DDR controller with a write-combine feature. A (very) rough performance estimate therefore would obtain at least a quarter of the Zynq-Ultra+ performance. The measured speed2 trends indicate a much worse

[6] The laptop has an Intel (Quad) Core i7-7820HQ CPU with a processor base frequency of 2.9GHz (max turbo frequency of 3.9GHz) (launch date Q1 17)

performance. In fact, its lowest read performance is close to 200MB/s and therefore close to the designated ethernet bandwidth (125MB/s). This is a serious bottleneck in event building. Building up a client routine that uses the ethernet efficiently will be challenging. The arithmetic elaboration results of the speed2 routine are summarized in Table 11.

Table 11 Arithmetic performance according to speed2. Speed2 is a single core thread that does not use AVX/NEON instructions. Recap that the listed values are from Zynq devices with a speed grade of -1. In case of the Z7k this means a CPU base frequency of 667MHz, in case of the ZU+ this means a CPU base frequency of 1.2Gz (Table 2). Other speed grades might provide a better performance. The laptop has an Intel (Quad) Core i7-7820HQ CPU with a processor base frequency of 2.9GHz (max turbo frequency of 3.9GHz) (launch date Q1 17). The used Raspberry Pi 4 has a quad core ARM Cortex A-72 CPU with a processor frequency of 1.5GHz. The ACAP devices of Xilinx use a dual core ARM Cortex A-72 CPU with a processor frequency of 2.0GHz.

MIZZI MOPS type	**Zynq-7k**	**Zynq-Ultra+**	**Raspberry Pi4**	**Laptop**
int	398.972	1331.987	1517.971	2177.029
short int	372.018	1310.323	1508.977	2296.541
double	117.527	205.643	412.565	1203.165
float	105.448	186.678	439.380	1084.195

The speed2 double arithmetic performance of the Zynq devices is significantly lower than the performance of Laptop and Raspberry Pi4. In a big.LITTLE configuration out of A-72/A-53 ARM Cortex CPUs the Raspberries A-72 represents the high performing CPU and the A-53 the energy economical one. This explains the performance margin of the Raspberry Pi4. Note that speed 2 does not determine the possible maximum arithmetic performance. The possible maximum performance may be much higher, but this implicitly requires the usage of NEON instructions and efficient parallel core processing. This is not in all applications possible. [19] and especially [20] evaluate the possible maximum arithmetic performance. In HPGe spectroscopy double is the preferred data type. NEON uses 128bit register, in the best case it therefore can operate two double entries in parallel. In appendix chapter A.10 a rough bandwidth estimation of a simple analysis chain with the results of Table 11 was tried. Such a procedure would enable an on-board γ-ray spectrum determination and hence the read-out server needs not to do the corresponding analysis. Even more important would be the possible reduction of the ethernet traffic, as an additional software signal chain of sufficient resolution could be used to “reject” the ethernet transfer of less promising events. The signal chain of the appendix chapter only uses basic shaping operations and hence represent an essential shaping strategy that is not sufficient to achieve the best possible resolution. It therefore is not eligible for the improved “ethernet transfer rejection” of traces. Strongly simplified the results of the appendix chapter say that in general cases the possible ethernet spare gain is not worth the much higher costs of the shaping. In some cases, with a sufficient low event rate, it still might be a feasible solution. Multithreading, more aggressive vectorizing (like 512bit -SVE2) and using the next ARM CPU generation could change the situation.

6.4 Setup for performance evaluation of ACP-CMA

The designed FC 2.0 ACP-CMA framework was designed for both Zynq device series, therefore the implementation was a tradeoff that had to consider the significant architecture differences between the Zynq devices. The performance elaboration uses besides the default ACP-CMA framework (firmware + user space device drivers) a "dummy" extension that enables a performance elaboration of the ACP-CMA. The analysis routine that determines the bare ACP-CMA performance setup is included in the default FC2.0 device driver library and was originally developed by Thomas Kihm too. In the scope of this thesis not all elaboration points are required. The used routine is a derivation of Kihm's bare performance routine. The "unnecessary" performance measurement points were removed, and the measurement time was adjusted on the same conditions like in the coming HP-DMA performance elaboration routine. A condensed description of the setup is given in chapter 6.4.1.

6.4.1 Bare ACP-CMA performance setup

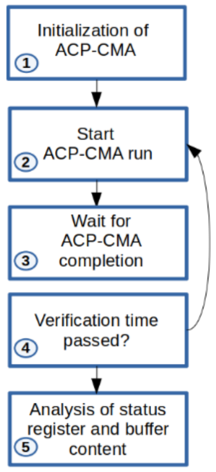

Initialization of ACP-CMA
1
Start ACP-CMA run
2
Wait for ACP-CMA completion
3
Verification time passed?
4
Analysis of status register and buffer content
5

Figure 37

The setup is rather simple (Figure 37). Besides the ACP-CMA framework, merely the µs counter of the CPU firmware block is required. In the initialization position ("1" in Figure 37) the ACP-CMA, the time management and round count variables are allocated. In the second step a ACP-CMA transfer of a selected length is started by the CpuCMARead() function of the ACP-CMA framework. The elaboration routine immediately invokes the CpuCMAWait() function ("3"). As soon as the completion of the transfer is detected, the wait function returns, and the routine internal "round (counter)" variable is getting incremented. Afterwards the routine checks the current time status. If the demanded measurement time is passed, the routine outputs the round counter value. Otherwise, the routine starts another ACP-CMA transfer round by jumping into position 2. As the measurement time is known, the ACP-CMA performance can be obtained by the round counter value and the selected transfer size (equation (2)).

$$BW_{ACP-CMA} = \frac{round \cdot len \cdot 4B}{t_{meas}} \tag{2}$$

The 4B factor in equation (2) is motivated by the 32bit data width of the ACP-DMA in the FC2.0 prototype design. In principle in Zynq-7000 devices enable a 64bit ACP port implementation, in the Zynq-UltraScale+ devices even a 128bit wide ACP port is possible. The actual ACP-CMA performance is dependent on the interconnect style (between ACP-CMA and port) and of the asserted AXI transfer/transaction signals (especially AXCACHE, AXLEN, AXSIZE and WSTRB). Thomas Kihm's ACP-CMA implementation is fixed to a transaction size of 64B and does not permit a modification of the AXI control signals, neither it permits a change of the interconnect implementation. In order to verify the parameter selection of the current ACP-CMA implementation modified ACP-CMA designs were also developed and elaborated by me.

6.4.2 Setup for ACP-CMA event building performance evaluation

In the event building performance evaluation setup not all implemented blocks of the firmware are used (Figure 38). The pulse detection block is not required as in dummy mode the pulse detection triggers also can be provided by the ADC dummy block unit. Using the ADC-Dummy Block as signal source instead of a real HPGe-detector signal source offers several advantages. For instance, such elaborations are not limited by the constraints of a real FADC card. The available FADC-Card “DC2266A” only offers one channel and is therefore not eligible to check multi-channel designs. The event rate of a real setup is also harder to control. Another advantage is the possibility to circumvent the trigger limitations of the Pulse detection firmware block. The current design offers three ACP-CMA event building styles (“via GP/LPD”,”headeracp” and “headerintrace”). Of these styles the performance of the “headeracp” mode are presented in this chapter as it considers all limitations that are required in high resolution HPGe-spectroscopy.

As the final performance of a FC DAQ also depends on the ethernet network setup and server performance, achieving the total event building maximum performance of the Zynq device was never the main intention of the development process. The current read-out routine on the server lacks properties that are required to optimize the ethernet traffic in a multi DAQ board environment and furthermore lacks fast message verification skills. The Read-Out server routine already includes payload checks and the permanent hard-drive storage of the prompted payload (at bottom the trace and trace header data). A severe disadvantage of the setup is the used laptop whose network interface and CPU IO-bandwidth are not able to compete with the performance of a real server. As this influences the ethernet message transfer it also influences the performance of the event building routines.

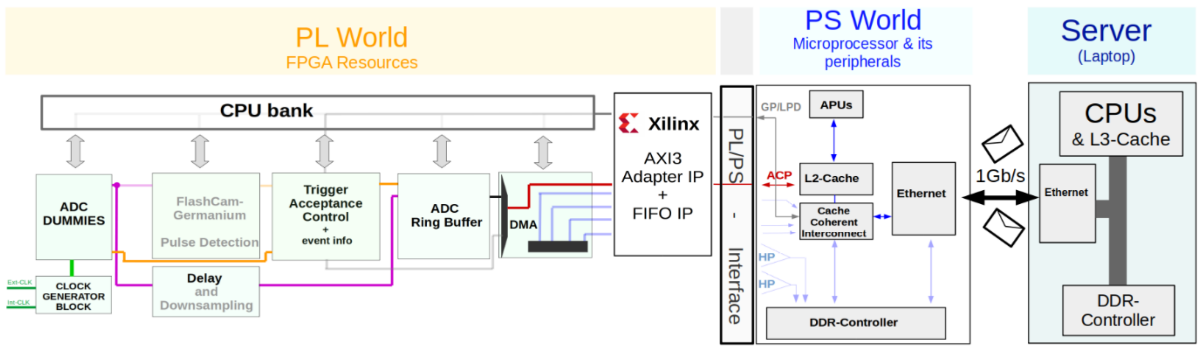


Figure 38 Firmware blocks & read-out server in event building performance investigation setup. Though the GP/LP-IO unit is implemented in the figure, it was not used for trace header transfer in the investigated setup. In the used “headeracp” mode all event data (trace + trace header) passes the ACP port. The trace header data passes the GP/LPD port in the not investigated “via GP/LPD” mode.

6.5 Results for performance evaluation of ACP-CMA

The Zynq PS/PL port performance elaboration is a focus of various publications ([19], [21], [22], [23], [24], [25]). All publications have in common that their setup tries to determine the maximum bandwidth of the port. The port bandwidth is just one parameter that determines the total bandwidth of a DAQ. In DAQs the event building bandwidth depends strongly on the CPU access bandwidth. In the listed publications an event building is mostly not a concern. Their suggested "optimal" procedure (to gain their maximum performance) therefore must not consider the event building routines CPU access bandwidth to the port data. For (most of) them a starving CPU is still fine. An exception of this rule is [19] and [23], but still some important aspects are omitted. In most real applications, also in the FC2.0 prototype, the remaining CPU access bandwidth is decisive. If the remaining CPU access bandwidth is not sufficient the port cannot be used – even if its maximum performance is superior compared to other ports. The listed publications are also focused on a single Zynq Series (either Zynq-UltraScale+ or Zynq-7000). Not all setup optimizations are transferable between the Zynq series. The FC2.0 DMAs must be applicable to all Zynq devices. The FC2.0 design just can yield performance improvement techniques when they are applicable in both devices.

6.5.1 Default ACP-CMA performance

In Figure 39 and Figure 40 the bare ACP-CMA performance elaboration results are summarized. The figures illustrate the effective write (into L2-Cache) performance of the ACP-CMA for different transfer sizes and does not correct the influence of the ACP-CMA header tasks time τ (time for setting the ACP-CMA transfer register and polling). The comparable speed2 caching trend is given by the read trend lines (in Figure 36). A performance elaboration requires a pertinent gauge. In the next chapter 6.5.2 the effects of ACP-CMA design modifications are compared with the results of other publications. In order to enable a fast implementation of the FC2.0 ACP-CMA for firmware designer newbies, the ACP-CMA core utilities are shielded by protective measures. The chosen AXI Interconnect interleaving and the chosen ACP-CMA data width are a key point in these protective measures. A resulting maximum performance loss is acceptable if the maximum performance is still much larger than the DAQ systems performance bottleneck. In case of the FC2.0 prototype this is the performance of the 1 Gb/s ethernet link. Equation (3) represents the expected maximum bandwidth of an ideal DMA after software overhead compensation.

$$BW_{Ideal\_Maximum} = f_{Op} \cdot w_{Data} \qquad (3)$$

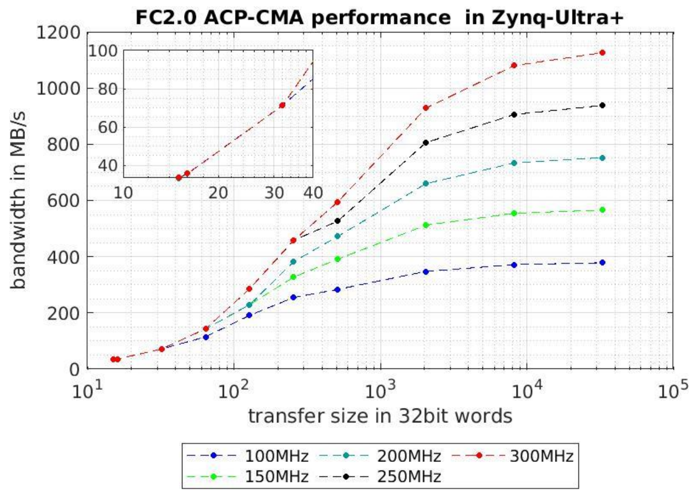


Figure 39 FC2.0 ACP-CMA performance in Zynq-Ultra+. The possible bandwidth of the ACP-CMA depends on its operation clock frequency (indicated in the legend). Other authors also compare the ACP port with the GP/LPD port performance for little transaction sizes (mostly less than 64 words). Similar measurements were done in the pre-evaluation of this thesis. The results are published in appendix chapter A.14. In case of the ZU+ the LPD port has a read performance of 20.3 MB/s. For a small transfer size the LPD-IO is hence more performant than the ACP-CMA. This is another reason that advantaged the "endevtno" prompting via the GP port.

The ACP-CMA effective performance trend in the ZU+ device is ensuing the expected trend (Figure 39). Like predicted by equation (3), the maximum performance is always slightly less than $\sim 4B \cdot f_{ACP\_Clk}$. In case of the 200 MHz default ACP-CMA design this means a maximum bandwidth of 752 MB/s. The strong variation of the effective bandwidth is mainly caused by the ACP-CMA header jobs. A correction of the header job influence would harmonize the performance results. As the header jobs are always required their influence compensation is not legit. A performance investigation of the LPD-IO will show that it offers a read performance of 20.3 MB/s (appendix chapter A.14). A firmware design therefore should offer an efficient technique to also read-out the event-information ring buffers ("trace header information") via the ACP-CMA and not only the trace data (out of the ADC Ring buffers). The solutions name in this thesis is "headeracp" mode (chapter 4.2.4). Note, that in the ACP port a simple transfer size increasing does not guarantee an increase in the effective bandwidth. At some point the cache cannot buffer all data of the transfer anymore and performance deterioration through cache trashing will occur. [24] determine this thrashing threshold at a transfer size of approximately 256 kB (64kWords). As the FC2.0 ACP-CMA also has to consider Zynq-7k devices its maximum ACP-CMA transfer size is limited to 128 kB.

The ACP-CMA trends (Figure 40) in case of the Zynq-7k devices is in accordance with the expectations. The official ACP port specifications for the Zynq-7k series state an IF_ACP_Clk frequency of 150 MHz and a maximum write bandwidth of 1.2 GB/s [14 p. 659][7]. As in the FC2.0 prototype the ACP-CMA data width is 32bit, but not 64bit like the ACP port width, a rough ideal write maximum bandwidth of 600 MB/s

[7] Xilinx does not give any explanation for the limitation of the operation frequency to 150MHz. In fact, this is surprising as the complete PS part of the ACP-L2-SCU-APU chain is operated with the APU base frequency (667MHz). In case of the ZU+ devices an IF_ACP_Clk equivalent specification (of Xilinx) could not be found.

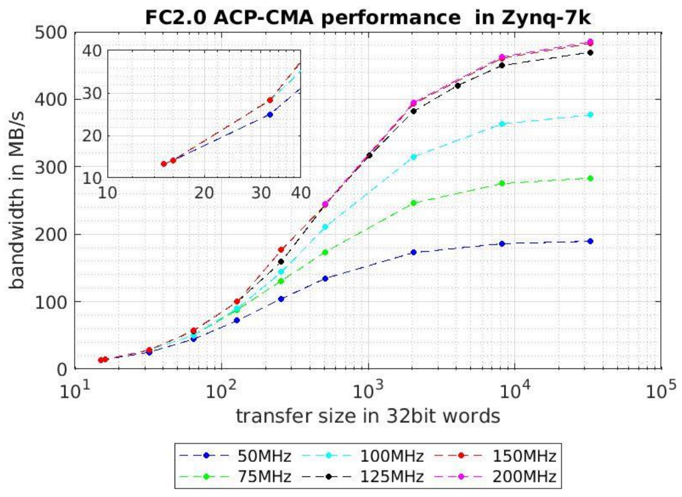


Figure 40 FC2.0 ACP-CMA performance in Zynq-7k. The legend indicates the used PL operation frequency of the ACP-CMA.

expected. In practice the performance is limited to ~470 MB/s (at an operation frequency of 150 MHz as consequence of IF_ACP_Clk). The unproportionally low bandwidth increase between the 125MHz and the 150MHz trend is a consequence of the chosen AXI interconnect implementation. The ACP-CMA is via the Xilinx AXI Interconnect IP connected to the ACP port. The implemented Interconnect adapts the 32 bit data AXI4 communication of the ACP-CMA in 64bit AXI3 traffic. The Interconnect-ACP Port link is only operated with 125MHz. In pre-investigations an implementation of a 150MHz clock could not achieve significant performance gains. As a 125MHz design is simpler to synthesize, it was finally prioritized.

6.5.2 Possible ACP-CMA variations and their performance

The actual ACP-CMA design in the FC2.0 prototype is motivated by feasible thoughts and limitations of the Zynq devices. For instance, it was the development aim to have the same ACP-CMA interface on both Zynq devices series. This procedure has many advantages. For instance, the data input width is equal in all designs and consequently just one ADC Ring Buffer design must be developed. This sounds trivial, but the inherited simplifications in the design process are far going and even affect central points of the trigger system. Unfortunately, unification also has its drawbacks. A unified interface prevents implementation of device specific assets and therefore can be inferior to an adjusted rival. As equation (4) already implies, the possible maximum bandwidth depends on the data width. The ZU+ devices have a wider data input width (128bit) than the Z7k devices (64bit). A device specific solution therefore could enable a higher maximum bandwidth. The publications listed in this chapter's introduction used device specific DMAs. Their determined maximum write performance is listed in Table 12.

Table 12 Gauged PS/PL port performance of various publications. Note that the Zynq devices have several HP ports and therefore some of the publications also investigate the total performance of DMA designs that use more than a single HP port. As their maximum performance strongly depends on the chosen design the corresponding results are not listed in Table 12.

Publication	**Zynq-Device**	**Operation Frequency in MHz**	**Port**	
			ACP	(Single) HP
[19]	Z7k	125	833 MB/s [rw=1665.9 MB/s]	854 MB/s [rw=1708.5 MB/s]
[22]	Z7k (Z-7020)	100/110	751 MB/s	839 MB/s
[23]	Z7k	167	-	1243 MB/s
[24]	ZU+	300	~3.3 GB/s	~4.8 GB/s

All listed publications declare an ACP port bandwidth that is much faster than the determined FC2.0 prototype ACP-CMA bandwidth. This is partly explainable by the software overhead, that in some publications is counterbalanced. The results in Figure 39 and Figure 40 include the software overhead as in practice it is also not possible to bypass its time loss. A further point that slightly diminishes the determined performances, is the time loss due to the periodical time inquiry. In the suggested setup any kind of time loss affects an incorrect decrease of the determined ACP-CMA bandwidth. But in overall still a significant gap between the ACP-CMA and the publications exists.

The performance gap must be explained by differences in the firmware (for instance by differences in the chosen AXI signal properties) or in the software (different device driver design). A promising firmware "patch" in the Zynq-7k devices is the AXI transaction size and the data width of the ACP-CMA state machine. The remaining properties of the state machine stay unchanged. Figure 41 depicts bandwidth performance trends of such modified 64bit ACP-CMAs. In case of the Z7k devices burst sizes that are a

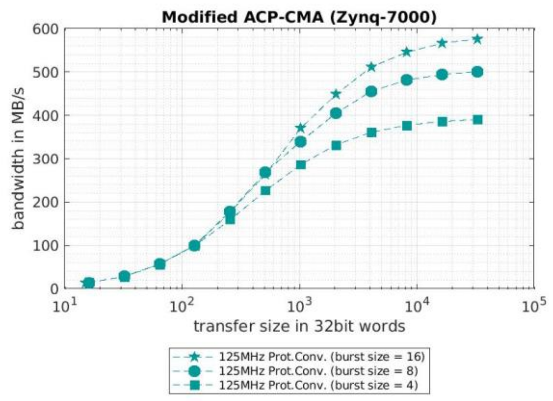


Figure 41 Zynq-7k ACP-CMA bandwidth after modification (I). The chosen default ACP-CMA implementation (used in chapter 6.5.1) uses the AXI Interconnect IP from Xilinx to tackle burdens that are resulting from the ACP-CMAs 32bit data width. AXI Interconnects are (sometimes) capable to intercept typical newbies errors of upstream DMA units and therefore act in FC2.0 as security measure. The burden compensation of the AXI interconnect implementation is not total, the performance is still degenerated. Hence it was decided to design a pure 64bit-ACP-CMA. This ACP-CMA is connected via a simple "Protocol Converter IP" (from Xilinx) to the ACP port. Furthermore for test tries the burst size (AXLEN) of the ACP-CMA state machine was varied.

multiple of the cache line size are permitted. A transaction of 4 64bit words has in total 32B and therefore represents the size of a cache line (square trend in Figure 41). Such an ACP-CMA state machine has the lowest performance. Especially note that the DMA state machines of Figure 41 use a protocol converter IP to interconnect directly with the ACP port. In the default FC2.0 prototype setup an AXI interconnect is between them. AXI interconnects significantly influence the transaction. The modifications of Figure 41 could increase the performance to 576.0 MB/s (instead of 469.7/485.0 MB/s (125 MHz/150 MHz)). The performance of such an ACP-CMA framework is still less than in the listed publications. The remaining performance gap is therefore likely a device driver issue. Several device driver implementation possibilities exist, and lack of time prevented their experimental implementation. But the used user space device driver approach via mmap is a well-known and still a common approach. Its main drawback is security and a breach of the Linux philosophy, but not

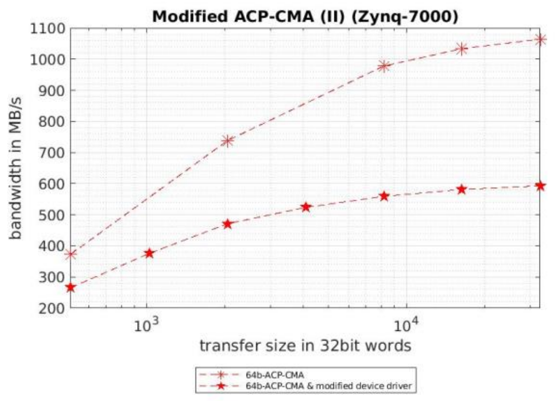


Figure 42 Zynq-7k ACP-CMA bandwidth after modification (II). In this setup the fastest firmware design out of Figure 41 is used for both measurements. The measurements only differ in the used device drivers polling method. The polling mechanism of the default ACP-CMA device driver uses more prompts than Figure 21 implies. As these additional prompts are not necessary, they are omitted without substitute in the modified device driver source. The additional polling was done in a way that no effects were expected.

performance analysis will lead the developer to the conclusion that the chosen polling approach in the ACP-CMA has unanticipated drawbacks. The performance of a modified ACP-CMA framework is illustrated in Figure 42. Now the maximum ACP-CMA performance is 1063.7 MB/s (instead of 592.9 MB/s (at an operation frequency of 150 MHz)). The ACP-CMA optimization in Zynq-UltraScale devices requires reverse engineering scrutinies as the official ARM documentation only guarantees proper function of transaction sizes of exactly one cache line size (64 B) and misses a lot of desired information. In case of the ZU+ devices documentation recommends a data width of 128 bit. The performance results of such a modified ACP-CMA state machines that are directly connected to the ACP ports of a ZU+ device are shown in Figure 43. Dependent on the device driver setup (polling!) the

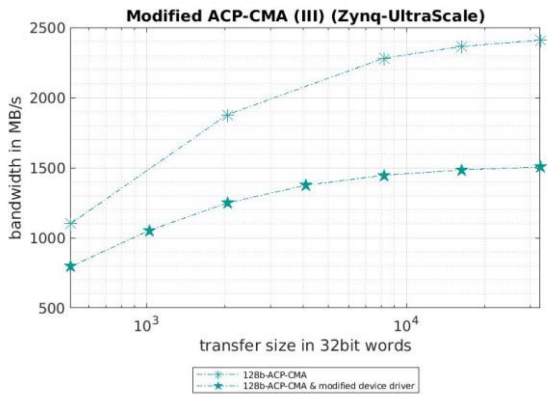


Figure 43 Zynq-Ultra+ ACP-CMA bandwidth after modification (III)

maximum performance is up to 2.41 GB/s (at an operation frequency of 200 MHz). The maximum performance of the modified ACP-CMA in ZU+ devices is still less than the given maximum performance of [24]. [24] published results whose software overhead influences were counterbalanced and does not explain its polling schema. In this chapter it was shown that modifications of the developed ACP-CMA framework easily can boost its performance. A corresponding survey for a “practical” setup will be done in chapter A.15. As the modifications would also require extensive modifications in the ADC Ring Buffers and in the Trigger System (the latter changes are expected to be more strenuous), they were never implemented in the final FC2.0 prototype.

6.5.3 ACP-CMA event building performance results

In the previous chapters the bare ACP-CMA performance was investigated. In the corresponding setups the ACP-CMA had no signification interaction with the CPU. Neither the CPU nor the Ethernet Controller tried to retrieve the ACP-CMA data. The ACP-CMA never had to share the resources of the cache system. In a real Trace-Read-Out application the interaction with concurrently executed tasks (like "event building) is decisive for the total performance of the read-out routine. The ACP-CMA performance embedded in an event building routine is especially important for final firmware design of the desired FC2.0 DAQ. For instance, the event building performance limits the possible number of channels and also decides the required buffer size that is necessary to guarantee an assigned mean dead time limit. Its influence on the total firmware design is therefore compelling.

The ACP-CMA event building performance setup, whose performance is illustrated in Figure 44 and Figure 45, was introduced in chapter 6.4.2. It is a FC2.0 prototype design that has 4 ADC dummy channels. The corresponding measurements were done with realistic settings which an included hard-drive storage of the transmitted event data. The shown performances are maximum performance, this means that for their achievement a significant dead-time fraction due to completely loaded buffers was accepted.

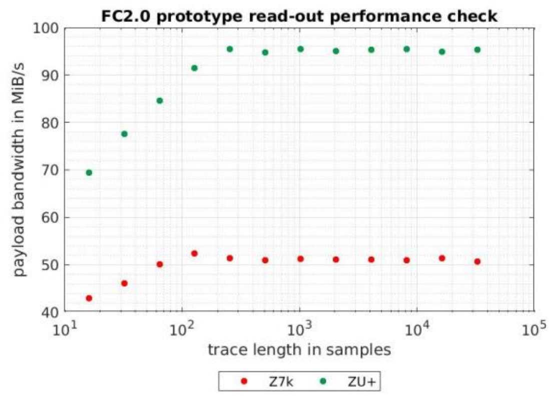


Figure 44 Payload bandwidth of the FC2.0 prototype (using "headeracp" option and "Global Multi Channel" Mode with 4 ADC dummy channels). The measurements used realistic settings and the results are therefore adaptable to a long-term measurement with real detectors. To achieve the illustrated values a significant dead time fraction was accepted. Measurements that must prevent any deadtime only can provide lower bandwidths. The ZU+ measurements were done with the Trenz Module TE0803-04-4AE11-A, the Z7k measurements were done with the Trenz Module TE0715-04-30-1C.

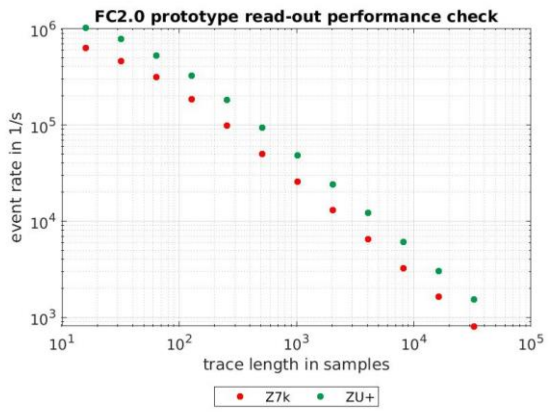


Figure 45 Event rate maximum of the FC2.0 prototype (using "headeracp" option and "Global Multi Channel" Mode with 4 ADC dummy channels). The measurements used realistic settings and the results are therefore adaptable to a real experiment with real detectors (for instance the TUM HPGe-detector measurements in chapter 8.6). To achieve the illustrated values a significant dead time fraction was accepted. Measurements that must prevent any deadtime only can provide lower event rates.

Figure 44 shows the superiority of the "ZU+" version of the FC2.0 prototype DAQ. The Z7k/ZU+ firmware designs are very similar, they mostly differ in their operation frequency and event buffer capacity. The latter is a consequence of the differences in their available FPGA-RAM resources. The ZU+ performance trends were obtained by an ACP-CMA operated by 200 MHz, the Z7k performance trends used an ACP-CMA that was operated by 125 MHz. The poor read-out performance is mainly caused by the used PS 1 GB/s ethernet option. In case of the Z7k devices several investigations confirmed the poor PS ethernet results (appendix chapters A.11 - A.12). The given maximum performance is likely close to the real maximum performance of the Z7k PS ethernet controller. In case of the ZU+ the derivations from the theoretical value are likely partly caused by unfavorable ethernet transfers during the read-out process (appendix chapter A.15). A more enhanced procedure therefore possibly could increase its performance. The Zynq devices also have much more powerful ethernet capabilities, but in the prototype mainly two points advocated the "PG 1 Gb/s Ethernet" solution. First of all, the required device drivers were already implemented in the used Linux kernel. Secondly the RJ45 plug of the used carrier boards is only connected to the PS ethernet pins of the Zynq devices. Though both issues in principle could be circumvented, it was decided to use only the PS 1 Gb/s ethernet in the prototype development. Especially in the case of the Z7k devices with multigigabit transceivers a reevaluation of the "PS 1 Gb/s ethernet fixation decision" could be done. Its PL ethernet opportunities likely can abrogate its poor "PS 1 G/bs ethernet" performance.

In the prototype the poor PS ethernet prevented the determination of the maximum event building performance of the Zynq devices. In order to extrapolate the possible maximum performance, further investigations were necessary. The ACP-CMA is an asynchronous DMA framework. This means that the CPU returns immediately after the ACP-CMA transfer started and hence the CPU can, with some restrictions, concurrently do the event building procedure. The event building procedure (building up the read-out server demanded ethernet buffer structure of Figure 29) mainly consists out of fast cache operations (event data movement via memcpy() invocation and messen_info structure update jobs). Only the prompting of the actual "endevtno" value (of the messen_info structure) requires a slow GP/LPD-IO unit transfer and is therefore not a fast operation in the cache. In order to determine the number of this jobs which can be done concurrently to an ACP-CMA transfer an investigation method was developed.

The method exploits the "Bare ACP-CMA performance" setup of chapter 6.4.1. In chapter 6.4.1 the ACP-CMA performance was inferred on the determined round time of a setup cycle (steps 2- 4 in Figure 37). If a within an ACP-CMA concurrently active operation increases the in chapter 6.4.1 determined ACP-CMA round time, we can deduce that the active operation needed more time than the ACP-CMA transfer. Consequently, we also know that the operation achieved its maximum workload that it can finish during a concurrently executing ACP-CMA. The procedure of the new method is for several reasons not totally perfect, but it enables a feasible estimation. Its results are shown in Figure 46 and Figure 47. Figure 46 shows the possible number of subsequent memcpy() (data transfer) operations that are possible in a single equal sized ACP-CMA transfer. Figure 47 depicts "endevtno" equivalent results.

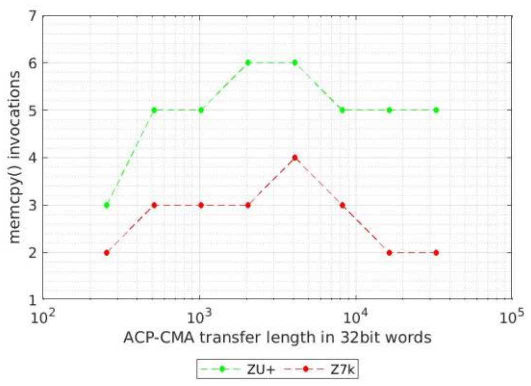


Figure 46 Possible number of memcpy() invocations during an ACP-CMA transfer of same size

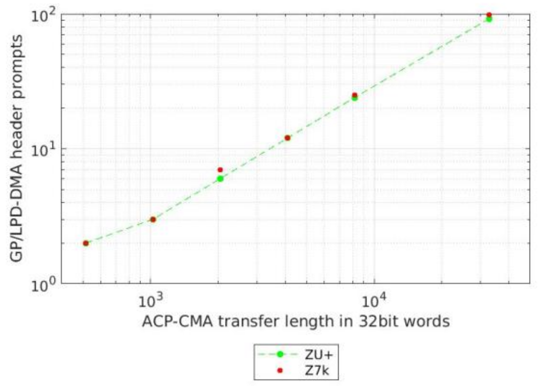


Figure 47Possible trace header invocations via the GP/LPD-IO during an ACP-CMA transfer. The efforts to prompt the trace header and the "endevtno" status value via the GP/LPD-IO are not totally equivalent. In the focus of the trend discussion of this section a conflation of both efforts is a good approximation and therefore constructive.

The maximum bandwidth (752 MB/s in the 200 MHz ZU+ design (Figure 39)) can be achieved when the event building routine can continuously prompt the largest ACP-CMA transfer size. Figure 46 and Figure 47 show that for such transfers the concurrent build up of the reply message in the ethernet buffer is possible. In principle an ACP-CMA that is operated at 200 MHz therefore could provide a 752 MB/s event data stream to its ethernet unit. Adopted to the setup of Figure 44 these means that the ZU+ design easily should exceed the 100 MB/s threshold and has a realistic bandwidth target of 120 MB/s (instead of the achieved 95 MB/s). This kind of extrapolation is critical as it does not consider all real-life restrictions. Typical real life design points that can cause a significant (negative) derivation from the extrapolation result are

- ➢ Read out server and the Zynq devices have an upper ethernet packet performance limit that cannot be exceeded. Simply increasing the ethernet packet ("command messages") number (per time) to achieve the ethernets link maximum bandwidth is therefore not a feasible solution. The transmitted ethernet packets must use the optimal ethernet packet size to achieve the best ethernet performance.
- ➢ The best ethernet packet size depends on many parameters. An instance parameter is the buffer size of the ethernet controller of the receiver. Obviously, the used setup (switches, …) of the ethernet network also influences the optimal ethernet packet size and even the DAQ settings (for instance the trace length) can influence it. A smart read-out server routine is capable to determine the optimal ethernet packet size.
- ➢ The effective number of event data that can be buffered in the DAQ boards. The possible buffer depth depends on several parameters (seamless retriggering required, trigger domain setup, trace length, …).

The listed suspension points are just a first sight overview. Implementing a DAQ whose performance seeks to achieve the extrapolated “maximum event building performance” must consider much more details. For instance, in the previous explanation to “maximum event building performance” extrapolation it was assumed that the event building routine must only prompt the “endevtno” value by the low performance. This assumption requires an ACP-CMA design that is capable to attach the header information (out of the event information ring buffers) automatically to its corresponding trace data. In the FC2.0 prototype the mode is called “headeracp”. Why the event header information prompting via the GP/LPD-IO is not an option in a high performance DAQ was mainly motivated by the poor GP/LPD-performance that significantly influences the event building process of events with small traces. This is not a general assumption that the “via GP/LPD” is always inferior to the “headeracp” procedure. This impression is mainly the result of the chosen read-out protocol and not discussed advantages of the “via GP/LPD” procedure. But in most of the cases the “headeracp” mode will be the backbone of the event building.

Also note in the extrapolation a “continuous” large ACP-CMA transfer was demanded (and not only a large ACP-CMA transfer). Continuously operating the ACP-CMA in a real application is much more challenging than one might think. Consider that the event building commands are given by the value of the “transmiteuntilevento” parameter and this parameter is a decision of the read-out server. “Prefetching” capabilities in the event building routine (running on the Zynq) could be a possible counterpoise. Designing such procedures unfortunately proved more challenging than expected and is also not urgent as a relevant fraction of the ZU+ missing performance (in Figure 44) can be explained by other procedures.

6.6 Setup for performance evaluation of HP-DMA

The design of the HP-DMA that was used for the continuous HP-DMA evaluation was already illustrated in Figure 20 (page 38). Its surrounding firmware design in the continuous HP-DMA evaluation is delineated in Figure 48. Every HP-port of the PS/PL interface is connected to its own DMA and monitoring logic. The ports can be operated and verified independently of each other. Every port has its own HP-DMA buffer in memory.

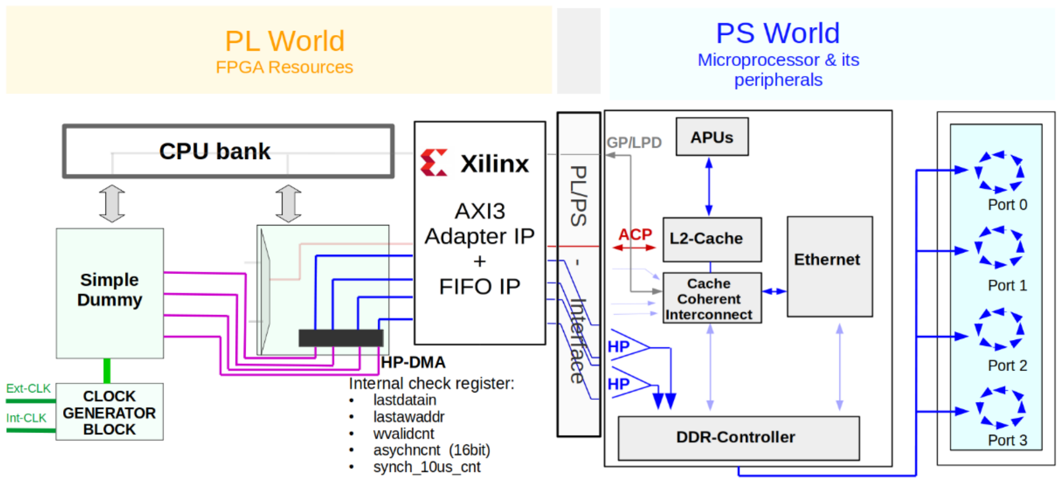
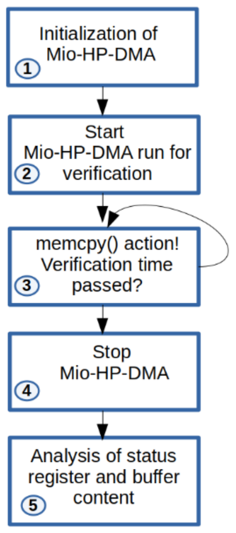


Figure 48 Firmware setup of HP-DMA Performance Evaluation. The asychncnt register counts the interrupted transfers of a connected HP-port. A transfer is interrupted when the signal wnext is deserted (Figure 19). In case of an actively operated HP-DMA this is equivalent with a desertion of the HP ports wready output signal. As in the HP-DMA Performance evaluation setup the server routine is not required (only specific client routine is running), the "omniscient" server of Figure 10 is not implemented in this figure.

Besides the possible bandwidth, the interrupted transfer rate is decisive. In principle, a transfer interruption can be caused by the HP-DMA or by the HP-ports control signal. In the setup it can be guaranteed that the HP-DMA can continuously provide valid dummy data and therefore its wvalid signal during the active verification time is always asserted. A deserted wnext signal (Figure 19) indicates an interrupted transfer and is monitored by the (16bit) asychncnt status register of the HP-DMA monitoring logic. If in a test run more than 65536 interrupt cycles at a port occur, the corresponding asychncnt status register monitor logic cannot count the exceeding amount. Note that already a single interrupted transfer (during the active verification time) ultimately will result in in a design rejection that use a similar or higher bandwidth.



① Initialization of Mio-HP-DMA
② Start Mio-HP-DMA run for verification
③ memcpy() action! Verification time passed?
④ Stop Mio-HP-DMA
⑤ Analysis of status register and buffer content

Figure 49 Continuous HP/memcpy() memory access investigation

The (Zynq) boards software routine that does the HP-DMA verification is quite easy as it must not communicate with a read-out server anymore. Its procedure is delineated in Figure 49. In the first step the routine initializes the HP-DMA and its corresponding monitoring logic. In the initialization step amongst others the burst size and the buffer size are selected. In the next step, the routine must enable the HP-DMA. The DMA is now running without any further CPU support. The HP ports wready signal has a higher desertion probability in cases that the memory is charged with further memory accesses. The performance evaluation results should be comparable with real life bearings in that the memory is significantly charged with CPU accesses ("event building"). The CPU now tries to operate continuously memcpy() system calls until the given verification time is passed (step 3). The memcpy() call always tries to copy the complete HP-DMA memory buffer to a static buffer of the same size. As the memcpy() call should retrieve the actual data and not matured cache data, every memcpy() call requires a previous call of the CpuSyncDMA() function. Explicitly note that during the memcpy processing time the HP-DMA is continuously in active mode and therefore continuously writes into the memcpy()'s source buffer! When the demanded verification time is passed, the routine can disable the HP-DMA. In the final step the routine must do the performance analysis (on basis of the corresponding monitoring registers) and outputs its results. The most important results of the performance analysis are the determined memcpy() "bandwidth" and the determined number of data transfer interrupts (labeled as "asychncnt" in Figure 20).

The main aim of this work is to find a HP-DMA setting that enables a continuous ("isochronous") data transfer with a sufficient high-bandwidth. It is not the primary intention to determine the maximum best-effort bandwidth of non-continuous data transfers. Such a specification is useless for the intended operation style in FC2.0. Furthermore, this topic was already elaborated in the publications of Table 12. In contrast to the ACP-CMA whose possible transfer settings are largely given by the cache line and cache size conditions, the possible HP-DMAs transaction settings differ significantly. Preliminary investigations indicate that the burst size has a major influence on the transfer interruption rate. How effectively the HP data streams can enter the memory also depends on the actual CPU and Ethernet controller access rate. In the designated event building routine the major CPU memory workload is caused by the memcpy() operations. All these supposed influences must be scrutinized systematically. Table 13 lists the selected parameter values that were elaborated with the suggested method.

Table 13 Parameter settings in HP-DMA evaluation . It was tried to measure every parameter value (listed in the table) in every possible combination. The HP-DMA design in the Zynq-Ultra+ device can be synthesized until 300 MHz. The Zynq-7k devices had synthesis with an operation frequency until 150 MHz. This limitation of the Zynq-7k devices is highlighted by green accentuations. Performance measurements with a burst size of 128 and 256 were not done in all possible combinations. This is the reason why the values were highlighted by a yellow accentuation. In all measurements the HP-DMA had a data width of 32bit.

System clock	Active HP-DMAs	Burst size	Transfer size [32bit words]
50	1	8	512
100	2	16	2048
150	3	32	8192
200	4	64	32768
250	-	128	65536
300	-	256	-

It was tried to determine the memcpy() bandwidth and the interruption cycle number for all possible parameter combinations. Exception of this rule are the transaction sizes of 128 and 256. They were only implemented for important limit cases. The Zynq-7k FPGA fabric is less timing friendly and therefore the HP-DMA performance evaluation design implementation was limited to operation frequencies until 150 MHz. The Z7k and the ZU+ module from Trenz not just differ in their in the PS implemented DDR controller system, they also differ in their implemented DDR memory type and size (Table 3). The DDR memory size should not have an effect on the results of this test setting. The memory type and its implementation are decisive for the performance of the DDR controller and thus also for the HP-DMA. A comparison of the Trenz selected memory type and the PS DDR Clock Domains Performance of the Zynq devices (given in [16] a. [15]) confirms that the selected memory will not decrease the maximum performance of the PS DDR controller. A tampering of the setups performance by a poor DDR memory selection is therefore unlikely.

To check the influence of a parallel running ethernet connection, a setup with an independent concurrent ethernet transfer via “nc” was used (appendix chapter A.11). In the “(nc) ethernet” setup of the “HP-DMA performance evaluation” additionally the HP-DMA performance evaluation routine is running. The performance evaluation routine is not actively informed about the parallel running ethernet evaluation chain and consequently does not have any kind of IPC with it. The evaluation routine therefore still outputs its determined memcpy() bandwidth without any consideration of the simultaneous ethernet resource consumption.

6.7 Results of HP-DMA

The discussion of the results is split in two subchapters. In the first subchapter the continuous (“isochronous”) transfer performance of the HP-DMA is elaborated. In the second subchapter the remaining available memcpy() access bandwidth on memory is elaborated. Both evaluations are based on the same measurement and are solely a different representation of the results.

6.7.1 Interrupt rate in dependence on the parameter settings

This subchapter is dedicated to the feasibility of the continuous HP-DMA. In FC2.0 a rigorous approach is pursued. As soon as the elaboration measurements monitor a single "non-continuous" situation, a feasible implementation with equivalent bandwidth is rejected. In the first measurements only the HP-DMA and the memcpy() invocations in the elaboration routine will proceed memory accesses. In real FC measurements ethernet transfer is required. As ethernet transfer have strict timing requirements that they must consider, the Zynq ethernet controller DMAs need a low latency, high priority access to the memory. Ethernet transfers make HP-DMA transfers especially "non-continuous" prone. All figures (Figure 50-Figure 52) in this subchapter have the same composition. The y-axis indicates the operation frequency of the HP ports, the x-axis the (ARM AXI4) burst size of the HP-DMA (interaction with the state machine was introduced in Figure 18). Do not mix this burst size with the memcpy() transaction size. Like already mentioned all corresponding memcpy() measurement results are condensed in a single interrupt result value in these figures. The corresponding "(non-continuous) interrupt" results relate to the bubble symbol's color and size. A green bubble symbol indicates a measurement that was interruption free and ergo a continuous transfer. Blue bubbles represent measurements that monitored at least one clock cycle with a deserted wnext signal and ergo represent a non-continuous transfer.

The Zynq-7k and the Zynq-Ultra+ differ significantly in their implemented memory system and maximum FPGA fabric utilization efficiency. In order to enable an elaboration design that can comprise both series without any adaption, the interrupt counter was limited to 16 bit. For the chosen measurement time of 60 s this means that every second an average of 1092 clocks with a deserted "wnext" ("non-continuous") signal can be monitored. At an operation frequency of 100 MHz this means that approximately every 100,000 clocks a deserted wnext signal can occur[8]. If this limit is exceeded, the counter will stop to increment. This is important as the following figure result values relate the sum of all enabled HP channels. Every frequency operation/ enabled channel amount pair is elaborated with 5 different memcpy() transfer sizes (Table 13). As illustrating every measurement is not possible in a single plot, the information must be condensed. The figures in this subchapter indicate the "interrupt" sum of all corresponding memcpy() transfers measurement by the size of their bubble. A bigger symbol size means more interrupts. In the analysis routine the presented figures are obtained by equation (4). Again, note that this is a tendency due to the counter limitations and measurement/channel summation.

[8] One might think that a 1bit interrupt counter ("1 deserted clock") might be sufficient to prove the failure of the setting. This is in principle right but does not consider all real-life points ("How certain is such a result?"). A much better impression of the real-life feasibility can be obtained by trend analysis. To do this, a sufficient counter width is required.

$$S_I = \sum_{n=1}^{N}\left(\sum_{m=1}^{M} A_{n,m}\right) \tag{4}$$

N is the number of enabled HP-DMA channels, M is the number of checked memcpy() transfers sizes and $A_{n,m}$ is the corresponding 16 bit value of the asynchncnt register ("deserted wnext clocks").

The usage of the determined "interrupt" sum (equation (4)) as decisive decision parameter is critical and should not be obeyed without any contemplation. A continuous design means that the corresponding design has an "interrupt" sum is zero. But in reverse, a determined "interrupt" sum of zero does not necessarily mean that the design always behaves as "continuous". This is only the case when the corresponding evaluation measurement had an infinite time duration. Obviously, such an evaluation measurement never will be done and therefore the "continuous conclusion uncertainty" of the interrupt sum also depends on the selected evaluation time. In this thesis the large number of elaborated design settings limited the measurement time to (5×)60 s. If a setting in the measurements of this thesis is on the edge to continuity, the corresponding "interrupt" sum has a significant uncertainty.

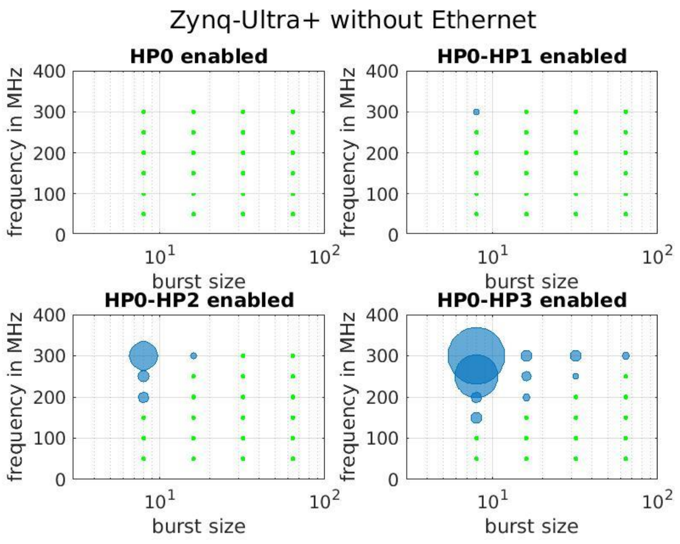


Figure 50 ZU+ continuous transfer interruptions dependent on operation frequency and burst size. The y-axis indicates the operation frequency of the HP-DMA (in the FPGA-Fabric), the x-axis indicates the burst size of the HP-DMA (AXLEN in chapter 4.2.1). In all plotted measurements the concurrent memcpy() invocation were still ongoing. The corresponding memcpy() performance is depicted in Figure 53 - Figure 55 but is not a subject in the current chapter.

Figure 50 is for concurrent HP-DMA/memcpy() memory access setups that have no significant ethernet traffic and therefore only an intermediate step for the final HP assessment in FC 2.0. If only one HP port is enabled, its traffic is continuous. As in the most experiments burst sizes bigger than 8 (32 bit) words per transaction are acceptable, the same is true for two enabled HP ports (left bottom). Note that in principle a burst size of 16 would enable an error free operation of all dual HP port setups. But in practice situations can occur that lead to a significant increase in the memory charge (for instance a remote user logs in via ssh and starts another routine). Such situations can provoke corrupted traces and are notoriously famous as a hard to debug issue. Thus, a plenty of spare is the only solution in FC. Instead of installing a bandwidth of 2.4 GB/s (“HP0-HP1 at 300 MHz”) with a burst size of 16, a burst size of 64 is chosen. This rule is not a hard limit for other Zynq designs. In fact, the data in the appendix will even unveil (restricted) strategies for continuous transfers with a bandwidth of 4.8 GB/s. Figure 50 implies that such a setup is not possible. This must not be true, if the designer considers further limitations. The interruption results for a measurement with concurrently enabled 1 Gb/s ethernet transfers is illustrated in Figure 51.

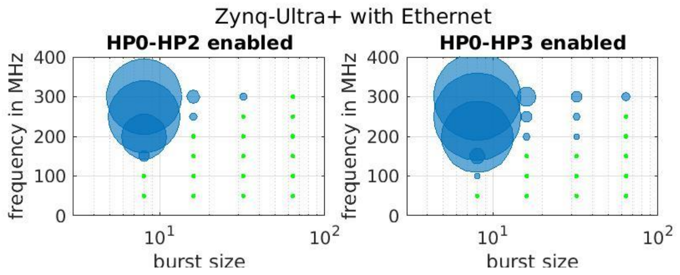


Figure 51 ZU+ continuous transfer interruptions with parallel ethernet operation. Same procedure like in Figure 50, but with additionally enabled ethernet transfer (~118MB/s (Table 30 in appendix chapter A.12) – the ethernet performance specification relates to a ethernet setup where the HP-DMA was disabled. The observed ethernet performance in the corresponding measurements of Figure 51 was slightly less, but always comparable). Note that the event building and the ethernet transfer implementation of the setup are not comparable with the required procedure in a real Zynq client routine. As ethernet data stream (not connected to any event data stream!) in this setup has the same priority as it would have in a real client routine, the measurement still enables reasonable estimations of the performance of a real routine.

Precheck measurements with one or two enabled HP-DMAs and concurrently running (~118 MB/s) ethernet showed a behavior (with slight worsening) that is compatible to the previous measurements without ethernet. Such settings are feasible for real experiments without any limitation. Therefore, these measurements were not extensively elaborated with corresponding enabled ethernet measurements and are consequently not a part of Figure 51. Like expected, parallel usage of 1 Gb/s ethernet “kills” setups that were already close to the edge and increases the non-continuous clock amount of the already failed ones. As summary can be concluded that the Zynq is capable to tackle a continuous HP-DMA bandwidth of 2.4 GB/s and a memory friendly parallel running 1 Gb/s ethernet transfer routine.

The results of the Zynq-7k modules are more sinister (Figure 52). Even without any additional memory access charge the determined “pure” PS ethernet performance was only 39 MB/s (Table 30 in appendix chapter). During the Figure 51 measurements the ethernet performance sometimes was drastically poorer. The HP ports have roughly a continuous bandwidth of 0.8 GB/s. This performance is just sufficient for 4 16bit ADCs that are operated with a clock of 100 MHz. A cheap design solution that can compete with FC1.0 costs and continuously writes the ADC streams into the DDR memory is not possible. Consider that the Zynq-7k modules use single channel DDR3 memory and therefore cannot compete with the Zynq-UltraScale+ that uses dual channel DDR4 memory.

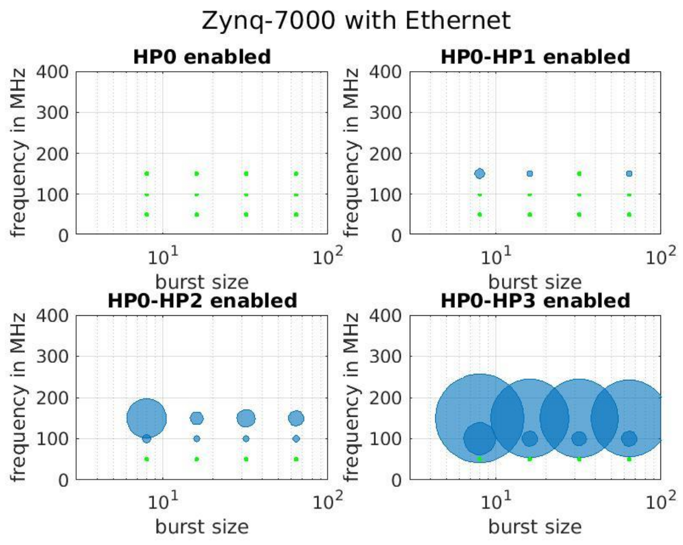


Figure 52 Z7k continuous transfer interruptions with parallel ethernet operation. The ethernet stream in this setup was always below 39MB/s (Table 30 in appendix chapter A.12). The poor PS ethernet controller implementation is one of the Z7k key disadvantages. The official PS ethernet device driver of the Z7k does not permit Jumbo Frames and hence has a significant disadvantage in compare to the ZU+ device driver.

6.7.2 Memcpy performance

Independent on the kind of the HP-DMA implementation (continuous/non-continuous) the remaining memcpy() performance determines the possible event building performance. The elaboration in this chapter uses the same setup and the same data like the previous subchapter 6.7.1. The achieved memcpy() performance always depends on the memcpy() transaction size. Unfortunately, there is no easy rule that describes the observed performance behavior. In the "interrupt" elaboration it is legit to merge/add up all interrupts of a specific implementation as the measurement conditions not really care – in continuous setups a "interrupt" is strictly forbidden in all transfer sizes. The memcpy() performance is decisive in event building of high performance measurements. Event building routines typically guess that the used DMA offers best efficiency at large transaction sizes. The detailed data shows that this simple rule is not true for the developed continuous HP-DMA designs and strongly can mislead into the wrong direction. A routine that wants to enable the maximum performance must therefore consider details that surmount the aspiration of this subchapter. Figure 53- Figure 58 condenses the determined results under various elaboration focuses. The y-axis and the x-axis are identical to the previous chapter, the color of the markers represent the bandwidth of the memcpy() operation. All setups that had interrupts are additionally marked by a red circle.

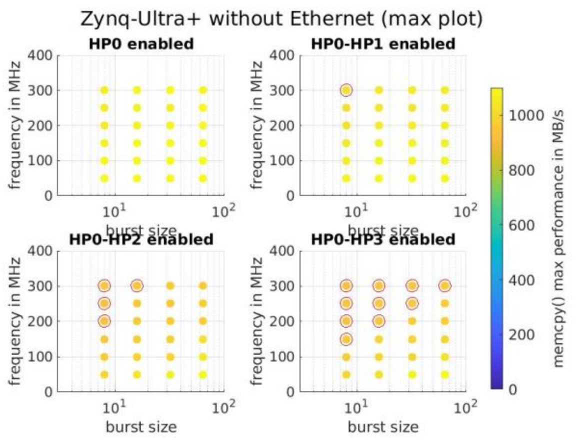


Figure 53 ZU+ best memcpy() bandwidth during active run of HP-DMA. The y-axis indicates the operation frequency of the HP-DMA (in the FPGA-Fabric), the x-axis indicates the burst size of the HP-DMA (AXLEN in chapter 4.2.1). The burst-size of the HP-DMA is independent of the memcpy() transfer size. Without an enabled HP-DMA channel memcpy() (including cache invalidation) has a max performance of 1183.9 MB/s.

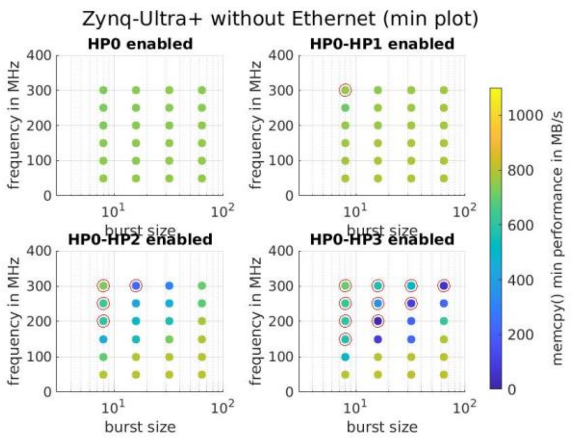


Figure 54 ZU+ worst memcpy() bandwidth during active run of HP-DMA. The y-axis indicates the operation frequency of the HP-DMA (in the FPGA-Fabric), the x-axis indicates the burst size of the HP-DMA (AXLEN in chapter 4.2.1). The burst-size of the HP-DMA is independent of the memcpy() transfer size. Without an enabled HP-DMA channel memcpy() (including cache invalidation) has a min performance of 811.7 MB/s.

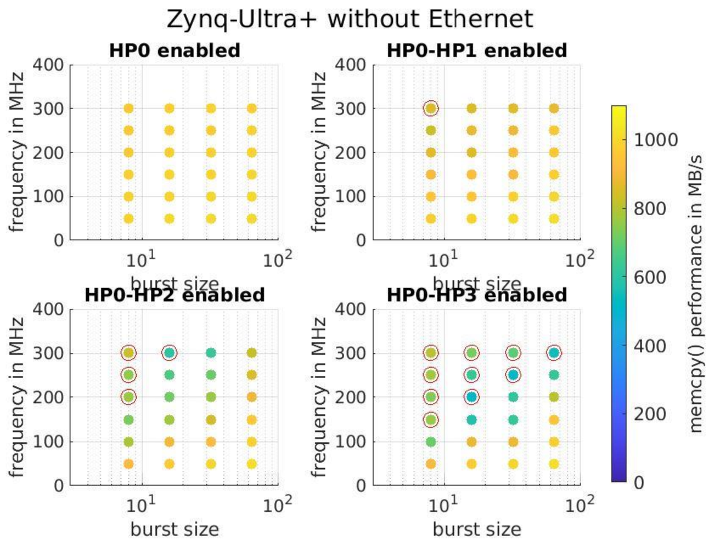


Figure 55 ZU+ mean memcpy() bandwidth during active run of HP-DMA. Without an enabled HP-DMA channel memcpy() (including cache invalidation) has a mean performance of 1055.8 MB/s.

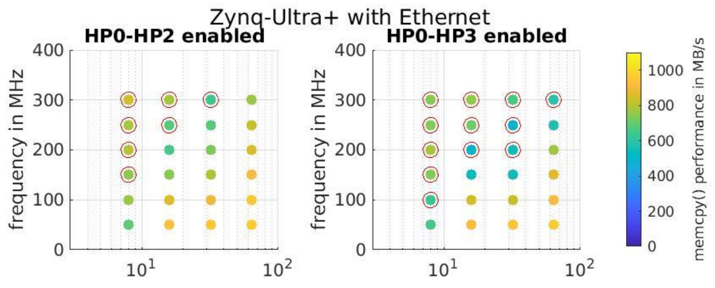


Figure 56 ZU+ mean memcpy() bandwidth during active run of HP-DMA and parallel ethernet operation

Figure 53 shows the determined maximum memcpy() performance of the corresponding designs. The maximum memcpy() performance is always bigger than 930 MB/s and therefore significantly outperforms the default ACP-CMA for all feasible operation frequencies. Unfortunately, the suggested continuous HP-DMA implementation in the FC2.0 prototype cannot consider the required restrictions of such a maximum number. The numbers of Figure 55 consider the restrictions of a continuous HP-DMA operation in FC2.0 more accurately. In chapter 6.7.1 as rough advice was given, that the suggested continuous HP-DMA implementation should not be feed with a larger input bandwidth as 2.4 GB/s. In such a setup the feasible mean memcpy() performance (in Figure 56) still is around 570 MB/s or higher (depend on the actually selected burst size). Condensing all discovered properties of the suggested continuous HP-DMA implementation in concludes in following golden rules:

(1) A design that uses a burst size of 64 and considers the recommended maximum HP input bandwidth of 2.4 GB/s will have a memcpy performance of ~940 MB/s. This performance is sufficient if a 1-2 Gb/s ethernet link is used.

(2) In the recommended range (burst size 64 and maximum HP-DMA bandwidth of 2.4 GB/s) the memcpy() performance increases with the selected transaction size.

(3) If the recommended range is exceeded the memcpy performance will drop until the HP-DMA input bandwidth causes interrupts. As soon as the “interrupt” frontier is passed, the QoS gives up the CPU access constraints and the memcpy() performance will rise up again.

(4) The HP-DMA input bandwidth has a smaller influence on small memcpy transactions than on large memcpy transactions.

(5) With a burst size of 64 and small memcpy() transaction sizes (approximately below/equal 2048 words) independent of the HP-DMA input bandwidth no interrupt will (very likely) show up. As big transfer can be built up out of small transfers, this point could (but needs not) be an eligible loophole.

(6) In the recommended range the (1 Gb/s) ethernet usage reduces the memcpy() performance about ~100 MB/s. As in the recommended range a lot of spare in the memcpy() performance is available,

the memcpy() performance loss due to additional ethernet consumption is only weakly sensitive on the final DAQ performance.

The listed golden “continuous HP-DMA” rules are only valid in ZU+ devices.

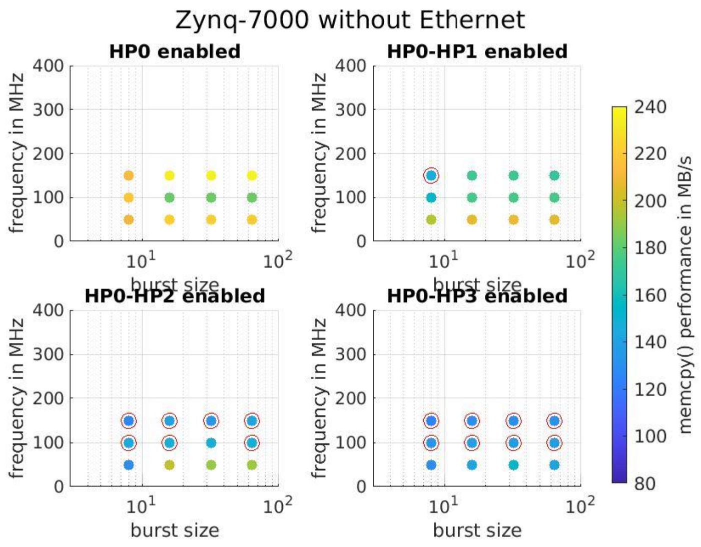


Figure 57 Z7k mean memcpy() bandwidth during active run of HP-DMA. Without a enabled HP-DMA channel, the mean memcpy() performance (including cache invalidation) is 254 MB/s.

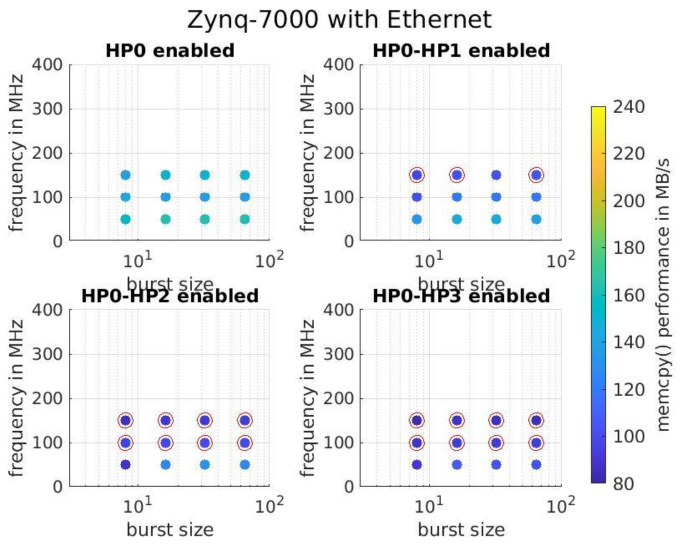


Figure 58 Z7k mean memcpy() bandwidth during active run of HP-DMA and parallel ethernet operation

In Z7k devices memcpy() performance (relating to the HP-DMA continuous framework) cannot compete with the performance of the ACP-CMA. Just in special applications that demand a bigger buffer capacity than the available FPGA-RAM of the devices, the implementation of the continuous HP-DMA could be motivated. In case of the Z7k the following golden rules were observed:

(1) Zynq-7k has not the clear trends like the Zynq-Ultra+

(2) In the Zynq-7k the effective HP-DMA bandwidth is always inferior the effective ACP-CMA bandwidth.

(3) The continuous HP-DMA (with a data transfer into external memory) does not offer loophole/enhancement possibilities that justify a more elaborated scrutiny.

6.8 Summary

The Zynq family of Field Programmable Gate Arrays (FPGA) offers a large variety of chips in terms of number of pins, programmable logic volume (PL) and integrated CPU processing system (PS). Hard-wired common CPU peripherals (DDR controller, Ethernet, SPI, …) are always available. The PL/PS regions of the Zynq can exchange data via the so-called PS/PL interface that has several kinds of ARM AXI protocol compatible ports (GP/LPD, HP, ACP, …) that differ in their system implementation. The device cost typically scales with the number IO pins and internal FPGA-RAM/programmable logic volume (Table 9). The Zynq family currently is sub split in two series: Zynq-7000 (Z7k) and Zynq-UltraScale + (ZU+). The lowest speed grade of Z7k devices use an ARM Cortex-A9 32 bit CPU with an operation frequency of 667 MHz, the ZU+ series has ARM Cortex-A53 64 bit CPUs with an operation frequency of 1200MHz. All Zynqs have in common that their ARM CPU and memory system is capable to run a default Linux system.

The goal of the overall Zynq evaluation project is a data acquisition (DAQ) that can accommodate several experiments with different requirements on digitization frequency, number of ADC bits, trace length and triggering schemes and rates. The focus in this thesis is on the readout for germanium detectors and to a minor extend also for photomultiplier tubes (PMT). Though their differences, the required DAQ of such experiments has also many similarities. In these DAQs the detector waveforms that contain the desired physical information are digitized by an ADC, locally buffered on the board, and subsequently transmitted by an Ethernet transfer that is controlled by a Read-Out server. The thesis investigation builds on the experience gained with the existing system called FlashCam, labeled as FC1.0. The new project builds on the above mentioned modern Zync FPGAs. The company Trenz Elektronik offers pluggable Zynq FPGA module that are the investigation focus of this thesis. The Trenz module cannot be operated on their own, they require so called “carrier boards”. The developed prototype, named as “FC2.0 prototype” uses a Trenz developed carrier board, but for the future FC2.0 DAQ it is intended to develop a customized version. This is important, as in the current setup the signal routing between the Carrier board and the FPGA module prevents an implementation of all Zynq pins. Unfortunately, in some Trenz modules the poor signal routing choice considers even the precious multi-gigabit transceiver (MGT) pins. This prevents an implementation of Xilinx PS-PL/PL ethernet approaches. In the FC2.0 prototype only the Ethernet components of the PS volume can be used.

Besides the technical aspects, also the costs of an experiment are important. This especially considers the designated experiments of FC2.0 where several hundred channels are implemented. In FC1.0 a Germanium detector channel costs around 280 € - a similar price was initially pursued in FC2.0. Costs can be reduced by implementing cheap devices - that commonly have less FPGA-RAM and consequently have a lower event data buffer capacity - or by increasing the channel number per device. IO-Pin number and FPGA-RAM size are not the only distinguishing features of the devices. The devices furthermore differ in their possible maximum (clock) operation frequency (given by their speedgrade), clock tree architecture, possible IO styles (especially the serialization/deserialization features), performance of their CPU and many other key properties. The costs are also the main reason to use the PS ethernet, as it does not require the precious MGT pins of the Zynq devices.

The implementation of the DAQ workloads is distributed in several subtasks (Figure 11):

- The read-in of the continuous ADC raw data stream is done by the “ADC-Data-Read-In” Block. The block uses special IO serialization/deserialization features of the FPGA fabric that can be extended by a logic. The required procedure is given by the ADC and the capabilities of the deserialization circuit of the device. The block relies on the generated system clocks of the “Clock Generator Block”.
- The extraction of the waveforms and their Zynq internal buffering is a task of the “Pulse detection”, “Trigger Acceptance Control” and the “ADC Ring Buffer” block. The task is completely done in the PL part of the Zynq device. The “ADC Ring Buffer” block is used to buffer the accepted waveforms, the “Event Information Ring Buffers” contain the corresponding trace header information. The pulse detection block must detect the pulses in the raw ADC sample stream, the “Trigger Acceptance Control” block must transform the trigger pulses of “pulse detection block” in trigger pulses (“frame trigger”) that start a correct waveform read-in into the ADC Ring Buffers. As in the future the relation between the accepted waveforms is a subject of the investigation, it is important that the DAQ also extracts their system clock accurate trigger time (buffered in the corresponding event information ring buffers). The FC2.0 prototype firmware derivatives the time out of the “Clock Generator Block”. The chosen size of the ADC Ring Buffer and the Event Information Ring Buffers is one of the most critical aspects in system design. Is the buffer size to low, the likelihood of a DAQ deadtime rises, is the buffer size to large only expensive Zynq devices can be used to implement the design (or a decrease in the channel number per device accepted).
- The transmission of the requested event data and all corresponding data transfer workloads are done by the “Processing System” of the Zynq Device. In this job the DMA, CPU and Ethernet Controller

performance and interaction is decisive. The event data (trace + corresponding event information like event time) is in the suggested DAQ structure only accessible by the PS/PL-interface DMAs (GP/LPD-IO unit, HP-DMA, ACP-CMA). In a typical Germanium detector application, a trace comprises at least 256 16 bit samples and not more than 8192 samples (512-16kB). The trace header size is much smaller – the trace header of FC has a default size of 40B. As in FC the trace header contains redundant information, a further decrease is possible. The GP/LPD-IO has a particular position as it is the DMA that set/gets the register values of the "CPU bank" firmware block that effectively controls/read-outs the status and control registers of all other firmware blocks. To access the trace data the ARM CPU must operate the ACP-CMA/HP-DMA respectively. In the continuous HP-DMA mode, the CPU can access the trace header data by the GP/LPD-IO. In the implemented ACP-CMA an event information prompting via the slow GP/LPD IO is not a requirement (but possible if for some reasons preferred). As proxy that transacts the decisions of the read-out server, the ARM CPU controls the trigger system. The management tasks include deallocation of event buffer space, read-out of prompted event data and forwarding of the actual occupation status of the buffers. This information is part of the reply/request messages of the developed read-out protocol.

- The chosen DAQ structure require a read-out server that supervises and controls the read-out process on basis of the received ethernet messages. In the FC2.0 prototype system the read-out server performs the online/offline analysis.

In this thesis the focus was on PS/PL interface ports of the Zynq devices and their usability in a DAQ for Germanium/PMT applications. Several port performance evaluations with different aspects are already published. Considering the Zynq-7000 device family, [19] is likely the most quoted one. The publications have in common that their performance determination setups not necessarily consider restrictions that apply in a real application. Many publications that investigate the PS/PL interface bandwidth do not consider the CPU access to the port transferred data. In particular tasks, like in the event building of the FC2.0 prototype, the CPU data access performance is decisive. Achieved performances of investigation setups that do not consider the CPU data access bandwidth are therefore more theoretical, than feasible. In this thesis it is tried to determine the port performance and the corresponding CPU access bandwidth.

All Zynq devices have ACP, GP/LPD and the HP ports. In principle the ACP as the HP port could be used to adapt the classical non-continuous DMA transfer setup of Figure 11 where the event data is buffered in the available FPGA-RAM (event information ring buffers and ADC ring buffers). In contrast to the ACP port that is directly connected with the Cache system, the HP ports enable continuous writes into the external memory. Designs that use HP ports hence can relinquish on the ADC Ring Buffer and the Delay Pipeline unit. As both components are the main consumer of the FPGA internal RAM and FPGA-RAM is one of the main cost drivers of the Zynq devices such designs can enable a significant decrease of the costs or allow designs that would not be possible due to the limited FPGA-RAM.

In the FC2.0 prototype the GP/LPD-IO is used to set/get the register values of the CPU bank. The architecture of the GP/LPD-IO was designed for low latency access into the FPGA volume and is not eligible for high performance transfers – for instance the read (FPGA → CPU) performance of the LPD port is only 20.3MB/s (Table 24). This is far away from the maximum performance of the ZU+ HP/ACP ports that is in the range of GB/s. The GP/LPD-IO therefore should only be used in cases that do not have a more performant alternative. The GP/LPD-IO is optimal for word status polling tasks. The performance investigation of the GP/LPD-IO is covered in appendix chapters A.13 - A.14.

Continuous writing to the external memory with the HP-DMA might lead to data losses if the transfer rate is too high. For the ZU a transfer rate of 2.4 GB/s to the dual port DDR4 RAM was achieved with parallel reading of the memory by the ARM CPU (Figure 51). This rate has a significant safety margin before data losses occur - under consideration of additional measures/restrictions even higher interrupt-free input bandwidths are possible (in the conditions of Figure 51 ~4.0 GB/s). But the required restriction may violate a condition of an application and are therefore not a general solution. In the evaluation setup the CPU could access and transfer the HP-DMA data in the memory with a memcpy() bandwidth of ~940MB/s. The parallel CPU access performance is important for the final event building performance. 940MB/s is more than a single PS 1Gb/s Ethernet controller can transmit und thus the 1Gb/s ethernet is the design bottleneck. In a Zynq-7000 (by default with single port DDR3 controller) the implementation of a continuous HP-DMA is only feasible in special cases, as the parallel operating CPU only can access the memory data with a low bandwidth (can be less than <120MB/s) and furthermore the maximum continuous HP-DMA performance is much lower (~0.8GB/s). Remember that waveform buffering on external memories via continuous data streams is not new and was introduced long before the advent of the Zynq devices. In these classic approaches in the FPGA a DDR controller IP is getting implemented and via the IO pins of the FPGA the external DDR memory is fed with the sample stream. Enclustra (Zurich, Switzerland) and Trenz Elektronik also offer such kind of Zynq modules that have a separate memory system that is only accessed by data streams of the FPGA. In this system uncontrolled CPU access do not occur and consequently it does not need such a high security spare as the developed continuous HP-DMA approach. For the FC2.0 DAQ the approach is not eligible as it occupies precious IO pins.

The ACP-CMA has a maximum performance of 1060MB/s in the Zynq-7000 devices (chapter 6.5.2). In case of the Zynq-UltraScale+ devices the maximum performance is 2.4GB/s. In contrast to the HP-DMA, the ACP-CMA (channel) destination buffer is already in the cache. The HP-DMA requires a combined cache-invalidation-memcpy() procedure to transfer the HP-DMA data from the memory to its cache destination address (in chapter 5.2 labeled as user space “ethernet buffer”) and consequently has a lower CPU access rate to the transferred data.

In practical DAQ operation also the pure ethernet performance is important. In pre investigations the required CPU performance for various PS ethernet transfers via netcat was monitored (appendix chapter A.12). In these pre-investigations the maximum PS ethernet transfer bandwidth in Z7k devices only could be gained, when the ethernet operation consumed at least 50% (up to 90%) of the computing time of a single CPU core. The ZU+ devices only need 20% of the computing time of a single core to gain their maximum PS ethernet performance. The determined Z7k ethernet operation behavior is adverse for high performance DAQ designs as the CPU computing time is required for the event building jobs and should not be dissipated for ethernet controller operation.

In the event building performance investigation chapter 6.5.3 the “headeracp” mode of the FC2.0 firmware was used (results illustrated in Figure 44). As in this mode the ACP-CMA reads-out the ADC Ring Buffer and the Event Information Ring Buffers, the slow GP/LPD-IO must not be used to prompt the trace header information. If the event building routine does not consider such a detail, its event building performance will decrease much faster at small trace lengths. In the setup of Figure 44 the used Ethernet link between Zynq and Server was the design bottleneck. Although the design used the poorest “Ethernet” option of the Zynq devices and does not use any (possible) multi-threading, still a bandwidth of approximately 95MB/s (ZU+ (51MB/s in Z7k)) was available. For small trace lengths in both measurements a lower bandwidth was determined. The “bandwidth” decrease for small trace lengths is in “headeracp” mode not a must, in the shown measurements it was a consequence of the chosen event ring buffer size. In practice a larger event ring buffer size could be chosen to tackle the issue. To estimate the “Ethernet bottleneck free” maximum event building performance of an ACP-CMA setup in chapter 6.5.3 an indirect approach that monitors the processing time was used. This approach concluded that an ACP-CMA setup can offer a maximum event building performance that agrees with the determined “pure” default ACP-CMA performances of chapter 6.5.1. Note that in this “bottleneck” free maximum event building performance investigation was assumed that the read-out protocol does not induce event building interruption times where the ACP-CMA and/or the Ethernet cannot be operated. In practice the chosen read-out modes do. To overpass the ethernet “bottleneck” limitations special MGT IO pins of the Zynq device can be used. These are however not available/conveyable on the investigated Trenz Zynq Modul/ carrier board combinations.

Figure 39 and Figure 40, that illustrate the effective ACP-CMA bandwidth in dependence on the selected transfer size, suggests that the optimal read-out performance requires the prompting of large ACP-CMA transfers. A DMA therefore must be capable to copy several traces in a single DMA transfer – otherwise settings with small trace lengths (“small ACP-CMA transfer sizes”) would show a performance that is significantly poorer than the performance of settings with a large trace length. The implemented ACP-CMA has such a capability. In this suggested optimal procedure, the influence of the Ethernet transfer is not considered. An Ethernet transfer has an optimal Packet Size. If the Majority of the Ethernet transfers misses the optimal Packet Size, the bandwidth of the already poor Ethernet link will fall further. The current read-out state machine of the server routine and the read-out protocol design are not optimized for these issues. Obviously, the restriction applies on all DAQs whose read-out and control are done by ethernet and thus also valid for FC1.0. Before concluding with a final performance comparison of the pros and cons of the possible FC2.0 FPGA devices, a repetition of the central FC1.0 design decisions is useful.

- The FC1.0 design only could offer the low channel price as it did not try to implement an external DDR memory. The reasons were only the high IO-pin consumption of the memory and not the price of the memory itself[9]. But this also meant that the used Microblaze CPU only can use the FPGA internal RAM as memory. As FPGA-RAM is precious, the Microblazes software routine needed to be written in a way that their operation requires as less memory space as possible. As the routine still must handle the same event building and initialization requirements like the FC2.0 prototype, writing such a bare-metal application is a challenging job. Obviously, such a routine has not sufficient memory space to implement extensive test routines or “gimmicks” like an integrated INL determination or signal analysis.
- In the FC1.0 design the “Fast-Simplex-Link” bus of the Microblaze is used to read-out the Ring Buffers and to forward the event-built data into the Ethernet controller buffer. The “Fast-Simplex-Link” is directly connected with the ALU registers of the Microblaze CPU and not just with the Cache like in the case of the ACP interface in the Zynq devices. Furthermore, the FSL bus is used to set/put the registers of the “CPU block” of the FC1.0 design. The FSL-DMA therefore also replaces the GP/LPD-IO of the FC2.0 prototype design.
- The IO-performance of the Microblazes “fast simplex link” can be increased linear with its operation frequency. In FC1.0 the Microblazes is operated by a 100 MHz clock. As in the meantime the FPGA architecture further matured, an operation frequency between 200-300 MHz should be a reasonable performance target for the new UltraScale+ Virtex devices (officially Xilinx states much higher maximum operation frequencies [26], but implementing them in such a dense design proposal like FC2.0 looks unlikely). Such a design would have an ADC Ring Buffer Access bandwidth between 0.8-1.2 GB/s or a combined read/write performance of 0.4-0.6 GB/s. As the buffer of the Ethernet controller block is a possible direct destination of this combined read/write performance even the ultra-performant proposed MPIK ACP-PL Ethernet Option is not unrivaled anymore. In contrast to the FSL-DMA-Ethernet collaboration block of the FC1.0 design, the “MPIK

[9] FC1.0 has 24 ADC channels per Spartan-6 device – every IO-pin of the device is important!

ACP-PL Ethernet" option has the disadvantage that it has not been implemented yet and therefore is only a concept that might (very likely) work.

In principle the next FC2.0 DAQ also could adapt the FC1.0 design to a new FPGA series without a hard-wired processing system. Thus, in the final comparison of Table 14, the Xilinx Virtex UltraScale+ series will also be considered.

Table 14 Final summary of the Zynq device evaluation results

	Zynq-7000 (FC2.0 prototype DMA design)	**Zynq-UltraScale+ (FC2.0 prototype DMA design)**	**Virtex UltraScale+ (FC1.0 DMA Design)**
CPU	ARM-Cortex-A9	ARM-Cortex-A53	Microblaze
External DDR required	Yes (The external DDR memory is not a hard requirement to operate the ARM CPU, but the external DDR is a hard requirement for the FC2.0 prototype design)		No (FC1.0 Design does not require external DDR memory)
OS	Default Linux (other OS also available)		Bare Metal (Linux is available for Microblaze, but in FC1.0 implemented Microblaze does amongst other things not have sufficient memory for a Linux OS operation)
Multithreading possible	Yes (if two cores are available)		In FC1.0: Single Core
FPGA-Manager available	Yes		No
Double operation performance and pulse spectroscopy eligibility of CPU	~10 % of Intel i7 desktop CPU (2.9GHz)	~20 % of Intel i7 desktop CPU (2.9GHz)	In FC1.0 not useful - Speed 2 is also not executable!!! (Depends on selected Microblaze – a typical Microblaze is not intended for DSP tasks)
Continuous writes into external memory	HP-DMA 0.8GB/s as upper limit recommended	HP-DMA 4.0GB/s as upper limit recommended	In FC1.0 Design: No external memory available, but in principle also a HP-DMA like DMA possible (if external memory available)! Virtex UltraScale+ has interesting HBM option!

	Zynq-7000 (FC2.0 prototype DMA design)	**Zynq-UltraScale+ (FC2.0 prototype DMA design)**	**Virtex UltraScale+ (FC1.0 DMA Design)**
Write into CPU cache	ACP-CMA Up to 1GB/s	ACP-CMA Up to 2.4GB/s	The Microblaze in FC1.0 does not have a Cache as its memory is implemented in FPGA-RAM
DMA Read from/into CPU ALU registers	DMA does not have direct access into the CPU ALU registers!		FSL-DMA $BW_{Max} = f_{Clk} \cdot 4B$ f_{Clk} Operation frequency of Microblaze (likely between 200-300MHz, the official maximum operation frequency is 682MHz [26] (requires Virtex-UltraScale+ FPGA of highest speedgrade))
DMA bandwidth dependence on transfer length	Significant – transfer size optimization is a must	Significant – transfer size optimization is a must	Negligible
Main consumer of FPGA-RAM	Delay line (optional), Event Information and ADC Ring Buffers (optional)	Delay line (optional), Event Information and ADC Ring Buffers (optional)	Microblaze, Delay Line, Event Information and ADC Ring Buffers
Ethernet **(nc = ethernet transfer via netcat)** **(Xilinx PL Ethernet options rejected due to cost reasons)**	PS Ethernet (nc: ~39MB/s), PS-PL Ethernet, MPIK-ACP-CMA-Ethernet-IP (not developed yet)	PS Ethernet (nc: ~119MB/s), PS-PL Ethernet, MPIK-ACP-CMA-Ethernet-IP (not developed yet)	MPIK-FSL-DMA-Ethernet-IP (implemented in FC1.0)

The bandwidth of the Zynq-UltraScale DMAs significantly exceeds the pure DMA bandwidths of the Zynq-7000 and the Virtex-UltraScale+ designs. In practice the Event Building Performance matters. Due to its highly efficient FSL-DMA the old FC1.0 firmware design is still competitive in event building tasks – but this assumption strongly depends on the achieved Microblaze operation frequency.

In the written thesis the interaction between the Zynq device resources and the application requirements was not a focus. In practice the device resources and costs affect the DAQ development process. Example application designs that use the developed FC2.0 prototype firmware functionalities are discussed in appendix chapter A.19.

7 ADC nonlinearity

The possible influence of the ADC nonlinearity on the reconstructed energy of HPGe detectors was already mentioned in chapter 2.2.4. Based on different approaches modern HPGe-spectroscopy experiments try to correct the ADC nonlinearity influence. The efficiency of these corrections depends on the accuracy of the determined ADC nonlinearity characteristics and other parameters. The availability of a precise ADC nonlinearity determination method is hence essential in high resolution HPGe spectroscopy. Established nonlinearity determination methods like the histogram test with linear ramp or sinewave fitting approaches are well known in HPGe-spectroscopy. For some time, ADC nonlinearity determination evolved without facing major obstacles. As signal source common shippable sinus wave generators or ramp generators could be used. Nowadays this is at least a money issue, as in the last years signal generators resolution stagnated, but the ADC resolution increased. Classic ADC nonlinearity determination methods are limited by the accuracy of their signal sources if not a second accurate compare ADC in the test setup is available. A target in the project of this thesis therefore was the development of a method that simply relies on a default signal generator out of the shelf. This chapter mainly intends to explain the new developed method and its differences to methods of “competing” collaborations.

In this chapter an overview of the investigated ADC nonlinearity determination will be given. In subchapter 7.1 the definitions of the differential and integral nonlinearity will be introduced. Subchapter 7.2 recapitulates the terms and requirements of the “classic” histogram setup that is strongly dependent on the pulsers distortion. The chosen ADC nonlinearity investigation approach of the GERDA collaboration is presented in subchapter 7.3. A tiny fraction of the chapter is also dedicated to another alternative method that in the first investigations (in 2019) looked promising. The official procedure of the Majorana collaboration is introduced in subchapter 7.4. The “Majorana” procedure is an interesting alternative procedure that was published lately in [8]. Subchapter 7.5 is dedicated the new developed “FC1.0 INL” method and its successor, the “FC1.0 INL via DNL” method. The final subchapter is a contemplation about the INL trend requirements of a simple INL correction.

7.1 Definitions of differential and integral nonlinearity

The target of the differential nonlinearity (DNL) and integral nonlinearity (INL) is a simple description of the real ADC transfer function in look-up table style. A simple example for an ADC transfer function is in

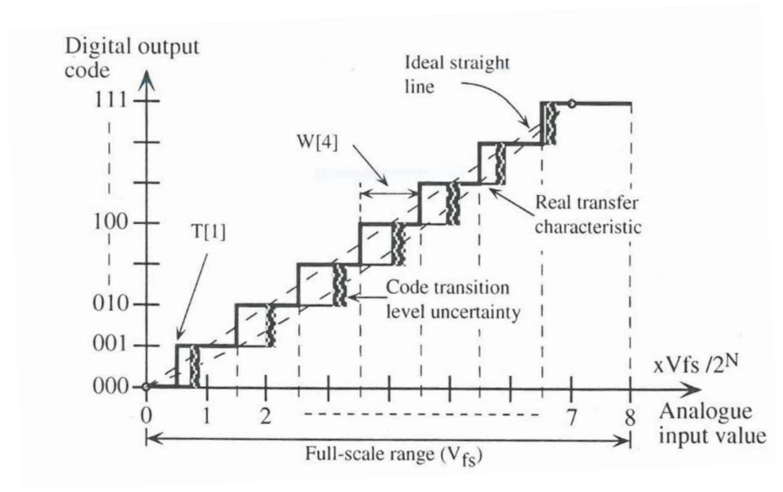


Figure 59 Typical transfer characteristic for a binary coded unipolar converter [27 p. 26]

Figure 59. In Figure 59 the code transition level voltage of code i is the voltage level at which the ADC with equal probability will output a code of i-1 and i. A more general approach considers a transition level as value of the ADC input that causes half of the digital output to be greater than or equal to i, and half less than i [27 p. 28]. The definitions of ADC gain and offset describe the global relation between a straight line and the ADC transfer function. In a perfect ADC an ideal straight line that connects all regular specified midstep values is assignable [27 p. 24]. In a real ADC transfer function a straight fit line exists that fits the transfer characteristic according to a specified criteria. This thesis assumes a least-square fit straight line which best approximates the measured transfer characteristic according a least-square fitting algorithm. Other criteria are the end point definition or the minimum-maximum approach [27 pp. 24-25]. The determined values of the ADC gain and offset depend on the chosen criteria. The integral nonlinearity (INL) and the ADC gain and offset can be used as down-to-earth description of the ADCs transfer functions behavior. The integral nonlinearity of a gain and offset error free ADC can be defined according to equation (5).

$$INL[i] = \left(\frac{\left(U_c[i] - U_{c_{ideal}}[1]\right)}{\Delta U_{c_{ideal}}}\right) - (i-1) \tag{5}$$

$U_{c_{ideal}}[1]$ is the ideal transition voltage of the second ADC channel, $U_c[i]$ is the real transition voltage of code i, $\Delta U_{c_{ideal}}$ is the ideal input voltage difference between two code transitions and therefore represents the width of an ideal ADC code bin. The ideal input voltage difference $\Delta U_{c_{ideal}}$ can be calculated with equation (6)

$$\Delta U_{c_{ideal}} = \frac{U_{FR}}{2^N} \tag{6}$$

U_{FR} is the input voltage range of the ADC. N is the number of ADC bits. Between the DNL and the INL a relationship according to equation (7) exists.

$$INL[i] = \sum_{i=0}^{i} DNL[i] \tag{7}$$

The DNL is defined as the portion between the actual step width between two adjacent code transition levels and the associated ideal input voltage difference (equation (8)).

$$DNL[i] = \frac{(U_c[i+1] - U_c[i])}{\Delta U_{c_{ideal}}} - 1 \tag{8}$$

In equation (8) the probability assumptions of the model are hidden in the transition voltages. A direct DNL relation is obtained by equation (9).

$$dnl[i] = \left(\frac{n_{real}[i]}{n_{ideal}[i]}\right) - 1 \tag{9}$$

$n_{real}[i]$ is real number of counts which were determined at channel i, $n_{ideal}[i]$ is the expected number of counts at channel i. The advantage of this definition is its usability in a nonlinearity determination method which is suitable for high precision ADCs, the so-called histogram test. Equation (9) uses the fact, that between the transition code voltages and the histogram counts a relationship exists (equation (10)).

$$\frac{n_{real}[i]}{n_{ideal}[i]} = \frac{U_c[i+1] - U_c[i]}{\Delta U_{c_{ideal}}} \tag{10}$$

7.2 Classic histogram test

In a classic histogram test the ADC transfer function is determined by a statistical analysis of the measured codes. The test therefore is based on equation (9). If the ADC input signal is well-known, it is possible to deduce the expected count histogram *n_{ideal}*. A triangular waveform which slightly exceeds both ends of the range was often used as test signal in the advent of ADC characterization (Figure 60). A triangular waveform signal shows a uniform distribution. This means that all codes have equal probability of occurrence and as consequence every channel of an ideal ADC will show the same number of counts. The ADC digitizes the triangular input signal with the sample frequency F_S. A capture frequency ("F_S/K") given fraction of the digitized data is used to update the code hit value at the corresponding memory address. The so generated real count histogram n_{real} is than finally analyzed. In the suggested approach it is important that the used wave generator has a better nonlinearity behavior than the ADC under test. For nowadays high-resolution ADCs with a bit amount of 14bits or higher the pulser requirement is non-trivial. For instance, the waveform generator "SDG6022X" [29] sine wave signal has a minimum harmonic distortion of -65dB (~10.5bits). The "SDG6022X" is not a deliberated poor choice, actually its distortion performance is a good

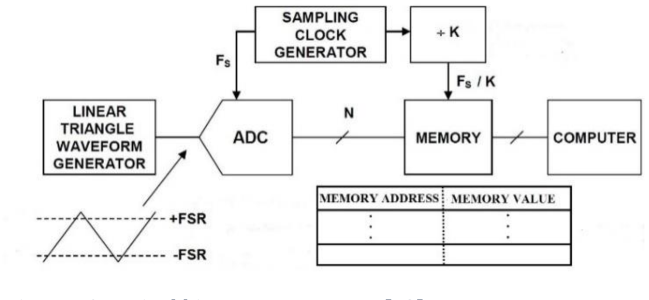


Figure 60 Typical histogram test setup [28].

representation of the nowadays arbitrary function generators. Note that using a simple wave generator setup is not the only solution, more enhanced linear histogram setups use integrated servo-loops to precisely control current sources. These current sources feed via an integrator circuit the ADC input. Obviously building such a setup is a non-trivial task too. For a couple of reasons major ADC manufactures guides (for instance [30]) and technical literature (for instance [27]) recommend the usage of sine wave generators. The authors mainly claim that sine wave generators are superior as it is possible to increase their signal quality by filtering. In sine-wave setup narrow-band AC filtering is possible and recommended. In theory this increases SNR by noise reduction and harmonics mitigation. In practice distortion attenuation by filtering has its limits, especially when the costs (and space) of the setup are also decisive. The officially specified distortion performance of the DS360 [31], an ultra-low distortion bench-top signal generator that is used for instance in audio development, is better than -100 dB whether the signal bandwidth in the operated setup is below 20 kHz. The DS360 also can be used in a classic INL determination setup. FADC cards with audio ADCs mostly have a well matching bandwidth and thus simply must relate to the DS360 to determine their INL trend. The FADC cards in this thesis have a bandwidth which is several magnitudes higher. Filtering is therefore still in a DS360 INL determination setup essential. Internet search for semiconductor basing sine wave oscillators suggests that a feasible, but still very ambitious, distortion limit is roughly in the range of -100...-110dB (17±1bits). The point is that every implemented circuit, whose performance might be sufficient, but is not gauged by the manufacturer, must be verified before it is used for INL measurements. Trusting circuit simulations is not an option, as in practical implementation various errors can occur. Verification would require a bandwidth matching ADC (or other eligible instrumentation) with higher resolution. For nowadays high-speed ADCs with a resolution of 16bits (or greater) in an extensive investigation no “classic histogram equivalent” publications were discovered that could reliably exclude a noticeable influence of the pulsers distortions on the determined INL trend. Hence the author of this thesis considers the manufactures general assumption “use a sine wave setup to determine the INL” as outdated. The assumption was trustful in times were the investigated high-speed ADCs only had a resolution of 12bits or less. For the nowadays high-speed 16bit ADCs the classic setup is not the promoted simple solution anymore. Therefore, it was decided that a “classic” approach is not a practical solution in this thesis though it still might be possible to implement. So, it was decided to use “non-classic histogram” procedures to determine the nonlinearity behavior of the FC2.0 ADC nominees.

7.3 INL determination in GERDA

The eligibility of classic histogram approach (chapter 7.2) depends on the available waveform generators and on the DAQ capabilities. The GERDA INL determination setup (Figure 61) offers an opportunity to tackle the issues and only requires instrumentation that is commonly available in an electronic workshop. The approach is basically using equation (5).

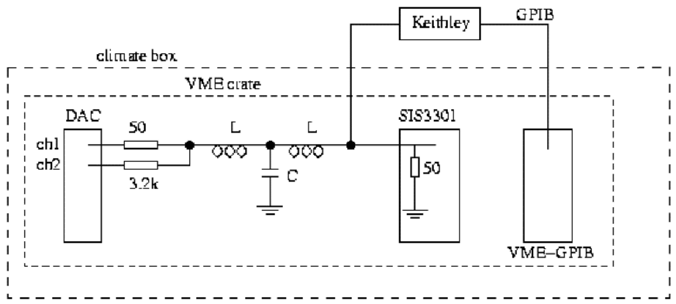


Figure 61 Setup for testing the GERDA FADC [32]. In this figure the GERDA 14b FADC is represented by the label "SIS3301". The "Keithley" is a DMM with 6.5 digits resolution (~20b). As DAC a default power supply was used. Every (used) power supply output channel has a resolution of 13b. The effective DAC fine resolution of ~19b is the result of the chosen resistances in the voltage "(pseudo) adder" circuit.

In the approach the FADC is connected to a remotely controllable "DAC" that outputs a desired DC voltage level. The issued voltage level is measured by the ADC under test ("SIS3301") and a concurrently operated high resolution DMM ("Keithley"). The analysis routine scans (and stores) the ADC output values and the DMM value. As soon as both measurements are completed, the analysis routine (via remote control) instructs the DAC to increment the DC level. The routine repeats this cycle until the desired ADC input voltage range is covered. After the measurement is completed, the analysis routine has two results vectors with each of a size that is equivalent to the total number of issued DC voltages. One result vector contains the stored ADC output values, the other vector has the stored DMM values. In both vectors the position relates to the issued DC voltage.

In the GERDA approach histograms are not used in the INL analysis. In the first analysis step the scalar response between the determined ADC output values and the DMM values is determined (done via linear regression). On basis of the returned fit parameter, the values of a corresponding straight fit line are computed. The nonlinearity error of every ADC output value is obtained by a subtraction of the corresponding straight fit line value. The total minimum error of the determined INL trend is given by the DMM error (systematic uncertainties + noise) and the noise level of the ADC output signal.

The lack of histograms is not a particular property of the GERDA INL determination approach. In HPGe-spectroscopy it is a common procedure to check the available FADC cards with special high-precision pulsers. An example, for such a high-precision pulser is the PB-5 from Berkeley Nucleonics Corporation (San Rafael, CA). The pulser is used in many HPGe-spectroscopy experiments and workshops to check the DAQ systems. As it has an 16bit resolution [33], its resolution is not sufficient anymore for the actual state-of-the-art ADCs in HPGe-spectroscopy that also have 16bits. But for FADC cards with a resolution less equal or less than 14b it is still a feasible option. An INL determination with the PB-5 pulser therefore could record the measured pulse amplitude (digitized signal) and the selected pulse amplitude. The latter would replace the DMM value of the GERDA approach. In pre investigations of this thesis this approach was elaborated with intensive efforts. In contrast to the "GERDA" approach the "PB-5" approach could enable an investigation of the dynamic nonlinearity behavior. Appendix chapters A.16-A.17 are designated to it.

7.4 INL determination in Majorana

The INL determination developed for the Majorana experiment is an enhanced histogram approach. In the classic histogram approaches (chapter 7.2) the “expected count histogram” trend is calculated on the supposed signal properties (shape, frequency, amplitude, …). In the Majorana approach the analysis routine tries to determine the “expected count histogram” trend from the measured traces. The developers of the approach claim that their suggested procedure might enable the usage of waveform generators that in common INL determination approaches are not eligible due to their poor linearity characteristics [8]. Another advantage of the proposal is its eligibility to determine dynamic nonlinear behavior. Their used setup is (slightly modified) illustrated in Figure 62. The setup uses two waveform generators, of which one is used to generate a slow sawtooth wave and the other is used to generate a fast triangle wave. The sawtooth signal has an amplitude that covers the complete range input of the ADC under test and a slow frequency of 0.1 Hz. The fast triangle signal covers a tenth part of the ADC FR and has a high frequency of 1333 Hz. Such a combined input signal setting is in principle also possible in the “classic” histogram test setup. The differences apply to the suggested sample acceptance and the expected count histogram determination method. In the Majorana method, the FADC is triggered by the start of the rising or falling part of the tringle. A trace of 2000 samples (20 µs) is read out. An example can be seen in Figure 63. The focus of the Majorana publication is on the INL correction of the used GRETINA FADC card and not on its suggested INL determination method. An elaborated prove and deduction of the limits of the suggested method is not included. The authors emphasize that the setup has to comply to following key assumptions and restrictions (all are inherited out of [8])):

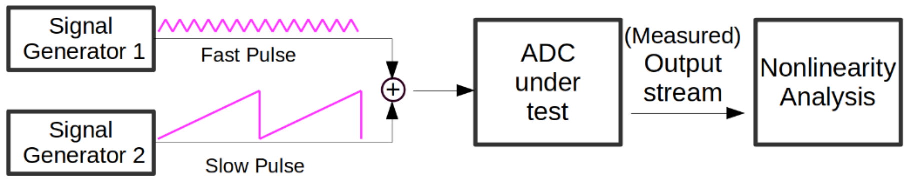


Figure 62 INL determination proposal of the Majorana Collaboration. The shown pulse shapes are used in the setup of [8]. In principle the method is also possible with pulses of different symmetry. Different symmetries will be used in the Majorana FC2.0 setup.

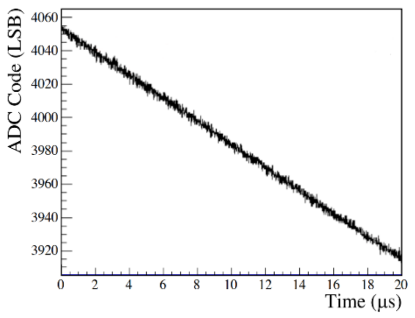


Figure 63 A typical measured waveform [8]. A trace roughly embraces 140 ADC codes.

(1) *The slope of the waveform is determined by the slope of the fast ramp, and its overall ADC offset is determined by the location along the slow ramp.*

(2) *… the bounds of the histogram are rounded by noise. To remove the latter effect, we discard the code counts for ADC values close to the boundaries.*

(3) As the slow ramp shifts the voltage offset of the fast ramp, each ADC code traverses varying locations along the fast ramp. As a result, the DNL we compute averages over any nonlinearities of the fast ramp, making our method insensitive to imperfections in the fast ramp linearity.

(4) The change in the slow ramp is much less than 1 ADC code (LSB) over the digitization window, and the computed DNL corrects for the measured frequency with which a particular ADC code is traversed, so that our method is also insensitive to any nonlinearities in the slow ramp.

(5) For each digitizer channel, we integrate the measured DNL curve, and fit and subtract away any overall slope to obtain the INL curve, ...

Points (1) and (2) must be considered to determine a "expected count histogram" and a "real count histogram". Points (3) ("all included fast pulser codes hit all ADC codes with the same magnitude") and (4) try to legitimate the implementation of non-perfect waveform generators in the setup. Point (5) ("slope gain fitting and correction") must be considered in all methods and is therefore not a particular feature of the Majorana method. In order to understand all details of the method, the author David Radford (ORNL, Oak Ridge-Tennessee) sent us his program for evaluation.

Important for the understanding of the Majorana method, are the chosen signal generator and DAQ trace record settings. The DAQ must be capable to capture traces of 2000 subsequent samples at a fixed pulse position (t_{ab} in Figure 64). Remember that point (4) does not exclude other eligible trace lengths, although the publication lists 2000 samples as trace length. The Majorana publication setup guarantees the fixed position by the implementation of the pulsers trigger signals. The used FC2.0 prototype DAQ setup has not enough ADC channels to also implement this procedure. The FC2.0 prototype setup therefore must internally trigger the fast pulses. A change of the fast pulsers triangle symmetry to a horizontally flipped sawtooth with an upstream baseline level signal part is sufficient (Figure 65). In the thesis this shape also will be called "triangle **pulse**" shape. Consequently, the FC 2.0 Majorana implementation is not capable to investigate the dynamic ADC nonlinearity behavior that is caused by rising signals. To prevent internal trigger issues in the FC 2.0 Majorana setup

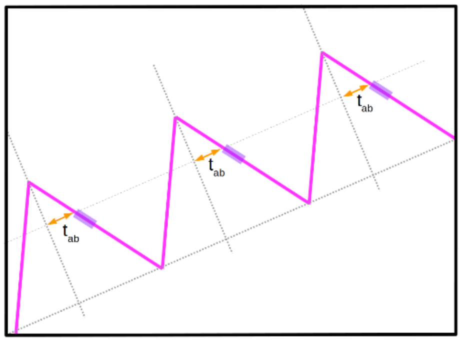


Figure 64 Fixing trace start position by t_{ab}

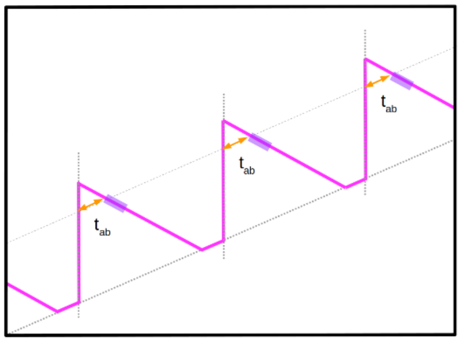


Figure 65 Meaning of t_{ab} in a triangle pulse

also the shape of the slow pulse is changed from a sawtooth to a triangular one. The slow shape signal level increase during the time duration of a trace is approximately 0.05/0.2 LSB (14b/16b ADC). This is much less than 1 LSB and therefore the Majorana Publication setup complies to point (4). A trace length that covers the complete incline of the fast pulse would violate point (4) as in this time range the signal level of the slow pulser rises about 0.6/2.4 LSB (14b/16b ADC).

DAQs have a limited bandwidth that they can handle. Therefore, a DAQ not necessarily can accept all fast pulses of the measurement run. Obviously, this influences the ADC code count distributions and hence the histogram generation processes must consider this issue. The Majorana procedure tries to tackle the issue by expected code count monitoring on basis of the DAQ accepted traces.

Every trace is analyzed in the same way. In the first step the algorithm obtains the trace-middle ADC code (of the current trace t) by equation (11).

$$mid_t = floor\left(\left(s\left[\frac{M}{2}-20\right]+s\left[\frac{M}{2}-10\right]+s\left[\frac{M}{2}\right]+s\left[\frac{M}{2}+10\right]+s\left[\frac{M}{2}+20\right]+2\right)/5\right) \quad (11)$$

M represents the trace length (2000 samples in the Majorana publication setup) and s[k] is the ADC code of sample (trace position indices) number k. In practice equation (11) uses ADC digitized samples, thus the result mid_t is interfered by the ADC nonlinearities. In the next step, the algorithm determines the ADC codes that in the current trace will be excepted for analysis (equation (12) and equation (13)).

$$bottom_t = mid_t - \left(\frac{Range}{2}\right) \quad (12)$$

$$top_t = mid_t + \left(\frac{Range}{2}\right) \quad (13)$$

Range is a (trace region) limiter that was introduced to consider point (2). The accurate value of the Range parameter depends on the chosen trace length and on the noise level of the setup. The value must be less than the actual ADC code range of the trace. Therefore the “Range” parameter value is given by the pulse amplitude, fast pulser frequency, the trace length and the noise. In the Majorana publication setup (14b ADC) a Range value of 70 is used, a similar setup with an ADC with higher resolution must increase its Range parameter accordingly. The $bottom_t$/top_t noise influence prevention approach might not work for $bottom_t$/top_t values that are close to extremum codes of the ADC. Such traces are immediately omitted and are not used to update the histograms. To ease the relations the following histogram update equations do not depict the restriction. The trace update process of histograms (also labeled as “histogramming”) is given by equations (14) and (15).

$$n_{ideal}[i] = n_{ideal}[i] + 1 \qquad \text{for } i \geq bottom_t \text{ and } i \leq top_t \quad (14)$$

$$n_{real}[s(k)] = n_{real}[s(k)] + 1 \qquad \text{for k } \epsilon \ \{k < M \text{ and } k \geq 0\} \text{ if } s(k) \geq bottom_t \text{ and } s(k) \leq top_t \quad (15)$$

i represent ADC codes and k symbolizes the sample position in the trace.

The trace recording (and the corresponding intermediated analysis) is done until the desired uncertainty level is achieved. It is recommended that the measurement time comprises an integer multiple of a slow pulser period. After the completion of the trace record, the final steps of the INL determination are done. The Majorana procedure obtains the INL trend by applying equations (16) – (17).

$$norm = \frac{\sum n_{ideal}[i]}{\sum n_{real}[i]} \tag{16}$$

$$DNL[i] = \frac{n_{real}[i] \cdot norm}{n_{ideal}[i]} - \qquad \text{for } i \geq 0 \text{ and } i < 2^N \tag{17}$$

The desired INL trend can be achieved by applying equation (7) (chapter 7.1). The INL trend of equation (7) relates not necessarily to a straight fit line. To satisfy the requirements of point (5) the INL trend must relate to a straight fit line.

7.5 FC1.0 INL determination nominee approach in FC2.0

The “FC1.0 INL” procedure is a MPIK in house development that was mainly dedicated to the FC1.0 DAQ system. The development of the initial “FC1.0 INL determination nominee” approach was initiated in a group meeting of Jim Hinton’s GERDA/LEGEND subdivision at MPIK. The successor “FC1.0 INL via DNL” approach was developed exclusively by the author of this thesis after the initial approach proved not to be eligible to investigate the ADC nominees of FC2.0. In this thesis both FC nonlinearity determination approaches are optimized for the conditions of the FC2.0 prototype DAQ.

One tasks in the FC1.0 development process was the development of an ADC nonlinearity determination method that can be used in an experimental environment and does not need high resolution pulsers. Instead of assuming a give shape of the pulser signal the proposed method tries to measure the exact shape with the FADC under test. The exactly known shape can then be used to measure the ADC nonlinearities. The setup is illustrated in Figure 66. Like the Majorana approach it uses two generators. The fast signal is a sawtooth pulse with a recommended amplitude height of a tenths of the ADC full range. The saw tooth signal is explicitly allowed to have nonlinear behavior. Note that the mentioned amplitude height is a recommendation and not a hard requirement. As the settings of sawtooth pulse stay unchanged during measurement, also potential nonlinear effects on the trend should not change. The slow signal is a DC signal whose value is controlled by the INL measurement routine. By changing the issued DC offset, the

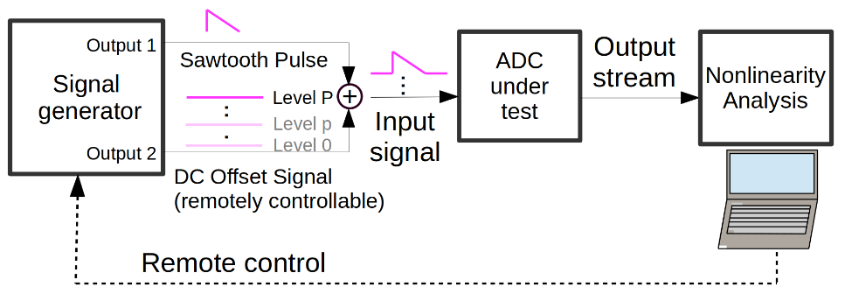


Figure 66 Setup INL determination nominee in FC2.0

measurement routine is capable to shift the sawtooth pulse. Through this procedure the fast pulse signal covers the full Range of the ADC and consequently is capable to cover all ADC codes. The sawtooth pulse was explicitly permitted to have nonlinearities – even much greater nonlinearites than the ADC under test owns are legit in the INL determination proposal. In the first step the accurate, non-linearities affected saw-tooth signal trend must be determined, otherwise a correct determination of the "expected count histogram" is not possible. To determine a sufficient good approximation of pulser issued saw-tooth trend, T. Kihm proposed following procedure.

At every DC level the digitized sawtooth pulse will be affected by the ADC nonlinearities. This means that the nonlinearities of digitized trace can be caused by the ADC or by the pulser. Note that the ADC nonlinearities at every offset level have a different influence on the digitized signal, but by the overall measurements (called merge(d) trace(s)) with different DC offsets, the ADC nonlinearity should average out. Remaining nonlinearities of the combined merged trace are expected to only originate from the pulser. This means that this merged trace should represent a good approximation of the accurate saw tooth pulse trend.

A perfect cancellation of all ADC nonlinearity effects in the merged trace would require several conditions that cannot be meet in a feasible way (limited by feasible maximum measurement time). It is vaguely assumed that the remaining ADC nonlinearity effects at some iteration point are much smaller than the real ADC nonlinearities and therefore just slightly influence the result of his method. A small (systematic) error is acceptable, as every feasible fast measurement will have uncertainties due to noise.

The INL determination starts after the accurate pulser shape determination ("merged trace"). The INL determination can be done on basis of the buffered data of the "merged trace" determination. As the data volume is significant, not every DAQ system may be able to buffer it. In this case the INL determination can also repeat the measurements at all previous DC levels to harvest the required trace data again. The "INL" determination step only differs from the "merged trace" determination step by its knowledge of the accurate pulser signal trend. The INL determination step needs the accurate pulser signal trend that it can calculate the expected trace (expected signal trend) at every DC level. The expected signal trend and the real signal trend then are used to update the count histograms. The elaborated analytical description of the merged trace determination and the INL determination step will be done in subchapter 7.5.1.

The procedure was extensively investigated and verified with the FC1.0 DAQ. A determined INL example trend is illustrated in Figure 67. Difference between the original FC1.0 setup and the suggested FC2.0 modification (Figure 66) merely considers the pulser. The FADC card of the FC1.0 DAQ is capable to precisely adjust the voltage level ("DC level") of the differential input signal lanes. In the real FC1.0 setup the waveform generator must only output the fast triangle pulse on output 1 and does not need an

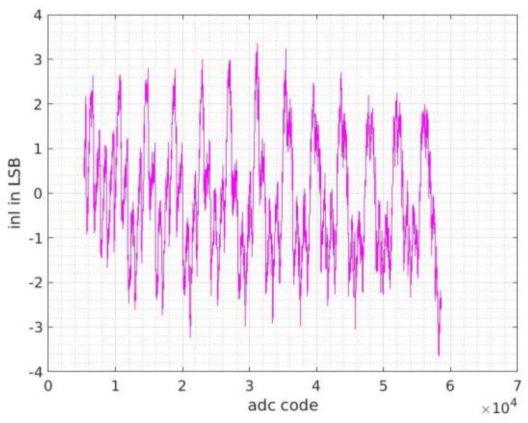


Figure 67 Example for a determined INL trend of interleaved (16bit) FC1.0-Germanium channel. The figure is a courtesy of T. Kihm (MPIK, Heidelberg).

additional pulser output ("output 2") to control the DC offset level of the final signal. The DC offset level control can be done much faster and more accurate by the FC1.0 FADC card itself.

The verification used the determined INL trends to improve the nonlinear behavior of separately measured pulser spectrums. As a linearity improvement by nonlinearity correction is only possible in cases at which the inputted INL trends describe the FADCs nonlinearity behavior, the success was considered as verification of the developed INL determination method. In order to dispel the last doubts, the measurements were repeated with a different waveform generator. The measurement series with the new pulser perfectly confirmed the outcome of the first measurement series with the initial waveform generator. A rough estimation of the achieved resolution gain (by INL correction) between the corresponding FC and Majorana publications results also favored the FC 1.0 procedure. Therefore, the developed modification of Figure 66 was considered as the INL determination approach that most likely will deliver the most precise results.

7.5.1 The FC1.0 INL determination

In the suggested approach a correct determination of the baseline level in all steps is essential. Why this is the case, will be revealed in later section. At this point it is sufficient to know that the required analysis information includes the baseline level. The baseline level must be extracted out of the recorded signal trend. The possible approaches for the supposed combined signal are summarized in Figure 68. The first approach, highlighted by "b1", is the most simple one. For a given signal region in front of the rising pulse edge, the mean value is determined. In Figure 68 the region is fringed by the labels "b_start" and "b_end". The mean value of the region relates to the baseline level value. The remaining influence of the noise depends on the region size. The second approach, highlighted by "b2", is more elaborated. It fits a straight line on the falling edge of the sawtooth function. The fit parameters enable an estimation of the signal maximum. When the accurate saw tooth peak amplitude is known, the signal maximum is sufficient to calculate the baseline level. All approaches use ADC recorded trace data and thus have the issue that their baseline level results are influenced by the ADC nonlinearity. In the simulations also the analog pulser signal and its accurate baseline level is available. In the simulations this baseline determination procedure is called "id" (acronym for "ideal"). In real life it may not be possible to determine a baseline value of equivalent precision.

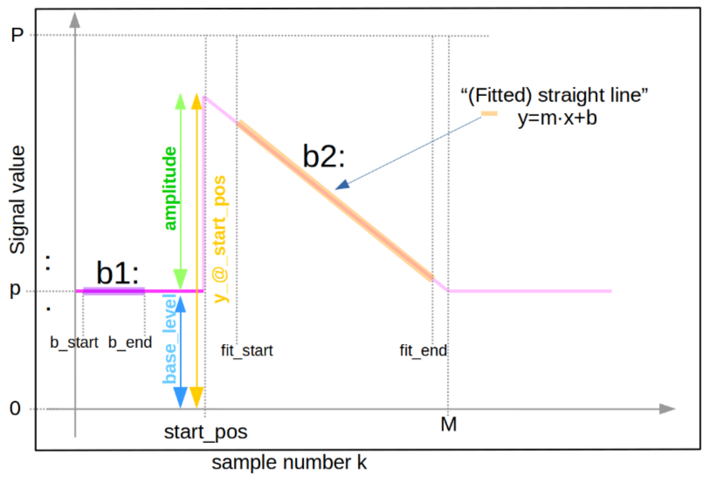


Figure 68 bX(p) baseline determination approaches. Instance illustration at baseline level p. The trend shows the triangle pulse (default pulse shape in all INL experimental setups of this thesis) and some of the possible relations(b1 a. b2) to determine the baseline level. The waveform (samples position given by the k indice) at position "start_pos" should have the rising edge of the triangle pulse. The waveform has a length of M samples.

A merge trace measurement that uniformly includes all possible (infinite) baseline levels (→"Majorana procedure"), is expected to offer the highest accuracy. In a real setup this is not possible as the available measurement time is limited and the DC offset resolution is confined. The measurement therefore starts at a specific start baseline level ("(DC) Level 0" in Figure 66), then uniformly increases the DC level value until it achieves the defined maximum baseline level value ("(DC) Level P" in Figure 66)[10]. During the whole measurement the fast triangle pulse signal (issued by "output 1" in Figure 64) is unchanged. The baseline start level must be selected sufficient high – a random noise fluctuation that includes a voltage level

[10] The explanation relates to Figure 66 where the ADC does not have a differential input and the signal combination is therefore done by an adder circuit. The real setup has an ADC with differential input and thus the DC signal must behave inverse to change the combined signal behavior identically to Figure 66.

of a saturated ADC code output should never occur in practice. The same is true for the selected maximum baseline level. As benchmark was decided solely to permit signal trend extremum values that are at least 20 noise RMS values away of the ADC saturation voltages. In case of the maximum baseline level ("Level P") the peak value represents the corresponding extremum value. The baseline level maximum must be restricted in a way that the recorded signal shape (in the trace) is not truncated (does not include saturated sample codes). This simplifies the following analysis steps a lot, as independent of the issued baseline level the same regions of the recorded traces can be used.

At every baseline level several traces are recorded to mitigate noise (uncertainty) effects. The mean trend of all recorded N traces at baseline level p is labeled as "average trace" (of baseline level p) in the scope of this thesis. Equation (18) describes the relationship between the recorded traces and the average trace.

$$average\_trace(k) = \left(\sum_{n=0}^{N-1} trace\_n(k)\right) \Big/ N \tag{18}$$

k is the sample position in the trace, M is the trace length (number of samples per trace).

The merge trace is the mean trend of all baseline corrected average traces. Consider that the average traces are also required in the INL determination step and consequently should be buffered. If it is not possible to buffer them, all average traces must be measured in the INL determination step again.

$$merge\_trace(k) = \left(\sum_{p=0}^{P} average\_trace(p,k) - bX(p)\right) \Big/ P \tag{19}$$

P is the number of scanned baseline levels, *bX(p)* is the determined "b1"/"b2" baseline level at baseline level *p*. In the ideal approach the merge trace would represent an accurate trend of the fast pulser signal. This would require a scan strategy in that all ADC nonlinearity shape deflections cancel each other out. Such a strategy is in principle only possible with a trace record strategy that can handle saturated sample codes or uses a dynamic saw tooth amplitude management. As in the latter the pulsers nonlinearity influence on the pulse shape changes, it is not really an option. In the INL determination step the now available merge trace enables a determination of the expected signal trend (equation (20)).

$$expected\_trace(p,k) = bX(p) + merge\_trace(k) \tag{20}$$

Explicitly note that the expected trace determination relies on the determined baseline level value *bX(p)* and therefore depends on the chosen baseline level determination approach ("b1" and "b2" (Figure 68)). In principle the available average trace and expected trace could be used to update the "expected count histogram" and the "real count histogram". But this is not the approach that the suggested procedure pursues. Instead of using an "real count histogram" it uses an "variation histogram". In the proposed approach the difference between the expected trace and the measured average trace is calculated. Every difference is considered as (unwanted) variation in which the sign matters. As every position in the difference trend can be assigned to a particular ADC code, it is possible to update the corresponding variation histogram element $var\_histo(i)$. Updating means that the determined difference ("variation") is added to the corresponding variation histogram element. The expected count histogram ("n_{ideal}") update has a procedure similar to the updating of the real count histogram in the classic approaches, but instead of using the bare sample values, it must use the expected trace values as histogram addresses (the histogram address was labeled as memory address in Figure 60). The described procedure can be summarized by equations (21) -(23).

$$diff\_trace(p,k) = expected\_trace(p,k) - average\_trace(p,k) \tag{21}$$

$$\begin{aligned} var\_histo\left(floor\big(expected\_trace(p,k)\big)\right) \\ = var\_histo\big(floor(expected\_trace(p,k))\big) + diff\_trace(p,k) \end{aligned} \tag{22}$$

$$n_{ideal}\left(floor\big(expected\_trace(p,k)\big)\right) = n_{ideal}\left(floor\big(expected\_trace(p,k)\big)\right) + 1 \tag{23}$$

$floor\big(expected\_trace(p,k)\big)$ obtains the histogram address whose element must be updated accordingly.

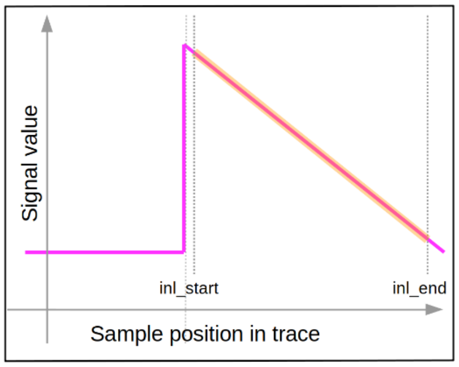


Figure 69 inl_start/inl_end parameter defination

The INL determination procedure only considers (trace) samples out of a well-defined region in the falling edge trend of the signal (Figure 69). These region limits are given by the parameters *inl_start* and *inl_end*. Such a fixed region definition is only feasible, when the rising edge of the saw tooth also has a constant position in the trace (that is provided by the DAQ). In contrast to this region limit, the merge trace equations comply to the complete trace. In principle it is also feasible to limit the merge trace determination on the same region[11]. Note that in the classic histogram approach the histogram update process needs not include a *floor* operation as it comprises only integer numbers. Using a *floor* operation can result in a precision loss. After the completion of the measurement, the proposal just needs to obtain the corresponding INL Trend (equation (24)).

$$INL[i] = \frac{error\_histo[i]}{n_{ideal}[i]} \tag{24}$$

[11] This is a potential opportunity to decrease the required operation number and ethernet transfer size. Everything that enables a decrease in the computation/data transfer time, is appreciated as it ultimately reduces the total measurement time. Reductions in measurement time also decreases potential temperature drift issues of a (practical) INL determination setup.

Again, the INL trend of equation (24) relates not necessarily to a straight fit line. To satisfy the requirements of point (5) the INL trend must relate to a straight fit line. None of the listed “FC1.0 INL” equations, with exception of the “b2” baseline determination approach”, is exclusively related to a triangle pulse signal. A not discussed benefit of the “FC1.0 INL” procedure is its capability to use all (periodical) output shapes of a real pulser. Pre-investigation measurements exploited this capability and for instance also used an exponential pulse shape. As the exponential pulse shape is identical to the intended detectors pulse shapes such a nonlinearity determination procedure could possibly offer a better reproduction of the experimentally observed behavior. The nonlinearity behavior of real ADCs is dynamic, this means it also depends on the signal shape.

7.5.2 The FC1.0 INL via DNL determination

Every resulting address of Equation (20) depends on the determined baseline value. As the determined baseline value is influenced by the ADC nonlinearities, a negative influence on the determined INL trend is possible. A non-intuitive opportunity to decrease the (possible) negative effects of the nonlinearity biased baseline value, was found by equation (25). Equation (25) transfers the “FC1.0 INL” method in a DNL determination method.

$$diff2\_trace(k) = diff\_trace(k-1) - diff\_trace(k) \tag{25}$$

$$\begin{aligned} &var2\_histo\left(floor\big(expected\_trace(p,k)\big)\right) \\ &\qquad = var2\_histo\big(floor(expected\_trace(p,k))\big) + diff2\_trace(p,k) \end{aligned} \tag{26}$$

Explicitly note that “var2_histo” update still involves the expected trace and consequently the ADC nonlinearity baseline bias still should be present.

The above listed equation (26) inherently assumes a triangle/sawtooth as fast pulse shape (signal shape with uniform hit probability)[12]. The original equation (22) does not require an assumption on the fast pulse shape. Experimental setups that based on this equation therefore could use an exponential pulse as fast shape. The relation between the variation2 histogram and the DNL is given by equation (27).

$$DNL[i] = \frac{var2\_histo[i]}{n_{ideal}[i] \cdot \Delta_{Pulse}} \tag{27}$$

$$INL[i] = \sum_{i=0}^{i} \frac{var2\_histo[i]}{n_{ideal}[i] \cdot \Delta_{Pulse}} \tag{28}$$

[12] In principle deviant shapes, like an exponential pulse, are also possible – in this case the declination correction (part of equation (27)) must be part of the “var2_histo updating” (equation (26)). This procedure is more challenging as the declination depends on the samples position. In the alternative procedure this also must be considered – assuming a mean declination (like it is done by the introduced Δ_{Pulse} parameter in equation (27)) is not sufficient anymore.

Δ_{Pulse} represents the declination of the (fast) saw tooth. It is assumed that the accumulated sum of equation (27) has a better “averaging out” performance of the baseline ADC nonlinearity bias than the unaccumulated INL procedure of the initial FC1.0 method. The prove of this assumption is still outstanding.

7.6 INL correction in HPGe-spectroscopy

In cases the ADC nonlinearities significantly change the energy linearity of the determined energy spectrums, a linearity improvement (of the energy spectrum) can be achieved by applying ADC nonlinearity corrections. In principle the correction can be applied on the unshaped raw sample stream (“V1” in Figure 70) or on the determined energy value of an event (“V2” in Figure 70). To understand latter an understanding of the signal shaping is required. The optimal shaping of energy signals in HPGe-spectroscopy is not investigated in this thesis. An investigation about optimal shaping strategies is for instance done in [3] and [34]. Typical HPGe-detector signal processing chains determine the γ-ray energy on basis of the pulse amplitude height of the shaped signal. HPGe-detector signal processing chains can include a pole-zero compensation unit and an upstream baseline subtraction (“baseline restoration”) unit. The subsequent shaper unit (in Figure 70 a trapezoidal shaper) would otherwise show unfavorable undershot/overshoot behavior. A pile up-rejection unit is a possible approach to circumvent the disadvantageous effects of pulse interference at heigh event rates. In Figure 70 is it not implemented as the example setup is only intended for experiments with a low event rate and in such cases the pulse interferences are unlikely. The “Max Search” unit extracts the amplitude height out of a defined pulse region via an extremum search. The determined energy value than can be used to update the spectrum (“histogram”). Readers that are not familiar with the topic can find additional information about common shapers in the appendix. A slightly more elaborated signal processing chain will be used in chapter 8.6.

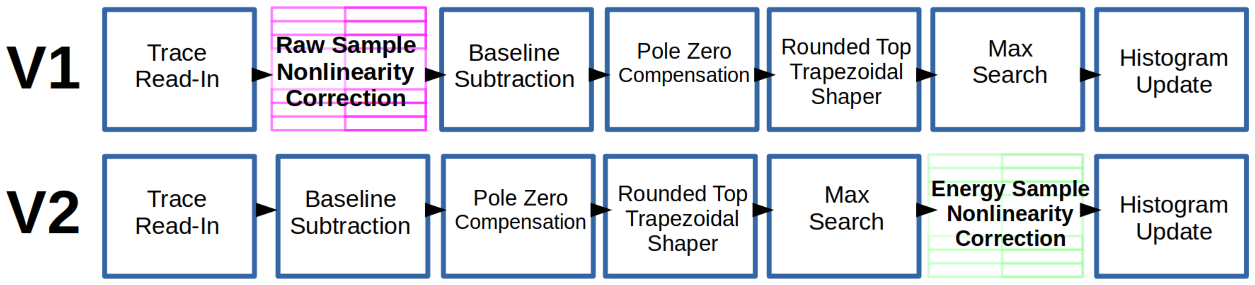


Figure 70 Typical HPGe-detector signal processing chain

In the former approach “V1” the correction must be applied on all samples[13] of the trace, in the latter only the determined energies basis parameter (baseline level and determined pulse height of energy signal) must be corrected. The “V1” raw data sample procedure is described by equation (29).

[13] A more precise description of the “V1” procedure would state: “all samples that have influenced on the peak height of the shaped pulse must be corrected” – this are not necessarily all samples of the trace (remember the rough

$$s_{corrected} = s_{uncorrected} - INL_s(s_{uncorrected}) \tag{29}$$

s is the ADC code value of a raw sample., $INL_s(s_{uncorrected})$ is the determined INL value of the corresponding ADC code. Consider that the uncorrected raw sample must be an integer value. The corrected sample value can be a real number. This hinders a firmware adaption in FC2.0 as the signal processing of FC "pulse detection block" is based on integer numbers. A future relocation of the task in firmware could be appealing as a significant amount of CPU time can be saved. As such an implementation would consider several blocks of the FC2.0 prototype firmware (and the read-out efficiency) a relocation into firmware cannot be recommended as general solution. The "V2" energy signal procedure is described by equation (30) – equation (32). The energy signal correction must also consider the baseline level as the baseline level during measurement may vary.

$$E_{Pulse_{Cor}} = E_{Pulse_{uncor}} - INL_E\left(E_{Pulse_{uncor}}, g\right) \tag{30}$$

$$E_{Basis_{Cor}} = E_{Basis_{uncor}} - INL_E\left(E_{Basis_{uncor}}, g\right) \tag{31}$$

$$E_{Cor} = E_{Pulse_{Cor}} - E_{Basis_{Cor}} \tag{32}$$

In the "V2" procedure $E_{Pulse_{uncor}}$ is the returned "energy" signal value of the "Max Search" function, g is the gain between the unshaped (raw sample) ADC signal and the energy signal (input signal of the "Max Search" function). The value of $E_{Pulse_{uncor}}$ is a result of shaping operations and therefore influenced by several ADC codes. The approach may not be eligible to improve the spectrums (energy) nonlinearity for all possible ADC nonlinearity trends.

The shaping can improve the resolution far beyond the given ADC data width and thus the effective data width of the energy signal is not equivalent to data width of the unshaped ADC raw data stream anymore. The energy signal value can be an intermediate value that is somewhere between the value of two adjunct ADC codes. As the nonlinearity trend only is available in discrete ADC code steps, this inevitable enforces the usage of interpolation operations to calculate the corrected value. The interpolation error can be reduced by decreasing the step size and accordingly increasing the extend of the available INL table (input array in interpolation)[14]. Such a correction requires a INL trend table of accordingly adapted size. The transfer of the ADC nonlinearity trend in an energy signal nonlinearity trend furthermore must consider the effective gain of used shaping. An ADC correction method that uses interpolation operations has a poorer

"2..3×shaping time" rule. As the accurate determination of this range is also challenging, in the "V1" approach of the thesis all samples of a trace are corrected

[14] In the "GERDA" approach the INL trend table size depends on the DC level number that was measured. In this procedure an extend of the INL trend table (and hence the possible correction quality) "only" requires a higher number of measurements. In the Majorana approach the INL trend table size is limited by the given ADC bit width.

performance than an equivalent correction method that does not need interpolation (or a mathematical similar operation) and uses the same input INL trend.

The ”raw sample” correction procedure has the advantage that the raw sample codes are identical to the ADC’s codes and thus an INL trend of the corresponding ADCs only must comprise all integer (code) values of the ADC. If the INL trend tables ADC code column relates to integer numbers, the samples INL correction without an interpolation is possible. If not, as in the case of the resulting GERDA INL trend table, an interpolation operation is also required in the “raw sample correction” procedure.

8 ADC nonlinearity investigation setups and their results

The suggested “Majorana” method and the “FC 1.0” procedures differ significantly from default setups. In their derivation some key limitations were motivated but never precisely proven. If they prove incorrect, some of these key limitations can even affect a total failure of the method. Though in the corresponding publications no obvious error could be found, the situation discomforted the author of the thesis. Further contemplation about the methods even aggravated the concerns. A technique to prove or disprove the suggested INL determination procedures was required. Practical setup verifications are critical as the real INL trend of the ADC under test is mostly unknown and therefore a simple determination of the INL trend of an ADC does not permit any conclusion. Is it a bad ADC, is it defected by the pulsers nonlinearities, is the determined INL trend the consequence of another FADC card flaw or is it caused by a misbehavior of the chosen procedure? Mostly this is hard to conclude. A possible quality benchmark for determined INL trends is the improvement of the energy linearity of a measured γ-ray spectrum after INL correction. Another approach is the adaption of “INL correction” approach on a setup with a perfect high resolution pulser. These verification approaches are capable to determine a quality priority of the methods for a given setup, but they do not give an insight about systematic errors (of the INL determination procedure) and obviously potential setup misbehavior cannot be distinguished by misbehavior of the ADC. This is critical, as for instance the misbehavior of FADC input amplifiers can depend on the electrical properties of the source and therefore depends on the chosen setup. To estimate the reliability of the procedures therefore was decided to investigate their behavior extensively by simulations.

Chapter 8.1 - 8.3 is dedicated to the INL determination simulations. In chapter 7 the influence of noise and the pulsers nonlinearities was already emphasized. A quantitative discussion of this influence parameters was not done. Therefore, it was decided to develop simulation routines that enable a precise analysis of the suggested INL determination methods. All INL simulation and result analysis tasks were done in MATLAB R2021b (MathWorks, Natick-Massachusetts).

Chapter 8.4 - 8.7 is dedicated to the practical implementation and verification of the suggest new methods. As verification procedure the nonlinearity correction of a HPGe-detector γ-ray spectrum was chosen. The verification included a real-life HPGe-detector measurement campaign with the new FC2.0 prototype DAQ at the TUM physic facility. A low distortion pulser of sufficient resolution was not available and hence the pulser verification approach could not be used to verify the determined INL trends. As additional verification a comparison with the INL trend result of the GERDA method was done. Chapter 8.8 is a concluding summary of the INL determination results.

8.1 Simulation setups and procedure

Before discussing the simulation setup, it is useful to remember the purpose of the simulations. First of all, the proposed algorithms should be verified. Furthermore, the simulations are intended to scrutinize the effects of noise and pulser nonlinearities on the methods. The method that returns the given INL trend most accurately, is a reasonable nominee for the feature implementation. The important key words in the last sentence are "given INL Trend". A real ADCs behavior is not independent of trend history of its input signal ("memoryless"), this means that the determined INL value of a real ADC can be influenced by the signal properties. A useful assessment of the determined INL trend (by method X) with such a real ADC would require a preceding determination of the expected INL trend with such a signal. In principle this is possible, but it requires complex models and does not offer any advantage. If a method works with real non-memoryless ADCs, a method also works with memoryless ADCs and vice versa. Hence it is sufficient to show that INL determination method X is eligible for memoryless ADCs. In a memoryless ADC model every derivation between the determined and the given INL trend appoint to an issue in the applied method.

8.1.1 Nonlinearities in the simulation

The Majorana INL determination and the INL determination nominee in FC2.0 have in common that their used test signal consists out of a fast pulse and a slow signal combination. In "Majorana setup" Figure 62 and "FC1.0 setup" Figure 66 the ADC input signal was made by a differential input "junction" of two waveform generator. Depending on the chosen method, one nonlinearity source may cause more harm than the other. To probe the signal source effects, both INL determination methods were simulated with different generator settings. The feasible ones are illustrated in Figure 71. Each output signal of such signal generator

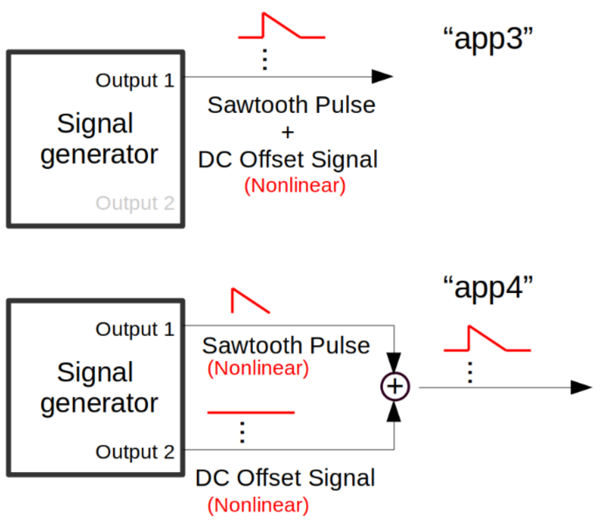


Figure 71 Simulation setup nonlinearity probing . A not included simulation setup is the perfect pulser setup (called "app0") where both outputs of the signal generator show an ideal linear behavior.

setups could be generated by its own DAC. In the "app4" setup (bottom) both pulsers have a nonlinear behavior. The upper "app3" setup in that both signal parts are generated by a single DAQ in the signal generator looks similar, but in fact it violates the Majorana Approach and the FC 2.0 nominee strategy. The effects of this violation will be elaborated in the corresponding simulations. A frequently used simulation setup that is not include in Figure 71 is the perfect ideal setup in that both outputs have a perfectly linear signal. In the simulation results chapter such a setup will be labeled as "app0" or as perfect pulser setup.

8.2 General simulation procedure

Generating a nonlinear ADC model can be done via an internal transition voltage table. This procedure reflects equation (9) and (10). In ADC models this approach is not an issue as the underflow/overflow transition voltage are given by the defined real Full Range of the simulated ADC. The chosen road to determine the ADCs model internal transition voltages is therefore

1 Determining a real count histogram (of the simulated ADC) for a perfectly uniformly ("hit") distributed input signal. The generation should be done in a way that the determined real count histogram demonstrates real ADC characteristics.
2 Determining the corresponding expected count histogram
3 Resolving the corresponding DNL trend with equation (9)
4 Using the relationship between DNL and the transition voltages (equation (10)) to determine ADC input voltage/output code interpolation look-up table. The look-up table essentially represent the nonlinear ADC model. In MATLAB the function *interp1* has the demanded behavior.

The pulser nonlinearities can be defined in a similar way. The only significant distinction is the used interpolation method for the look-up table readout. The INL determination methods under scrutiny are not identical, but their simulation processes have the following steps in common.

1) The desired nonlinear models are loaded and passed to the corresponding ADC/pulser model functions.
2) On basis of the recorded trace data and the corresponding intermediate analysis results, the routine determines the INL trend of the method under investigation. The determination of an INL Trend

includes the gain and offset compensation ("straight line fit"). By comparing determined and given INL trend an error assessment of the INL determination approach can be done. Comparing the given against the determined INL trend is just useful when both are straight line fit corrected and both fits comprise the same ADC code region. Remember that not all INL determination approaches are capable to include all ADC codes and consequently their straight-line fit cannot include all ADC codes.

3) To assess the error of the method the differences between the given and the determined INL trend are investigated. As several setups are under scrutiny, the error assessment must be automated. Concerning this matter, the error terms "mean INL error" and "mean DNL error" will be defined.

Figure 72 and Figure 73 show possible difficulties of the last step of the error analysis. Figure 72 depicts an example for a given INL trend (red line) and a determined INL trend (magenta line). Over a larger ADC code range both histograms seem to agree very well. The inset (in Figure 72) shows the two distributions for a small range of ADC codes and reveals a relative shift. The difference between the given and the determined INL trend of Figure 72 is delineated as red line in Figure 73.

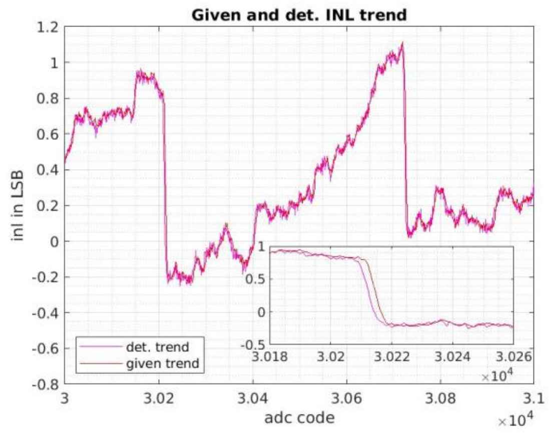


Figure 72 Example of a given and determined INL trend

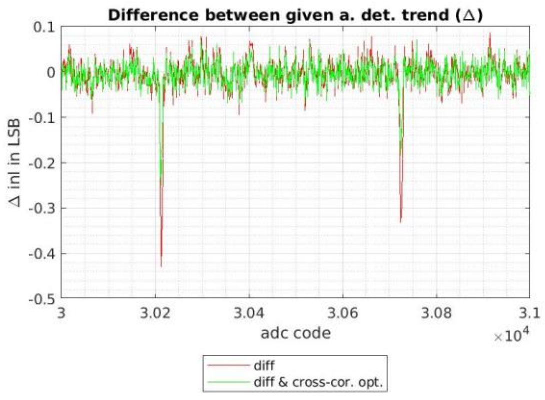


Figure 73 Difference between given and determined INL trend

Although the (red marked) difference plot in Figure 73 is properly calculated, it does not consider the discovered shift ("delay") between the given and determined INL trend (in Figure 72). To quantify the quality of a method such shifts should be taken into account. An opportunity to determine the constant shift is given by the cross-correlation operation. MATLAB offers for this purpose the *finddelay* function. The difference trend after the shift correction with *finddelay* is indicated by the green line in Figure 73.

The difference trace is obtained by equation (33). In case of the "cross-correlation optimization" approach (green line in Figure 73) the determined INL trend in equation (33) is the already delay compensated INL trend.

$$diff(code_x) = inl\_given(code_x) - inl\_\mathrm{det}\,(code_x) \tag{33}$$

$$mean\_inl\_err = \left(\sum_{code_x = slf\_start\_code}^{code_x = slf\_end\_code} abs\big(diff(code_x)\big) \right) \Big/ 2^N \tag{34}$$

$$mean\_dnl\_err = \left(\sum_{code_x = slf\_start\_code+1}^{code_x = slf\_end\_code} abs(inl\_given(code_x) - inl\_given(code_{x-1}) - (inl\_det(code_x) - inl\_\mathrm{det}\,(code_{x-1}))) \right) \Big/ (2^N - 1) \tag{35}$$

slf_end_code is the top code that limits the straight-line fit region, slf_end_code is the bottom code that limits the straight line fit region.

8.3 Simulation results

Although the FC1.0 and Majorana approach only use essential mathematical methods their quality and sensitivity to negative effects are hard to estimate. In the beginning the simulation focus was only on the "FC1.0 INL" method. The findings of this simulations are condensed in chapter 8.3.1. In the Majorana simulation chapter 8.3.2 the investigation is limited to a setting that agrees with David Radfords recommendations (chapter 7.4) and a proposed "acceleration" setting. A final method compare by simulations is done in chapter 8.3.3.

8.3.1 FC1.0 results

Initially many key aspects of the proposed procedure were not understood. For instance, it was unclear

- Which triangle pulse amplitude height is best suited?
- Which baseline determination (b1, b2) method is superior?
- Does the ADC nonlinearity bias of the baseline significantly harm the "FC1.0 INL" method?
- How does noise influence the determined INL trend?

Reasons why the ADC nonlinearities deteriorate the merge trace trend were already listed in chapter 7.5. As the used rationales do not enable an educated guess on the scale of the deteriorations, the simulations of this chapter were necessary. The first simulations proved that the initially intended "FC1.0 INL" method has its lacks that prevent a general implementation in ADC investigation setups. At this time the Majorana publication was not released and consequently it was not possible to implement its approach as alternative method. The development of enhanced method was necessary. By this motivation the elaboration of the "FC1.0 INL via DNL" method was initiated. Figure 74 – Figure 77 shows its results for a simulation that used the settings of Table 15. The simulation result does not enable a certain assumption about the quality of the procedure. The determined INL or DNL errors will depend on the INL input pattern. This means that the error might be small for some patterns but large for other patterns with similar INL magnitude.

Table 15 Simulation settings of Figure 74 – Figure 77

Pulser Nonlinearity Model	App4
ADC Model	ADC Model 1 (red marked “given INL trend”, for instance depicted in Figure 74)
Frequency fast pulser	~3000Hz (only in simulation)
Noise Level in LSB RMS	1.2
(Max) Variation of the pulse position in T_S	1.0
Baseline start level (in LSB)	200 (exactly)
Baseline end level (in LSB)	58850
Total amount of histogram absorbed traces	~84k
trace length *M*	65536
Triangle pulse amplitudes (in LSB)	6144
offset_inc (in LSB)	50
rising edge position (sample in trace)	8192
inl_start (trace position)	8592
inl_end (trace position)	65536
b_start (trace position)	1
b_end (trace position)	6000
Baseline approaches:	B1
Straight line fit start code (“slf_start_code”)	700
Straight line fit end code (“slf_end_code”)	64700

Pay heed that in a real setup the pulsers DAC clock phase may drift randomly from trace to trace. This phase variation affects in the sampling system an additional sampling error. In the “FC1.0” simulation it is assumed that this phase variation is not bigger as $2\times0.5\cdot T_S$ (in Table 15 the property is listed as “Variation of the pulse position”). The restriction to 1 T_S is motivated by the assumption that the simulated DAQ has a perfect pulse detection and hence would compensate higher shifts. In the simulation the phase variation can be implemented by a linear interpolation. In this procedure the phase can differ from trace to trace, but during the interval of a trace the DACs clock phase is constant.

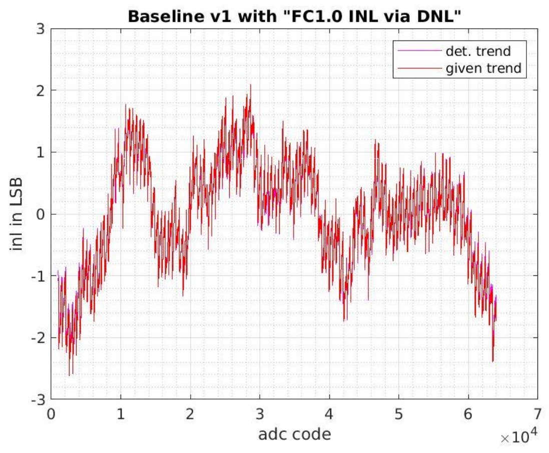


Figure 74 Simulation result of "b1 FC1.0 INL via DNL" method (with a pulse amplitude of 6144 LSB and a (DC) offset increment size of 50 LSB). The determined INL trend (magenta line (hardly visible as covered by the given INL trend)) is almost identical to the given INL trend (red line) of "ADC Model 1". A zoom-in plot (Figure 75) indicates that though the high coverage both trends differ significantly.

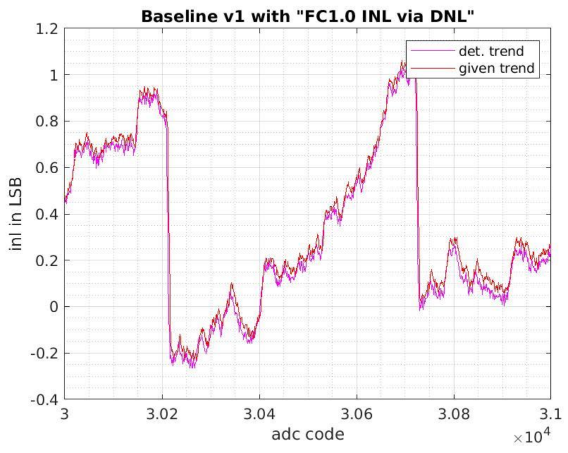


Figure 75 Simulation result of "b1 FC1.0 INL via DNL" (zoomin) (with a pulse amplitude of 6144 LSB and a (DC) offset increment size of 50 LSB). In the beginning of the simulation campaign, it was arbitrarily decided that all automatically generated zoom-in plots will cover the ADC code range from 30k to 31k. As the error behavior also depends on the code region, the trend of the zoom-in plots is biased. An unbiased error is obtained by the determination of the "mean INL error" (equation (34)). This is also automatically done by the MATLAB analysis routine.

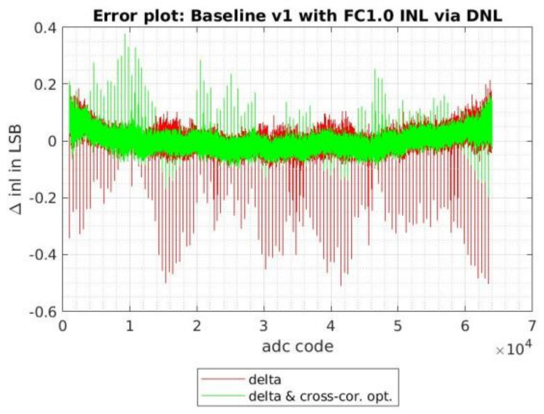


Figure 76 Error trend of the "b1 FC1.0 INL via DNL" simulation. The red colored trend is the difference between the given and determined INL trend of Figure 74. The green colored trend is the cross-correlation optimized difference trend (chapter 8.2).

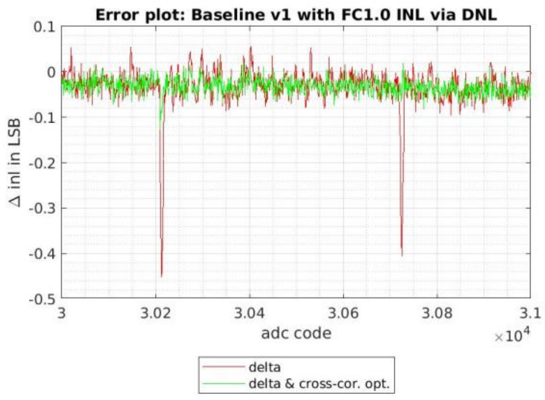


Figure 77 Zoom-in error trend of the "b1 FC1.0 INL via DNL" simulation

The red marked error trend of Figure 76 reveals that the "unoptimized" determined INL trend of the method struggles with the accurate reproduction of abrupt INL trend changes. The optimized trend (green line) is capable to reduce the issue but cannot completely abandon it. Notice that the used optimization step is not useable in practice as in real setups the real INL trend of the implemented ADC is unknown. Another detail considers the determined error behavior of the ADC codes that are part of the measurements at the initial and the final DC offset level. In the case of this simulation this roughly considers the code regions 200-6350 and 58850-65000 (Table 15). In the following parts of the thesis these regions will be called “entrance”/”exit” region. In the “entrance”/”exit” region the determined INL trend has a significant higher error as in the remaining ADC code regions of the plot. In Figure 76 this is solely indicated by the bigger variations of the error trends, but a detailed scrutiny also would reveal higher systematic anomalies. This can be explained by the poorer statistics of the code regions and by defects in the merge trace. In the merge trace building it is not possible to cover all merge trace positions uniformly with the ADC codes of the “Entrance”/”Exit” region.

A noticeable fact of all realized simulations is the de-facto failure of the “bX FC1.0 INL” approaches (Figure 78). As in the “FC1.0 INL” procedures the samples noise is distributed over all included codes, the approaches should offer the smallest noise uncertainty and consequently also have the lowest INL errors (detailed numbers will be given in chapter 8.3.3). In the simulations this premise was only valid for the “id FC1.0 INL” approach that is hard to transfer in a real setup. The “id FC1.0 INL” approach uses an unbiased baseline value, therefore the suspicion arose that the poor performance of the “bX FC1.0 INL” approaches was mainly caused by their ADC nonlinearity biased baseline value. Due to this reason the “FC1.0 INL smart” approach was developed. This approach requires buffer space for all recorded average traces. As the “FC1.0 INL” approaches base on the same input data like the “FC1.0 INL via DNL” approaches, it is possible to calculate in a pre-step the supposed INL trend with the suggested “bX FC1.0 INL via DNL” procedure. This pre INL trend has an error that is much higher than the INL trend of the “id FC1.0 INL” procedure. But obviously a baseline value that is corrected by its rough nonlinearities contributions is closer to reality as an untouched biased baseline values and therefore harms less its resulting “FC1.0 INL” trend. In the proposed “FC1.0 INL smart” approach the final INL determination is done like in all “FC1.0 INL” styles but in contrast to default procedure, the smart approach uses a nonlinearity corrected baseline value to determine the expected trend signal. The result INL trend of the smart approach is represented in Figure 79, the INL trend of the default “FC1.0 INL” procedure (without baseline correction) is presented in Figure 78. Both simulations were done with the settings of Table 15 and only differ in the analysis procedure.

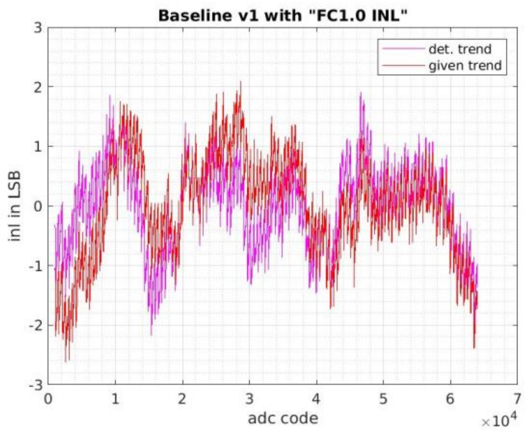


Figure 78 Simulation result of the "b1 FC1.0 INL" approach. The result was obtained by exactly using the same pulser data output stream like in Figure 74.

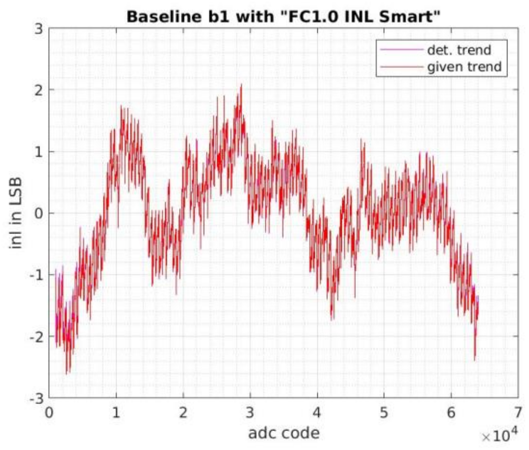


Figure 79 Simulation result of the "b1 FC1.0 INL smart" approach. The result was obtained by exactly using the same pulser data output stream like in Figure 74.

The smart approach is clearly better than the default "bX FC1.0 INL" procedure. Using equation (34) to its results would frequently return a better error value than the "bX FC1.0 INL via DNL" approaches, but its error is still higher than the error of the "id FC1.0 INL" method. Figure 80 shows a zoom in of the determined INL trends. The "b1 (FC1.0) INL smart" trend is similar to the trend of "id (FC1.0) INL", they differ mostly by the offset between them. In simulations it is easy to show that the scale of the offset depends on the remaining nonlinear bias of the baseline value that is used to determine the expected trace (expected signal) trend.

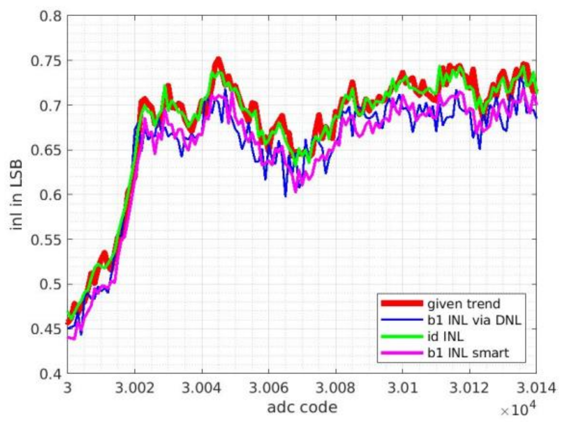


Figure 80 Local magnification into the plots of Figure 74, Figure 78 and Figure 79.

The previous discussion of the "FC1.0 INL smart" approach was fixed to the "b1" baseline determination procedure. In the "b1" approach a limited ADC code subset determines the resulting baseline level value. This is dangerous as samples whose nonlinearity is corrected by the INL trend of the "FC1.0 INL via DNL" approach still have a significant random nonlinearity error. A modification of the suggested "FC1.0 INL smart" approach on the "b2" baseline determination procedure therefore can be useful. As it determines the baseline level on a large ADC code subset, the (slight) random nonlinearity errors of the "FC1.0 INL via DNL" approach can cancel each other out in their impact. The MATLAB codes that correspond to the "bX FC1.0 INL smart" approaches are available in the attached source codes. The simulation results (Figure 74-Figure 77) points up that the "b1 FC1.0 INL via DNL" approach is capable to reproduce the INL trend of the ADC under elaboration, but the determined DNL behavior is charged with a significant error share. In several points of this thesis, it was already supposed that the noise level has influence of the accurateness of the determined DNL trend. To elaborate the noise influence precisely, a simulation series with the settings of Table 19 was proceeded. The results are shown in Figure 81.

Table 16 Settings of the noise evaluation simulation series

Pulser Nonlinearity Model	Perfect Pulser ("app0")
Method	"id FC1.0 INL", "b2 FC1.0 INL via DNL"
Noise Level in LSB RMS	$1.2 \cdot \left(\sqrt[2]{2}\right)^N N \epsilon \{0,1,2,3,4,5\}$
Total amount of histogram absorbed traces	$60000 \cdot 2^N N \epsilon \{0,1,2,3,4,5\}$
Amplitudes $A_{sawtooth}$(in LSB)	8192

It is important to note that the simulation conditions were selected in a way that all simulation runs (corresponding to Table 16) have the same statistical noise uncertainty. A statistical investigation, that covers all introduced methods, it will be seen that the determined INL trend results of the "FC1.0 INL via DNL" approaches still have a significant remaining uncertainty due to remaining noise uncertainties. As it is the intention of the simulation series to elaborate the systematic DNL error, Figure 81 illustrates only the results of the "id FC1.0 INL" approach whose remaining uncertainty magnitude is not significantly influencing the plotted trends. The plots derivations between the given INL trends (red line) and the determined INL trend (magenta line) must therefore represent a systematic smoothing of the determined DNL properties.

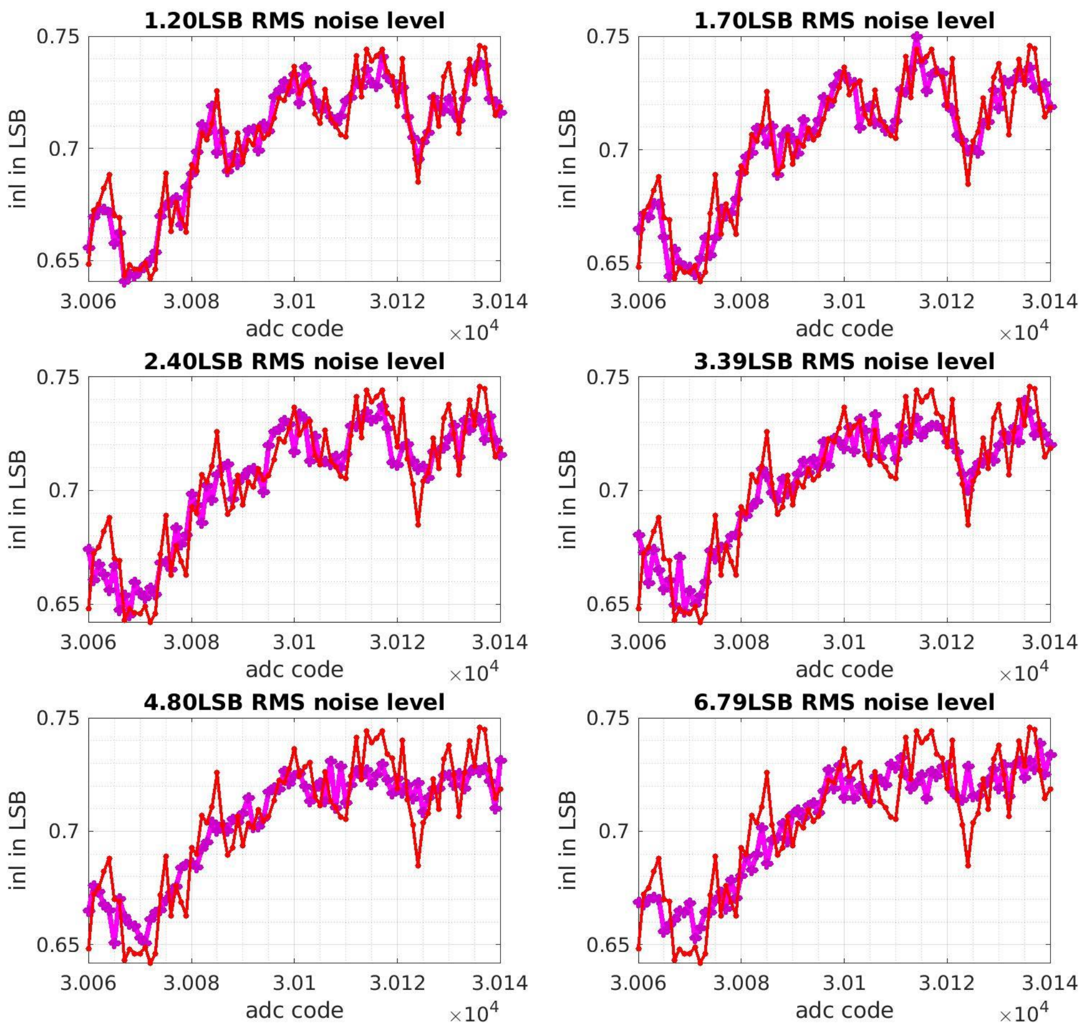


Figure 81 Noise influence on nonlinearity (I). With increasing noise level (noise level is indicated in the titles) an average sample is the mean result of a wider ADC code range. (In cases of larger noise) The average trace sample value is affected by a greater number of ADC nonlinearities and thus its nonlinearity reproduction is getting a stronger smoothing ("moving average filter with a bigger window size"). As consequence the herewith determined INL trend also should be "smoother" and therefore its DNL trend too. This behavior can be seen in the subplots of this figure.

In the above Figure 81 the noise smoothing of the "DNL trend" can be seen. Noise attenuates the DNL swings and consequently noise also has feedback on the error of the determined INL trend. The noise effects

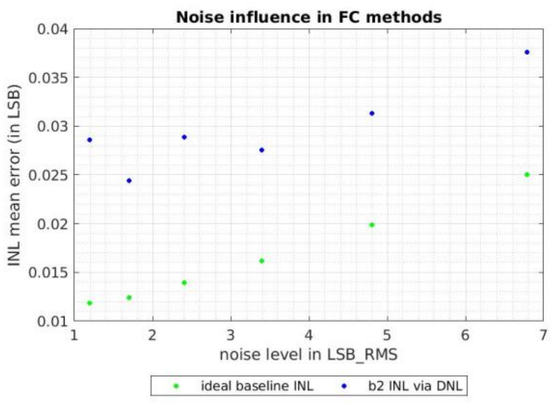
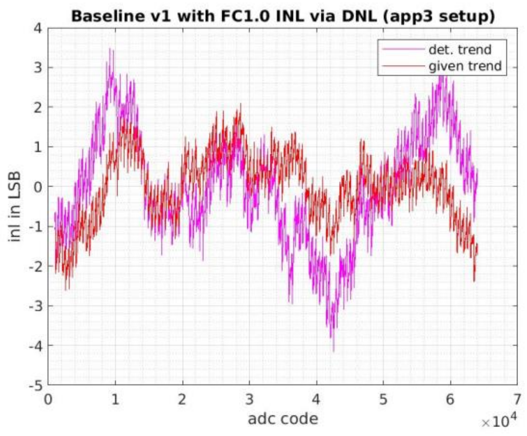


Figure 82 Summary noise influence on determined nonlinearity

on the mean INL error are depicted in Figure 82. In case of the “b2 FC1.0 INL via DNL” approach, the noise interference is engulfed by remaining noise uncertainty and other systematic issues of the method as long as the noise level is below or close to 3LSB_RMS.

In addition to noise, the outcome of an INL measurement can be influenced by the nonlinear behavior of the pulser setup. Except the already investigated "app4” setup, the single output “app3” setup (Figure 71) would be a feasible option. The INL trend result of the “FC1.0 INL via DNL” procedure with such an “app3” setup is illustrated in Figure 83. The used settings are equivalent to previous simulation settings. Remember that the “app3” setup is not capable to determine a merge trace trend that reproduces



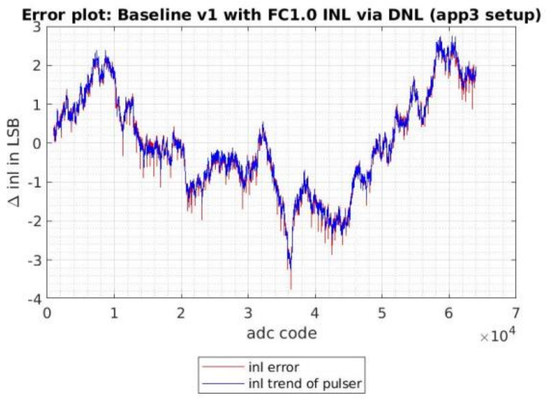


Figure 83 Simulation result of the “b1 FC1.0 INL via DNL” procedure with an “app3” compatible pulser setup. In the “app3” setup (Figure 71) only a single output of a single pulser is used.

Figure 84 Error plots of the “app3” setup simulation (Figure 83). The determined error (red line) agrees with the given INL trend of the pulser (blue line). The “app3” pulser setup is hence not eligible for ADC nonlinearity determination, although its implementation would be less challenging.

the signal trend of the nonlinear fast pulse. The determined INL trend (magenta line in Figure 83) of the “app3” setup significantly differs from the given INL trend (red line in Figure 83). The results of the error determination are illustrated in Figure 84. The blue line in Figure 84 is the INL Trend of the used pulsers output. The red line, that is mostly covered by the blue line, represents the determined INL error trend (determined without any kind of optimization). This error behavior is expected and the “app3” setup is therefore not eligible for a practical adaption.

8.3.2 Majorana results

The Majorana setup was developed and published by the Majorana Collaboration (Abgrall, Allmond, Arnquist, & alia, 2021). In the simulations of this chapter the original Majorana Collaboration procedure (chapter 7.4) was investigated for the proposed "app4" nonlinear pulser setup (Figure 71). The only significant derivation between the proposed Majorana and the simulated procedure affects the number of recorded traces, the given noise level and the given pulsers INL trend. The latter was unpreventable, as the corresponding pulser data sheets do not contain the accurate INL trend behavior. The modification of the noise level (curtailed on the same level like in Table 15 ("Settings of FC Methods")) and the sample number dwindling are mainly motivated by simulation effort reductions. Another measure to save computing time is the abdication of a DAC clock phase that contrasts with the FC1.0 simulations is not considered anymore. The results of a corresponding simulation are depicted in Figure 85 -Figure 90. In the simulation pulser nonlinearity behavior like in the proposed "app4" setup was induced. Figure 85 and Figure 86 show the determined INL trends.

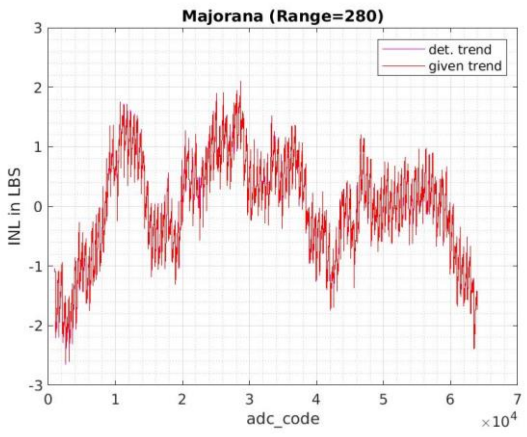


Figure 85 Simulation results with the Majorana Collaboration procedure

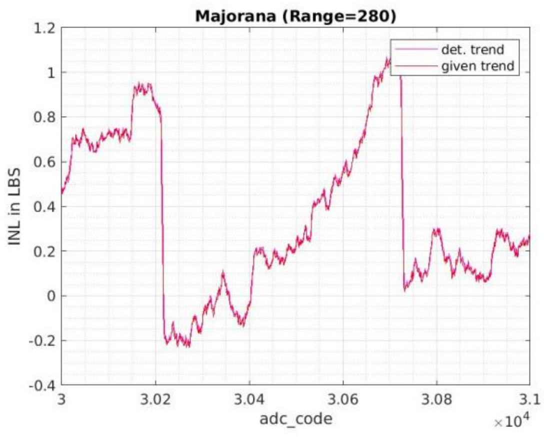
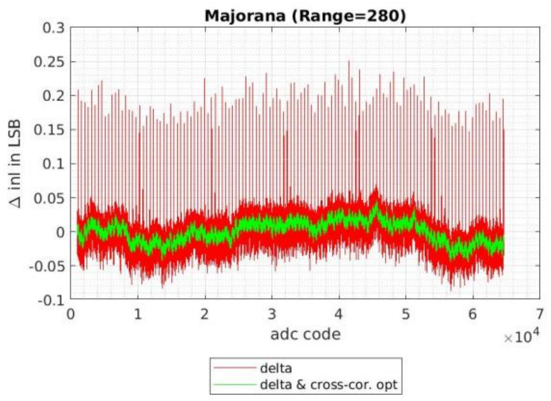


Figure 86 Zoom-in in the simulation results with the Majorana Collaboration procedure

A simulation series consisting out of 4 single Majorana simulations with the previously mentioned settings provided a mean "mean INL error" value of 0.0112 LSB and a ("mean INL error") standard derivation of 0.0017 LSB. The Majorana procedure simulation results outperform the recommended "b1 FC1.0 INL via DNL" approach which has a mean "mean INL error" value of 0.025 4LSB and a standard derivation of 0.0030 LSB. Note that this performance comparison was done for a noise level of 1.2 LSB_{RMS}. At higher noise level the Majorana method may even more outperform the FC1.0 nominee approaches, as the Majorana method does not use the concept of average traces and therefore is not frail to systematic noise interferences on its determined DNL trend.



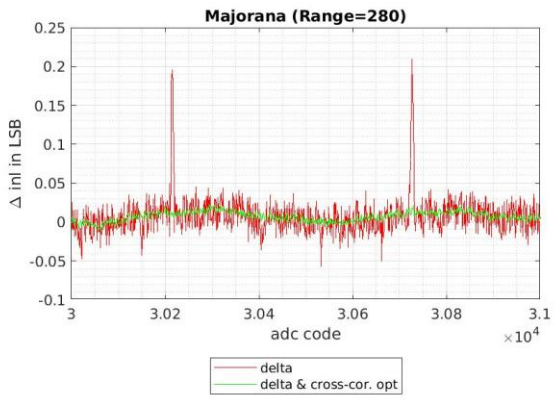


Figure 87 Error trend of the "Majorana (Range=280)" simulation

Figure 88 Zoom-in the error trend of the "Majorana (Range=280)" simulation. The error spikes belong to codes that represent transition points in the ADCs approximation architecture. At these points the change in the nonlinearity behavior is typically quite distinct (→Figure 86).

Figure 87 and Figure 88 illustrate the determined error trend. Their "unoptimized" red line trend has its spikes only on code values that belong to a code that represents a transition point of the ADCs approximation architecture. At such codes commonly a dramatic change in the nonlinearity level occurs. In [8] the published error plot also shows a similar pattern. The "optimized" green line trend only follows the basis error of the "unoptimized" trend. Potential reasons for the remaining basis error are the remaining noise uncertainties, but simulation results also imply that the nonlinearity behavior of the ADC itself also has an effect. In the simulations the base error behavior was dependent on the selected RANGE parameter settings. The remaining "basis error" won't have significant influence in in HPGe-detector experiments that consider INL correction. Hence a final investigation of the base error behavior was not done.

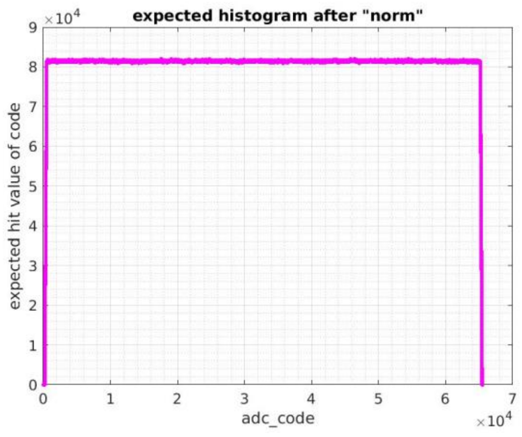


Figure 89 expected count histogram after applying "norm" correction factor. The "norming" (equation (16)) is required as without it the total counts of the expected histogram would dissent to the total counts of the real count histogram (David's expected count histogram updating does not consider the declination (and the resultant ADC code hit probability) of the saw tooth).

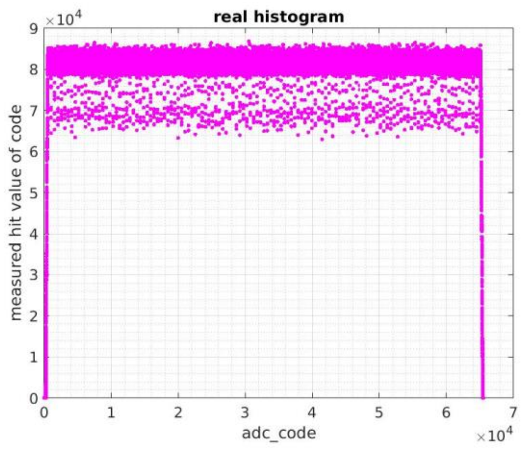


Figure 90 real count histogram (associated to the expected count histogram of Figure 89). The real count histogram and the "normed" expected count histogram are used to determine the DNL trend (equation (9)). The determined DNL trend is the basis of the shown INL trends.

Figure 89 and Figure 90 show the resulting histograms. The expected histogram (Figure 89) has with exception of the entrance and exit codes a uniform distribution.

The major burden of the published Majorana concept is the required measurement time due to the high total trace demand. This burden could be fixed by selecting a larger trace length and a corresponding RANGE parameter value. Fixing the original Majorana collaboration settings to a "faster converge" setting is for instance possible with the settings of Table 17.

Table 17 Setting of the "faster converge" Majorana simulation

Triangle wave frequency in Hz	1333 (750µs (→375µs))
Saw tooth frequency in Hz	0.1
Triangle amplitude$_{P2P}$ in LSB	~6553.6
Trace length in samples	32000 (320µs (→85.3% of 375µs))
ADC Sample Frequency in MHz	100
Resulting ADC codes per trace	~5600
Noise level in LSB$_{RMS}$	1.2
Variation of the pulse position in LSB RMS	0.0
RANGE parameter	4480
ADC Model	ADC Model 1
Pulser Nonlinearity Model	In agreement to App4

For the same statistically noise uncertainty the corresponding simulation runs require only a sixteen of the total trace number of the previous "genuine Majorana" simulation setup. An ideal measurement of fast converge setup of Table 17 is accomplished after approximately 2.6 mins. In contrast to that an ideal measurement with the original Majorana Collaboration settings (and the same histogram accepted total sample number) would require roughly 42 min. An example INL trend of the accelerated setting is shown in Figure 91. The “mean inl error” value of Figure 91 is superior to the default setting “mean inl error” of Figure 85. Also note that a performance comparison limited to the “mean inl error” basis is not priceless, as the represented code range in Figure 91 is smaller than in Figure 85. In the “fast converge” setting deliberately all ADC codes below 4480 and larger as 61506 are omitted. Including them would consequence in a INL trend that has in these ranges a deteriorating systematic error that significantly outperforms the remaining uncertainties of the genuine Majorana approach.

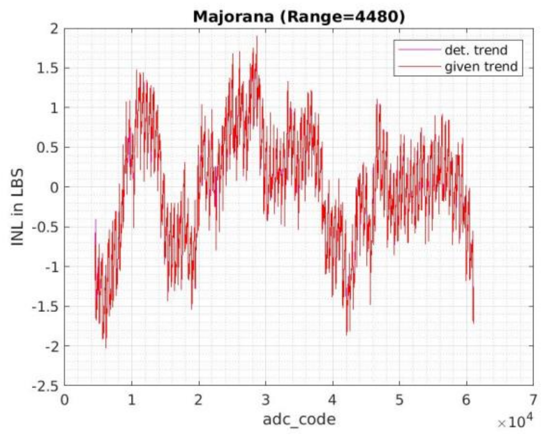


Figure 91 Simulation result with the fast converge Majorana approach. The faster converging behavior is harnessed by the selection of a larger Range parameter value (4480 instead of 280) and a sufficient (larger) trace length.

This systematic error is so distinctive that the corresponding “mean INL error” is ~0.06 LSB (instead of ~0.01 LSB in Figure 91). As in Germanium spectroscopy not the nonlinearity trend of the complete ADC FR is required, the ADC Code range reduction of the accelerated setting is in the case of this thesis acceptable.

In the “FC1.0” procedure the noise level influenced the possible accuracy of the DNL trend reproduction. For the Majorana method similar simulations were done. It was not possible to prove a similar effect. This outcome was expected, as in the Majorana procedure a DSP operation that is similar to the “averaging operation” is not implemented. Consequently, DNL trend smoothing in the Majorana procedure is not expected. In the “FC1.0” simulations also the influence of the pulser setup nonlinearity type was done. The investigation was repeated with the “app3” pulser setup in the Majorana simulations. The result is illustrated in Figure 92 and confirms that a saving of outputs (“DACs”) is not an option. Like in “FC1.0” an “app3” pulser setup is not eligible for the Majorana procedure.

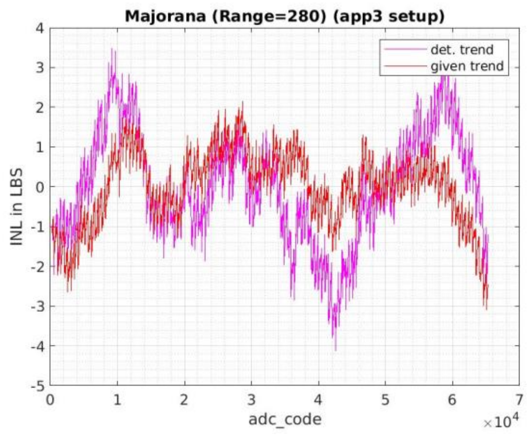


Figure 92 Simulation result of the “Majorana” procedure with an “app3” compatible pulser setup

8.3.3 Compare between Majorana and FC1.0 methods

An investigation of the ADCs "memory" behavior ("dynamic nonlinearities") requires several measurements with different amplitude[15]. In order to exclude an inaccurate determined nonlinearity behavior due to the procedure's internal flaws several simulations with different triangle pulser amplitude settings were done. The remaining simulation settings are unchanged and are still like in Table 15. The "mean INL error" results (equation (34)) of the simulations are shown in Figure 93. The depicted values are the mean value of four independent simulations. All "FC1.0" and "Majorana" simulation results base on a roughly comparable sample number. Figure 94 shows the corresponding uncertainty represented by the doubled value of the determined standard derivation.

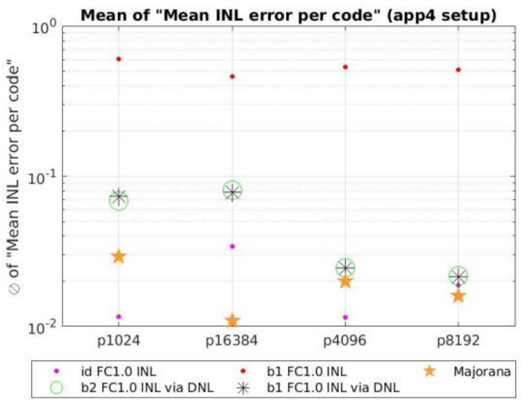


Figure 93 Evaluation of triangle pulse amplitude influence on INL error (I). The plot shows the determined mean value of the executed simulations (four per each delineated marker in the plot)(in total 80 simulations). The x-axis indicates the selected amplitude of the triangle pulse/shape. In the simulations of the FC1.0 procedures an offset increment of 50LSB was chosen. The value was chosen as the future pulser setup (chapter 8.4.1) has (roughly) a similar DC level resolution. In simulations also could be shown that a smaller increment size does not affect an error dwindling that in practice rewards the additionally require efforts.

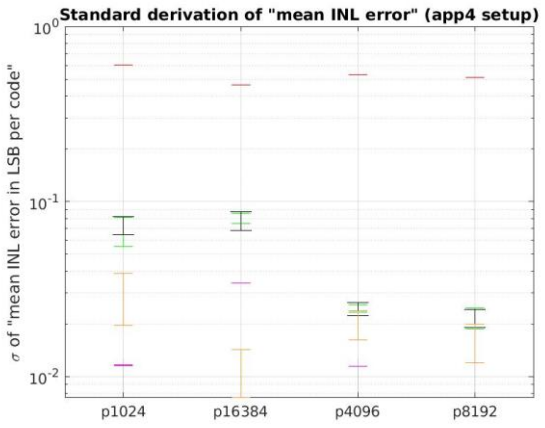


Figure 94 Evaluation of triangle pulse amplitude influence on INL error (II). Same simulations like in Figure 93. The error bars length indicates the (twofold of the) determined standard derivation of the executed simulations (four per error bar). The procedure is given by the color (color identical to Figure 93).

The Majorana procedure always outperforms the "bX FC1.0 INL via DNL" procedures. The result differences between the b1/b2 approach are negligible and therefore the much humbler "b1" approach is recommended. In all "FC1.0" procedures the error increases with the pulse amplitude. This is expected, as the "entrance/exit" issue is related to the height of the pulse amplitude. In case of the "FC1.0 INL via DNL" procedure the error also increases with decreasing amplitude. The underlying error is also in the

[15] The memory behavior is not only influenced by the signal amplitude. For instance, the real-life behavior of a non-memoryless ADC is influenced by the slope direction (negative or positive – sometimes discussed as ADC nonlinearity hysteresis) and the pulse frequency. In this discussion the focus is on the signal amplitude, as the used simulated ADC model is memoryless and thus the signal amplitude should have no influence. An influence of the signal amplitude on the determined INL error is therefore an implication of a method error. A frequency change in the simulations cannot induce an error (if properly selected), hence such simulations were not done.

corresponding “FC1.0 INL” implemented, but the differences in the data processing only amplify the affects in the “FC1.0 INL via DNL” approach. In a perfect setup the chosen pulse amplitude must be a multiple integer amount of the chosen offset increment step size. Only in this case the histograms will have no internal pulse pattern. As in a real setup it is hard to accurately fix the pulse amplitude, in the simulations the relation was deliberately violated. An example of the induced histogram pattern is represented in Figure 95. Note that the simulations of Figure 95 the same pulser settings and ADC nonlinearity model were used, but the merge trace building differed significantly (to exclude any noise uncertainties). This is legitimated as the simulations main intention was the visualization of the pattern’s magnitude. The patterns magnitude directly relates to the determined magnitude of the “FC1.0 INL via DNL” error trend. Initially it was assumed that this (expected histogram) peaks have no influence as – like expected – also similar patterns in the real count histogram were observed. Maybe histogram deflection caused by noise can explain the mutual non-cancellation.

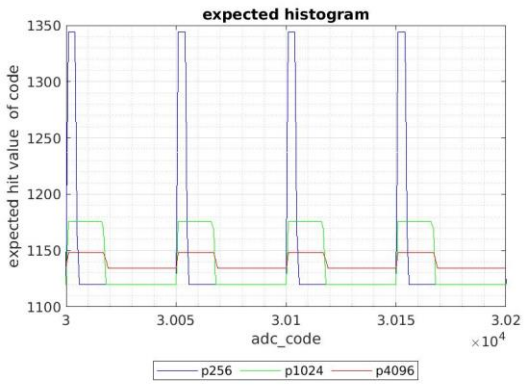


Figure 95 Instance for a histogram pattern that is induced by a maladjustment between offset increment and pulse amplitude (only in “FC1.0” procedure / plotted results were generated in an ideal setup simulation – in practice such a setup is not possible)

In the first not shown simulation series different ADC nonlinearity trends were used. The results implied that the merge trace is influenced by the ADC nonlinearities and therefore not pristinely represents the accurate pulser signal trend. To check the influence of the ADC nonlinearity magnitude, a simulation series with the specification of Table 18 was elaborated. In this series the already known nonlinearity trend of “ADC Model 1” was used (red marked “given INL trend” in previous figures of this section). The simulations differ in the used scaling factor applied on the given DNL trend of “ADC Model 1” (equation (37)) and consequently have a different <u>absolute</u> mean INL value[16].

[16] Definition of absolute mean INL value: $abs\_mean\_INL = \left(\sum abs(INL_{Code_x})\right) / 2^N$

Table 18 Settings of the INL magnitude influence evaluation simulation series

Pulser Model	Perfect pulser ("app0")
ADC Model	ADC Model 1 (Red marked "given INL trend" in previous figures of this chapter)
Used scaling factors **("dnlfac" in equation (36))**	0.25, 0.5, 1.0, 1.5,2.0,2.5,3.0,4.0
Noise levels in LSB_{RMS}	1.2
Amplitudes $A_{sawtooth}$(in LSB)	4096
offset_inc (in LSB)	10

$$DNL_{Given_{Simulation}} = DNL_{Given_{Model}} \cdot dnlfac \tag{36}$$

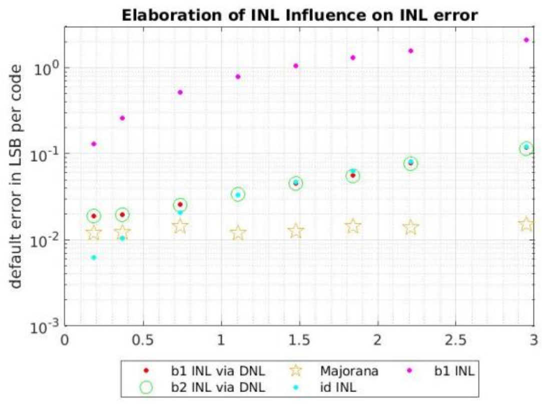


Figure 96 Evaluation of INL influence on mean INL error (I). The corresponding simulations used the settings of Table 18. The x-axis indicates the absolute mean INL value of ADC Model 1 after applying the scaling factor ("dnlfac") on it. The y-axis indicates the already known "mean INL error per code" (equation (34)) of the determined INL trend.

The results of Table 18 simulation series are presented in Figure 96. Condensing the key results of all simulations concludes to the following summary list:

- The Majorana and the FC1.0 procedure are capable to reduce the pulsers nonlinearity effects on the determined INL trend
- The error of the FC1.0 procedures depends on the ADCs nonlinearity behavior
- In the FC1.0 procedure a maladjustment between pulse amplitude and offset increment has negative effects. The magnitude of the negative effects depends on the method ("INL" or "INL via DNL") and the pulse amplitude.
- The Majorana procedure also has systematic issues, but surprisingly the determined "mean INL error" trend of the INL magnitude influence simulation series (Figure 96) was varying around a fixed value. The systematic error behavior of the Majorana procedure is therefore significantly different from the systematic error behavior of the "FC1.0 INL via DNL" procedure. Both procedures use a DNL determination approach.

- The originally intended “FC1.0 INL” procedure is not eligible to determine accurate INL trends. In special cases its results still can be used for nonlinearity correction.
- By its averaging operation the “FC1.0” procedure smooths its DNL trend. The noise level is decisive for the smoothing strength.
- The “FC1.0” procedure has two kinds of baseline determination (“b1”/”b2”). As the “b2” approach has more error sources, the “b1” approach is preferred.
- In the ideal case the “FC1.0” procedure has sufficient buffer space (for all determined average traces) in memory. If not sufficient buffer space is available, the average traces must be measured in the merge trace determination step and in the INL determination step. Besides a time loss, this also affects the precision of the procedure. In cases where not sufficient buffer space is available, the “b2” baseline determination approach should be preferred.
- The reproduction results of the Majorana approach are better than the results of the “FC1.0 INL via DNL” approach
- In general, the published Majorana procedure is slower than the FC1.0 procedure. This can be an issue as with measuring time also the perils of temperature drift issues can increase. In order to keep the suggested setups simple, an implementation of a climate box was rejected.
- In some cases, it is possible to accelerate the Majorana approach by simply increasing the trace length and the Range Parameter. The acceleration expense is an abandonment of some Code range of the ADC.

8.4 Setup of the used experimental INL determination approaches

Already in the INL measurement introduction the main design aspects of the setup were mentioned. It must be cheap and must be applicable in an experimental environment. In the best case the setup only requires a poor wave generator that approximately outputs the demanded signal shape and no further special features. Such setups are the proposed FC1.0 nominee setup or the Majarona Collaboration approach. All investigated approaches used as FADC card the Evaluation board "DC2266A" from Analog Devices [35] (Figure 97). As ADC it has the Ltc2107, a 16bit ADC that runs up to 210 MHz and has a transition noise of 3 $\mathrm{LSB}_{\mathrm{RMS}}$ and a typical INL value of ±1.6 LSB [36]. A Ltc2107 typical INL trend with its striking double peak behavior is depicted in Figure 98. The DAQ in the setup, is like in all experimental chapters of this thesis, still the FC2.0 prototype DAQ. In contrast to the firmware of "event building performance" setup, the firmware must handle only a single channel. As in this setup the firmware obviously must input the ADCs output stream, the firmware needs the "ADC data read in" unit and cannot (actively) use the ADC dummy as substitute. As the DAQs firmware and software design is of minor importance for the INL determination approaches results, their precise involvement is not presented in this chapter.

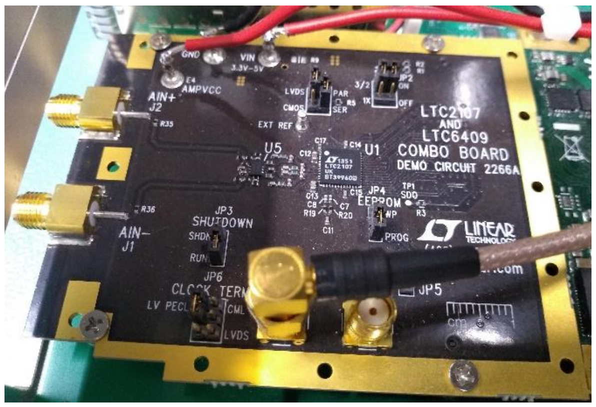


Figure 97 Evaluation board "DC2266A" (implemented in FC2.0 prototype)

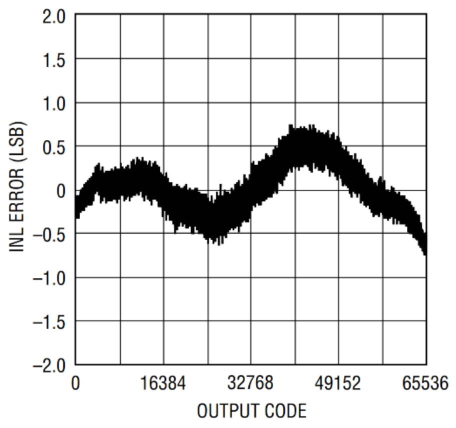


Figure 98 (Analog Devices official) typical INL trend of ADC Ltc2107 [36]. Note the double peak behavior of the INL trend.

As "cheap" waveform generator mainly the SDG6022X from Siglent was used in the experiment [29]. The selection of the waveform generator was mainly motivated by its low costs and appealing specifications. In first pre investigations with the FC1.0 DAQ Thomas Kihm achieved impressing results that concluded in the final affirmation of the SDG6022X as future main pulser. Unfortunately, in the subsequent development process several FC2.0 prototype adjustments were necessary. These adjustments affected serious changes.

An especially critical point of the waveform generator selection is its nonlinear behavior as data sheets rarely provide the required detailed information. In the simulation chapter 8.1.1 four nonlinearity composition models were introduced (Figure 71). The models assumed that every pulser output generates its signals on its own DAC and has no additional electronic component that is capable to add a specific DC level. In practice this must not be the case. Real pulsers commonly must cover a full signal range that is much higher than the FR of a common DAQ. A waveform generator therefore still requires analog electronics whose gain may depend on the chosen generator settings. Some of the given data sheet accuracy specifications therefore may only be valid for a special setting case. The noise behavior also may be dependent on the selected shape settings.

8.4.1 FC1.0 nonlinearity determination setup

The used setup is a slight modification of the originally proposed setup and illustrated as condensed overview in Figure 66. Instead of using an adder circuit to combine the signal generator outputs, the outputs are connected to the differential inputs pins of the FADC channel under test. This is similar to the procedure in the published Majorana paper. The differential input behavior effectively replicates the adder circuit. A drawback of such a simple adder emulation is that the effectively applied output common mode voltage must meet the demanded input common mode voltage range of the FADC card. Note that this limitation is always the case, even if the differential FADC card is feed with a single ended single and its remaining pin is connected to ground. A photo of the real setup is shown in Figure 100. The Siglent SDG6022X was used as signal generator. It has two independent, but synchronize able, non-floating outputs. The (signal) polarity of each output can be controlled. Its most important specifications are summarized in Table 19. The SDG6022X is capable to output the demanded triangle pulse shape.

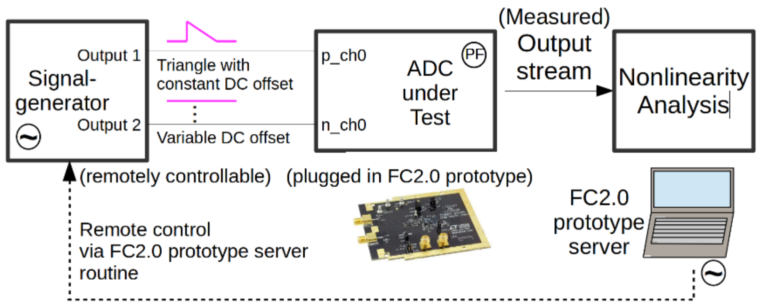


Figure 99 Basic principle sketch of the FC2.0 prototype implementation in the FC1.0 nominee approach

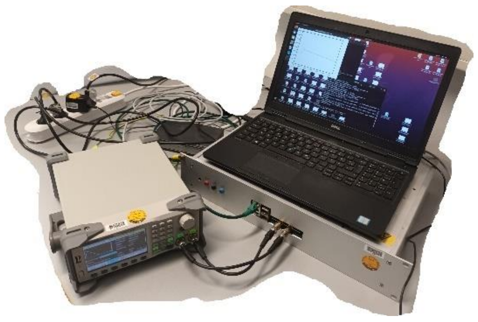


Figure 100 Real FC2.0 prototype setup in the FC1.0 nominee approach. The FC2.0 prototype DAQ and the FADC are hidden in the housing. The laptop (with a default Intel desktop CPU) is the used FC2.0 prototype read-out server.

The measurement and the signal generator control are done by the available FC2.0 server routine. Appendix chapter A.9.3 is designated to the internal procedure in the server routine. Their available options and their usage are listed in Table 20.

Table 19 Key specifications of the “SDG6022X” waveform generator [29]

Property	
Harmonic Distortion (0-1MHz)	-65dB
Ramp	
Output Range in V (50Ω load)	0.001-10
Frequency	10^{-6}-5×10^{6}
Linearity	1% (Peak output)
DC	
Output Range in V (50Ω load)	-5-+5
Accuracy	±(1%+2mV)
Pulse	
Rise time (10%~90%)	2ns
Overshoot	3%

Table 20 Used command options of the FC2.0 prototype server routine (1). To reduce space consumption only the essential command options are listed in this table.

Command	
-wlength	Selects the trace length
-doanalysis	Enables the analysis path of the server read out routine
-ana_start_offV	Determines the start voltage level (“baseline_level_start”) of the DC signal (output 2 in Figure 99)
-ana_end_offV	Determines the final voltage level (“baseline_level_end”) of the DC signal
-ana_inc_offV	Determines the voltage increment size (“offset_inc”) of the DC signal
-ana_pulse_ampV	Amplitude of the “triangle” signal (output 1 in Figure 99)
-ana_pulse_freqHz	Frequency of the signal
-ana_tracesperaverage	Number of traces which are used to determine an average trace (“N”)
-ana_baselinestart	“b_start” in Figure 68
-ana_baselineend	“b_end” in Figure 68
-ana_inltracestart	“inl_start” in Figure 69
-ana_inltracesend	“inl_end” in Figure 69

Command	
-tsel	Signal selection. "-tsel 1" returns the raw ADC data in the trace (default setting), "-tsel 2" returns the Gaussian shaped ADC signal (done in the pulse detection block of the firmware).
-srprop	Selects the effective sample rate after downsampling. In the used setup the selection of "-srprop 3" prompts an effective sample rate of 25 MHz (as mean value of 8 samples). The default setting ("-srprop 0") is without downsampling (setup sample rate of 200MHz). More information in appendix chapter A.2 available.
-ana_filenameaveragemerge	Determines the filename of the merge trace file.
-ana_mizziinl	INL determination in "FC1.0 nominee" style is done. This step requires the input of an available merge trace filename (for instance via "-ana_filenameaveragemerge dateiname.bin"). If no merge trace is available or input "-ana_mergeinl" is not used the INL determination run is not possible.
-ana_mergeinl	If the passed file name of option "-ana_filenameaveragemerge" not exists and the option "-ana_mizziiinl" is selected, the routine starts a merge trace determination before the (subsequent) INL determination run. The determined merge trace will be stored under the chosen name of the "-ana_filenameaveragemerge" option.
-ana_sdg6022X_classicoffV	Much more settings than presented were elaborated. Setting "-ana_sdg6022X_classicoff" is essential to get the demanded signal behavior of Figure 99. If setting "-ana_sdg6022X_classicoff" is not used, the outputs of the signal generators will behave different.

The "brief" mode and the "bX mizzi smart" approaches currently require the support of MATLAB and are not implemented in server routine. The used MATLAB Routine can access the average trace files of the server routine and analyzes them in "brief" or in "smart mizzi" approach procedure compatible way.

8.4.2 Majorana setup

The FC2.0 prototype Majorana setup illustrated in Figure 101 is a modification of the suggested setup of the Majorana Collaboration. Using a triangle pulse as fast signal was necessary as the used FADC card only has a single channel and therefore the setup is not capable to use the outputted trigger signal of signal generator. This means that the pulse detection block of the FC2.0 prototype firmware must trigger the fast pulse shapes. An advantage of this modified signal shape is the better agreement to the nonlinear behavior of the used FC1.0 setup. As a triangle pulse shape does not have an inclining ramp part, only the declining ramp part of the triangle pulse can be used for INL analysis. As modification penalty the determination of the INL hysteresis properties is not possible.

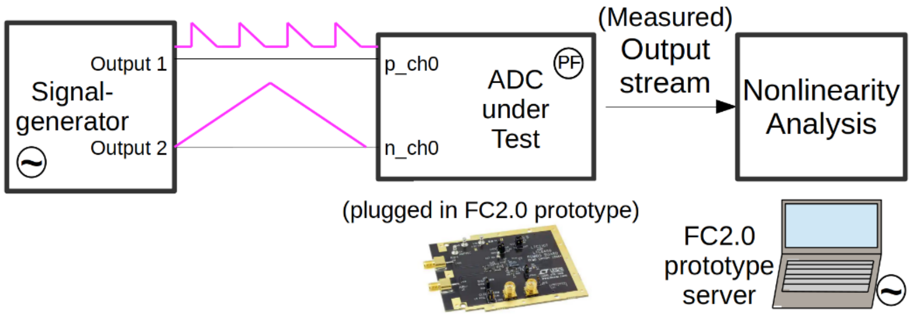


Figure 101 Majorana Setup for FC2.0 prototype. In the FC2.0 setup of the Majorana procedure a triangle pulse (as fast signal) and a triangle shape (as slow signal) are used as signal shapes.

In contrast to the suggested FC1.0 setup the Majorana Setup does not require the remote control of the signal generator. The miscellaneous parts of the used setup are otherwise identical to the FC1.0 setup. Additionally required options of the server read out routine for the Majorana setup are listed in Table 21.

Table 21 Used command options of the FC2.0 prototype server routine (2)

Command	
-ana_davidinl	Enables the Majorana INL Analysis functions in the server read out routine.
-ana_davidrange	Determines the value of the RANGE parameter (equation (12) - (13)) and therefore determines the size of the accepted ROI whose samples are accepted for the INL determination. The parameters value must not violate the restrictions out of the selected trace length and the selected “-ana_davidmeanpos” value.

Note that the signal generator setting must be done manually and therefore no corresponding server routines commands exist.

8.4.3 GERDA setup

The original GERDA setup (Figure 61) was developed for an ADC with a single ended input signal. The FC2.0 adapted version is presented in Figure 102. The investigated FADC card "DC2266A" is intended for differential input signals. The modified "DC adder" circuit to generate the fine-grained DC signal is depicted in Figure 103. Remember that the circuit is also responsible for noise and hum attenuation. Both factors are especially critical in the GERDA setup. As "DAC" the power supply SPD3303X and as DMM the SDM3065X (both from Siglent) were used. In the chosen implementation the SDM3065X has roughly a precision of ~5.5 digits (~19 bit) and is therefore close to its official maximum resolution of 6.5 digits [37]. Both devices support remotely control and read-out via SCPI (TCP/IP link, not raw ethernet like FC). The instrumentation control and data read-out are done by the "FC2.0 prototype server" laptop that is not visible in Figure 102. The most serve difference to the setup of the GERDA collaboration is the relinquishment of the climate box. The climate box initially was introduced to suppress temperature drift issues. In the thesis the GERDA setup was implemented to verify the determined global INL trends of the Majorana and FC1.0 procedure. It was not the intention to determine the DNL trend of the ADC under investigation. Thus, it was decided to limit the scanning to ~160 DC voltage levels. Such a measurement will take less than 25 min and consequently temperature drifts should not be a serious issue. In this big step procedure, the "adder circuit" is implemented due to its noise filtering capabilities.

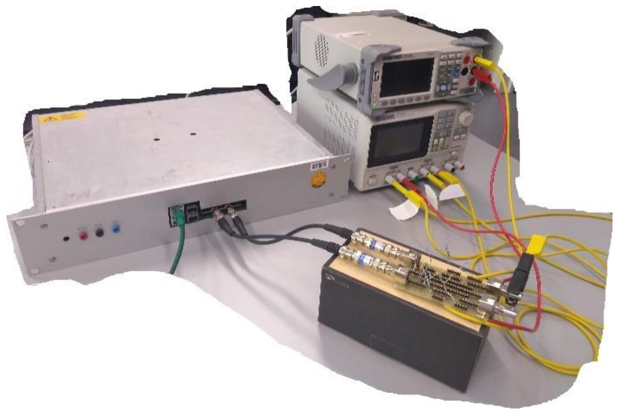


Figure 102(FC2.0 prototype) GERDA setup to verify the determined INL trends. The upper right instrumentation is the DMM (SDM3065X), the lower right device is the power supply ("DAC" in Figure 61). The modified "adder" circuit is on the top of the black box (in the foreground) and the FC2.0 prototype housing can be seen on the left.

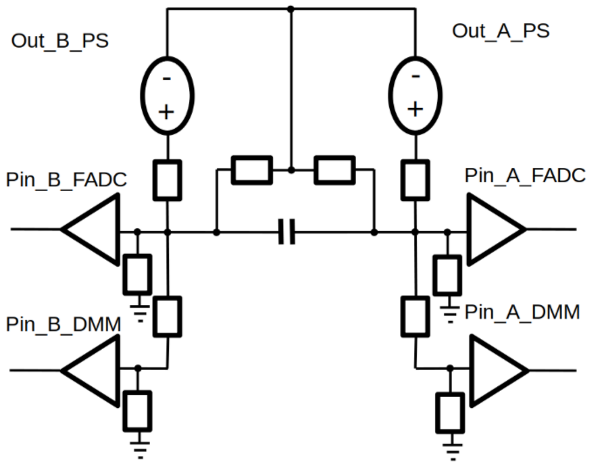


Figure 103Schematic of the modified GERDA "adder" circuit. This schematic is simplified. In the implemented circuit the resistors have customized (unbalanced) values given by the FADC/DMMs input range and the output range of the power supply.

The nonlinearity analysis of the "GERDA" FC2.0 prototype setup does not need any adaptions.

8.5 Results of the experimental INL determination setups

The results of the FC1.0 INL determination setup are discussed in chapter 8.5.1, the results of the Majorana setup are examined in subsequent chapter 8.5.2.

8.5.1 FC1.0 INL determination setup

In the simulations it was shown that a waveform generator with "app4" nonlinear behavior is eligible to be used as signal generator. In the "app4" style the two signal outputs of the waveform generator are independent. In practice this must not be the case. It is even possible that the output "cross talk" of a waveform generator depends on the chosen setting combination. Such dependencies are not certainly correctable by the methods under investigation and therefore might degenerate the resulting INL trend. Another important design point of the simulations was the noise influence. As a real waveform generator is a combination of several amplifiers and DACs whose contribution to the output signal may depend on the setting, also the noise level of a real waveform generator may depend on the chosen pulser settings. In the first step it was tried to determine the best possible pulser setting by a measurement series with specifications of Table 22. The chosen voltage ranges of Table 22 are inspired by the datasheet specifications of the used "DC2266A" FADC card.

Table 22 Pulser pre investigation measurement series. In all measurements the triangle shape ("fast pulse") had a selected amplitude of 0.1 V and a frequency of 425 Hz. The chosen offset increment was 1mV. The used settings selected special options that trigger an enhanced shaping (via gaussian shaping). This shaping reduces noise and consequently enables faster measurements. Though the input of the already shaped signal 360 traces were used to determine a single average trace. As the gaussian shaping besides the "averaging" also "attenuates" the DNL structure the setting is not recommended for the final measurements and therefore not discussed in detail. Every measurement was repeated at least three times to reduce statistical uncertainties.

ICM in V ("Output 1")	Start Voltage Level ("Output 2")	End Voltage Level ("Output 2")
-0.5	-1.056	+0.056
-0.25	-0.808	+0.306
0	-0.556	+0.556
+0.25	-0.306	+0.806
+0.45	-0.104	+0.999
+0.7	+0.144	+1.256
+0.8	+0.244	+1.356
+0.9	+0.344	+1.456
+0.999	+0.444	+1.558
+1.1	+0.544	+1.656
+1.2	+0.646	+1.756
+1.5	+0.946	+2.056

Figure 104 illustrates the observed noise behavior of the measurements. An ideal pulser would always

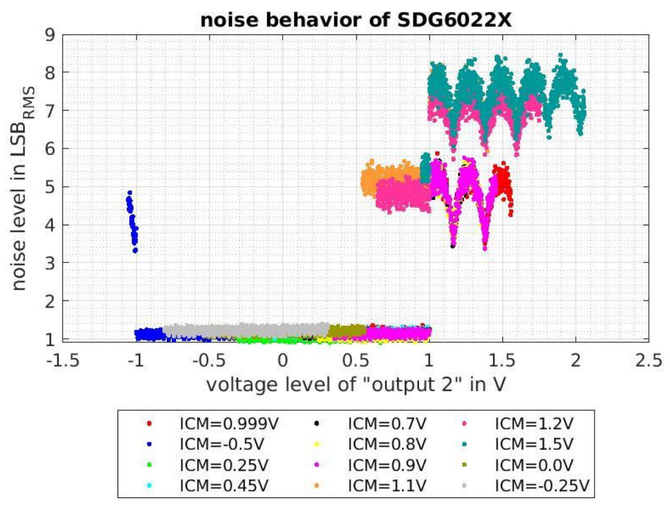


Figure 104 Investigated noise behavior of SDG6022X. Thy y-axis indicates the signal noise level in LSB_{RMS} after passing a gaussian shaper with an integration time of approximately 10µs. The noise level of the unshaped raw signal is much higher – for instance the raw ICM=0V signal has a noise level of $3.9LSB_{RMS}$ (instead of $1.2LSB_{RMS}$).

have the same noise level and therefore its noise level would be independent of the selected voltage settings (given by “ICM” in Table 22). The high noise levels are partly a result of artificial noise that is deliberately added by the SDG6022X. Unfortunately, all tries to disable the artificial noise induction were not successful. This is especially painful, as the artificial noise in the required signal bandwidth likely has an insufficient “corrupted” non-random distribution. Signal filters that depend on real random noise therefore may show a poorer performance[17]. As noise degenerates the determined INL trend result (chapter 8.3.1), Figure 104 implies that the setup should be operated with the voltage conditions of “ICM=0V” or a setting close by. Determining the best “cross talk” setting is more challenging. In the setup we can only detect cross talk by its degenerative influence on the linearity properties of the setup. This means that the ICM setting with the lowest nonlinearity is possibly the pulser setting that has the lowest “cross talk” and the pulsers outputs in this setting have a behavior that is comparable to the “app4” simulation models. The INL trend result of such a setting can represent the genuine INL trend of the ADC in the setup. This point is critical and hazy as with the ADC one of the significant nonlinearity sources of the system is the virtual subject under investigation. The ICM setting that achieves the lowest INL trend magnitude is therefore not necessarily the setting with the best “cross talk” behavior. In the worst case the setting achieves the best mutual cancellation of the pulser “cross talk” and ADC nonlinearity effects[18]. The resulting trend then strongly differs from the real ADC INL trend that is the subject of the investigation. Figure 105 depicts the resulting INL trends of the measurements of Table

[17] In the FC1.0 procedure average traces are used. “Averaging” effectively attenuates (white) noise by averaging it out. If the noise frequency distribution is not uniform (as likely in the artificial noise case), noise attenuation may be poorer. Such an artificial noise behavior therefore may be capable to generate an uncertainty-noise dependency that implies a systematic error in the method. Via a “noise level after shaping” analysis also the suspicion of the “corrupted” artificial noise was investigated (not discussed in this thesis).

[18] A further possibility to check the “cross talk” supposition is the substitution of “output 2” by an output of another additionally implemented SDG6022X signal generator. As the pre-investigations indicated an excellent designed SDG6022X this additional implementation burden was initially rejected. With the size of the setup, ground induced issues and other EMI interferences are thriving potentially. Reducing the setup size is therefore useful. The investigated noise behavior of Figure 104 changed the situation significantly. In the meantime, a new setup is under elaboration.

22. Remember that the used “FC1.0 INL via DNL” approach requires the declination of the triangle pulse (Δ_{Pulse} in equation (28)). In all real measurements the triangle declination is determined by a linear fit on the corresponding merge trace region.

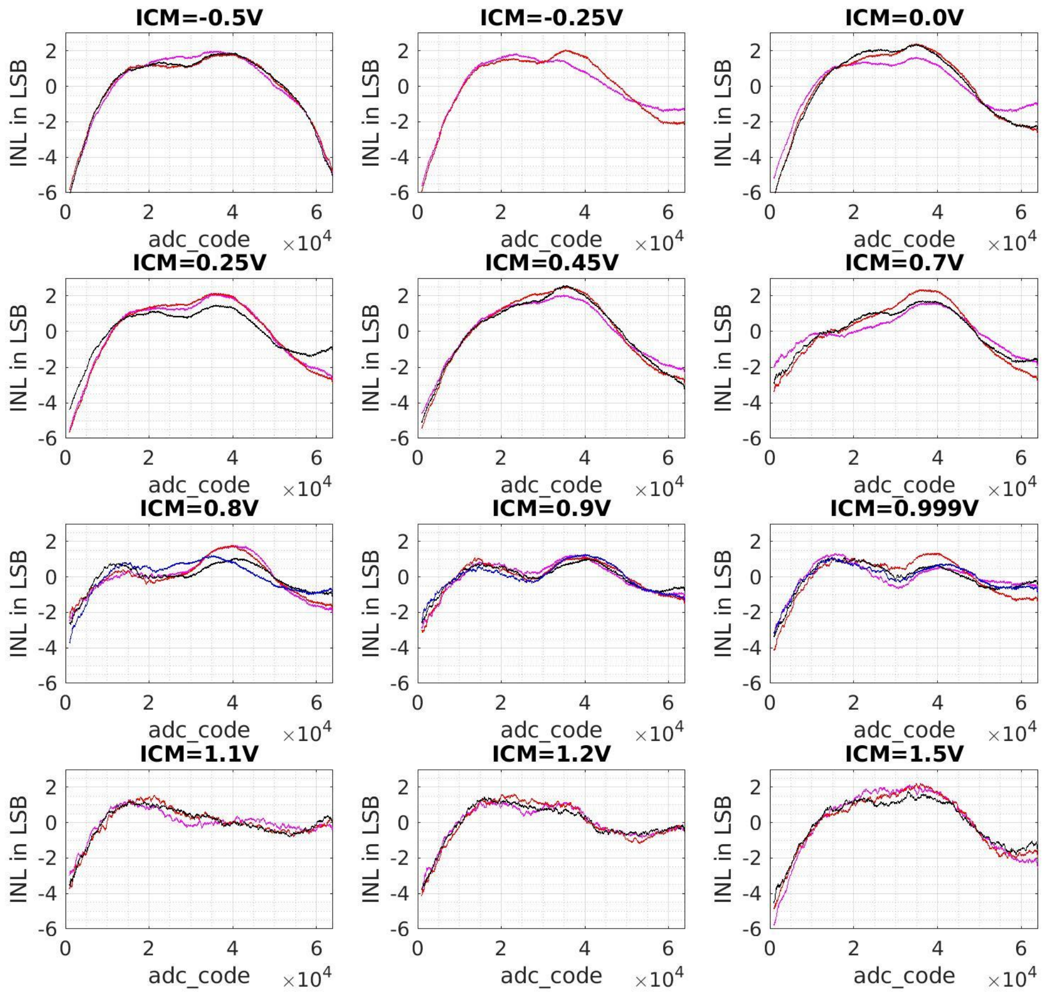


Figure 105 Determined INL trends in dependence of the used ICM voltage

The setting “ICM=0.9V” achieves the INL trend with the lowest magnitude and therefore could (but most not) represent a pulser operation condition that is close to the “app4” behavior (two signal outputs without any interference). Another indication that sublines that the “ICM=0.9V” setting is likely the best, is the high similarity between its determined INL trend and the instance INL trend of the Ltc2107 data sheet. A certain conclusion is not possible therefore the assumption that the setting is the best is only preliminary. Note that the “ICM=0.9V” settings conditions also have the best agreement with the supposed ideal operation conditions of the implemented ADC driver op-amp on the FADC-card. The “ICM” Voltage therefore needs not only influence the nonlinear “cross talk” behavior of the waveform generator, but it might also influence the nonlinearity properties of the implemented ADC driver op-amp.

The pre-investigation of the setup is concluded by the measurements of Figure 105. The final INL measurements used a slightly changed signal generator setting of the previous “ICM=0.9 V” measurements, at which the frequency of the fast triangle pulse was increased to 3300 Hz. The remaining pulser settings are identical. Furthermore, the measurements need to use the unshaped raw ADC data signal as input and should not use the pre gaussian shaped signal as measurement setup of the pulser pre investigation. As the raw signal has a higher noise level, an increase of the trace number per average trace is also required. In the following measurement whose results are discussed in the next subsection 2400 traces per average trace were used.

In the simulation setup it was elucidated that the determined INL trends are deduced from the expected count and error histogram. Examples of determined histograms are plotted in Figure 106 and Figure 107. The error histogram (Figure 107) is split in two parts with different uncertainties. All ADC codes below 30000 are measured by pulser settings that cause a higher noise level and consequently the corresponding part of the error histogram has a higher uncertainty (variations in the “DNL error”).

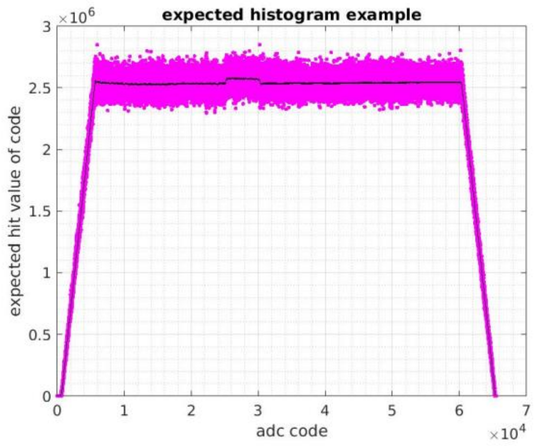


Figure 106 Instance of an expected count histogram

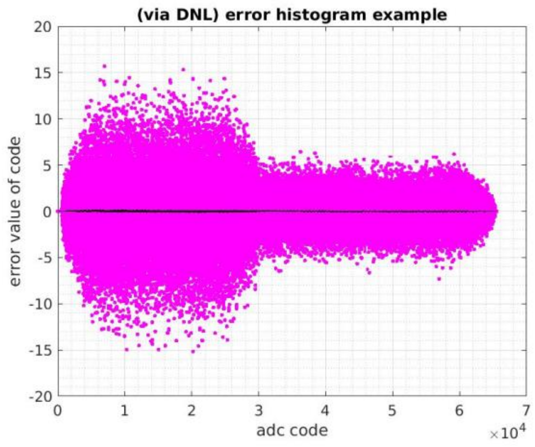
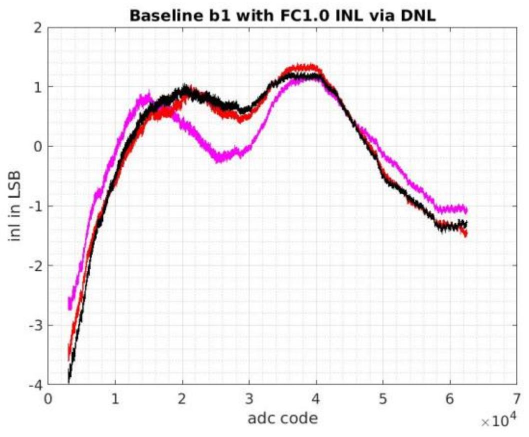


Figure 107 Instance of a “FC1.0 INL via DNL” error histogram

The histograms of Figure 106/Figure 107 deduce the magenta INL trend in Figure 108. The additional INL (“verification”) trends of Figure 108 were determined in subsequent measurements with the same settings.



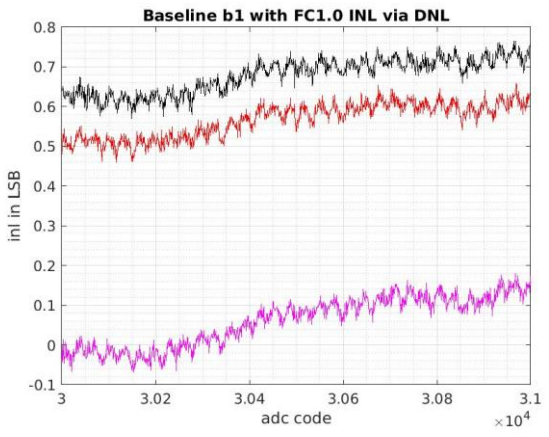


Figure 108 INL trends determined with the SDG6022X and the "FC1.0 INL via DNL" approach. The terminating straight line fit correction started at ADC code 3000 ("slf_start_code") and stopped at ADC code 62500 ("slf_end_code"). The reduction of the gain-offset determination region was necessary as the noise level in the real setup is much higher than in simulated setup and furthermore the used measured settings covered a smaller ADC code range.

Figure 109 Zoom-in in the determined INL trends of Figure 108

The magenta INL trend in Figure 108 has still a high agreement with the determined INL trends of the pre investigation setting (subplot "ICM=0.9V" in Figure 105). The trends of the verification measurements (black and red trend in Figure 108) differ significantly. In principle the difference could be explained by drift effects in the setup, for instance by nonlinearity inducing thermal drifts of the voltage reference or by some drifts in the waveform generator. The available data did not enable a certain conclusion of the source. The issue is a major blow as the main objective of the developed method was its brief measurement time. It was intended that in this brief measurement time drift issues have a minor effect on the resulting INL trend.

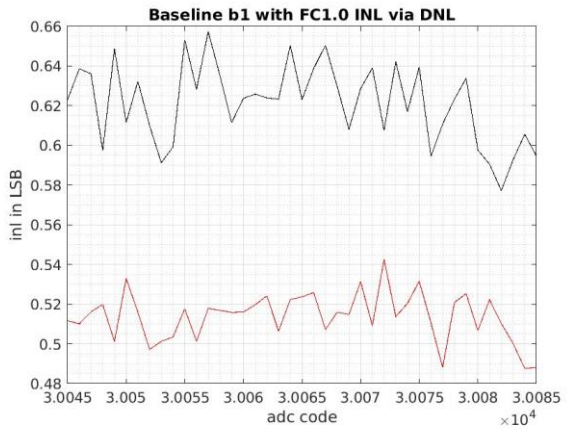


Figure 110 Further zoom-in in the determined INL trends of Figure 108

INL trends of sufficient precision can be used to determine the ADCs internal architecture. Unfortunately, the zoom-in plots (Figure 109 and Figure 110) are not reliable enough to determine the architecture of ADC under investigation. Comparing the different trends reveals that they differ in every magnification level. The noise influence simulations (Figure 82) only implied issues in the magnification levels of Figure 110. In total the results of the FC1.0 nominee setup cannot meet original expectations. The malperformance is likely caused by issues of the used waveform generator "SDG6022X" and not caused by an unfavorable analysis setting.

In the simulation chapter 8.3.1 the failure of the initially intended “FC1.0 INL” method was resolved. Applying the “b1 FC1.0 INL” approach on the same raw data sets of Figure 108 obtains the INL trends of Figure 111. The assignment between color and the corresponding raw data set is unchanged to Figure 108. Like expected, the determined trends significantly differ from the data sheet example. Strangely the DNL behavior (“uncertainty variations”) of the trends partly varies in a way that is not compatible to the observed noise behavior of the pulser.

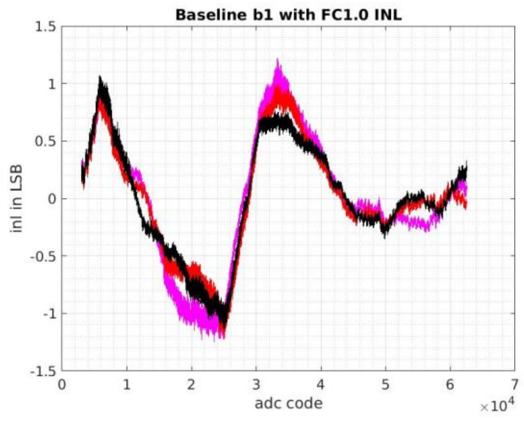


Figure 111 INL trend determined by the “FC1.0 INL” analysis procedure of T. Kihm The used (average trace) data sets (and their color assignment) are the same as in Figure 108, but the analysis procedure is changed to “b1 FC1.0 INL”.

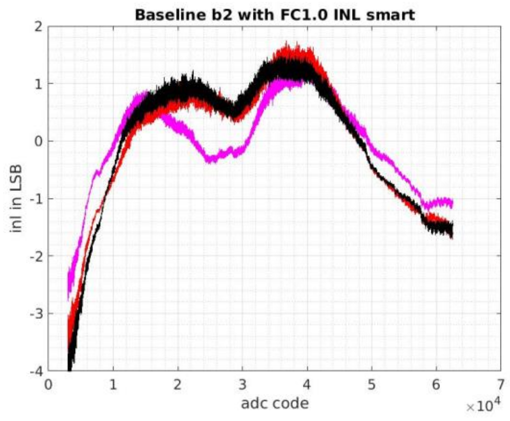


Figure 112 INL trend determined with the “b2 FC1.0 INL smart” procedure. The used (average trace) data sets (and their color assignment) are the same as in Figure 108, but the analysis procedure is changed to “b2 FC1.0 INL smart”. In contrast to the expectations, the “b2 FC1.0 INL smart” procedure is not eligible to reproduce the local nonlinearity behavior (can be seen as strong dithering of the trends). The (not shown results of the) “b1 FC1.0 INL smart” procedure faces similar issues.

In the simulation chapter the “b2 FC1.0 INL smart” approach was introduced. Applying the “smart” procedure on the raw data again returns the INL trends of Figure 112. Like forecasted by the simulations, the “FC1.0 INL smart” approach is capable to conserve the global INL behavior. But surprisingly it fails in the reproduction of the local INL behavior. In the plot this is connoted by the strong dithering of the trends. The not shown results of the “b1 FC1.0 INL smart” approach show a similar behavior. In practice the “smart” approach faces an unknown issue that was of minor significance in the simulations.

8.5.2 Majorana setup

The Majorana setup uses as slow signal a saw tooth and not a DC signal like the FC1.0 nominee setup. As in the FC1.0 nominee setup the signal settings strongly influenced the noise level of the SDG6022X output signal it was decided to redo the noise level-output signal level investigation also for the saw tooth setting of the modified Majorana setup (Saw tooth amplitude of 1.11 V_{PP} and a frequency of 0.1 Hz). The determined noise level trend presented in Figure 113 is almost constant, but also unfavorable high. A noise level of 16 LSB_{RMS} would require 65536 traces to obtain an average trace with an effective noise level of 1/16 LSB_{RMS}. Examples for expected count and real histograms of the modified Majorana approach (with a RANGE parameter of 3600) of this thesis are illustrated in Figure 114 and Figure 115. Note that a setting that agrees to the setup of the published Majorana paper must use a RANGE parameter value of ~280 LSB. In contrast to the FC1.0 approach the Majorana method does not (significantly) adulterated the DNL trend as it uses an unsophisticated real count histogram determination approach. The real count histogram (Figure 115) enables conclusions on the ADCs internal structure. In case of the ADC Ltc2017 the digitization is likely done in three approximations.

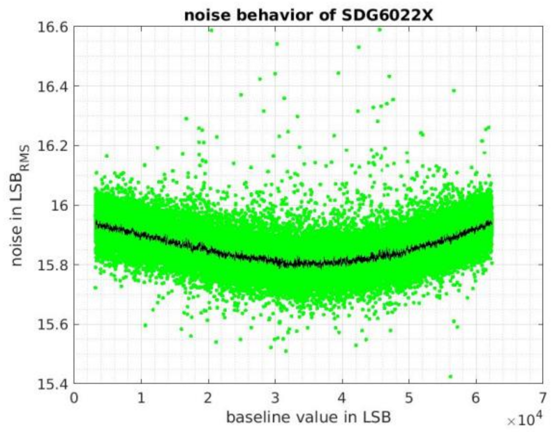


Figure 113 Saw tooth noise level in dependence on the saw tooth signal level

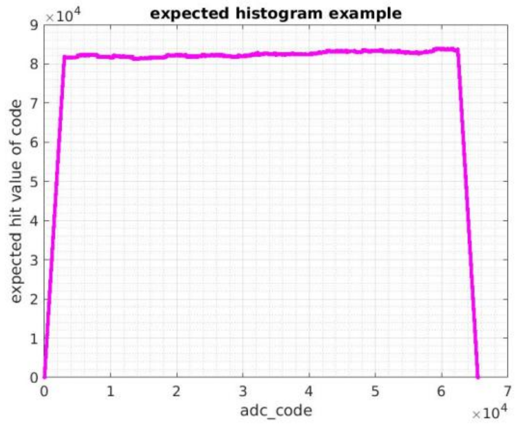


Figure 114 Instance for an expected count histogram in the modified Majorana approach. The corresponding measurement used a RANGE parameter of 3600. The code “hit” value magnitude in Figure 114 is much smaller than in Figure 115. This can be explained by the missing “norm” factor correction (equation (16)).

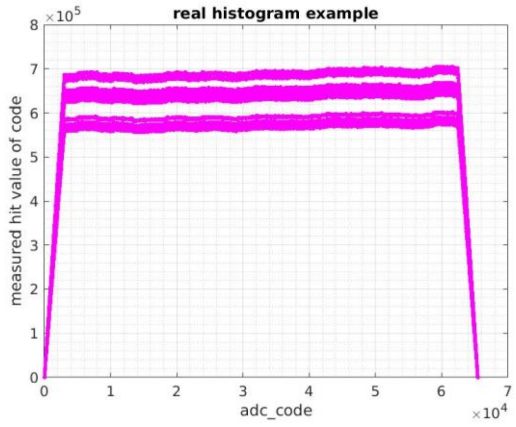


Figure 115 Instance for a real count (hit) histogram in the modified Majorana approach. In the published results of the FC1.0 nominee approach the average hit rate of a code was $2.6\times10^{6.}$ (Figure 106).

Figure 144 shows that the FC2.0 DAQ was capable to (almost) uniformly scan the ADCs full range and consequently no beat frequency issue did arise. Not every DAQ can offer such a performance. The results of such DAQs may be distorted, as the uniform sampling is a requirement to correct the negative influence of the fast pulsers nonlinearities.

Figure 116- Figure 119 illustrate two examples for the determined INL trends with the Majorana approach. In all figures the same data sets with identical measurement settings were used. The results have a high reliability and fluctuate much less than the determined INL trends of the FC1.0 approach. The “stable” behavior of the Majorana trends could be confirmed in several measurements and is therefore not coincidence.

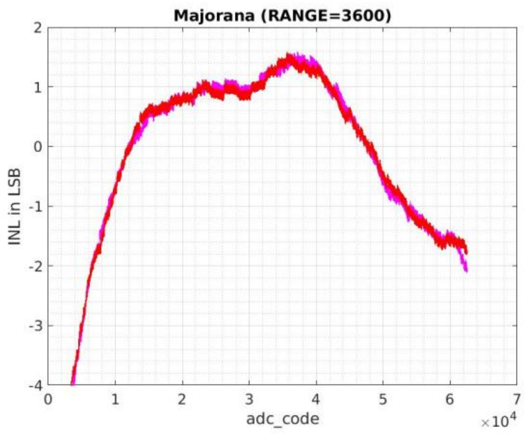


Figure 116 Determined INL trends with the Majorana Approach (I). In case of this figure the almost complete code extend of the measured (histogram) data sets was included in the corresponding final analysis. The final straight line fit of the plotted trends uses a slf_start_code value of 3000 and a slf_end_code value of 62500.

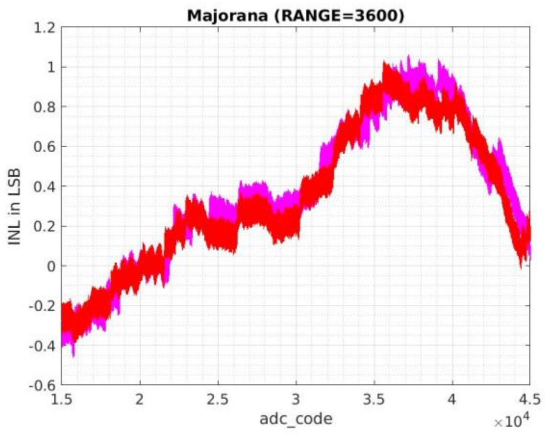


Figure 117 Determined INL trends with the Majorana Approach. In case of this figure only the measurement data (histogram) parts that belong to the code range between 15.000 and 45.000 were processed in the corresponding final analysis (straight line fit and subsequent INL trend correction). The remaining code parts were excluded as it is assumed that their behavior is interfered by an unknow error source.

But the Majorana results is always missing the supposed double peak behavior and never achieves the smaller INL trend value of the FC1.0 nominee approach. In Figure 116 intimations of the double peak exist. In order to prove (or disprove) the double peaks existence the analysis corresponding to Figure 117 was started. It uses the same data sets, but in the final INL Analysis (“straight line fit”) just the ADC codes in the range of 15000 – 45000 are included. This procedure is motivated by suspicion that codes outside of this area are strongly influenced by a nonlinear driver behavior and this behavior effectively covers the double peak. Figure 117 disproves the double peaks existence. In order to confirm the supposed architecture of the Ltc2107 (ADC under test), the INL trends zoom-in plots of Figure 118 and Figure 119 were generated. In them a basic pattern of the 8×4 basic structure can be seen. As in Figure 117 a slight structure repetition pattern of 2048 code width can be seen, the determined INL trend confirms the initial suspicion that the ADC design uses three approximations to digitize the analog input voltage (3bit×2bit×11bit).

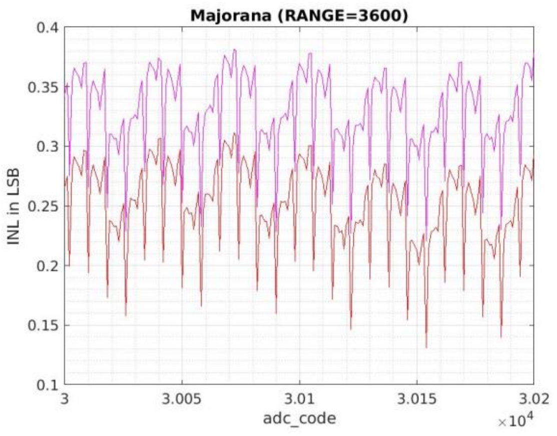


Figure 118 Zoom-in in the determined INL trend of Figure 117

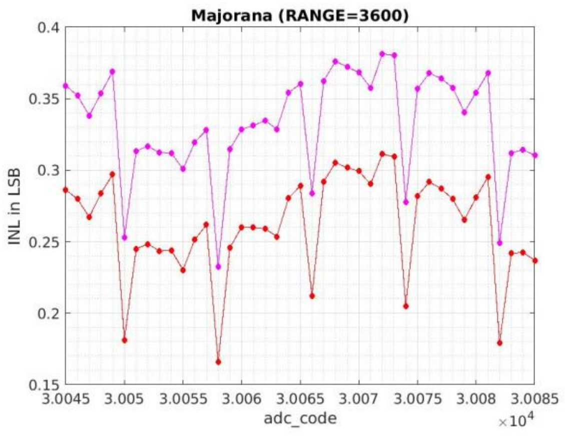


Figure 119 Magnification of the determined ADC basis structure

8.5.3 GERDA results

In this thesis the GERDA setup was introduced to verify the determined INL trends of the FC1.0 procedure. In the GERDA setup the nonlinearity of the issued DC voltage is vanquished by the ~5.5-digit resolution (~19 bit) of the implemented DMM (mainly a result of the issued noise level). The mean ADC code of every DC level is determined out of ~13.1M samples and consequently its standard derivation is much lower than the 1 LSB_{RMS} of the raw ADC code stream. The error of the determined INL trend therefore should be less than ~0.125 LSB and is consequently negligible. The determined GERDA procedure INL results are shown in Figure 120. Figure 121 is a zoom-in into the same data.

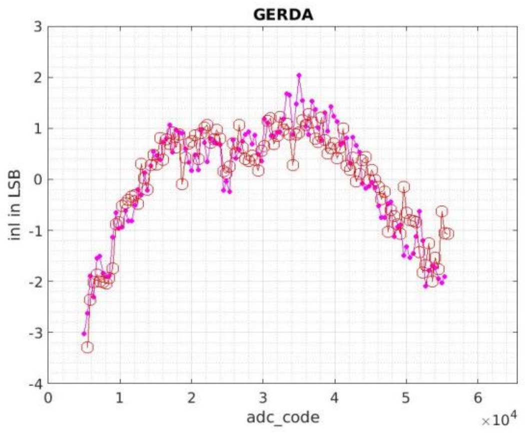


Figure 120 Determined INL trends with GERDA procedure The plot shows two independent INL measurements (with identical settings) that were done with the GERDA setup.

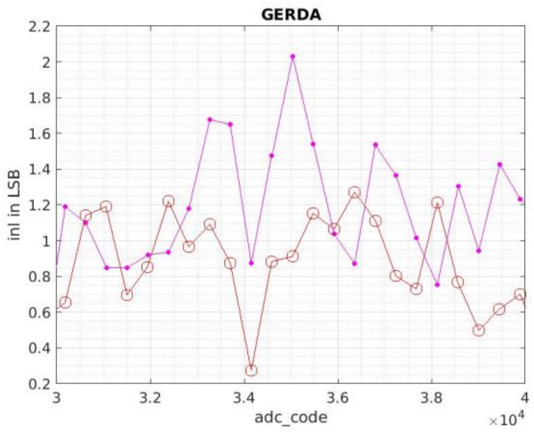


Figure 121 Determined INL trends with GERDA procedure (zoom-in). The Ltc2107 has a typical DNL smaller than 0.25 LSB [36]. The noise induced uncertainty is around 0.125 LSB. As the issued DC voltages can have a drift and consequently in the second measurement series a vicinity code can be meet, the DNL will cause (the impression of) a deviation (in a magnitude of ~0.25 LSB) between both delineated INL trends in Figure 121. A look into Figure 121 shows several ADC codes where both trends have a deviation bigger as 0.375 LSB. This is a hint that another effect also influences the measurement results.

Both trends of Figure 120 are not eligible to prove/disprove the existence of the “FC1.0 procedure” proposed double peak structure. Both trends agree with the rough global behavior of the FC1.0 and Majorana approach. Surprisingly the determined INL trends differ stronger as one may expect. This should not be caused by the method. The internal design and behavior (calibration, dithering, …) of the Ltc2107 is unknown. That may explain the missing reproducibility.

8.6 Setup for a γ-ray spectrum measurement with a HPGe detector

In this chapter the focus is on the HPGe-detector measurements with the FC2.0 prototype which were done at the TUM physic department (Munich-Germany). The initial intention of these measurements was a quality assessment of the FC2.0 prototype and not a utilization as INL verification backup. The FC2.0 prototypes data acquisition proceeded without any issue and therefore can be seen as another verification of the developed design. The measurements confirmed older signal quality measurements which were done at MPIK. In the result discussions (chapter 8.7) the determined signal quality has a minor part. The focus in this chapter is clearly on the determined INL trends verification (done in an offline analysis). In the scope of this thesis, it is not useful to discuss the used (digital) signal shaping strategies, (analog) amplifiers and the typical issues of the used HPGe-detector type, as still potential error sources would remain that prevent an accurate comparison with the FC1.0 DAQ. Simply said, the determined FC2.0 prototype signal quality still can compete with FC1.0 ADC, likely it is even capable to overtake it.

The HPGe-detector setup (Figure 122) is like the "Nonlinearity Determination" setups of chapter 8.5. The major difference is the signal source and the final signal analysis chain to obtain the γ-ray spectrum.

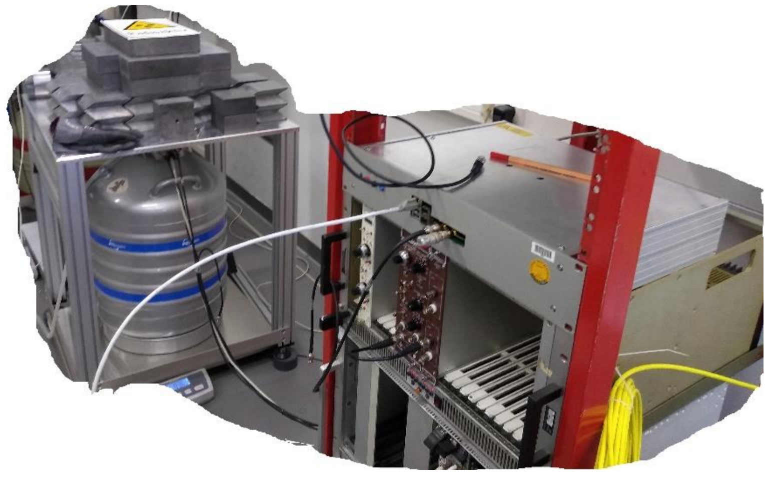


Figure 122 FC2.0 prototype and the used HPGe-detector in the TUM setup. The HPGe-detector is shielded by lead vessel to reduce the background. The measurements were done at the physics department of the TUM university (Munich-Germany) as at MPIK was no comparable HPGe-detector available. In the FC2.0 prototype the Trenz Modul TE0803-04-4AE11-A and the FADC card "DC2266A" were implemented. The implemented firmware design agrees to Figure 11.

The HPGe-detector and the thereon placed γ-ray radiation source are not visible in Figure 122. It only shows the corresponding Dewar vessel and lead tiles (to prevent background that can be induced by external radiation). The positive channel input of the FADC card is directly connected with the single ended output signal of the preamplifier (by the black cord in Figure 122). The negative input of the FADC card (still "DC2266A") is terminated against ground. In the TUM setup therefore only the half input range of FADC card can be used and a SNR quality loss is the consequence.

The read-out routine of the FC2.0 prototype offers the opportunity to concurrently determine the γ-spectrum out of the received traces. In this setup the procedure is different. Like in the "Event Building Performance Evaluation"" setup (chapter 6.5.3) the read-out server routine stores the traces of all accepted events on the read-outs server hard drive. After the accomplishment of the read-out routines job, an also available analysis routine reads in the stored traces and determines by a given signal processing chain the corresponding γ-ray spectrum. The signal chain has besides the default shapers (Gaussian a. Trapezoidal), a Pole-Zero Compensation filter (including DC correction) and an extremum finder. Storing the traces and doing the analysis afterwards offers the advantage of multiple analysis. By checking the influence on the filter settings of the spectrum peaks behavior sometimes a conclusion on potential DAQ or detector issues is possible.

8.7 Results of γ-ray spectrum measurment

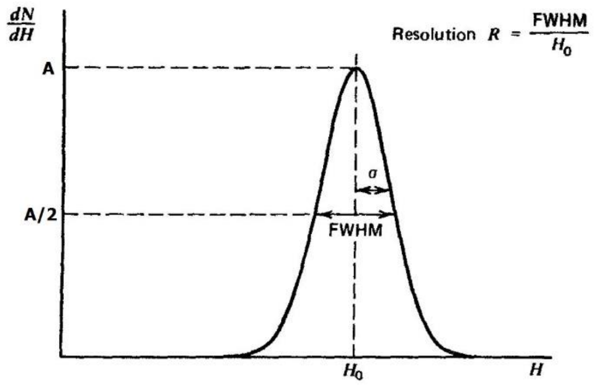


Figure 123 Connection between FWHM and resolution for a Ge-detector and mono energetic gamma rays [2 S. 114]

The precise measurement settings of the final analysis data set are listed in Table 23. The γ-ray spectrum generation was done by the FC2.0 prototypes analysis server routine. The underlying DSP C library was used in various publications of the GERDA collaboration and its DSP basics are discussed in appendix chapter A.18. The nonlinearity and resolution determination, on basis of the generated spectrums, were done by a MATLAB Routine. In MATLAB the essential fitting and peak detection functions are implemented by default. Its energy model assumes that the background subtracted γ-ray line's peak (in a spectrum) can be described by a Gaussian shape (Figure 123 and equation (37)). In a perfect matched setup with well-behaving detectors this assumption is a good approximation [2 S. 114]. Other authors (for instance [34]) use the more precise Levenberg-Marquardt model that besides the Gaussian also includes a step-like shelf and a Hypermet function. This approach has the advantage that a previous background cancellation is not necessary. In the MATLAB routine the fitted standard derivation σ relates to the FWHM parameter ("Full Width Half Maxima") simply by equation (38). The resolution can be obtained by dividing FWHM by the γ-ray energy of its peak.

$$G(H) = \frac{A}{\sigma \cdot \sqrt{2 \cdot \pi}} exp\left(-\frac{(H - H_0)^2}{2 \cdot \sigma^2}\right) \tag{37}$$

$$FWHM = 2 \cdot \sqrt{2 \cdot ln2} \cdot \sigma \approx 2.35 \cdot \sigma \tag{38}$$

Table 23 Measurement setting of final TUM data set "RawEthernet_NewServer_Approach_25_Sep_20_08:43:05.bin". A simplified version of the used signal processing chain was already introduced in chapter 7.6. In contrast to the simplified version of chapter 7.6 the signal chain of the analysis routine also has a simple "pile-up rejection" implemented. The analysis routine has several procedures to do the baseline subtraction. In the case of this thesis only the baseline level values that were determined by the firmware implemented baseline follower are used.

Data set	RawEthernet_NewServer_Approach_25_Sep_20_08:43:05.bin
Detector	(Inverted Coax Type)
Preamplifier Output	Single Ended Output Signal
FADC Board	DC2266A
Full Energy Range of Setup	~6.3 MeV
Radiation Sources	Co60 & Th228
Event Rate (including detected but not accepted events)	~610 Hz
Event Number	500.000
Fall time τ of preamplifier (DAQ (as PZ))	~137 µs
Shaper Types (Analysis Routine)	Gaussian and rounded top trapezoidal (TZ)
Shaping Times (Analysis Routine)	3-30 µs
Shaping Time (DAQ)	5 µs
Baseline determination approach	Baseline follower output
Pile up rejection	In analysis routine
Trace length	32768

Note that the used analysis routine does the pile up rejection on basis of header information which is provided by the FC2.0 trigger acceptance block firmware (appendix chapter A.5). At the low event frequencies of the TUM setup a pile up interference is unlikely and hence even an analysis that accepts events with pile up would obtain similar FWHM results. In Figure 124 the spectrum with the best determined FWHM of the 1333 keV peak of Co60 is illustrated. It used a rounded top TZ shaper with an integration and collection time of 15 µs. Such a long shaping time is unexpected and hence a hint that the setup has a potential noise issue. Therefore, also a setup error evaluation was done. Due to limited time the noise source(s) could not be identified. Another potential error source, likely less important than the previously mentioned noise, are charge trapping effects of the used detector. As these errors are result of the setup and not of the FC2.0 prototype, a discussion of them is not done. Figure 125 shows the corresponding peak and a Gaussian fit that was used to determine its FWHM value (2.09 keV).

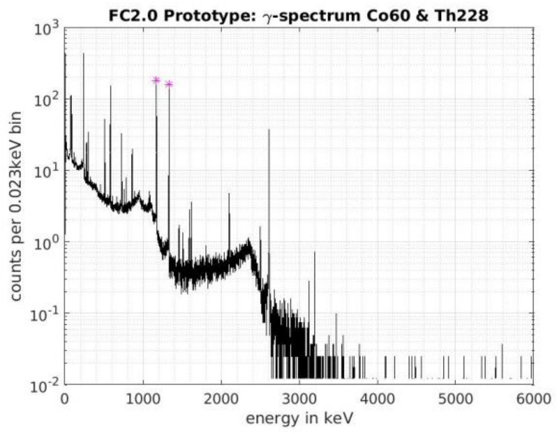


Figure 124 γ-ray spectrum of the dataset "RawEthernet_NewServer_Approach_25_Sep_20_08:43:05.bin" with a trapezoidal shaper. The peaks that were marked by stars are the automatically recognized key peaks of the spectrum. As their energy is known, they can be used to check the linearity of the DAQ.

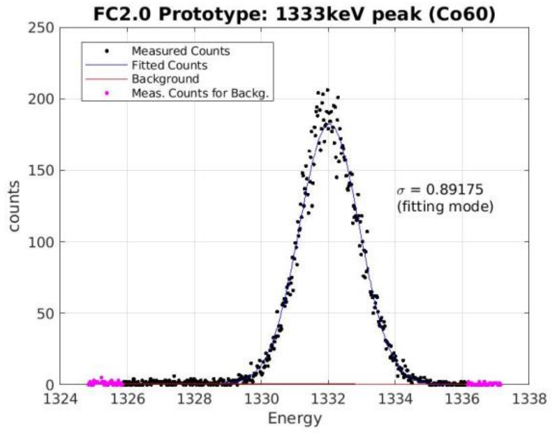


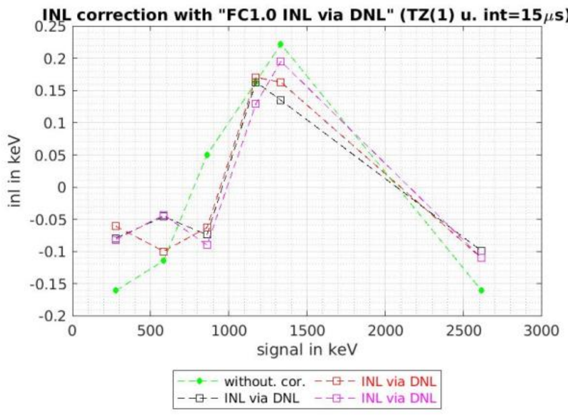
Figure 125 1333keV peak and its determined Gaussian fit A feasible approach of the Gaussian peak model assumes that the spectrum background is removed. Thus, the MATLAB routine also tries to determine the background level (red line in figure) before it determines the corresponding Gaussian fit of the peak (blue line in figure). The detailed procedure is described in [3].

The FWHM value is mostly given by the noise behavior of the pre-amplifier and detector specific effects. A frequently important HPGe-detector issue is its charge trapping behavior (tightly connected with the ballistic deficit issue in signal shaping). Charge trapping ultimately influences the shape of the γ-ray spectrum peaks. Hence charge trapping can influence the linearity of the γ-rays spectrum determined energy relation[19]. Consequently, a flat top trapezoidal shaper might enable a better resolution in future analysis runs. Another error source that can degenerate the linearity is the electronics nonlinearity. In the Majorana publication [8] the determined INL trend was amongst others verified by its capability to correct the nonlinearity of a measured γ-ray spectrum[20].

In this thesis we will pursue the same approach to assess the determined INL trends of the "FC1.0" chapter 8.5.1 and the "Majorana" chapter 8.5.2. As verification data set the γ-ray spectrum out of Figure 124 was chosen. In order to correct its ADC nonlinearity behavior the determined INL trends of Figure 108 (page 136) and Figure 116 (page 139) were used. The results of the "V1" INL correction approach (procedure described in chapter 7.6) are shown in Figure 126 and Figure 127.

[19] Note that the charge trapping only explains why the investigated TZ shapers have a better FWHM performance than comparable gaussian shapers. Significant charge trapping effects that comprise a time of 15µs are rather unlikely. The high integration time of 15µs is likely a consequence of the induced noise. In an investigation measurement – that differed from the shown TUM setup and hence the investigated results are not totally adoptable – serious "noise" frequencies in the kHz region could be confirmed. It's also possible to estimate the magnitude of the charge trapping effects by considering the low side tails of gamma-ray peaks (spectrum must be determined by a shaper without ballistic deficit correction (→ "gaussian shaper)). The results of this estimation confirmed that in the TUM setup noise is the main contributor of the FWHM degeneration.

[20] Another verification approach in [8] used a precision pulser. At MPIK a pulser with sufficient resolution is not available. Pay heed that [8] only needed to verify the determined INL trend of a 14bit ADC and not a 16bit ADC like in this thesis.



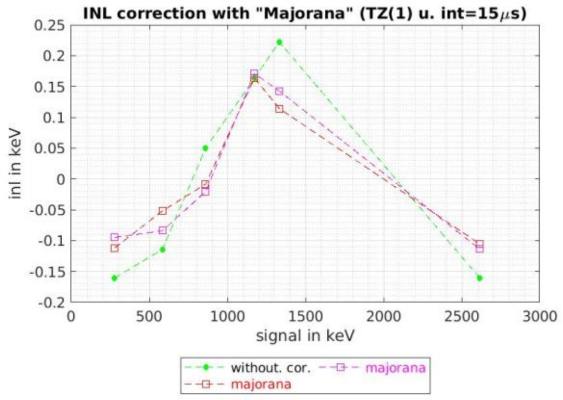


Figure 126 Nonlinearity of the determined γ-ray spectrums (I). The green trend line indicates the uncorrected nonlinearity trend of Figure 124. The red, black and magenta lines show the nonlinearity trend after its ("V1" approach (chapter 7.6)) correction with the same colored INL trends of Figure 108 (page 136). To do the nonlinearity determination the MATLAB routine must extract the peak position of the Co60 & Th228 out of the determined gamma ray spectrums (example is illustrated in Figure 124). To simplify the routine only the 6 major peaks of the spectrum are observed. As the routine knows the accurate energy of the peaks, it can determine their nonlinearity (deviation to the linear fit).

Figure 127 Nonlinearity of the determined γ-ray spectrums (II). The green trend line indicates the uncorrected nonlinearity trend of Figure 124. The red and magenta lines show the nonlinearity trend after its ("V1" approach) correction with the same colored INL trends of Figure 116 (page 139).

The determined INL trends are narrowly eligible to correct the ADC nonlinearity influence on the energy relation of the γ-ray spectrum. The remaining nonlinearities could be caused by the leftover nonlinearities of the chosen correction strategy that assumes a static, memory-less ADC nonlinearity behavior. Another potential issue are the systematic errors of the determined ADC INL trends itself. As the simulations imply that the systematic errors magnitude is in the case of investigated ADC (Ltc2107) just a slight fraction of the real INL trend this cause is merely unlikely. In the discussion of the determined INL trends it was already emphasized that the determined trend is likely strongly influenced by the input amplifier of the used FADC card. As the electronic conditions of the input amplifier in the TUM setup (only usage of one of the differential input pins of the FADC card and CM voltage of 0 V) and the INL determinations setup (both input pins of the FADC card are feed with an active signal and a CM voltage of 0.9 V) the differences in the gain behavior of the FADC cards input stage are likely significant. The nonlinearity trend of the input amplification stage in both setups likely differs strongly and therefore the total INL trends of both setups are not really comparable. Obviously, this harms an INL correction. Another significant error source of the setup are the potential charge trapping effects of the detector. Their "nonlinear" peak shape distortions cannot be corrected by the ADC nonlinearity.

8.8 Summary INL determination

The nonlinearity of the ADC(s) digitizing the detector signal(s) is a limiting factor in particular for precision spectroscopy, where nowadays 16 bit high speed ADCs are used. Various methods were developed to characterize and correct the nonlinearity, either by measuring the differential nonlinearity (DNL) (the input voltage step to go from one ADC code to the next (equation (8)), or directly the integral nonlinearity (INL) (the deviation from a linear relation between input voltage and ADC code (equation (5)).

- A well-monitored DC voltage can be applied to the input, and the resulting mean ADC code measured. DC voltage meters with precision exceeding that of a 20 bit ADC are readily available. This technique directly measures the INL and was already used by the GERDA collaboration (chapter 7.3). It is referred to as the GERDA INL technique. The main disadvantage is that stepping the DC voltage through the entire ADC range with bit-level resolution is quite time-consuming.
- Much faster are classic histogramming methods where a voltage ramp that covers the full input range of the ADC is digitized, and the values are histogrammed. Assuming a perfect ramp, the "count" frequency at which a given ADC code appears is proportional to the voltage step between this value and the next, i.e. measures the DNL. The INL is obtained by integrating the DNL, ignoring parts that can be absorbed into a redefinition of ADC gain and offset. The difficulty of such an approach is that "perfect" voltage ramps with better than 14 bit resolution are challenging and expensive to generate.
- To circumvent the "perfect" voltage ramp requirement, new methods were developed. Effectively these methods claim: In the histogramming method, the nonlinearity of the voltage ramp can be compensated by using a voltage ramp that covers only a small part of the ADC range, superimposed on a voltage offset that is slowly swept through the entire ADC range. Since – averaged over all offset values – all voltage levels of the ramp contribute to the counts for a given ADC value, ramp nonlinearities are expected to average out.
  - In Majorana procedure this is achieved by superimposing a fast triangular waveform with amplitude of about 10% of the ADC range, onto a slow sawtooth sweeping the entire range ("Majorana INL technique" (chapter 7.5).
  - In the "FlashCam 1.0 INL technique" (chapter 7.5), a sawtooth pulse (labeled as "triangle pulse") of about 10% amplitude is superimposed onto a DC level, that is stepped through the full ADC range in sufficiently fine steps.

The Majorana procedure stands out as their developers explicitly claim that their method is also eligible to determine the dynamic nonlinear behavior of the ADC. The INL might also depend on the slew rate of the signal. For germanium spectroscopy a pulser would be needed that mimics a detector-like pulse: a sharp rise of a few 100 ns and a long exponential decay with typically 100 µs decay constant. The shape must be stable and the amplitude of this pulser must be controlled to much better than 16 bits ("PB-5 technique"). This was attempted by feeding a controlled, mechanically switched voltage – a precision DC source connected via a mercury relay – into a passive shaper network. A main disadvantage of this technique is again the time consumption.

Validating the determination and correction of ADC INL in an experiment is non-trivial (e.g. [8]); known energies of gamma-ray lines can e.g. be used to check the linearity. In experiments where signals are read out with two gain channels, the comparison of amplitudes in the range where the two channels overlap can be used to monitor the quality of the INL correction.

In this thesis various methods of DNL and INL determination were enhanced and adapted on the FC 2.0 prototype system. Before the approval of the latest "Majorana" code, the self-deployed "FC1.0 INL via DNL" method was the most accurate procedure that was available at MPIK. A further novel aspect of this thesis is that – for the first time, to the knowledge of the author – extensive simulations were carried out to compare the different techniques. The simulations not only included the ADC with its INL, but also nonlinear pulser distortions, noise superimposed to the input signal and phase shifts between ADC and ramp.

The results of the simulations proved the feasibility of the developed procedures, but it also demonstrated that the procedures have systematic issues besides the uncertainty induced by noise. The determined INL trend in the simulations have mean total error that is much lower than 0.1LSB (typically ~0.03LSB). The Majorana technique offers a slightly better result than the "FC1.0 INL via DNL" procedure and is also less dependent on the INL trend of the ADC under investigation. The suggested FC1.0 successor procedures are partly capable to overtake the performance of the Majorana technique, but their implementation in a real setup is much more challenging than the original FC1.0 technique. The most serious disadvantage of the FC1.0 procedure is its DNL smoothing that depends on the noise level. The in [8] proposed Majorana approach has the disadvantage that its histogramming only can use a small fraction of the waveforms and additionally the selectable trace length is limited by the inclination of the slow sawtooth. This negatively influences the measurement time. In simulations also "fast measurement" settings were investigated that violate the restrictions of [8]. In these simulations could be shown that if some restrictions are accepted, the faster Majorana procedure can be a feasible option.

In the experimental setups the Majorana procedure had the most plausible results although its determined INL trend did not include all key properties of the data sheet example. The determined INL trends of the FC1.0 procedure showed a dependence on the common mode offset voltage (of the "ADC input signals") and varied significantly, but also showed key properties of the data sheet example that were missing in the determined Majorana trend. Accurate INL correction therefore implies that the common mode is either stabilized, or monitored. In the current setup the CM offset quality is unclear. The common-mode offset can be optimized towards minimal INL of the ADC. In all investigated methods the determined INL trend is likely influenced by the INL behavior of the input amplifier. Also the energy validation of the determined INL trends for all procedures still showed significant deviations.

Table 24 is a comparison of all relevant properties of the investigated INL determination method. Besides the already extensively discussed methods (Majorana, FC1.0, GERDA) the summary also comprises the "PB-5" method. In contrast to other procedures, it is limited by the pulsers nonlinearity behavior. This is also the main reason why it was not discussed in the main part of the thesis. The main part of the thesis is intentionally limited to procedures that are capable to compensate the nonlinearity of the used pulser(s). An extensive discussion of the PB-5 procedure is attached in the appendix (chapters A.16 - A.17).

Table 24 Summary of all investigated INL determination procedures

	Majorana	**FC1.0 INL via DNL**	**GERDA**	**PB-5**
Instrumentation requirement	Default arbitrary pulse generator	Default arbitrary pulse generator with remote control	Power supply (with remote control) & corresponding mixer circuit, high precision DMM (with remote control)	Ultra-low distortion pulser with exponential shape and fine (remotely) selectable amplitudes
Quality in Simulation	Mostly (slightly) better than FC1.0	Mostly (slightly) worse than Majorana	Not simulated	Not simulated
Obstacle that was not considered in simulation	Non-steady scanning of the ADC range may degenerate the performance method	Crosstalk effects were only superficially investigated (as "app3 setup")	Not simulated	Not simulated
Potential error sources in setup	Common mode offset voltage may not be sufficient stable Crosstalk effects	Accurate fast pulse shape may fluctuate or be dependent on time Crosstalk effects	-	Pulse shape is not stable enough and the precision of the amplitude may be much more lower than required
Dynamic nonlinearity	Investigation of the pulse amplitude and the slew rate influence possible (several measurement series required)	Investigation of the pulse amplitude and the slew rate influence possible (several measurement series required)	Only static nonlinearity	Pulse shape can be made similar to Pulse Signals of HPGe-Detectors (fast rise in the range of several (up to 100ns) nanoseconds, fall in several (up to 100) µs!

	Majorana	**FC1.0 INL via DNL**	**GERDA**	**PB-5**
Limits in ADC code range	“faster converge” approach requires the abduction of a significant code range	Only a small noise fringe must be abducted	Complete code range possible	Only a small noise fringe must be abducted
Determined INL Trend Table has Integer ADC Code	Yes	Yes	No	No
Precision depends on:	Noise, RANGE parameter[21]	Noise, Pulse Amplitude, ADC INL trend itself	Noise	Noise, Pulser Nonlinearity
Potential improvements	-	“bX FC1.0 INL Smart” / “id FC1.0 INL”	Substitution of power supply/DMM by high resolution DAC and high resolution ADC that are directly under the control of the client routien	“several regions” / “mercury relay & passive shaper network”

[21] In the thesis the influence of the Range parameter was not discussed. In Figure 93 the mean INL error of Majorana was dependent on the Pulse Amplitude. It can be shown that this is at least partly caused by the required changes of the Range parameter.

8.9 Final prospect

Not just professional Zynq modules are available. In the meantime, Zynq-7000 devices are eagerly used in the Maker Scene, e.g. the “Cora Z7” module. “Cora Z7” uses the simplest Z7k devices (Z-7007S) and is therefore cheaply available (151€). An important criterion in the Maker Scene is the price and the usability of the module. The “Cora Z7” is fully equipped and no Carrier Module is required. Obviously, such a cheap module can be interesting for particular experiments too.

In this thesis the focus was on SoC products of Xilinx. Xilinx continuously extends its product line, for instance Xilinx recently introduced the ACAP family (“Adaptive Compute Acceleration Platform”) (with an ARM-Cortex-A72 CPU). Due to its extensive amount on hardwired computing engines (corresponding signal streams are routable via a programmable Network on Chip), the ACAP architecture is hardly comparable with a classical FPGA architecture and therefore could be a better match for the initially intended “on-board-DSP” tasks. The ACAP devices are not intended to replace the Zynq SoC families – the extent of their FPGA fabric (especially FPGA-RAM and IO-pins) is significantly lower and thus they are not eligible as FC1.0 successor. Therefore, the initially pursued on-board processing FC DAQ is still not feasible and only a distant dream. Xilinx itself states as ACAP example applications data center networking, storage and compute acceleration, AI acceleration from edge to cloud and autonomous driving. In this thesis the ACAP family is hence a symptom of the high competition between Xilinx and Intel.

The competition is not only limited to the new ACAP devices, especially in the SoC product selection a high competition between Intel and Xilinx exists. Intel also has several SoC families that comprise an ARM-Cortex-A9 and Cortex-A53 CPU. The Intel “Cyclone V” and Xilinx “Zynq-7000” “Cortex-A9” SoCs are roughly similar and it is expected that the developed FC2.0 prototype firmware design should be adaptable between them. The “Cortex-A53” SoCs differ much more. For instance, in 2016 announced “Stratix 10 SoCs” the ACP port is missing. Its available ACE Lite Port is similar to the not discussed HPC ports of the ZU+ devices. As the ACE lite port is connected to the “on chip memory” of the Stratix 10 device, it enables an implementation of a “Mock-ACP-CMA”. With the current expertise it is hard to forecast the performance of such a “Mock-ACP-CMA” – there are specifications that imply a higher bandwidth, but also points that advocate the opposite behavior. An implementation of the developed FC2.0 prototype design into the Stratix 10 SoCs is therefore critical - but not impossible - and thus not intended.

In the Legend-200 experiment uses FC1.0-Germanium as DAQ. The FC1.0 DAQ complies with all requested requirements, mostly FC1.0 even exceeded them. Rather the performance of FC1.0 will likely cover the requests of Legend-1000. As with the expiring of the HESS experiment, MPIK will have a lot of free available FC1.0 boards that also can be used in Legend-1000. So currently no pressure exists to develop a FC1.0 Germanium successor (in the thesis called “final FC2.0 DAQ”). In the beginning of the thesis the situation was different, and Legend was one of the main objects of the new FC2.0 DAQ. The setup of the Legend-1000 is not ultimately decided yet, but currently it looks like the unexpected success of FC1.0-Germanium prevents the development of a FC2.0 DAQ for Legend-1000. This is not a lethal point against FC2.0 – the announced end of lifetime of the Spartan-6 devices still is unchanged and therefore MPIK still pressures to develop a new DAQ.

The investigation of the ADCs INL behavior has two main interests: First of all, it is tried to select an ADC whose INL behavior is not detrimental in Germanium-detector measurements, secondly it is intended to use the determined INL trend to improve the spectrums nonlinearity. INL determination is likely one of the most underestimated measurements tasks in ADC characterization. The task sounds trivial, but due to the tough requirements on modern high-speed ADCs that are used in Germanium spectroscopy the opposite is the case. In common guides, published by ADC manufactures, the perils are only considered superficially and in a way that their mitigation looks easily controllable. In practice this is not the case. My first expertise with INL determination of modern high-speed ADCs I acquired during my scholarship at Lawrence Berkeley National Laboratory (Berkeley, CA). At this time, I used an approach that is similar to the suggested Majorana procedure, but instead of a fast sawtooth pulse shape an exponential as fast pulse was used. The setup was promoted as tedious, but also as bulletproof and therefore looked like a good choice. It was not. The proposed Majorana setup picks up some of the assumed errors [3].

The “FC1.0 INL via DNL” and “Majorana” INL determination procedure have an impressive accuracy, but they also have systematic errors. As both procedures have a low mean error, the applied corrections procedures did not consider the systematic errors of the determined INL trends. The systematic errors likely play a minor part in the published results of this thesis. In the “Majorana” procedure the behavior of the systematic errors is currently investigated by further simulations. In noise-free simulations could be shown that the firstly assumed causes, only contribute a minor part to total error, but the simulations did not unveil the cause. The proposed INL determination methods hence offer still opportunities of more elaborated investigations.

Characterization the dynamic nonlinearity trend caused by an exponential pulse signal is tried by several groups by several approaches – e.g. by [38] – and was not limited to MPIK and LBNL. Effectively all exponential pulse setups known by the author struggle to achieve a reliable trend of a 16bit ADC. In principle it would be desired to transform the “Majorana” and “FC1.0” procedure in an approach that uses an exponential as fast pulse. In simulations such modified tries were successfully completed. Unfortunately, the modifications also increased the determined mean INL error value. Overturning the limits of these procedures will be the next challenge.

List of figures

List of tables

List of references

1. **Spieler, Helmuth.** *Semiconductor Detector Systems.* Oxford : Oxford University Press, 2005.

2. **Knoll, Glenn F.** *Radiation Detection and Measurement (4th edition).* New York : John Wiley & Sons, Inc., 2010.

3. **Schuett, Mario.** *Improving the energy resolution of the GRETINA digital electronics for high gamma-ray rates.* Karlsruhe : Hochschule Karlsruhe – Technik und Wirtschaft , 2017.

4. **Baker, Bonnie.** Optimize Your SAR ADC Design. *Texas Instruments.* [Online] 15. March 2022. www.ti.com.

5. **Kester, Walter.** ADC Input Noise: The Good, The Bad, and The Ugly. Is No Noise Good Noise? *AnalogDialogue (Vol. 40).* [Online] February 2006. https://www.analog.com/en/analog-dialogue/articles/adc-input-noise.html.

6. **Pelgrom, Marcel.** *Analog-to-Digital Conversation (Third Edition).* Cham : Springer International, 2017.

7. **Maxim Integrated.** Effects of Digital Crosstalk in Data Converters. *Application note 1761.* [Online] 04. March 2022. https://www.maximintegrated.com/en/design/technical-documents/app-notes/1/1761.html.

8. **Abgrall, N., et al.** ADC Nonlinearity Correction for the MAJORANA DEMONSTRATOR. *IEEE Transactions on Nuclear Science (Volume 68, Issue 3).* March 2021, S. 359-367.

9. **Xilinx.** Zynq-7000 SoC. *Product Selection Guide.* [Online] 08. July 2021. https://www.xilinx.com/support/documentation/selection-guides/zynq-7000-product-selection-guide.pdf.

10. —. UltraScale Architecture and Product Data Sheet: Overview. *ds890.* [Online] 16. March 2021. https://www.xilinx.com/support/documentation/data_sheets/ds890-ultrascale-overview.pdf.

11. **Schulz, Peter und Naroska, Edwin.** *Digitale System mit FPGAs entwickeln.* Aachen : Elektor Verlag, 2016.

12. **Xilinx.** Zynq UltraScale+ Device Technical Reference Manual. *ug1085.* [Online] 18. October 2019. https://www.xilinx.com/support/documentation/user_guides/ug1085-zynq-ultrascale-trm.pdf.

13. **Patterson, David A. und Hennessy, John L.** *Computer Organization and Design The Hardware/Software Interface ARM Edition.* Cambridge : Morgan Kaufmann, 2017.

14. **Xilinx.** Zynq-7000 SoC Technical Reference Manual. *ug585 (v1.12).* [Online] October 17, 2017. https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf.

15. —. Zynq-7000 All Programmable SoC (Z-7007S, Z-7012S, Z-7014S, Z-7010, Z-7015, and Z-7020): DC and AC Switching Characteristics. *ds187 (v1.20).* [Online] 15. June 2017. https://www.xilinx.com/support/documentation/data_sheets/ds187-XC7Z010-XC7Z020-Data-Sheet.pdf.

16. —. Zynq UltraScale+ MPSoC Data Sheet: DC and AC Switching Characteristics. *DS925 (v1.12).* [Online] 18. June 2018. https://www.xilinx.com/support/documentation/data_sheets/ds925-zynq-ultrascale-plus.pdf.

17. **ARM.** AMBA AXI and ACE Protocol Specification AXI3, AXI4 and AXI4 Lite. *ACE and ACE-Lite.* [Online] February 22, 2013. https://developer.arm.com/documentation/ihi0022/latest.

18. —. AMBA overview. *(ARM developer).* [Online] 24. September 2021. https://developer.arm.com/architectures/system-architectures/amba.

19. *Energy and Performance Exploration of Accelerator Coherency Port Using Xilinx Zynq.* **Sadri, Mohammadsadegh, Weis, Christian und alia, et.** Copenhagen and Stockholm : ACM, 2013. FPGAworld '13: Proceedings of the 10th FPGAworld Conference. S. 1-8.

20. **Belloch, Jose A., Germán, Leon und alia, et.** Evaluating the computational performance of the Xilinx Ultrascale+ EG Heterogeneous MPSoC. *The Journal of Supercomputing volume 77.* 2021, S. 2124–2137.

21. *Analysis and Comparison of Attainable Hardware Acceleration in All Programmable Systems-on-Chip.* **Sklyarov, Valery, Skliarova, Iouliia und alia, et.** Madeira, Portugal : IEEE, 2015. 2015 Euromicro Conference on Digital System Design.

22. **Göbel, Matthias, Elhossini, Ahmed und alia, et.** A Quantitative Analysis of the Memory Architecture of FPGA-SoCs. *ARC 2017: Applied Reconfigurable Computing.* 31. March 2017, S. 241–252.

23. *High performance readout module based on ZYNQ with giga ethernet.* **Xue, Tao, Gong, Guanghua und Li, Jianmin.** Strasbourg, France : IEEE, 2016. 2016 IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop.

24. *Analysis and Optimization of I/O Cache Coherency Strategies for SoC-FPGA Device.* **Min, Seung Won, Huang, Sitao und alia, et.** Barcelona, Spain : IEEE, 2019. 2019 29th International Conference on Field Programmable Logic and Applications (FPL).

25. *Evaluating the Memory Architecture of Next-Generation FPGA-SoCs for HPC.* **Goebel, Matthias, Clasen, Kai Norman und alia, et.** Dublin, Ireland : IEEE, 2019. 2019 International Conference on High Performance Computing & Simulation (HPCS).

26. **Xilinx.** MicroBlaze Processor Reference Guide. *UG984 (v2020.2) November 18, 2020.* [Online] 18. November 2020. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2021_1/ug984-vivado-microblaze-ref.pdf.

27. **Dallet, Dominique and Machado da Silva, José.** *Dynamic characterisation of analogue-to-digital converters.* Dordrecht, Netherlands : Springer, 2005.

28. **Kester, Walt.** *The Data Conversion Handbook.* Oxford, US : Newnes, 2005.

29. **Siglent.** Siglent SDG6022X 200MHz Function / Arbitrary Waveform Generator. *Siglent.eu.* [Online] 26. April 2022. https://www.siglent.eu/product/1138498/siglent-sdg6022x-200mhz-function-arbitrary-waveform-generator.

30. **Maxim Integrated.** Histogram testing determines DNL and INL errors. *Tutorials 2085.* [Online] 07. March 2022. https://www.maximintegrated.com/en/design/technical-documents/tutorials/2/2085.html.

31. **Stanford Research Systems.** DS360 — 200 kHz low distortion generator. [Online] 01. June 2022. https://www.thinksrs.com/products/ds360.html.

32. **Burenkov, A. und Schwingenheuer, B.** GERDA Scientific / Technical Report: GSTR-05-015. *Test Report for Struck SIS3301 and XIA Pixie-4 FADCs.* s.l. : Gerda Collaboration, 2007.

33. **Berkeley Nucleonics Corporation.** PB5 data sheet. *Instruction Manual, Rev. 4.* [Online] 08. March 2022. https://www.berkeleynucleonics.com/sites/default/files/products/datasheets/pb-5_datasheet_1.pdf.

34. **Salathe, Marco und Kihm, Thomas.** Optimized digital filtering techniques for radiation detection with HPGe detectors. *(2nd revision).* [Online] 2. March 2016. https://arxiv.org/abs/1504.02039.

35. **Analog Devices.** DC2266A . *LTC2107 and LTC6409 Demo Board.* [Online] 26. April 2022. https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/dc2266a.html#eb-overview.

36. —. Datasheet ADC Ltc2107. [Online] 26. April 2022. https://www.analog.com/media/en/technical-documentation/data-sheets/2107fb.pdf.

37. **Siglent.** SDM3065X Series Datasheet. *Version 2021.5.* [Online] 11. July 2022. https://www.siglenteu.com//wp-content/uploads/dlm_uploads/2018/04/SDM3065X_DataSheet_EN02G-1.pdf.

38. **Holcer, A., Michaeli, L. und Saliga, J.** DNL ADC Testing by the Exponential Shaped Voltage. *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT.* 3, 2003, Bd. 52.

39. **Xilinx.** Efficient Shift Registers, LFSR, Counters, and Long Pseudo Random Sequence Generators. *XAPP 052.* [Online] 7. July 1996. https://www.xilinx.com/support/documentation/application_notes/xapp052.pdf.

40. —. 7 Series FPGAs Memory Resources User Guide. *UG473 (v1.12).* [Online] 27. September 2016. https://www.xilinx.com/support/documentation/user_guides/ug473_7Series_Memory_Resources.pdf.

41. —. UltraScale Architecture Memory Resources User Guide. *UG573 (v1.8).* [Online] 14. November 2018. https://www.xilinx.com/support/documentation/user_guides/ug573-ultrascale-memory-resources.pdf.

42. —. Macb Driver. [Online] 09. 07 2022. https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18841740/Macb+Driver.

43. —. PS and PL based Ethernet in Zynq MPSoC. *(superseded) version v.36.* [Online] 03. September 2020. https://xilinx-wiki.atlassian.net/wiki/pages/viewpage.action?pageId=18841830&pageVersion=36.

44. **Nicholson, P.W.** *Nuclear Electronics.* New York : John Wiley & Sons Ltd., 1974.

45. **Radeka, V.** Trapezoidal filtering of signals from large germanium detectors at high rates. *Nuclear Instruments and Methods (Vol. 99).* 23. November 1971, S. 525-539.

46. **Smith, Steven W.** *The Scientist and Engineer's Guide to Digital Signal Processing (Second Edition).* San Diego : California Technical Publishing, 1999.

47. **Jordanov, Valentin T. und Knoll, Glenn F.** Digital synthesis of pulse shapes in real time for high resolution radiation spectroscopy. *Nuclear Instruments and Methods in Physics Research A 345 (1994).* 18. January 1994, S. 337-345.

48. **Hegyesi, G. und Lakatos, T.** Charge pulse restorer for resistor feedback preamplifiers. *Nuclear Instruments & Methods in Physicss Research Section A (Volume 356).* 28. September 1995, S. 444-446.

49. *DIGITAL PULSE PROCESSOR FOR ION BEAM MICROPROBE AND MICRO X RAY FLUORESCENCE 2-D AND 3-D IMAGING.* **Bogovac, M, Jaksic, M. und alia, et.** Wien : International Atomic Energy Agency, 2010. Instrumentation for digital nuclear spectroscopy (IAEA-TECDOC-1706). S. 9-24.

50. **Kihm, Thomas und Bauer, Christian.** FC250b. *User manual.* Heidelberg : MPIK, 1. December 2016.

51. **ARM.** Arm Cortex Cortex-A53 MPCore Processor Technical Reference Manual. *Revision: r0p4.* [Online] 2018. file:///C:/Users/mario/Documents/Promotion/Starter_Kit_TE0720_Workstation_20210406/ARM_Cortex-A53_MPCore_TRM.pdf.

52. **Horowitz, Paul und Hill, Winfield.** *The Art of Electronics (Third Edition).* Cambridge : Cambridge University Press, 2015.

53. **Radford, David.** *INL.* [Befragte Person] Bernhard Schwingenheuer. 02. February 2022.

List of publications

1. **The GERDA Collaboration: M. Agostini, A. Alexander, G. Araujo, A.M. Bakalyarov, M. Balata, I. Barabanov, L. Baudis, C. Bauer, S. Belogurov, A. Bettini, L. Bezrukov, V. Biancacci, E. Bossio, V. Bothe, R. Brugnera, A. Caldwell, S. Calgaro, C. Cattadori, A. Chernogorov, T. Comellato, V. D'Andrea, E.V. Demidova, A. Di Giacinto, N. Di Marco, E. Doroshkevich, F. Fischer, M. Fomina, A. Gangapshev, A. Garfagnini, C. Gooch, P. Grabmayr, V. Gurentsov, K. Gusev, J. Hakenmüller, S. Hemmer, W. Hofmann, J. Huang, M. Hult, L.V. Inzhechik, J. Janicskó Csáthy, J. Jochum, M. Junker, V. Kazalov, Y. Kermaïdic, H. Khushbakht, T. Kihm, K. Kilgus, I.V. Kirpichnikov, A. Klimenko, K.T. Knöpfle, O. Kochetov, V.N. Kornoukhov, P. Krause, V.V. Kuzminov, M. Laubenstein, M. Lindner, I. Lippi, A. Lubashevskiy, B. Lubsandorzhiev, G. Lutter, C. Macolino, B. Majorovits, W. Maneschg, L. Manzanillas, G. Marshall, M. Miloradovic, R. Mingazheva, M. Misiaszek, M. Morella, Y. Müller, I. Nemchenok, L. Pandola, K. Pelczar, L. Pertoldi, P. Piseri, A. Pullia, C. Ransom, L. Rauscher, M. Redchuk, S. Riboldi, N. Rumyantseva, C. Sada, S. Sailer, F. Salamida, S. Schönert, J. Schreiner, M. Schütt et alia.** "Search for exotic physics in double-β decays with Gerda Phase II", 4. September 2022, e-Print: 2209.01671 [nucl-ex]

2. **The GERDA Collaboration: M. Agostini, A. Alexander, G. Araujo, A.M. Bakalyarov, M. Balata, I. Barabanov, L. Baudis, C. Bauer, S. Belogurov, A. Bettini, L. Bezrukov, V. Biancacci, E. Bossio, V. Bothe, R. Brugnera, A. Caldwell, S. Calgaro, C. Cattadori, A. Chernogorov, T. Comellato, V. D'Andrea, E.V. Demidova, A. Di Giacinto, N. Di Marco, E. Doroshkevich, F. Fischer, M. Fomina, A. Gangapshev, A. Garfagnini, C. Gooch, P. Grabmayr, V. Gurentsov, K. Gusev, J. Hakenmüller, S. Hemmer, W. Hofmann, J. Huang, M. Hult, L.V. Inzhechik, J. Janicskó Csáthy, J. Jochum, M. Junker, V. Kazalov, Y. Kermaïdic, H. Khushbakht, T. Kihm, K. Kilgus, I.V. Kirpichnikov, A. Klimenko, K.T. Knöpfle, O. Kochetov, V.N. Kornoukhov, P. Krause, V.V. Kuzminov, M. Laubenstein, M. Lindner, I. Lippi, A. Lubashevskiy, B. Lubsandorzhiev, G. Lutter, C. Macolino, B. Majorovits, W. Maneschg, L. Manzanillas, G. Marshall, M. Miloradovic, R. Mingazheva, M. Misiaszek, M. Morella, Y. Müller, I. Nemchenok, L. Pandola, K. Pelczar, L. Pertoldi, P. Piseri, A. Pullia, C. Ransom, L. Rauscher, M. Redchuk, S. Riboldi, N. Rumyantseva, C. Sada, S. Sailer, F. Salamida, S. Schönert, J. Schreiner, M. Schütt et alia.** "Pulse shape analysis in Gerda Phase II", 1. April 2022, Eur.Phys.J.C 82 (2022) 4, 284, e-Print: 2202.13355 [physics.ins-det]

3. **The LEGEND Collaboration: N. Abgrall, I. Abt, M. Agostini, A. Alexander, C. Andreoiu, G.R. Araujo, F.T. Avignone, W. Bae, A. Bakalyarov, M. Balata, M. Bantel, I. Barabanov, A.S. Barabash, P.S. Barbeau, C.J. Barton, P.J. Barton, L. Baudis, C. Bauer, E. Bernieri, L. Bezrukov, K.H. Bhimani, V. Biancacci, E. Blalock, A. Bolozdynya, S. Borden, B. Bos, E. Bossio, A. Boston, V. Bothe, R. Bouabid, S. Boyd, R. Brugnera, N. Burlac, M. Busch, A. Caldwell, T.S. Caldwell, R. Carney, C. Cattadori, Y.-D. Chan, A. Chernogorov, C.D. Christofferson, P.-H. Chu, M. Clark, T. Cohen, D. Combs, T. Comellato, R.J. Cooper, I.A. Costa, V. D'Andrea, J.A. Detwiler, A. Di Giacinto, N. Di Marco, J. Dobson, A. Drobizhev, M.R. Durand, F. Edzards, Yu. Efremenko, S.R. Elliott, A. Engelhardt, L. Fajt, N. Faud, M.T. Febbraro, F. Ferella, D.E. Fields, F. Fischer, M. Fomina, H. Fox, J. Franchi, R. Gala, A. Galindo-Uribarri, A. Gangapshev, A. Garfagnini, A. Geraci, C. Gilbert, M. Gold, C. Gooch, K.P. Gradwohl, M.P. Green, G.F. Grinyer, A. Grobov, J. Gruszko, I. Guinn, V.E. Guiseppe, V. Gurentsov, Y. Gurov, K. Gusev, B. Hacket, F. Hagemann, J. Hakenmueller, M. Haranczyk, L. Hauertmann, C.R. Haufe, C. Hayward, B. Heffron, F. Henkes, R. Henning, D. Hervas Aguilar, J. Hinton, R. Hodak, H. Hoffmann, W. Hofmann, A. Hostiuc, J. Huang, M. Hult, M. Ibrahim Mirza, J. Jochum, R. Jones, D. Judson, M. Junker, J. Kaizer, V. Kazalov, Y. Kermaïdic, H. Khushbakht, M. Kidd, T. Kihm, K. Kilgus, I. Kim, A. Klimenko, K.T. Knoepfle, O. Kochetov, S.I. Konovalov, I. Kontul, K. Kool, L.L. Kormos, V.N. Kornoukhov, M. Korosec, P. Krause, V.V. Kuzminov, J.M. Lopez Castano, K. Lang, M. Laubenstein, E. León, B. Lehnert, A. Leonhardt, A. Li, M. Lindner, I. Lippi, X. Liu, J. Liu, D. Loomba, A. Lubashevskiy, B. Lubsandorzhiev, N. Lusardi, Y. Mueller, M. Macko, C. Macolino, B. Majorovits, F. Mamedov, W. Maneschg, L. Manzanillas, G. Marshall, R.D. Martin, E.L. Martin, R. Massarczyk, D. Mei, S.J. Meijer, S. Mertens, M. Misiaszek, E. Mondragon, M. Morella, B. Morgan, T. Mroz, D. Muenstermann, C.J. Nave, I. Nemchenok, M. Neuberger, T.K. Oli, G. Orebi Gann, G. Othman, V. Palusova, R. Panth, L. Papp, L.S. Paudel, K. Pelczar, J. Perez Perez, L. Pertoldi, W. Pettus, P. Piseri, A.W.P. Poon, P. Povinec, A. Pullia, D.C. Radford, Y.A. Ramachers, C. Ransom, L. Rauscher, M. Redchuk, A.L. Reine, S. Riboldi, K. Rielage, S. Rozov, E. Rukhadze, N. Rumyantseva,J. Runge, N.W. Ruof, R. Saakyan, S. Sailer, G. Salamanna, F. Salamida, D.J. Salvat, V. Sandukovsky, S. Schoenert, A. Schueltz, M. Schuett et alia.** „The Large Enriched Germanium Experiment for Neutrinoless ββ Decay: LEGEND-1000 Preconceptual Design Report", 23. April 2021, e-Print: 2107.11462 [physics.ins-det]

4. **The GERDA Collaboration: M. Agostini, A. Alexander, G. Araujo, A.M. Bakalyarov, M. Balata, I. Barabanov, L. Baudis, C. Bauer, S. Belogurov, A. Bettini, L. Bezrukov, V. Biancacci, E. Bossio, V. Bothe, R. Brugnera, A. Caldwell, S. Calgaro, C. Cattadori, A. Chernogorov, T. Comellato, V. D'Andrea, E.V. Demidova, A. Di Giacinto, N. Di Marco, E. Doroshkevich, F. Fischer, M. Fomina,**

A. Gangapshev, A. Garfagnini, C. Gooch, P. Grabmayr, V. Gurentsov, K. Gusev, J. Hakenmüller, S. Hemmer, W. Hofmann, J. Huang, M. Hult, L.V. Inzhechik, J. Janicskó Csáthy, J. Jochum, M. Junker, V. Kazalov, Y. Kermaïdic, H. Khushbakht, T. Kihm, K. Kilgus, I.V. Kirpichnikov, A. Klimenko, K.T. Knöpfle, O. Kochetov, V.N. Kornoukhov, P. Krause, V.V. Kuzminov, M. Laubenstein, M. Lindner, I. Lippi, A. Lubashevskiy, B. Lubsandorzhiev, G. Lutter, C. Macolino, B. Majorovits, W. Maneschg, L. Manzanillas, G. Marshall, M. Miloradovic, R. Mingazheva, M. Misiaszek, M. Morella, Y. Müller, I. Nemchenok, L. Pandola, K. Pelczar, L. Pertoldi, P. Piseri, A. Pullia, C. Ransom, L. Rauscher, M. Redchuk, S. Riboldi, N. Rumyantseva, C. Sada, S. Sailer, F. Salamida, S. Schönert, J. Schreiner, M. Schütt et alia. “Characterization of inverted coaxial ^{76}Ge detectors in GERDA for future double-β decay experiments”, 07. Juni 2021, Eur.Phys.J.C 81 (2021) 6, 505, Eur.Phys.J.C 81 (2021) 505, e-Print: 2103.15111 [physics.ins-det]

5. **The GERDA Collaboration: M. Agostini, A. Alexander, G. Araujo, A.M. Bakalyarov, M. Balata, I. Barabanov, L. Baudis, C. Bauer, S. Belogurov, A. Bettini, L. Bezrukov, V. Biancacci, E. Bossio, V. Bothe, R. Brugnera, A. Caldwell, S. Calgaro, C. Cattadori, A. Chernogorov, T. Comellato, V. D’Andrea, E.V. Demidova, A. Di Giacinto, N. Di Marco, E. Doroshkevich, F. Fischer, M. Fomina, A. Gangapshev, A. Garfagnini, C. Gooch, P. Grabmayr, V. Gurentsov, K. Gusev, J. Hakenmüller, S. Hemmer, W. Hofmann, J. Huang, M. Hult, L.V. Inzhechik, J. Janicskó Csáthy, J. Jochum, M. Junker, V. Kazalov, Y. Kermaïdic, H. Khushbakht, T. Kihm, K. Kilgus, I.V. Kirpichnikov, A. Klimenko, K.T. Knöpfle, O. Kochetov, V.N. Kornoukhov, P. Krause, V.V. Kuzminov, M. Laubenstein, M. Lindner, I. Lippi, A. Lubashevskiy, B. Lubsandorzhiev, G. Lutter, C. Macolino, B. Majorovits, W. Maneschg, L. Manzanillas, G. Marshall, M. Miloradovic, R. Mingazheva, M. Misiaszek, M. Morella, Y. Müller, I. Nemchenok, L. Pandola, K. Pelczar, L. Pertoldi, P. Piseri, A. Pullia, C. Ransom, L. Rauscher, M. Redchuk, S. Riboldi, N. Rumyantseva, C. Sada, S. Sailer, F. Salamida, S. Schönert, J. Schreiner, M. Schütt et alia.** “Calibration of the Gerda experiment”, 02. August 2021, Eur.Phys.J.C 81 (2021) 8, 682, e-Print: 2103.13777 [physics.ins-det]

6. **The GERDA Collaboration: M. Agostini, A. Alexander, G. Araujo, A.M. Bakalyarov, M. Balata, I. Barabanov, L. Baudis, C. Bauer, S. Belogurov, A. Bettini, L. Bezrukov, V. Biancacci, E. Bossio, V. Bothe, R. Brugnera, A. Caldwell, S. Calgaro, C. Cattadori, A. Chernogorov, T. Comellato, V. D’Andrea, E.V. Demidova, A. Di Giacinto, N. Di Marco, E. Doroshkevich, F. Fischer, M. Fomina, A. Gangapshev, A. Garfagnini, C. Gooch, P. Grabmayr, V. Gurentsov, K. Gusev, J. Hakenmüller, S. Hemmer, W. Hofmann, J. Huang, M. Hult, L.V. Inzhechik, J. Janicskó Csáthy, J. Jochum, M. Junker,**

V. Kazalov, Y. Kermaïdic, H. Khushbakht, T. Kihm, K. Kilgus, I.V. Kirpichnikov, A. Klimenko, K.T. Knöpfle, O. Kochetov, V.N. Kornoukhov, P. Krause, V.V. Kuzminov, M. Laubenstein, M. Lindner, I. Lippi, A. Lubashevskiy, B. Lubsandorzhiev, G. Lutter, C. Macolino, B. Majorovits, W. Maneschg, L. Manzanillas, G. Marshall, M. Miloradovic, R. Mingazheva, M. Misiaszek, M. Morella, Y. Müller, I. Nemchenok, L. Pandola, K. Pelczar, L. Pertoldi, P. Piseri, A. Pullia, C. Ransom, L. Rauscher, M. Redchuk, S. Riboldi, N. Rumyantseva, C. Sada, F. Salamida, S. Schönert, J. Schreiner, M. Schütt et alia. "Final Results of GERDA on the Search for Neutrinoless Double-β Decay", 18. Dezember 2020, Phys.Rev.Lett. 125 (2020) 25, 252502, e-Print: 2009.06079 [nucl-ex]

7. **The GERDA Collaboration: M. Agostini, A. Alexander, G. Araujo, A.M. Bakalyarov, M. Balata, I. Barabanov, L. Baudis, C. Bauer, S. Belogurov, A. Bettini, L. Bezrukov, V. Biancacci, E. Bossio, V. Bothe, R. Brugnera, A. Caldwell, S. Calgaro, C. Cattadori, A. Chernogorov, T. Comellato, V. D'Andrea, E.V. Demidova, A. Di Giacinto, N. Di Marco, E. Doroshkevich, F. Fischer, M. Fomina, A. Gangapshev, A. Garfagnini, C. Gooch, P. Grabmayr, V. Gurentsov, K. Gusev, J. Hakenmüller, S. Hemmer, W. Hofmann, J. Huang, M. Hult, L.V. Inzhechik, J. Janicskó Csáthy, J. Jochum, M. Junker, V. Kazalov, Y. Kermaïdic, H. Khushbakht, T. Kihm, K. Kilgus, I.V. Kirpichnikov, A. Klimenko, K.T. Knöpfle, O. Kochetov, V.N. Kornoukhov, P. Krause, V.V. Kuzminov, M. Laubenstein, M. Lindner, I. Lippi, A. Lubashevskiy, B. Lubsandorzhiev, G. Lutter, C. Macolino, B. Majorovits, W. Maneschg, L. Manzanillas, G. Marshall, M. Miloradovic, R. Mingazheva, M. Misiaszek, M. Morella, Y. Müller, I. Nemchenok, L. Pandola, K. Pelczar, L. Pertoldi, P. Piseri, A. Pullia, C. Ransom, L. Rauscher, M. Redchuk, S. Riboldi, N. Rumyantseva, C. Sada, F. Salamida, S. Schönert, J. Schreiner, M. Schütt et alia.** "First Search for Bosonic Superweakly Interacting Massive Particles with Masses up to 1 MeV/c^2 with GERDA", 1. Juli 2020, Phys.Rev.Lett. 125 (2020) 1, 011801, Phys.Rev.Lett. 129 (2022) 8, 089901 (erratum), e-Print: 2005.14184 [hep-ex]

8. **The GERDA Collaboration: M. Agostini, A. Alexander, G. Araujo, A.M. Bakalyarov, M. Balata, I. Barabanov, L. Baudis, C. Bauer, S. Belogurov, A. Bettini, L. Bezrukov, V. Biancacci, E. Bossio, V. Bothe, R. Brugnera, A. Caldwell, S. Calgaro, C. Cattadori, A. Chernogorov, T. Comellato, V. D'Andrea, E.V. Demidova, A. Di Giacinto, N. Di Marco, E. Doroshkevich, F. Fischer, M. Fomina, A. Gangapshev, A. Garfagnini, C. Gooch, P. Grabmayr, V. Gurentsov, K. Gusev, J. Hakenmüller, S. Hemmer, W. Hofmann, J. Huang, M. Hult, L.V. Inzhechik, J. Janicskó Csáthy, J. Jochum, M. Junker, V. Kazalov, Y. Kermaïdic, H. Khushbakht, T. Kihm, K. Kilgus, I.V. Kirpichnikov, A. Klimenko, K.T. Knöpfle, O. Kochetov, V.N. Kornoukhov, P. Krause, V.V. Kuzminov, M. Laubenstein, M. Lindner,**

I. Lippi, A. Lubashevskiy, B. Lubsandorzhiev, G. Lutter, C. Macolino, B. Majorovits, W. Maneschg, L. Manzanillas, G. Marshall, M. Miloradovic, R. Mingazheva, M. Misiaszek, M. Morella, Y. Müller, I. Nemchenok, L. Pandola, K. Pelczar, L. Pertoldi, P. Piseri, A. Pullia, C. Ransom, L. Rauscher, M. Redchuk, S. Riboldi, N. Rumyantseva, C. Sada, F. Salamida, S. Schönert, J. Schreiner, M. Schütt et alia. „Modeling of GERDA Phase II data“, 24. März 2020, JHEP 03 (2020) 139, e-Print: 1909.02522 [nucl-ex]

Appendix

The following appendix chapters are extended explanations to design issues that consider the firmware. As in the main part of the thesis it was not possible to show all details of the design (and all corresponding performance results) the most important issues are discussed in the appendix.

A.1 Firmware design: ADC data read in dummy design

A functional schematic of the ADC (data read in) dummy block design is depicted in Figure 128. The dummy block has its internal trigger signal (orange line) and the corresponding "shaped" ADC dummy signal as output signal (magenta line). The dummies status and control register are interfaced with the CPU via the CPU bank block.

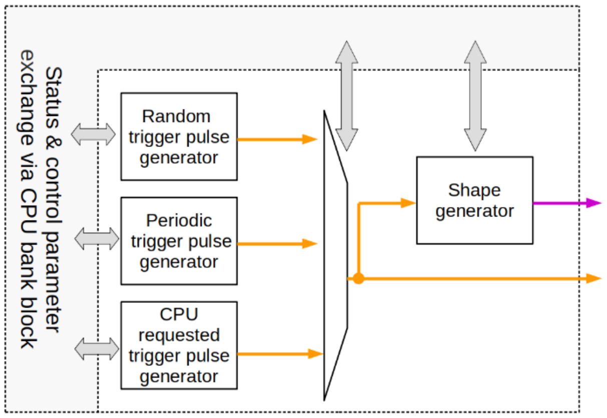

Status & control parameter exchange via CPU bank block
Random trigger pulse generator
Periodic trigger pulse generator
CPU requested trigger pulse generator
Shape generator

Figure 128 Functional subblocks of the ADC dummy

The dummy consists out of 4 major subblocks, at which three are responsible for generating triggers and one for the corresponding shaped signal output. In the scope of the thesis not all implemented subblock capabilities are important. In fact, the dummy was developed in several steps and the first dummy version solely had a "periodic trigger pulse generator", an elementary "CPU requested trigger pulse generator" and a counter as dummy signal output. The initially implemented "periodic trigger" block can generate trigger signals with a user selected frequency. The trigger signal can be used to initiate the output of a selected pulse shape type by the shape generator subblock. The dummy blocks have following function:

- (Pseudo) Random trigger pulse generator: Generates (pseudo) random trigger pulses with a user selectable frequency (feasible frequency range roughly between 10-10 MHz – theoretical range between 1-100 MHz (for a design with a system clock frequency of 200 MHz)). Default sub block design basis on a 28 bit Fibonacci linear-feedback shift register as described in [39]. The 28 bit are motivated by the assumed default system clock frequency of 200 MHz.

- CPU requested trigger pulse generator: Can cause a trigger signal immediately on the (board) CPUs command (via the CPU bank control register) and also has (simple) time-scheduled trigger features which can be programmed. The immediate CPU triggers are important for the DAQ maintaining services, the time-scheduled trigger features are a requirement for the verification of the “trigger acceptance control” block. The time-schedule programming is currently not possible via the server read out routine as the chosen ethernet read-out protocol structure does not consider its needs. The time-schedule programming is therefore given by the corresponding code instructions of the client routine.
- Shape generator block: Every trigger pulse which enters the shape generator subblock causes an output of a selected shape as dummy output signal. The shape behavior is user selectable. Besides a linear ramp, an exponential pulse or a special debug shape behavior exists. In verification test the “special debug shape” is of superior importance. Its signal must be eligible to enable a verification of the error free operation of the ADC Ring Buffer/trigger acceptance control block combination. Common misbehaviors of such a combination are the omitting of accepted pulses or an erratic seamless retrigger behavior. Obviously periodic trigger signals are not sufficient to verify retriggering situations. Therefore the “random pulse trigger generator” was implementation. A constant pulse shape triggered by the random trigger generator can be used to verify some trigger situations, but for instance it is not capable to deduce if no accepted pulse was wrongfully omitted. To do so the “history triangle pulse signal” was introduced. Its pulse amplitude is increased by every trigger. When the pulse amplitude achieves its maximum, the next incoming trigger resets it on its given minimum value and the procedure starts again. Furthermore, the triangle pulse signal is getting decremented every clock cycle until its value achieves zero (“bottom level of signal”). If the pulse shape generator block achieves a new trigger pulse before the “history triangle pulse signal” arrives its bottom level, it adds the new pulse on the remaining tail of the old pulse. This adding obeys the superposition principle and consequently the buffered signal trend of the “history triangle pulse signal” can be used to verify the correct operation of the ADC Ring Buffer/Trigger Acceptance Control Block combination. Note that this procedure has the disadvantage that some very important trigger signal combinations will show up only rarely in “random trigger mode”. Obviously, it must be assured that these critical situations are verified. This special trigger situation can be systematically verified by programming the “CPU requested trigger pulse generator” accordingly.

The verification of the triggered traces in “history triangle pulse signal” mode, can be done manually or automatically by analyzing the corresponding permanently stored traces. In case of this thesis the latter was done by a to this developed MATLAB routine. In the verification measurements (and consequently also in the corresponding MATLAB routines) also some of the “trigger acceptance control” block internal signals were included in the verification procedure. These signals are accessible via the available “+2bit mode” (mode will be discussed in chapter A.2 and chapter A.4.3). The selected signal forwarding (in the firmware) in case of the “trigger acceptance control” verifications measurements is shown in Figure 129. The ADC dummy unit was mainly developed for debugging and monitoring purposes of the “trigger acceptance” and the “ADC ring buffer” block, but it is also eligible for system bandwidth performance checks. Effectively in the published event building performance results of Figure 44 a “random trigger” and the “history pulse triangle signal” mode combination was used. None of its functionality is required in a practical setup, therefore the “ADC dummy block” is an optional option which not has to be implemented in every FC system.

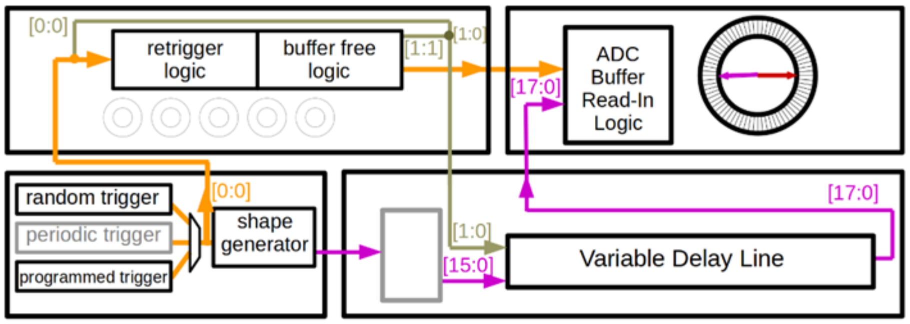


Figure 129 Signal forwarding in case of the “trigger acceptance control” verification measurements. The dark yellow line forwards the original trigger pulse signal of the dummy and the frame trigger signal of the “trigger acceptance control” block. In “+2bit mode” they are also buffered in the ADC Ring Buffers. The corresponding event building routine hence can read-out (and forward to the read-out server) the dummy signals (default traces) and the corresponding pulse trigger signals. The verification MATLAB routine thus also has an overview over the internal pulse trigger states.

A.2 Delay and down sampling block design

The “delay and down sampling” block requirements can be understood if its interaction with other system blocks is considered (Figure 130). The trigger pulse (orange line) out of the pulse detection unit has a temporal delay of several system clocks to its input signal (magenta line). Passing of the trigger acceptance control consumes further clocks – note that in the final FC2.0 system (with several boards in one clock domain) the trigger delay of trigger acceptance block will be dominant and strongly depends on the setup. Hence the trigger input signal and the data input stream (grey line) of the ADC Buffer Read-In have a significant delay. If the design would not be able to compensate this delay, it would not be possible to record the beginning of the pulse, as the ADC Ring Buffer starts with the trace record when it receives the corresponding input trigger and omits all input samples anteriorly.

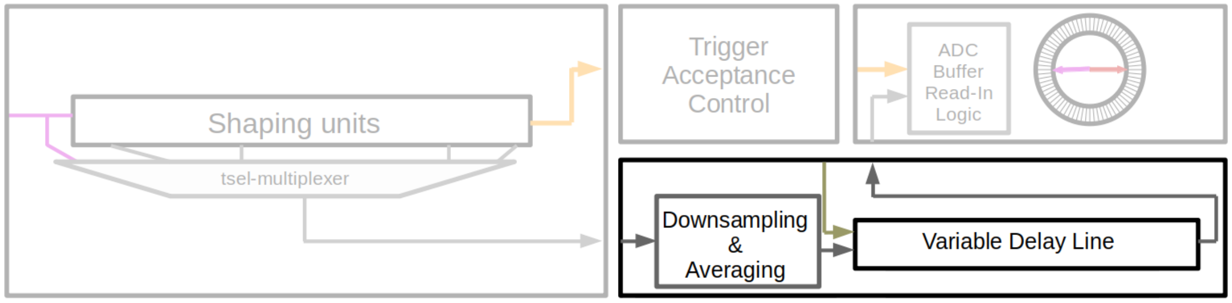


Figure 130 Delay and downsampling block and its system interaction. The initial block (with the shaping units and a corresponding signal-multiplexer) represents the implemented "pulse detection" block (modified version of FC1.0).

For signal processing reasons it is important that every buffered trace has a determined number of samples before its rising edge. The required pre-pulse length depends on the selected shaping parameters and must also be borne by the variable delay unit. In some cases, down sampling enables a signal with higher resolution and negligible loss of fast frequency information. In FC2.0 prototype the down sampling (and corresponding) averaging depth is controlled by the "srprop" ("sample rate proportion") parameter that enables a maximum averaging of 1024 samples (srprop=10). The relation between the effective frequency and the selected srprop value is determined by equation (39).

$$f_{eff} = \frac{f_s}{2^{srprop}} \tag{39}$$

A simplified setup of the "delay and down sampling" block is shown in Figure 131. In case of "srprop=4" the averaged output signal of the downsampling unit obtains a noise gain of two bits (uncorrelated input signals assumed). As the input signal has a width of 16 bit in the ideal case an effective 18-bit output signal can be generated. Therefore, it was decided to implement a "Variable Delay Line" (modified design out of FC1.0) with a data input and output width of 18 bit. This 18-bit signal can be buffered lossless in the ADC Ring Buffer block of chapter A.4.3.

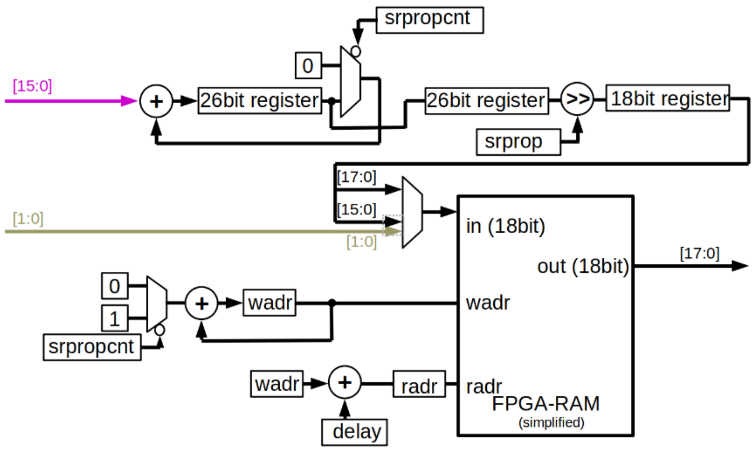


Figure 131 Internal signal processing example for a "Delay and downsampling" block with default settings. The variable delay unit can merge the 16 bit raw (dummy) channel signal (magenta line) with the corresponding debug signals (dark yellow line) of

the trigger acceptance control block. "srpropcnt" is an incremental counter with an effective width of the selected srprop value. The design sets a maximum srprop value of 10. The corresponding multiplexers will feed through a zero if srpropcnt is equal to zero, otherwise they will feed through the alternative signal/register. Note that the corresponding signal input selection, delay and srprop parameters in the initialization process are loaded via the CPU bank. In order to simplify the sketch, the corresponding components and connections were omitted.

The main motivation of the "18 bit buffering width" was not the possible resolution gain. Whether a real setup can meet the 18 bit resolution via down sampling depends on the linearity and SNR of the used ADC. If these requirements are not meet, it does not make sense to do an ethernet read-out of the gained bits – the default 16 bits are sufficient. In DAQ development the error-prone working of the trigger system is challenging to verify. In order to implement an efficient verification approach, the input logic of the variable delay line can merge the most significant 16 bits of the down sampling unit with a two bit debug signal from the trigger acceptance control block (dark yellow line in Figure 130 and Figure 131).

Another point that motivated a signal width limit of 18 bit is a consequence of the available FPGA-RAM components on Zynq devices. BRAM offers a.o. input widths of 18 and 36 bits [40] [41]. A signal width of 19 bit requires the usage of the 36 bit BRAM – 17 bits of this memory would not be used and therefore this implementation would be inefficient.

A.3 Firmware design: CPU bank block in design

A schematic of "CPU bank" block internal design is depicted in Figure 132. The draft design of the CPU bank block in the FC2.0 prototype firmware is identical to the CPU bank block design in FC1.0. The major change is the DMA setup and hereof derived software adaptions, the remaining design elements are mostly identical to FC1.0.

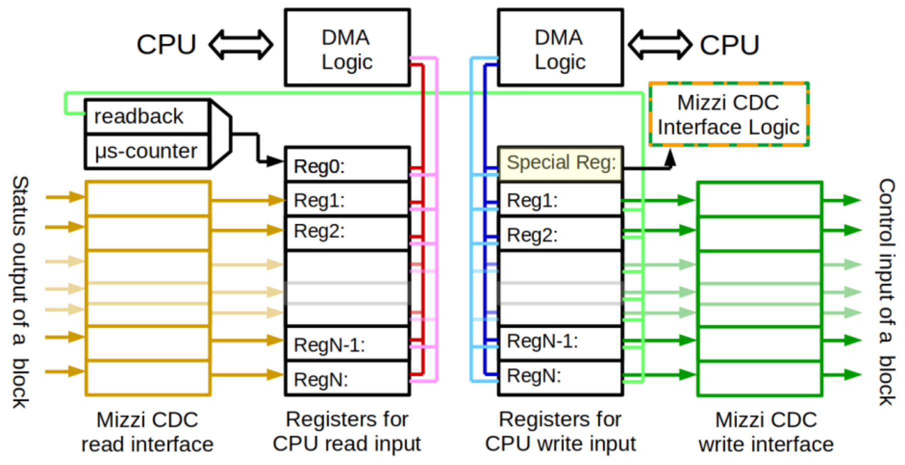


Figure 132 CPU bank in FC2.0 prototype design

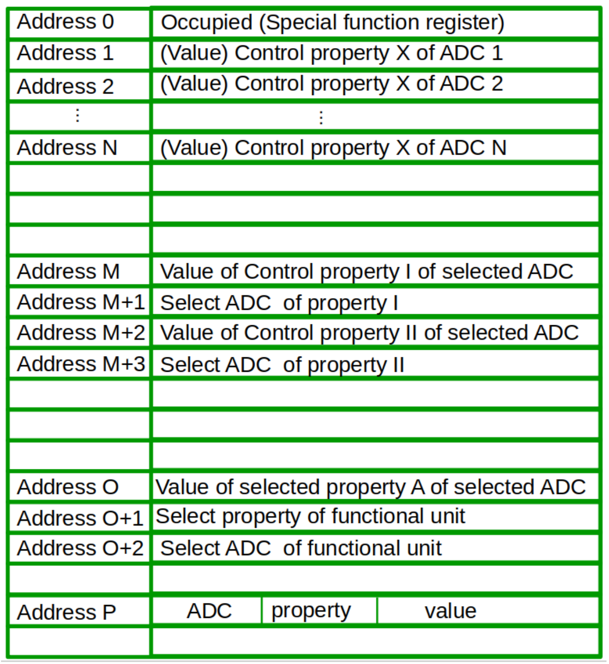
The setup is centered around of two multiplex/demultiplex register banks (black columns in the middle of Figure 132) who effectively buffer the status/control register values of all other firmware blocks. The CPU has access to the register values via a DMA logic. The DMAs in the CPU bank block uses the M-GP or the

M-LPD port of the Zynq device and are directly connected with the CPU. Note that the DMAs just permit single CPU reads/writes into the de/multiplex registers. The multiplexer register bank (left one – labeled as "Registers for CPU read input") is intended as CPU gate to the (firmware) blocks status registers, the demultiplexer register bank (right one – labeled as "Registers for CPU write input") is intended as CPU gate to the (firmware) blocks control registers. The multiplexer/demultiplexer design of the register banks is essential as it must be guaranteed that all registers must be loaded/unloaded (in the firmware blocks) at the exact same point of time (keyword: "concentrated/merged" loads/stores). Note that in both banks the first registers (with an index of 0) obey special purposes. Register 0 of the demultiplexer bank (highlighted by its yellow surface) conducts the logic that handles the CDC compliant word transfer through the "Mizzi CDC interfaces" and passes the multiplex register address whose content the CPU can read back via the (software) GetCpuOutreg function. Note that CPU banks reads and writes always require the correct handling of the CDC interfaces and therefore a default ISA load/store access is not sufficient. The FC2.0 software library contains for this purpose GetCpuReg and SetCpuReg functions.

In the FC2.0 firmware setup without a HP-DMA implementation ("default implementation") all blocks are driven by the same system clock frequency and therefore an additional CDC handler like the "Mizzi CDC interfaces" on first sight would not be necessary. In contrast to this false assumption the "Mizzi CDC interfaces" is also a requirement to transfer sequential CPU writes in the multiplex register bank effectively into an merged write operation whose changes concurrently appear at the corresponding "Mizzi CDC write interface" outputs (="atomic" store). The same is valid for merged ("concurrent") read operations out of the block status registers via the bank "registers for CPU read input".

Register 0 of the multiplexer bank (with an upstream multiplexer) contains depended on the selected (demultiplexer register) read back address (part of the special register of the demultiplexer bank) the corresponding readback value. An exception of this rule is the read back address of zero. As it does not make sense to read back the content of the read back control register ("special" register 0 of the demultiplexer bank) in this case register 0 of the multiplexer bank returns the actual value of the block internal 32bit µs counter.

A bottleneck of the CPU block is its available number of multiplex/demultiplex registers. Multiplex/Demultiplex register arrays are a memory storage where every of its registers has its own input/output respectively. In the PL area of the Zynq devices such array registers must be build up by cascading LuTRAM memory elements (BRAM is not eligible!). Forwarding signals in cascading setups is limited by the given time closures. This strongly limits the possible multiplex/demultiplex register array size. This has effect on the whole firmware design. For the control and status registers read out of all remaining firmware blocks just the limited address space which corresponds to the array size in the CPU block is available. In preliminary studies (for the Zynq-Ultra+ design) a feasible (de)multiplexer register array size was approximately 200 elements (at f_{system}=200 MHz) though bigger array sizes were possible. The FC2.0 prototype firmware consists of 7 blocks (Figure 11). With exception of the “clock generator” block, all remaining blocks have a register set for every given ADC channel. In FC2.0 every block needs approximately 5 registers per block control set[22]. A firmware design which embraces 8 channels therefore obtains ~240 control registers. If the design assigns every control register its unshared CPU bank address the design would not be convertible. As all control and status registers have to be accessible via the CPU bank block, a register address condense strategy was developed. In Figure 133 the tenor of the address condense strategy is shown. Recap that every register has a unique CPU bank address and has a width of 32 bit.



Address 0	Occupied (Special function register)
Address 1	(Value) Control property X of ADC 1
Address 2	(Value) Control property X of ADC 2
⋮	⋮
Address N	(Value) Control property X of ADC N
Address M	Value of Control property I of selected ADC
Address M+1	Select ADC of property I
Address M+2	Value of Control property II of selected ADC
Address M+3	Select ADC of property II
Address O	Value of selected property A of selected ADC
Address O+1	Select property of functional unit
Address O+2	Select ADC of functional unit
Address P	ADC property value

Figure 133 Address condensation strategies applicable on CPU register banks. The most efficient event ring buffer read out in FC2.0 prototype is possible with a modified “Address P” approach.

Without address condensation, every control word of a block control set has its own address (top example in Figure 133). Consider that a DAQ board in FC2.0 prototype has up to 12 channels (N channels in the top example) and every of these channels in the uncondensed approach requires its own control register set. In the first condense approach (second example in Figure 133) a channel select register is implemented. In this approach the pending property value is assigned to the selected channel (after the invocation of function SetCpuReg). In this approach it can be an advantage to reserve a special channel value (as input of the channel select register) to set in a nonrecurring operation, the corresponding control registers of all channels with the pending property value. A more matured address condensation approach is shown as third example.

[22] This is a rough guess and may change in the final version. The number was selected to motivate the issue. In the final design the channel number and the number of blocks also may differ (likely larger).

Besides the channel, the control register (labeled as property) has to be selected. The input selected property value is assigned the corresponding register that comply with the channel and property registers values. In some cases, the control registers do use a small fraction of the available 32 bits. As in the case of the FC2.0 prototype 4 bits for channel selection and a similar bit number for control property selection is sufficient, in these cases the complete write process could be covered by a single CPU bank address. This possibility is presented by the last example. The first example enables the highest bandwidth. Therefore, it is used for all control/status registers (especially important for “permtrgnum” a. “endevtno”) which determine the bandwidth of the final read-out (and control) process.

A.4 Firmware design: ADC Ring buffer design

The ADC Ring buffer task is to store all traces that were accepted by the upstream trigger acceptance control unit. In FC 1.0 the ADC Ring buffer was also a requirement for the CDC organization. The trigger acceptance control unit returns for every trace which is accepted exactly one trigger pulse. The ADC Ring buffer must not discard one sample of an accepted trace. The trigger acceptance control assumes that a continuous, lossless buffering is done. In FC2.0 every incoming trigger has to cause a nonparalyzed immediate read-in of “wlen” samples of the current issued input data stream – not even a single sample delayed read-in start is permitted.

The fundamental design idea of an ADC Ring buffer is rather easy (Figure 134). It requires a Read-In Logic that stores the input stream data at its designated ADC Ring Buffer address, a Ring Buffer and a Read-Out Interface that enables the available DMA to read-out the traces that were requested by the event building process. In order to guarantee that every incoming (“frame”) trigger pulse causes an immediate trace record, the read-in logic must operate without any concurrent CPU support (after accomplishment of its initialization in the init process). In some DAQ designs all implemented ADC Ring Buffers depend on a single trigger input. In FC2.0 prototype every channel has its own ADC Ring Buffer with its own independent trigger input. This enables for every channel its own trigger domain and hence is a requirement of the “Independent Multi Channel” mode. The read-out logic must be able to transfer the requested read out trace to the DMA in the temporal and sequential constraints the DMA expects. Commonly the DMA expects a read-out of a trace which is not identically to the one which was accepted at the latest. In case of ethernet transmission errors it even can be necessary to read-out a single trace several times. (Therefore) Every DMA requested trace read out process therefore must tell the ADC Ring Buffer read out logic the start address of the desired trace.

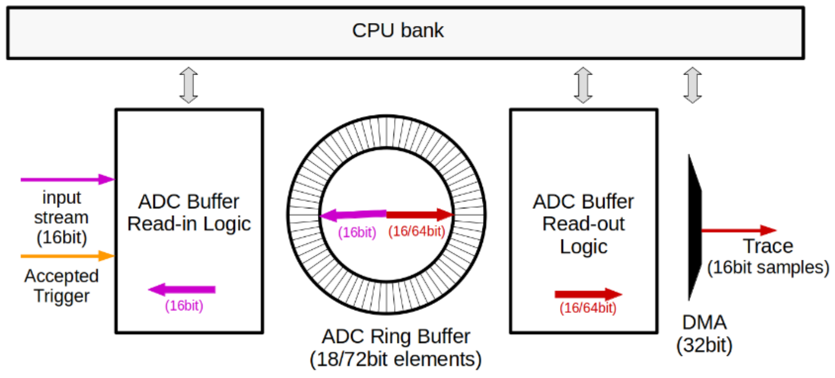


Figure 134 ADC Ring Buffer in FC. Recap that the Zynq devices offer different kind of FPGA-RAM. UltraRAM has a.o. a width of 72bit, BRAM has a.o. a width of 18bit. In case of FC2.0 prototype the DMAs always have a width of 32bit. Further common DMA widths (that are not used in FC2.0 prototype) in Zynq-Devices are 64 and 128bit.

The requirements of an efficient ADC Ring buffer implementation hinges on the available FPGA RAM, the CDC conditions and the DMA requirements. In FC 1.0 and in FC2.0 Zynq-7k prototype designs the ADC Ring Buffer can completely be built out of BRAM. BRAM in Zynq devices has a possible data width of 18bits and is therefore a good match to the given trace data width of 16bit [40] [41]. The smallest available input data width of UltraRAM, the FPGA-RAM that must be used in some Zynq-Ultra+ devices, is 72bit [41]. In order to enable a profitable usage of the UltraRAM memory an enhanced read-in/read-out logic is required, therefore the BRAM case is elaborated first. In FC 2.0 all DMA interfaces have data inputs/outputs with a width of 32bit. This is not a hard requirement as the PS/PL interfaces ports of the Zynq-devices enable also a width of 64bit or 128bit (Zynq-Ultra+).

Modern EDAs like Vivado support the FPGA designer in Ring Buffer design. As ADC Ring Buffers typically contain several thousand samples, the storage capacity of a single FPGA-RAM component is not sufficient. The concatenation of several FPGA-RAM components into a bigger interconnected RAM is automatically done by the EDA. The given FPGA-RAM components are eclectic – they can be implemented as FIFO, as Cache, as Ring Buffer, etc. The EDA can conclude on the most likely desired buffer type (Variable Delay Pipeline, Ring Buffer, …) by the (programmer) given clock lines and the appendant high level addressing logic. The remaining signal requirements of these memory types are elaborated by the EDA without any programmer interference. As an address space range figure of power of two requires the lowest operation performance, all ADC Ring Buffers in FC have a size of power of two.

The suggested first design ("Simple ADC Ring buffer design") is identical to the primarily implemented ADC Ring Buffer Block in the first (now superseded) firmware designs.

A.4.1 Simple ADC Ring buffer design

The first implemented ADC Ring buffer block was developed for Zynq-7k firmware. Zynq-7k devices do not have UltraRAM therefore the design considerations were focused on BRAM. Figure 135 shows its fundamental interaction. Striking in the design is the fact that all data inputs and outputs effectively have a width of 16bit. This avoids the implementation of complex synchronization mechanism.

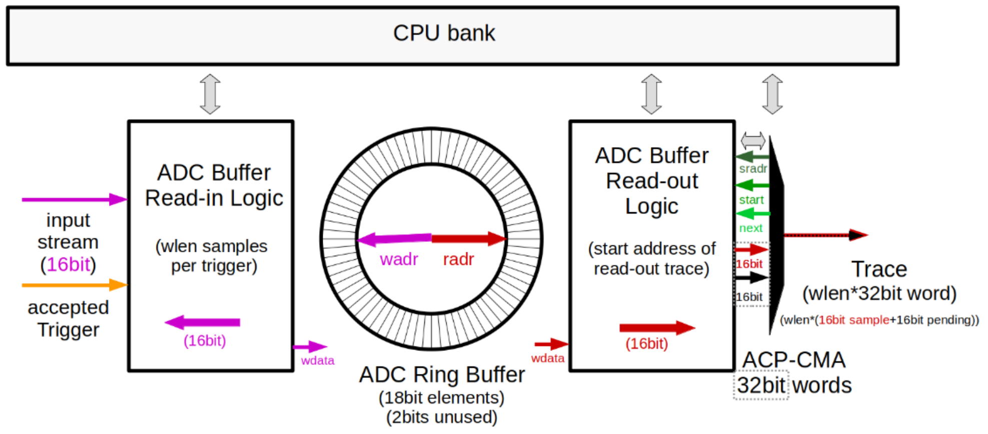


Figure 135 Simple ADC Ring buffer design

The "ADC Buffer Read-in Logic" subblock organizes the data read-in. The Ring buffer needs as input a valid memory (write) address and the corresponding input data value. Every sample write into the ADC ring buffer requires an automatic increment of the corresponding write address counter wadr. The write process has to stop after wlen samples and must automatically (re)start/continue for further/additional wlen samples when a trigger signal arrives. The value of wlen and the start value of wadr must be set in the initialization process via the CPU bank. Figure 136 depicts the corresponding state diagram of the Read-in logic. In [40] a. [41] it is specified that at the ring buffer address input, a non-combinational signal must be used. Consequently, the designated write address must be determined one clock cycle before the corresponding input data is issued at the BRAM input. Despite to that, immediate read-ins are not an issue, as the designated trace start address is always identical to the actual value of the wadr counter (this is a consequence of the automatic increment after every write). But this simple procedure also has a possible drawback. Caused by the automatic increment, a situation can occur that the start sample of trace that was not transferred to the read-out server yet is accidentally overwritten and consequently a corrupted trace was generated. Ultimately this unwanted situation can be prevented quite simply by uncomplete usage of the ADC rings buffer capabilities. Between the last trace which was not read-out yet and the last ensuing trace which can be buffered (in the ADC Ring Buffer) needs to be a free space of at least a sample (Figure 137). In practice a minimum free space of a trace length is chosen, as this permits a simpler design. This spare trace management is a simple software task and easily implementable.

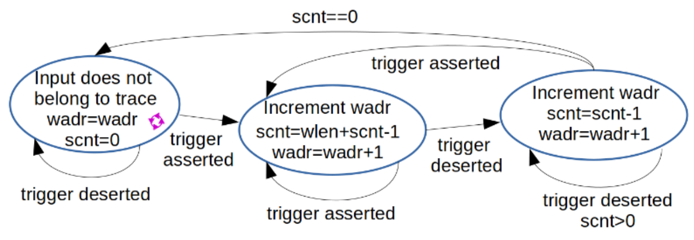

scnt==0
trigger asserted
Input does not belong to trace wadr=wadr scnt=0
trigger asserted
Increment wadr scnt=wlen+scnt-1 wadr=wadr+1
trigger deserted
Increment wadr scnt=scnt-1 wadr=wadr+1
trigger deserted
trigger asserted
trigger deserted scnt>0

Figure 136 States of the Simple ADC Ring Buffer Read In Interface. As the interface does not have a write next signal, it continuously writes into the ADC Ring buffer. This approach simplifies the (trigger caused) immediate write in, but also causes disadvantages like the "trace spare issue".

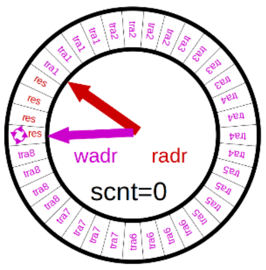


Figure 137 Trace Spare Issue . Example with a trace lengt (wlen) of 4 and an ADC Ring Buffer with a total size of 36 samples. The current write address (wadr) must not be identical with an address which was not successfully read-out yet (tra1-tra8).

The "ADC Buffer Read-out Logic" subblock organizes the data read-out. Its design requirements are driven by the connected DMA interface. In FC2.0 prototype for the ADC ring-buffer readout solely the ACP-CMA is used. The ACP-CMA interface was developed with focus on its potential interaction with the ADC ring buffer. In order to enable a simple interaction, it has in total only 4 signals at which 2 of them control the data transfer. The interface states and the corresponding interface signal behavior are depicted in Figure 138. The data input signal has a width of 32bit – this is exactly twice as much as the given sample width of 16bit. The remaining 16bits are padded with a constant value of zero. These bits do not contribute to the effective system bandwidth. Furthermore, it has three (control) output signals: sradr, start and next. "sradr" is the designated read-out buffer start address. An asserted start signal implies the read-out logic that the actually issued value of sradr is valid and the ACP-CMA expects likely in the next clock cycle already its first valid data input word. As soon as the start signal is deserted again, the issued word on the data input must be valid. The remaining part of the trace transfer is controlled by the next signal. Always when the next signal is asserted, the read-out logic issues the next trace word on the ACP-CMA interface data input.

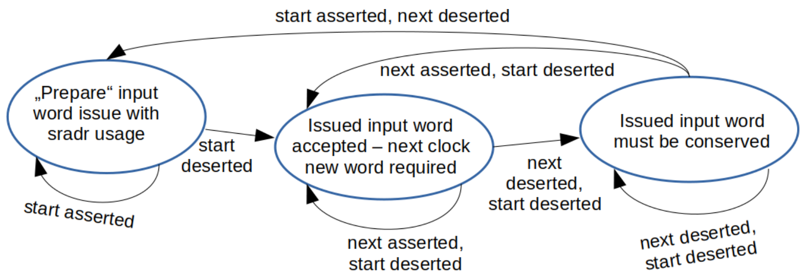

start asserted, next deserted
next asserted, start deserted
„Prepare" input word issue with sradr usage
start deserted
Issued input word accepted – next clock new word required
next deserted, start deserted
Issued input word must be conserved
start asserted
next asserted, start deserted
next deserted, start deserted

Figure 138 (Simplified) States of the ACP-CMA Interface. The implemented ACP-CMA has more states, but the required information is hidden and not accessible via the interface signal values. As the possible finer grain is without use in chapter A.4 it is not discussed. Note that the input conservation in state "next deserted, start deserted" assumes that the ACP-CMA reads out a Ring Buffer – if its task is different, a conservation (of the issued input word) may not be required.

The described setup has two points which need an elaboration. The "trace spare" requirement of the ADC Buffer Read In Logic, is remediable by a more elaborated Read In Logic. More sever is the inefficient data input usage of the ACP-CMA. Just 16 of its 32 bit are used in every transfer. This means that 50% of the available DMA performance used for the transfer of "trash" bits.

A.4.2 Efficient ADC Ring Buffer

The “Simple ADC Ring Buffer” designs drawbacks are getting eliminated in the “efficient ADC Ring buffer” proposal (Figure 139). The major change in this design is the change of the data width of the ADC Ring buffer. Instead of a width of one sample (16bit), it has width of four samples (64bit). Furthermore, the ADC Buffer Read-in Logic does not continuously write in the ADC Ring Buffer anymore. With determined sequence the input data stream is continuously written in a prebuffer. The prebuffer content is just in case of an accepted trace forwarded to the ADC ring buffer. When the ADC Ring Buffer Read In Logic was triggered it has to do a write address (wadr) increment every fourth clock cycle. In Figure 139 this synchronization effort between the Read-in logic and the ADC Ring Buffer is emphasized by the subsidiary signal wnext. The selected write address of the Ring Buffer is loaded with a new input data value, when the signal wnext is asserted. Otherwise, the old value stays unchanged. All other rules of the “Simplified ADC Ring Buffer” remain – identical to the simplified state diagram (Figure 136) every received input trigger causes a seamless read-in of wlen samples.

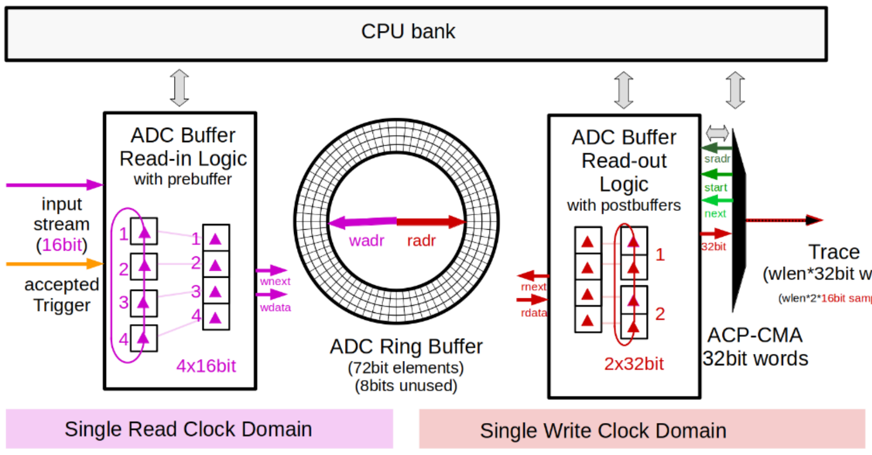


Figure 139 Efficient ADC Ring Buffer Design (Introduction Example). The Read-in logic has two “prebuffer” levels. In the first level a “cycling (16bit sample) buffer unit” (consisting out of four 16bit registers) is implemented, in the second level the merged samples are transferred in a 64bit prebuffer. In principle the proposed concept could also be used with 12bit samples (common sample bit width in PMT experiments), but in this case the proposed concept would work inefficiently (just uses 48bit of available 72bit). In this case the development of a 6×12bit design may be useful – note that the 6×12bit design would have a more difficult timing as it cannot use the counter overflow techniques (as $6<8=2^3$) which are used in the implemented design (there $4=2^2$).

The (suggested) read-out logic uses like the read-in logic internal buffers and a synchronization signal (rnext) for width conversion. The suggested “Efficient ADC Ring Buffer Design” is not the only possibility to gain the full DMA efficiency. But the proposal just requires one clock domain per operation side and can be implemented by BRAM and UltraRAM efficiently (as both have a possible data input width of 72 bit). Especially the latter is a big advantage, as the proposal therefore can be applied on all Zynq-devices.

The proposed “efficient” design seems perfect – but in practice it has a burden that de-facto prohibits its implementation in pulse spectroscopy. To understand this burden, it needs to be considered that it is impossible to say at which prebuffer "address" (the magenta numbers 1,2,3 a. 4 in Figure 139) the trigger shows up. Without assistance of a CPU bank register that holds the corresponding prebuffer address it would not be possible to determine the effective trace start position. A more pleasant approach that is possible without CPU support is shown in Figure 140.

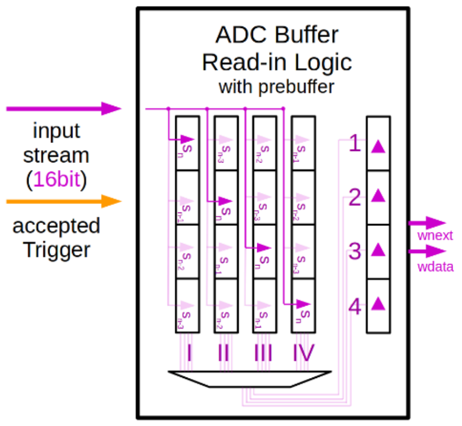


Figure 140 Read-in logic

Instead of one cycling prebuffer like in the first "efficient" proposal, four parallel cycling prebuffers are implemented. They are operated concurrently, whereas the internal buffer address of the issued input sample in every prebuffer has its own (constant) phase (sample s_n represents the current input sample in Figure 140). If a trigger arises, the read-in logic has to forward the cycling register (Roman numbers I, II, III or IV in Figure 140) where the trigger sample is buffered at the start address. Conclusively it is now possible to say that the trigger shows up at the prebuffer start address "1" (Arabic number 1 in Figure 140). The remaining behavior of the read-in logic is not influenced by the changes.

Recap again that the "efficient" ADC Ring Buffer design has two major advantages: The "spare trace" is not required anymore (higher buffer capacity) and the ADC Ring Buffer can be implemented with BRAM and UltraRAM (not only with BRAM). The design proposal is intended for 16bit samples but would also work proper with 18 bit samples (as 4×18 bit=72 bit (input data width of BRAM/UltraRAM in Zynq devices)).

A.4.3 FC 2.0 prototype ADC Ring Buffer design

The implemented “ADC Ring Buffer” block in the FC2.0 prototype is an extended version of the trigger influence compensated efficient ADC Ring buffer. All behavioral constraints are still valid. The development of the block was the final step in the firmware development. The block has a high interplay with the “trigger acceptance block” and conclusively struggles with the design principle of “maximum possible functional block independence”. A schematic which shows all relevant input and output signals is depicted in Figure 141.

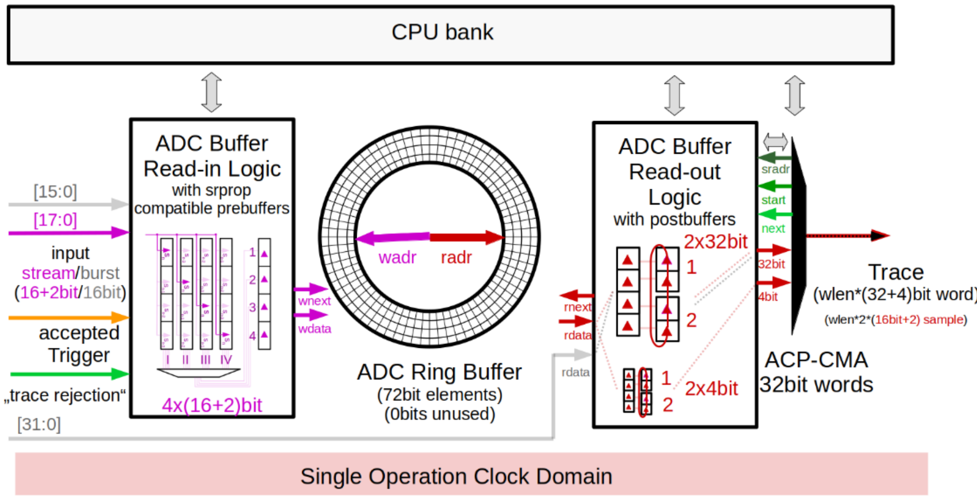


Figure 141 Final ADC Ring Buffer design in FC2.0 prototype. Besides wider channel signals (maximum width is 18bits) the block now can also handle event information bursts (grey lines). The temporal correct transmission of the event information bursts requires the consideration of the clock phase between the source (in the “trigger acceptance block”) and the destination (ADC Ring Buffer or ACP-CMA (depends on merge style)). The currently implemented trace rejection is not eligible for future implementation and was originally determined for a specific setup that seemed out of reach with the default trigger capabilities. New procedures made the initial setup superseded and consequently the insistence on a trace rejection was not existing anymore.

Before discussing the new design elements, a listing of the extended capabilities is useful. The extended capabilities are

- srprop compatible (chapter A.2)
- additional debug signal buffering and read-out (+2bits) (chapter A.2). In this mode every sample has effective 18bits, whereas 2bits belong to the debug signals and the remaining 16bits are the (dummy output) signal. Its objective is mainly verification purposes of the also implemented “trigger acceptance control” unit (already discussed in chapter A.1). In the latest version of the software code (not firmware code), that was used to determine the performance indications of this thesis, the “additional debug signal mode” event building is not implemented anymore.
- 18bit signal compatible (chapter A.2). The 18bit mode differs significantly (from the default 16 bit sample mode) in its event building procedure and the corresponding event building routine is currently not performance optimized. In order to implement this extension in firmware just an

increased number of ACP-CMA channels (and corresponding signal wiring) is required. The states of the ADC Ring Buffer Read-Out Logic are not influenced. The ethernet transfer and the (software) event building of 18bit mode have significant disadvantages in comparison of a 16bit sample process. In the actual software code, that was used to determine the performance indications of this thesis, the "18bit mode" event building is not implemented anymore.

- Trace rejection interface. During the record time of a trace, events can happen that entail a final rejection of this trace. User can enable/disable trace rejection interface during the initialization process of the Server Read Out routine. Trace rejection capabilities can influence the efficiency of the ADC Ring Buffer operation and hence can influence the dead time of the DAQ. If disabled, the trigger acceptance control logic will run in default "retrigger" mode (without any active event rejection). In default FC (1.0) the trace rejection process is a software management task. The ADC Ring Buffer interface of the FC2.0 prototype can handle the trace rejection in most of the cases (but not in all) in firmware. A CPU support in these cases is not required. The design is not finally verified yet (though never an issue was discovered) and can be considered as "curiosity" design that is not urgently deployed in the final FC2.0 DAQ system.
- Extended ADC Buffer Read-Out logic to enable a merged readout of the ADC Ring Buffer and the Event Information Ring buffers (one of the extended modes was labeled as "headeracp" mode in chapter 6.5.3). Merged trace/header read-outs can enable faster access of the desired information as they enable a bypassing of the slow GP/LPD-IO unit event information read-out. The setup and the states of this logic interface uses properties which are specific to the current ACP-CMA implementation and may have different time closures in the final ACP-CMA implementation.

The srprop ("downsampling") compatibility affects exclusively the ADC Buffer Read In Logic and is internally handled via additionally implemented "srpropcnt" counters like in chapter A.2. The 18bit signal read out was implemented (in firmware) via additional ACP-CMA channels. Note that eight 18bit signals require one additional ACP-CMA channels (8×4bits=32bits). An implementation of 8 18bit channels conclusively requires 9 ACP-CMA channels. The trace rejection interface is tightly coupled with the corresponding interface of the "trigger acceptance block.

The most important capabilities extensions in the final ADC Ring buffer design are the trace-header merge options. Remember that in FC every trace has its own header with the corresponding event information (10×32bit words). The event information is (partly) buffered in the Event Ring Buffers that are a subpart of the "trigger acceptance control". Initially it was intended to access the event information via the CPU controlled GP/LPD IO (Figure 11 (not implemented in Figure 142)). For typical read-out conditions of PMT experiments, the event read-out via GP/LPD-IO proved to be the performance bottleneck of the read-out procedure. This leaded to the development of trace-header merge options (Figure 142). The merged “trace-header” ACP-CMA readouts offer two more options.

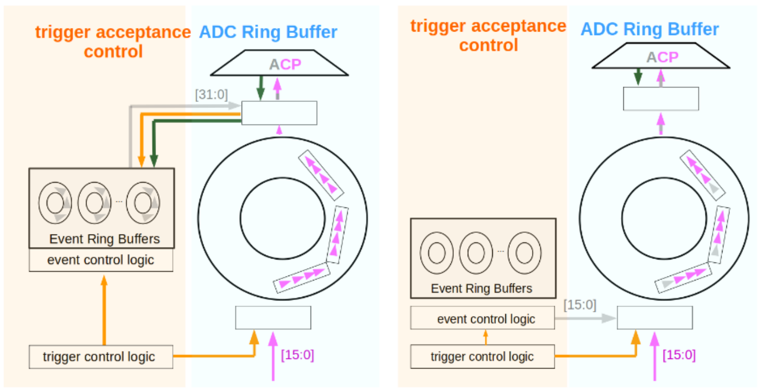


Figure 142 Signal flow in the merged “trace-header” ACP-CMA readouts. Grey lines represent the event information bursts links, the orange lines are trigger pulses (caused by the trigger acceptance control logic or the ADC Buffer Read-out logic), the dark green lines the sradr “instruction” and the magenta (input) line represents the 16bit signal (the merge options are limited to 16bit signals, though in principle 18bit signals would be possible). The rectangle below the ADC Ring Buffer represents the ADC Buffer Read-In Logic, the rectangle above the ADC Ring Buffer represents the ADC Buffer Read-Out Logic (of Figure 141). In Figure 142 the rectangles in the ADC Ring Buffer represent traces. In the left subplot the traces (in the ADC Ring Buffer) just contain samples (magenta arrows) of that event, in the right subplot the traces contain the samples (magenta arrows) and the event information (grey arrows). In later chapters the possible performance of left subplot will be discussed under the label “headeracp” (accessible via server read-out option “-headerintrace 2”). The read-out transfer which corresponds to the right subplot is available under the server read-out option “-headerintrace 1” and hence is discussed under the label “headerintrace”.

In the “headeracp” option the ACP-CMA reads out the event information directly out of the ADC Ring Buffers (left subplot in Figure 142). The timing constraints of the ACP-CMA interface impede an event information communication in AMBA AXI4 protocol style, therefore it was decided to transfer them in (design specific) “burst” link style (grey line). Before the Read-Out Logic starts with the read-out of the trace samples out of the ADC Ring Buffer it demands the transmission of the corresponding event information. The event ring buffers read out logic (not shown in figure) then sequentially conveys the required event information to the ADC Buffer Read Out Logic. No checks and handshaking in this request are done – the timing constraints are absolute, and the received information is interpreted in a fixed way dependent on their clock number difference between the time of the trigger broadcast and the reception. Conclusively all designs that use this merge logic must have the same clock phase between the Event Buffer Read Out Logic and the ADC Buffer Read Out Logic. This impedes a future modification of the block.

In the “headerintrace” option (right subplot in Figure 142) the merge of the trace and the trace header information is done in front of the ADC Ring Buffer. Hence in this approach the corresponding event information is additionally buffered in the ADC Ring Buffer. The event (information) ring buffers are not required in this approach. A deeper probing of the approach limits requires knowledge of the “(trace) framing strategy” of the trigger acceptance control block. Like the first trace-header merge option, it uses event information bursts (grey line) to invoke the header information the ADC Buffer Read-In logic. This impedes a future modification of the block again.

As all extension, with exception of the “headeracp”/”headerintrace” extension of the ADC Buffer Read-Out logic, in practice have a minor importance, their performance results are not included in the thesis. The “srprop” compatibility is used in some of the proposed “FC1.0” nonlinearity determination setups (chapter 8.5.1). Table 25 lists all ADC Ring Buffer specifications of the final implementation. The content of the status and control registers is only accessible via the GP/LPD IO unit.

Table 25 Possible ADC Ring Buffer settings. Note that not all settings are limited by firmware constraints, some are also (tighter) limited by software constraints. The specifications in the table assume the firmware operation via the developed software stack.

Number of elements per ADC Ring Buffer	2^m (maximum value of m is given by available FPGA-RAM of selected device) $$RAM\_SAMPLES = \frac{RAM_{bit}}{sample\_width_{bit}}$$ $$m = floor\left(\log_2\left(\frac{RAM\_SAMPLES_{available}}{channels}\right)\right)$$
Minimum (16bit sample) ADC Ring buffer size	2^{17}
Possible trace lengths (wlen)	2^n
Minimum trace length (wlen)	16 (2^4) (worst case – in “trace-header merge in ADC Ring Buffer” mode (“-headerintrace 1”)) (best case: 4 (2^2) – in all other modes)
Maximum trace length (wlen)	65536 (2^{16})
Seamless retrigger compatible	Yes
(Hardware) overwrite protection	No (pure software task (“permtrgnum”))
Status registers	Every channel: Reg0: evtno[31:0] (Number of traces whose read-in (into the ADC Ring Buffer) were (utterly) accomplished)
Control registers	Every channel: Reg0: wlen[15:0] (in default configuration) Reg1: srprop

In the read-out routine the status of the evtno register is of special importance. Its value indicates the number of trace read-ins (into the ADC Ring Buffer) that were already accomplished. This is exactly the desired behavior of the “endevtno” element of the messen_info control structure that is used to control the “Trace-Read-Out” transfer. The “Trigger Acceptance Control” block also offers status registers whose behavior complies with the desired “endevtno” behaviour. In practice the “evtno” registers of the “Triger Acceptance Control” block are preferred as they enable a faster Trace read-out (for the price of a more elaborated read-out procedure). The latest client routines are therefore prompting the “evtno” status registers of the “Trigger Accaptance Control” Block.

A.5 Firmware design: Trigger acceptance control design

In default FC (1.0+2.0prototype+2.0) the event analysis is done on the server. This means the server must get the signal trend in its region of interest as sampled trace. The “region of interest” criteria in HPGe-spectroscopy is given by the pulse shaping conditions of its signal processing chain. In FC-Germanium, where the actual baseline level is a part of the trace header, it is possible to condense all criteria’s in following simple thumb of rule:

The pre-pulse area must cover at least the time range of the selected shaping time, the post-pulse should cover at least the 1..2fold of the shaping time (depending on the given ballistic deficit).

Again, the above listed criteria are only valid if the baseline level is known. This is in FC and in the FC2.0 prototype the case. Determining the baseline value with sufficient precision out of trace that comply to the criteria is not necessarily possible. A trace readout that covers a bigger range of time does NOT contribute to a better resolution. It just consumes more bandwidth. The FC DAQ therefore must be capable to buffer a trace that covers a defined amount of pre-pulse samples and a defined number of post-pulse samples (Figure 143), but it also should try to avoid the buffering of any gratuitous samples.

It is the task of the trigger acceptance control unit to trigger the ADC Ring Buffer Read-In logic in exactly the right point of time that causes the requested trace read in. This is possible, when the Delay block (chapter A.2) and the ADC Ring Buffer block are initialized with proper settings. As the ADC Ring Buffer starts stream-buffering at frame trigger arrival, a variable delay unit (in upstream position) is essential for buffering of the pre-pulse trace region.

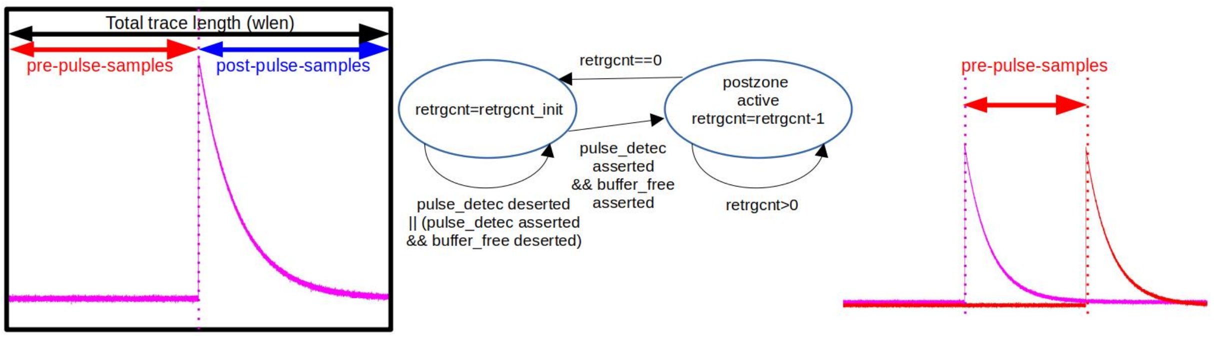

Total trace length (wlen)
pre-pulse-samples
post-pulse-samples
retrgcnt==0
retrgcnt=retrgcnt_init
postzone active retrgcnt=retrgcnt-1
pulse_detec asserted && buffer_free asserted
pulse_detec deserted || (pulse_detec asserted && buffer_free deserted)
retrgcnt>0
pre-pulse-samples

Figure 143 Pulse without any interference framed into a single trace. In FC 2.0 prototype the trace length (wlen) is determined by the ADC Ring Buffer block. The trigger acceptance control block initialization routine must know this value, but it is not allowed to influence it. The instance state machine only works in pile-up free experiment. Note that the given state machine requires a variable delay line with a delay of at least “pre-pulse-samples”. In the right subplot the magenta line is the signal that causes the pulse_detec trigger signal. The red line shows the same signal (with a delay of exactly pre-pulse-samples) after it passed the variable delay unit. This signal is the input stream of the ADC Ring Buffer.

The trace framed pulse in Figure 143 represents an ideal trigger situation where the trace is just charged with a single pulse. A corresponding state machine example for a DAQ which is used in pile up free experiments is shown to the right. Its pulse_detec signal represents the pulse trigger output signal of the pulse detection block. The counter initialization value “retrgcnt_init” relates to the sample number of the post-pulse area. The (trigger acceptance) state machine needs not consider the pre-pulse region, as in this region never a pulse will show up (assumption of a “(unfeasible) pile-up free experiment”). Note that a (variable) delay line operation with an eligible value is essential for the developed trigger acceptance logic – if the delay value is not proper set, the start position of the pulse is not on its intended location and the shaping criteria’s can be violated. In general trigger state machine – trigger state machines which cannot assume pile up free signals - a wrong selected delay value even can out of sync the whole system. In FC it is important that the trigger control unit just accepts traces when free ADC Ring Buffer is available – otherwise data corruption would be unavoidable. In the example state machine this is the task of the “buffer_free” signal.

In practice not all pulses obey the ideal signal analysis conditions which were given in Figure 143. In real life a HPGe-detector measurement faces “pile-up” situations there the read-out of a single trace is not sufficient. Such a case is depicted in Figure 144, where in the post-pulse area of the primary trace an additional event occurs.

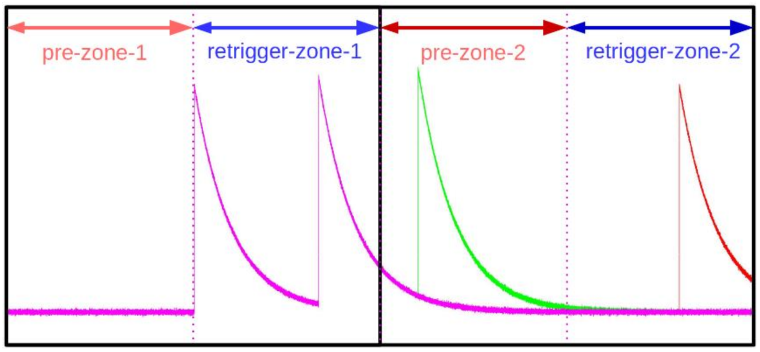


Figure 144 Primary trace (left subplot) and corresponding secondary trace (right subplot). Both traces have the same length (“wlen”). Therefore wlen=”pre-zone-1a”+”retrigger-zone-1”=”pre-zone-2”+”retrigger-zone-2”. As shaping in general demands that “retrigger-zone-1” has the same length like “retrigger-zone-2”, conclusively “pre-zone-1” has the same length like “pre-zone-2”. In chapter A.5.1 the retrg event information ring buffer is introduced. It stores the detected “retrigger” situations of the trigger acceptance control logic. The buffered “retrg” value is finally conveyed as trace header parameter cmd_num. In this case the first trace will have a cmd_num value of 5, the cmd_num value of the second trace depends on the “retrigger” situation. If no retrigger is detected (“green pulse”)the cmd_num value in the trace header is 4. Otherwise (“red pulse” situation and additionally free buffer space for the subsequent retrigger event must be available) the cmd_num value in the trace header is 5 again.

The second pulse in the first trace must provoke the buffering of a seamless subsequent trace. Without the subsequent trace a proper shaping of the secondary pulse would not be possible and conclusively its energy determination is afflicted with higher uncertainty. An ideal “trigger acceptance control” block therefore causes in such cases a seamless retriggering of an additional trace. In the subsequent trace again two event cases can occur (Figure 144 again). In the first case, highlighted as green pulse, the event in the subsequent trace shows up in the region that permits its complete shaping. This region is labeled as “pre-zon-2”. In the second case, highlighted as red pulse, the event emerges in a trace region which does not permit its designated shaping. Again, a seamless retrigger should be the consequence and therefore this area was labeled as “retrigger-zone-2”. Pay caution that the labeling in Figure 144 is partly confusing. It implies that both “pre-zone” areas are functionally equal (like the retrigger areas) – this is not true, as in the first “pre-zone” the occurrence of a pulse is forbidden. An example state machine that covers the desired state behavior of Figure 144 is illustrated in Figure 145.

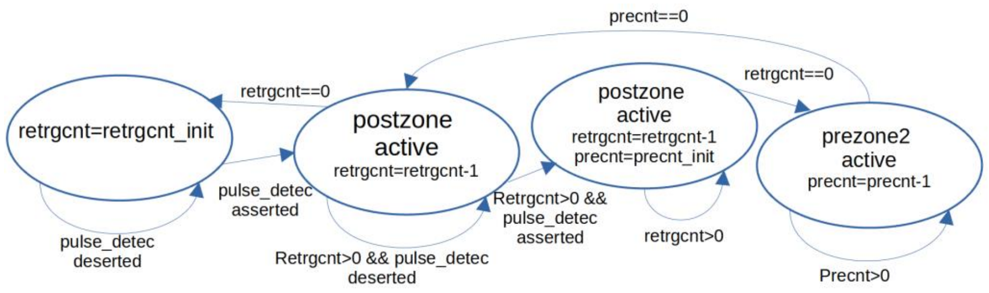

precnt==0
retrgcnt==0
retrgcnt==0
retrgcnt=retrgcnt_init
postzone active retrgcnt=retrgcnt-1
postzone active retrgcnt=retrgcnt-1 precnt=precnt_init
prezone2 active precnt=precnt-1
pulse_detec asserted
Retrgcnt>0 && pulse_detec asserted
retrgcnt>0
pulse_detec deserted
Retrgcnt>0 && pulse_detec deserted
Precnt>0

Figure 145 (Example) state machine of a (re)trigger acceptance control logic. The given state machine could replicate the (imperfect) behavior of Figure 144. The suggested state machine is matched on the given ADC Ring Buffer design of chapter A.4.2 and requires a variable delay line. Initialization values of counters (“…_init”) have to be selected in a way that all traces have the same length (“wlen”). Note that a real state machine must gather if the ADC Ring Buffer has free space to buffer traces. In a real design the condition “pulse_detec asserted” therefore must be extended by a condition which checks if free buffer space is available (like the buffer_free signal example in Figure 143).

It is worth to note that the instance state machine essentially consists out of counters and do not need any enhanced buffer implementation. Such tasks are ideal for FPGAs. An unresolved issue in our discussion is the length of the subsequent trace (Figure 146). Obviously in the subsequent trace not always will show up another event and thus a total trace length that is identical to post-pulse length of the primary trace would be feasible (as this saves samples and therefore system bandwidth). It would not be an issue to implement such a retrigger approach in firmware (thought to implement it in a fast accessible way also would not be easy). The antagonist of this flexible trace length approach is the server read-out routine. It is much easier to program a (high-performance) server-read-out (and analysis) routine that can rely on a measurement fixed trace length. Therefore FC uses in all trigger cases the same trace length ("wlen"). Conclusively the ADC Ring Buffer must also only handle one trace length ("wlen") in operation.

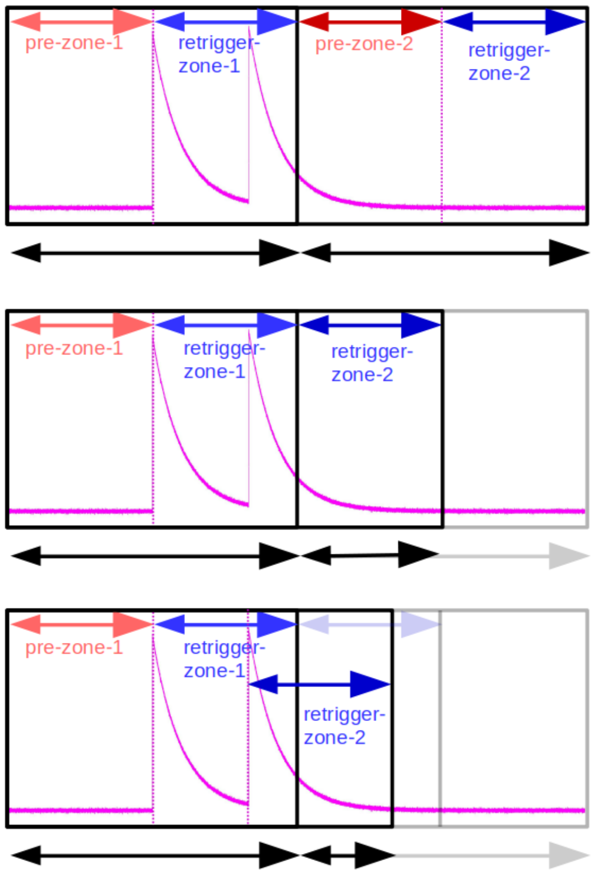


Figure 146 Possible retrigger trace extension designs

A state machine that just can handle the trigger patterns of Figure 143 and Figure 144 is not complete. In these patterns still possibilities exist, that harm an analysis. One of the still possible harmful situations is shown in Figure 147.

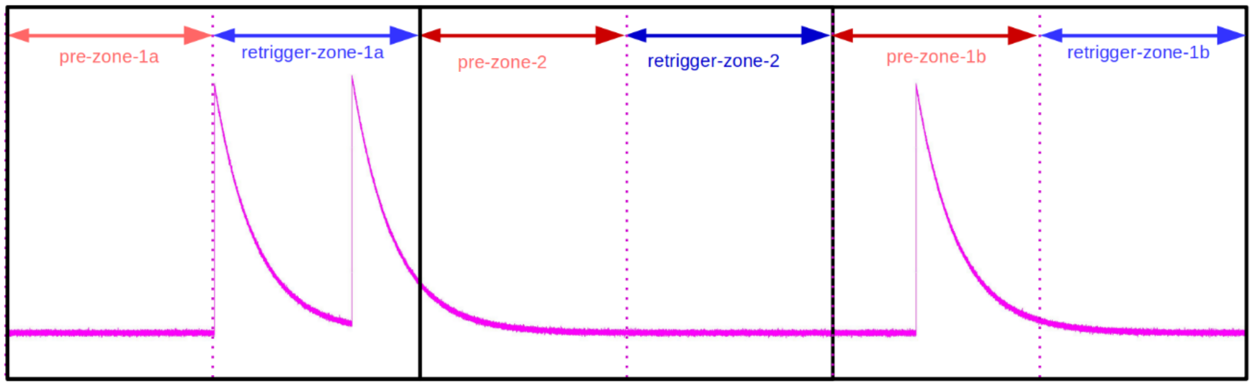


Figure 147 Incomplete trigger model In chapter A.5.1 the retrg event information ring buffer is introduced. It stores the detected "retrigger" situations of the trigger acceptance control logic. The buffered "retrg" value is finally conveyed as trace header parameter cmd_num. In this case the middle trace will have a cmd_num value of 6, the first trace a cmd_num value of 5 and the last trace (third trace) will have a cmd_num value of 4.

The first two traces show the initial situation out of Figure 144, there a pulse in the retrigger zone causes a seamless retrigger trace. In contrast to Figure 144, in the secondary trace no event shows up, therefore the retrigger machinery stops after this trace. The state machine of Figure 144 would immediately switch back in its first pulse trigger level that causes a trace read out in style “pre-zone-1 and retrigger-zone-1”. But such a compliant trace read out is not possible in situations where a further pulse appears before the pre-zone duration is gone. The situation is depicted in Figure 147. Again, a corresponding example state machine that can handle such situations would be build up by counters.

In the previously introduced examples it was assumed that for every trace buffer space is available. This is in FC the case if the actual event rate is (sufficient) below the systems maximum. In measurements with an exceeding event rate, not sufficient buffer space for every detected pulse is available. Thus, the trigger acceptance control logic cannot guarantee the “record” of all detected traces in the ADC Ring buffer. If no free buffer space is available, the trigger acceptance control logic must omit detected pulse triggers and prohibit their (frame matched) forwarding to the ADC Ring Buffer Read In Logic. In the most cases such obstruction will reside in retrigger events (left subplot of Figure 148). But sometimes this will meet a start pulse of an event series (right subplot of Figure 148). In FC2.0 the trigger acceptance control block has a special “buffer free” logic that provides the trigger control logic with a binary occupation state (“available”/”occupied”). The signal “buffer_free” (of the state machine in Figure 143) is the output signal of this subblock. The signal value is determined by firmware internal status register states and (read-out) server given “control” information.

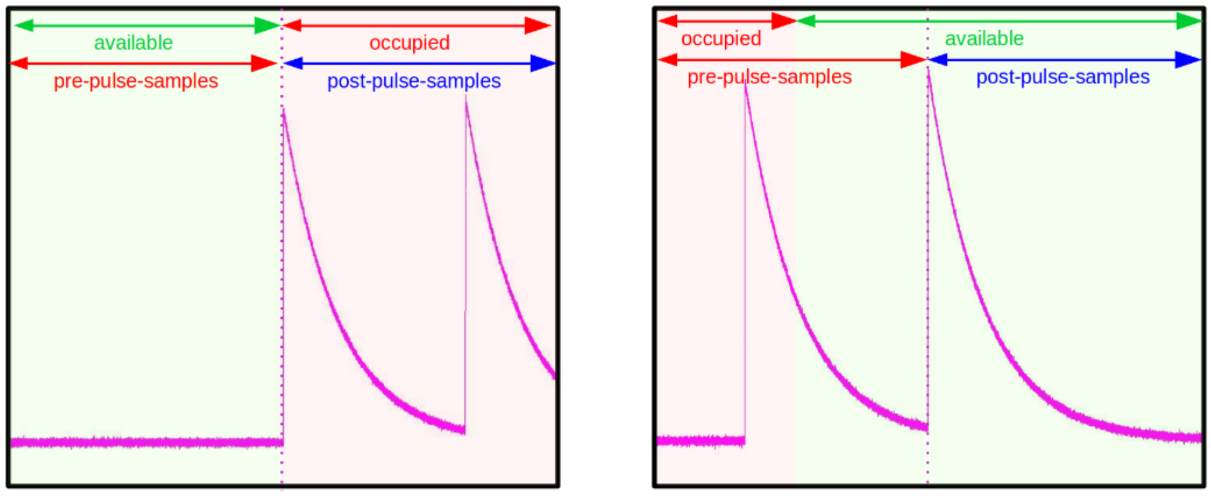


Figure 148 Possible trace record issues due to completely occupied buffer. As long as the signal “buffer_free” indicates that free space is available, it must be possible to buffer at least a complete single trace. If space for a single trace is (currently) available, the next received pulse should immediately cause a switch of the “buffer free” signal to the “occupied” state. This prevents from unwanted retriggering – though it is not the only approach to avoid it.

Before discussing deeper details of the “trigger acceptance control” block behavior, it is useful to remember the most important property of the ADC Ring Buffer block. Simply said, it immediately starts to store additional "wlen" (subsequent) samples for every incoming frame trigger pulse. With this knowledge, it should be decided when the "retrigger" logic optimally transmits its frame trigger output signal. The thought experiment will be done on basis of Figure 149 which contains the data input stream of the pulse detection block (magenta line) and the same delayed data stream as input of the ADC Ring Buffer block (red line). The time of the pulse detection (in the pulse detection block) is emphasized via the dotted (vertical) magenta lines. Furthermore, it is assumed that the trigger pulses (of the pulse detection block) immediately arrive at the retrigger logic inputs. The retrigger logic has to verify the "acceptance" of the incoming pulse detected triggers. Let's (arbitrarily) assume that this verification process in the start trace is immediately accomplished (and therefore P_1=D_1). The time duration of this verification process may depend on the trigger mode and the trigger history (start trace or a subsequent retrigger trace).

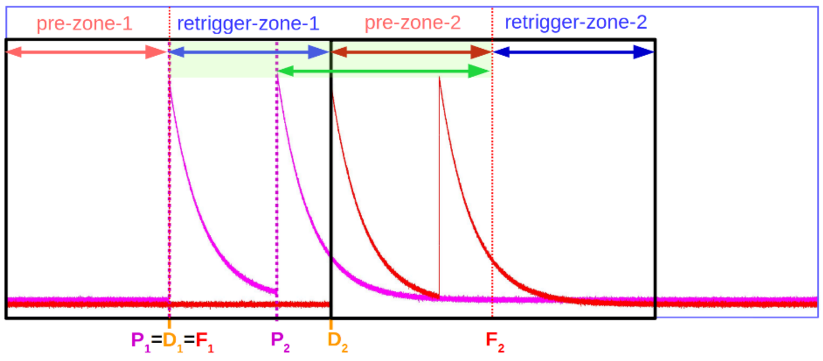


Figure 149 Signal delay exploitation of the seamless retrigger state machine

The delay between the ADC Ring Buffer data time input and the detection time (of the pulse detection block) is exactly "pre-zone-1" system clock cycles. In the retrigger logic therefore the first detected pulse ("P_1") must immediately cause a frame trigger output ("F_1"). In the example it was assumed that "retrigger-zone" has the same length like the "pre-zone", but this is not a hard requirement. The "retrigger-zone" (highlighted by the blue arrow) is the region there a pulse must cause the read-in of a seamless retrigger trace. For an incoming retrigger pulse in the first trace (in the time region of "retrigger-zone-1"), the retrigger logic therefore has a maximum decision time of the green (background) highlighted area. At the end of this time the retrigger logic needs to know if it transmits a frame trigger on its output. If the frame trigger is transmitted later, there is no seamless transition between the start trace and the retrigger trace. The frame trigger of the example retrigger pulse "P_2" therefore must be transmitted at the latest at time "F_2". The retriggering (of the trace read in in the ADC Ring Buffer) would also work when the frame trigger is transmitted earlier.

In practice the transmission time of the (retrigger) frame trigger is not as relaxed as previously derived. The reason is the derivation process of the event header information. In the previous thought experiment every pulse was assigned to its own trace. In real experiments more than one pulse per trace can occur. In this case it is not distinct which time should be accredited to the traces. The “occurrence time” of the retrigger pulse as time stamp for retrigger traces is not a solid time reference anymore. In FC2.0 prototype was decided to use the arrival time (at the ADC Ring Buffer) of the frame trigger as decisive time for the time information in the event header.

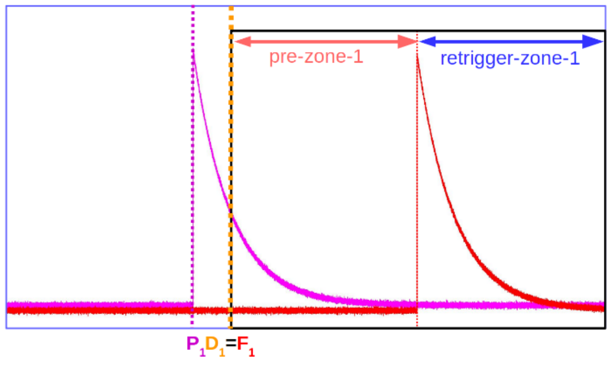


Figure 150 Compensation of delayed pulse “detection acceptance” via usage of the variable delay line (chapter A.2)

In the introduction besides “retriggering” as trigger style also the “pile up rejection” approach was introduced. “Pile up rejection” originally meant that as soon as shaped pulses have interference they are not permitted to contribute to the final result and thus the trace(s) containing them can be discarded. In FC2.0 prototype a more restrictive approach was chosen. As soon as a trace contains more than a single detected pulse, the trace is rejected. The discussion of the “pile up rejection” state machine is omitted in this chapter.

A.5.1 Components of the FC 2.0 Trigger acceptance control block

A simplified schematic of the internal design of the trigger acceptance control block is illustrated in Figure 151. The schematic shows a design that has *n* channels. The orange input lines are linked with the output of the pulse detection unit, the orange output lines conduct the frame trigger pulse that controls the corresponding ADC Ring Buffer. The magenta lines, which are not part of the trigger acceptance control block, conduct the corresponding (ADC) data input streams of the channels to the corresponding inputs of the ADC Ring Buffer block. Note that every channel has its own independent "(Single Channel) Trigger unit". This architecture enables a concurrent run of all channels where every channel has its own trigger domain (called "Independent Multi Channel" mode in chapter 4.1). In FC1.0 the multi-channel triggering situation is more like the FC2.0 prototype "Global Multi Channel" triggering via the "global channel trigger unit subblock". All "pulse detected" input wires are connected to their input at the "global trigger" subblock (Figure 151). The internal setup of the global trigger unit is an extended "Single Channel Trigger Unit" design with an enhanced upstreamed sum trigger logic. The frame trigger output (dark violet line) of the "global trigger" subblock can be forwarded to all ADC Ring Buffers (without any channel specific phase) and is therefore the basis of the setup of a global trigger domain that embraces several boards. The "global trigger" unit can be operated in a way that the ("pulse detected") trigger input of an (exclusive) single channel is used to generate concurrent frame triggers for all channels of the board. This setting will be the default setting in future "multi-channel mode" evaluations. In "single channel mode" evaluations only a single channel is operated via its "single channel trigger unit". An important task of the "trigger acceptance control" block besides the frame trigger scheduling is the management of the corresponding event information. All trigger subblocks have therefore corresponding logic and buffers. The "headeracp/headerintrace" readouts event information burst links are getting generated in the "trigger" subblocks and are denoted by grey lines (the burst links leave Figure 151 at the bottom). The initial design envisaged an exclusive event information readout via the CPU bank block (in thesis called "via GP/LPD" mode) (again highlighted by grey lines). Like in all other blocks the setting and read-out of the blocks control and status registers is done via the CPU bank too (thin black line).

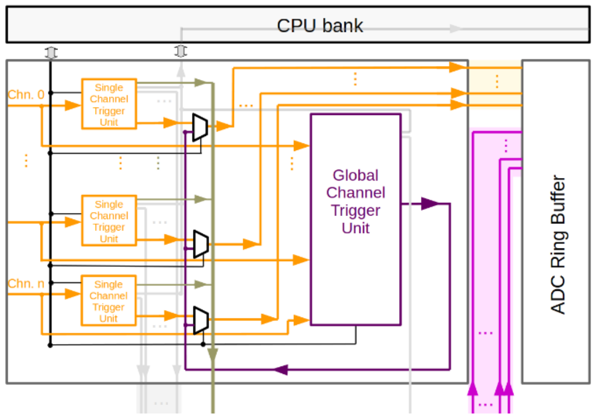

CPU bank
Chn. 0
Single Channel Trigger Unit
Single Channel Trigger Unit
Chn. n
Single Channel Trigger Unit
Global Channel Trigger Unit
ADC Ring Buffer

Figure 151 Schematic of the internal design of the trigger acceptance control block. Note that the schematic was simplified in order to enable a first sight overview. The shown routing is identical to the routing in the design but does not implement all routes and subblocks. For instance, the (unmentioned) trigger masking capabilities and the event burst links (required for "headeracp"/"headerintrace" mode) are omitted or incomplete respectively. Note that depending on the designated application, it can make sense to only read-out channels which were NOT triggered (of course this is not the default situation). In FC 2.0 prototype the control of the read-out behavior is done via trigger masking flags in corresponding (CPU bank) control registers.

For an understanding of the "trigger acceptance control" block, the logic and buffers that are implemented in the "trigger" subblocks are indispensable. All trigger subblocks consist out of a "retrigger logic", "buffer free logic" and a "event info management unit" (Figure 152). The "global channel trigger unit" has an additional (enhanced) sum trigger logic which is implemented in front of the "retrigger logic". The retrigger logic is responsible for a frame trigger behavior which is identical to the proposed "correct" trace framing in the introduction examples of chapter A.5.

The "buffer free logic" continuously feeds the retrigger logic with occupation state ("buffer_free") of the ADC ring buffer. An ADC Ring Buffer address is free when the occupying sample was successfully transferred to the read-out server. The "buffer free logic" has an attached dead time analysis unit which is not implemented in Figure 152. Prompting the dead time status during the "Trace-Read-Out" Phase of a measurement is not useful as the required CPU bank access consumes a significant amount of time. Thus, its prompting was not implemented in the "Trace-Read-Out" phase of the "dummy" event building performance measurements (chapter 6.5.3).

In the ADC Ring Buffers an evtno (status) was introduced. The trigger acceptance block has a (roughly) similar evtno (status) register for each channel (remember that this evtno register was called "endevtno" in chapter 5.2). When the "retrigger logic" is operated in "retrigger" mode, the evtno register indicates the number of frame triggers that were forwarded to the channels ADC Ring Buffer. The evtno trend must be expanding without any decline. If this would not the case, it would indicate an inconsistent system behavior (as consequence of a firmware design error) that forces an immediate shutdown of the server-read-out process. This regulation is also valid in the "pile-up rejection" mode (to bypass this issue the pile-up rejection design introduced additional "evtno" registers where some of thema are permitted to decline). The value of “evtno” never can be bigger as the value of “permtrgnum” – otherwise trace data in the ADC Ring Buffer would be corrupted.

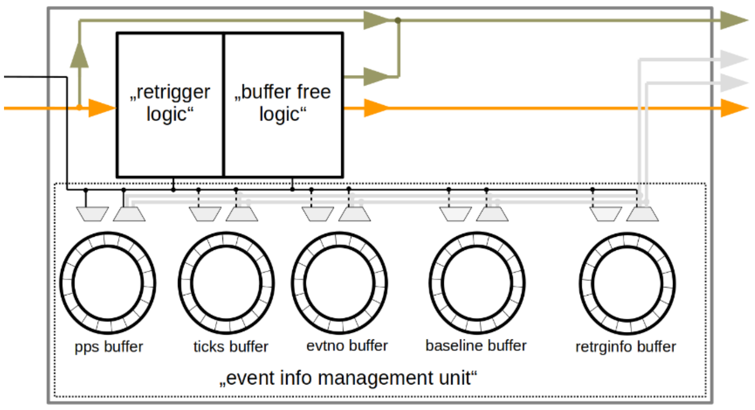


Figure 152 Setup of a “single channel trigger unit” subblock. The dark yellow lines denote the “debug”/”monitoring” signals, the orange lines stands for the trigger signals (in the input (left) the pulse_detec signal of the corresponding pulse detection unit, the output (right) is the frame trigger signal of the same channel) and the grey lines typify event information wires (for the GP/LPD IO unit or the event burst links (not distinguished in the figure)).

The “event info management unit” contains all information which is required to build up the event header of a trace (in three separated event buffers). The temporal information (“detection/framing time”) of the event can be derived from the pps and ticks ring buffer content. Both buffers contain the corresponding values out of the “clock generator” block of the firmware. The pps counter in the clock generator block counts the number of seconds since the successfully accomplished clock generator block initialization. The fine (temporal) resolution is derived by the (system clock) “ticks” information. The corresponding tick counter is designed in a way that it has an overflow every second and its (continuously running) increment operation is driven by the system clock. Recap that a typical system clock is 200 MHz and in these cases the tick counter has a value range of 0-199999. The value which is loaded into the event ring buffers (automatically done by the event buffer read-in logic) is the corresponding issued counter value at the time of the frame trigger transmission (of the “retrigger logic” block). In FC1.0-Germanium the baseline level is part of the trace header and thus a baseline ring buffer is required. The issued channel baseline value (at the ring buffer read-in logic) is out of the baseline follower of the pulse detection block (Figure 11). The information which is given by the “evtno” and the “retrginfo” buffer in principle could be derived on available information. Therefore, these event information ring buffers are not a hard requirement and could be omitted in order to gain a spare of FPGA-RAM. The “retrginfo” buffer content is discussed in the captions of Figure 144 and Figure 147. The “retrg” information has no effects on the measurements of chapter 6.5.3. The analysis routine of chapter A.9.3 used it, but fairly must be said that at this low event rates its effects on the determined resolution are negligible.

The design of all implemented event information ring buffers is identical. Independent of the actual input data width all event information ring buffers are configured for 32bit words. Their implementation is focused on BRAM as available FPGA-RAM. The most significant functional difference between the event information ring buffers and the ADC ring buffer is in the read-out logic design. The ADC Ring Buffer is exclusively read-out via the ACP-CMA, the event information ring buffers also can be read-out via the GP/LPD IO unit. In order to read-out the event ring buffers via the ACP-CMA, a (FC2.0 specific) event burst link approach was chosen. Table 26 lists all significant properties of the implemented "retrigger acceptance control" block.

The FC firmware design is mainly designated to global multi-channel operation. One optimization which was done, is the increased event information ring buffer size in the "Global Channel Trigger Unit". Instead of 128 elements (like the event information ring buffers of the "single channel trigger" unit), every event information ring buffer there has 512 (32bit) buffer elements.

In order to achieve down sample ("srprop") compatible behavior the state machines in the "retrigger logic" got corresponding (counter) extensions. These changes just need to change the effective operation clock of the "retrigger logic". The "headeracp"/"headerintrace" facilities (which are requirement for the same named ADC Ring Buffer capabilities (introduced in chapter A.4.3)) are implemented in the read-out logics of the event information ring buffers.

Table 26 Possible trigger acceptance control settings

Minimum trace length	16
Maximum trace length	65536
Maximum number of buffer able event information elements	128 ("Single Channel Mode" via Single Channel Trigger Units) 512 ("Global Multi Channel Mode" via Global Channel Trigger Unit)
Status registers	Reg0: evtno
Control registers	Reg0: permtrgnum Reg1: srprop

A.6 Firmware design: Practical implementation of continuous HP-DMA

In chapter 4.2.3 a simplified continuous HP-DMA state machine was introduced (Figure 18). The implemented state machine of the thesis chapters 6.6 and 6.7 even further simplified its implemented addressing logic part. In the designed HP-DMA test implementation for real PMT/HPGe DAQs the undertaken simplifications needed to be redone and alternatively an address logic that fully complies to Figure 17 was implemented. The most challenging part in the continuous addressing logic design is not the addressing logic itself. Such state machines are quite easy – even if they consider the correct framing conditions (the signal shaping requirements are still valid!). The difficulties in the FC2.0 prototype are tied to the "reconstruction" information, which is required to get a correct read-out of the subbuffer trace. A read out of the subbuffer that considers the time sequence of the samples requires the knowledge of the address of its oldest (first) sample of the trace[23]. A system that obeys the framing conditions of the FC2.0 prototype trigger control logic (chapter A.5) enables a determination of the trace start address (in the subbuffer) on basis of the determined pulse position address. The "reconstruction" information must therefore enable a conclusion on the issued subbuffer address at trigger time. This information must be buffered in an Event Ring Buffer similar to the already introduced Event header information Ring Buffers ("pps","ticks",…) (for instance depicted in Figure 152).

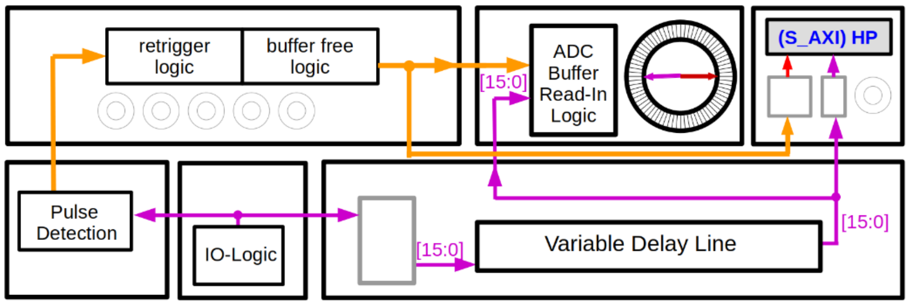


Figure 153 Interaction of the participated firmware blocks in the continuous HP-DMA setting. The variable delay line only has one delayed output signal. As the ADC Buffer Read-In Logic assumes an effective input delay of pre-pulse samples (remember at Figure 143). This contrasts with the continuous HP-DMA which assumes an effective input delay of zero (as it must jump after retrgcnt(_init) samples into the next higher subbuffer).

The continuous HP-DMA writes continuously into a subbuffer area until it receives a trigger pulse. A combination of the trigger acceptance control block and the HP-DMA enables a trace framing behavior that

[23] In practice it should not matter who determines the oldest (first) sample of the trace. De-facto (at least in seamless retrigger HPGe-spectroscopy) this also could be done by the server-read-out routine. But as the Server Routine has no access to the Trigger Signals of the Zynq Firmware it is much harder. Note that this assumption includes many premises about the "Trigger Acceptance Control" Blocks Trigger capabilities. In case of the "FC2.0 prototype DAQ" the "server reconstruction approach" could be critical in retrigger situations that start like the right subplot of Figure 148. As the "Server reconstruction" has in general a high analysis burden (that obviously can (but most not) limit the performance of the whole system), this approach in the FC2.0 prototype never was tried. In the (some kind similar) MATLAB Routine that verified the error free working of the "Trigger Acceptance Control" block much more information was available and even there the implementation of the trace analysis routine was not easy.

is identical to the trace framing behavior of the previously discussed trigger acceptance control/ADC Ring Buffer combination. Note that in the ADC Ring Buffer the trace read in starts with (the delayed) perception of the frame trigger pulse. The ACP-CMA also requires the “Variable Delay Line” in order to generate the “pre-pulse-samples” region of the traces (Figure 143). To comply with the “pre-pulse-samples” region regulation the continuous HP-DMA design does not need an additional “Variable Delay Line”. The HP-DMA FC2.0 prototype setup continuously writes into the memory – when the frame trigger arrives the pre-pulse samples are already buffered. After the detection of the frame trigger pulse a continuous HP-DMA will write further “retrgcnt” samples and then jump into the next subsequent subbuffer. In this ideal case the trace “framing” conditions are perfectly resolved. This ideal behavior is not always possible as address jumps into the next higher subbuffer are (roughly spoken) only possible with the start of a burst (remember the AXI4 protocol). In order to enable a continuous transfer, the next burst start address must be issued (and “accepted”) four transfer cycles before the accomplishment of the burst. In Figure 154 the effects of the discrete jump points on framing conditions shall be elaborated. Figure 154 illustrates a simple example in that a trace consists out of four burst transactions with 16 transfers per transaction. The total trace length “wlen” is therefore 64 samples. The (actually) oldest sample is stored at subbuffer address 0 (“Ad. 0”), the latest sample is tored at the subbuffer address 63 (“Ad. 63”). After accomplishing the write process at address 63, the continuous write process must jump into the next subbuffer and proceed there. Note that these addresses are random address positions. In practice every burst start address (in the subbuffer) has the same likelihood to be the oldest sample in the subbuffer. In this example a retrigger zone length of 16 samples is required (retrgcnt=16). Like the total trace length, the retrigger zone length must be a multiple of the burst size. The left subplot of Figure 154 shows a situation in that the trigger arrives four transfers after the start address of the corresponding burst. In this burst just 12 samples for the post-pulse region (“retrgcnt”) are available. In total therefore 8 samples violate with the ideal framing conditions. To get a post-pulse region that has 16 samples, the skip to the next subbuffer could be delayed to address 15. In this procedure the demanded post-pulse region length is achieved (in total 28 post-pulse samples), but the pre-pulse condition is violated (only 36 samples instead of the demanded 48samples). In total this approach has 24 samples that disagree with the ideal framing condition[24]. This approach is therefore worse than the original situation in the example. If the ideal framing condition is not achievable, it should be tried to get a trace with the highest possible agreement. But even in the case of the best possible tracing the sum of the pre- and post-pulse samples derivation can be up to the selected burst transaction size.

[24] Remember: A trace has exactly X sample in front of the detected rising edge of the pulse and precisely X samples afterwards. The trace always must have the same length of “wlen” samples and therefore wlen can be obtained by wlen=X+Y.

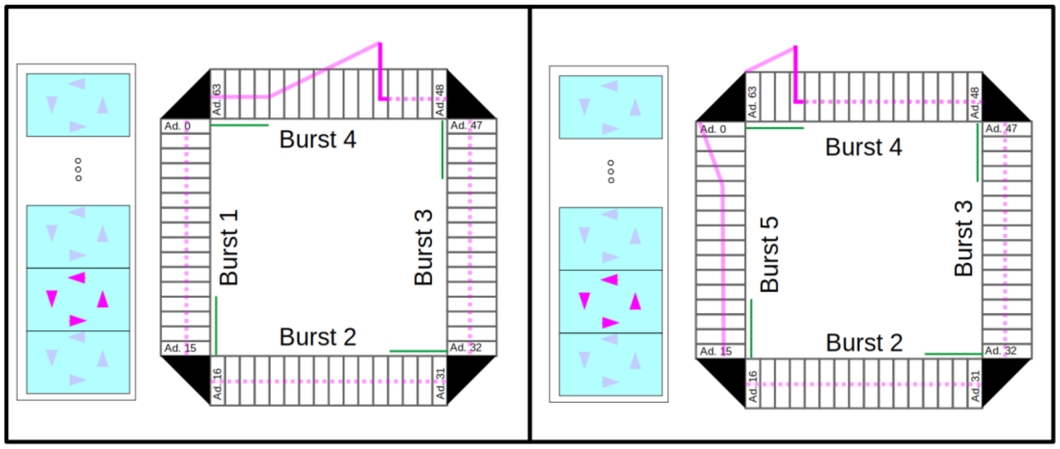


Figure 154 Continuous HP-DMA: Framing inaccuracy due to burst transaction size : In the example it is assumed that the post-pulse zone (also named as "retrgcnt") should have a region size which is exactly equal to the burst size (16samples in the illustrated example). Due to the "burst discretization" this is not always possible. The maximum derivation of the ideal framing is equal to the selected burst size. In the left subplot 8 samples disagree with the ideal framing conditions, in the right subplot 8 samples disagree too.

A possibility to dodge the "burst discretization issue" offers the subbuffer size. Instead of using a subbuffer size that is exactly equal to the demanded trace length, the subbuffer size (in the design) should cover the trace length and at least one additional burst. A simple subbuffer size extension can eliminate the issue. This situation is depicted in Figure 155. The knowledge of the pulse address (in this extended subbuffer) permits the read-out of traces that comply with the ideal framing rules.

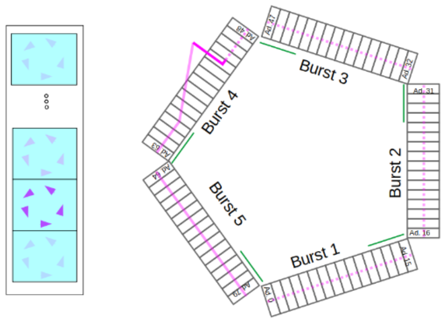


Figure 155 Continuous HP-DMA: Resolving framing inaccuracy via circle size ("Extended subbuffer size approach").

In the extended subbuffer design it is essential to know if the subbuffer contains a retrigger trace. The pulse address as "reconstruction" information is not sufficient anymore. The retrigger issue is solvable. More serious are the required hardware resources and their effects on timing closure[25] in synthesis. Remember that FlashCam 2.0 is limited on trace lengths that relate to the power of two (Table 25). Such control logics can be implemented gently with counter overflow approaches. Extended subbuffer sizes that are eligible for

[25] A FPGA design must consider like every digital circuit so called "Setup and Hold" times there the signal voltage levels must be in the defined low and high level voltage ranges of the corresponding digital electronic. The "Setup and Hold" times relate to the system clock. The "Setup and Hold" time effectively limit the possible operation frequency of a design.

the demanded trace lengths do not obey the power of two rule and consequently cannot rely on counter overflow approaches. Their “timing” is therefore more challenging and consequently it is more challenging to build up a high performance system with them. As the burst discretization issue leads to a maximum error equal to the selected burst transaction size and the future elaboration key point is on HPGe-detector setups, the implementation of the “extended subbuffer” approach was never started. In such setups the error is small in comparison with the total trace length and mostly much smaller than the given shape spare. In such setups a (slight) violation of the framing does not have significant influence on resolution.

One of the most important specifications was not discussed yet. In practice the time between the pulses digitization and its CPU accessibility should be as little as possible. The subbufer trace (on memory) is available for the CPU as soon as its last sample achieves its designated memory position. Many DMAs inform their CPU about such events via interrupts. Interrupts are not eligible for classic user space device drivers. If interrupts are not possible, the CPU has constantly to ask the DMA (or another functional unit) about the success/progress of its transfers. This procedure is called polling[26].

Apparently, the response channel of the AXI4 protocol looks like a reliable information source about the accomplishment status of the actually running transfer. This is an erroneous belief that enables corrupted traces. The transaction response (signal BRESP in AXI4) needs <u>not</u> refer to the transaction arrival at its final destination. In the FC2.0 prototype the selected signal values affect a transaction response of the AXI FIFO that is implemented between the HP ports and the DDR controller (in the PS area of the Zynq device). At this position the precise time closure of the remaining data forwarding (to the final destination on (external DDR) memory) is not reliably predictable. Thus in the case of the FC2.0 prototype, the BRESP signals cannot be used as reliable polling information.

[26] If polling is not done properly, the event building routine can invalidate the. The subsequent memcpy() transfer (of ethe event building routine) hence invokes outdated data. In this case the event building routine will transmit corrupted event data to the read-out server.

In the previous examples only a single HP port was used. A practical implementation that has several High Speed ADC channels likely has to implement several HP ports to achieve the required bandwidth. In chapter 6.7 designs with the same total HP-DMA bandwidth but differences in the implemented HP-DMA number also differ in their “continuous” write capability. Typically, a design with a smaller number of HP-DMAs shows a better “interrupt free” performance. This can be explained with the effects of the HP-DMA issued addresses on the DDRs controller efficiency. A DDR controllers’ efficiency (and therefore its available input bandwidth) depends on the issued address. If the addresses always are getting incremented the DDR controller can achieve a higher efficiency than a DDR controller that is fed with random addresses. As every DMA has its own address space (and consequently their asserted AXI addresses do not have any “interference”) the “randomness” of the DDRs controller input addresses is getting increased. Hence the DDR controller’s efficiency is decreasing, and the HP-DMAs struggle more to achieve a continuous transfer without write interrupts. Practical continuous HP-DMA designs that are on the fringe of continuity therefore can get a performance boost by the implementation of a (FPGA-RAM basing) prebuffers (in Figure 156 labed as “Burst Buffer”) that buffer several AXI transactions. For N transfer cycles the DDR controller then gets (incremental) addresses issued by “DMA 1”, afterwards it gets N incremental burst transaction are isssued by “DMA 2”. A rough example design that comprises two channels is depicted in Figure 156.

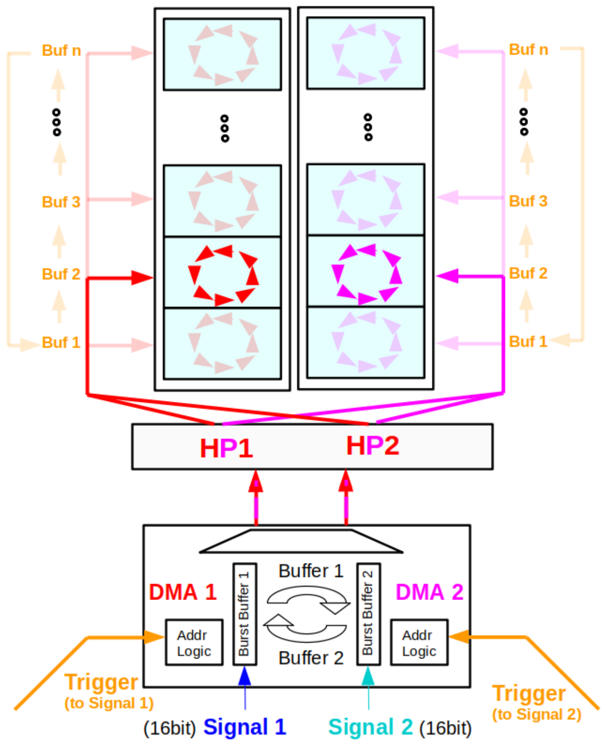


Figure 156 Continuous HP-DMA with upstream buffer logic to increase the “incremented address sequence length”

A.7 Software Design: User Space DMA Device Driver Implementation

Direct accessing of physical memory address, for instance via pointers, is for user space routines not possible as they are operated in a virtual (MMU given) address space. Devices just run correctly when their registers, that occupy a specific physical address, are set properly. To do this a correlation between the user's space virtual address space and the physical address device address space is required. In memory mapped files the correlation is organized by the OS. Therefore, classic user space device driver can be implemented via memory-mapped files. Linux offers for "correlated" memory accesses the "mmap" system call. Devices in Linux are threaded like classic data files. Before the "correlated" memory can be accessed via the mmap() call, it is important to instance the memory as file via the open system call. Like in all file instantiations, the memory access permissions and its synchronization behavior are determined by the chosen flags of the open system call. A functional implementation of the "mmap" functions requires the support of the whole FC2.0 toolchain. The firmware design and the Zynq PS/PL interface (address) mapping relates the DMAs operation address space to specific physical address ranges (Figure 157). The (user space DMA) device driver library must know these specifications. It is the task of the FC2.0 synthesis toolchain to do this correctly. Therefore the usage of the <mizzi-make> script as enhanced "make" substitute in the toolchain. Furthermore, the future running kernel must not occupy these addresses with anything. The kernel must know that these address ranges are for exclusive (firmware) DMA usage. Depending on the kernel version the reserved memory setting can be done via so called "Device Tree" files. In the compilation process a properly adjusted Petalinux toolchain therefore checks the design's device tree file after corresponding properties. The effective device tree of a system can be merged out of several distributed device tree "sub" files. In the FC2.0 Prototype setup the device tree file that has the reserved memory information is named "system-user.dtsi". The FC 2.0 toolchain must be able to adjust the corresponding values accordingly. The virtual (user space DMA) device driver library/framework approach is rather simple. First all (firmware block) DMAs must be initialized. In the initialization the "memory-mapping" (via the mmap system call) is done. This is done in the "init" function of the DMA device driver library. System memories distinguish in their read-write rights properties and in their coherence management. In UNIX the setting of the corresponding rights is done by the corresponding open function and therefore it cannot be chosen at mmap invocation time.

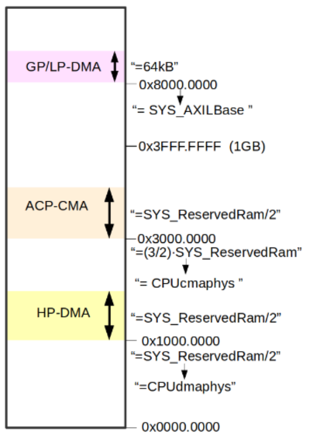


Figure 157 Physical address mapping of the Zynq CPU in FC2.0 prototype designs. The indicated physical addresses relate to the default design.

In principle the ACP-CMA data transfer(s) can be done by accesses of the returned (virtual address) pointer of the mmap() function. An unchecked, direct usage of these pointers has the risk that the (board) routine (falsely) tries to enter an address that was not included in the mapping of the initialization step. Trouble with (OS) signals and an abortion of the client routine can be the consequence. To avoid such errors the library has corresponding read and write functions that are easy to use. As the HP-DMA offers no hardware support for cache coherency, it may require cache control support via an additional device driver invocation. The library has therefore a “synch” function. The necessity of the “synch” invocation depends on the select synchronization flag (O_SYNC/O_ASYNC) in the open system call. Table 27 lists the chosen flags of the open function in dependence of the device and the DMA. The cache control has no influence on the correlation between physical and virtual addresses.

In case of the Zynq-Ultra+ FC2.0 design the “synch” function is implemented via inline assembly directives of the used gcc. Assembly directives can conflict with the actual privilege level of the routine (“User”/”Kernel”). In the Zynq-7k devices therefore the synch operation is already part of the corresponding memcpy() system calls.

Table 27 Selected properties of the (DMA) memory files

open() of	Properties
GP/LPD-IO unit	O_RDWR | O_SYNC
HP-DMA in Zynq-7k (ARMv7)	O_RDWR | O_SYNC
HP-DMA in Zynq-Ultra+ (ARMv8)	O_RDWR | O_ASYNC
ACP-CMA	O_RDWR | O_ASYNC

A.8 Software design: The FC2.0 ethernet read-out protocol

Remember that the read-out is done via command and status structures that were introduced in chapter 5.2. A FC2.0 measurement is split in several phases and therefore several structures exist. This chapter tries to elaborate ideas beyond these structures.

The command and the status structure of the “Trace Read-out” phase are roughly similar (The phase was originally labeled as “Measurement” phase, but as this was confusing the name was changed to “Trace Read-out” phase. The structure names of the “Trace Read Out” phase still remember on the old phase name (“messen” is German verb for “measure”)). The structures were a first draft in the development process and not all its parameters are getting used anymore. In the current protocol interaction just the parameters adc_num, simultrg, pps, ticks, startevtno, endevtno, transmiteuntilevetno and permtrgnum have an important function. The parameter names “pps” and “ticks” are misleading as they do not contain a time information – they are a redundance for the already introduced “request number” and are therefore would not be a must. The original intention was to design a protocol that also can be used with the first TCP-IP/UDP framework tries. Therefore these redundant parameters were implemented. The necessity of the parameters adc_num and simultrg depends on the mode. The actual server read-out routine uses them in an inefficient way. The parameter adc_num (in some modes) describes the channel which is assigned to the values this structure. In principle the “adc_num” parameter enables the implementation of a multi-channel mode in that every channel has its own (independent) trigger domain (do not mix this mode with global multi-channel domain mode, there all channels are integrated in a single trigger domain). The currently implemented multi-channel mode embraces all channel boards in a single domain and therefore would not need the “adc_num” parameter. In contrast to that, the parameters startevtno, endevtno, transmiteuntilevtno and permtrgnum are fundamental. The purpose of parameter “permtrgnum” (permitted trigger number”) was already unveiled in chapter 5.2. In chapter 5.2 the read-out and trace buffer management tasks were simply done on basis of the “permtrgnum”, “transmiteuntilevtno” and “(end)evtno” registers. In contrast to that, the developed protocol has three “evtno” parameters (“start…”,”end…” and “transmiteuntil…”).

```
struct messen{
unsigned int pps;
unsigned int ticks;
unsigned int startevtno;
unsigned int endevtno;
unsigned int transmiteuntilevtno;
unsigned int returnedevtcnt;
unsigned int permtrgnum;
unsigned int expectedevtno;
};

struct efbp_messen_command{
unsigned int adc_num;
struct messen evtstatus;
int simultrg;
};

struct efbp_messen_information{
int unify;
int adc_num;
int evtno;
struct messen evtstatus;
int simultrg;
struct debug debugparam;
};
```

Figure 158 Structure of messen_cmd ("struct ebp_messen_cmd") and messen_info ("struct efbp_messen_information")

The server needs a parameter that instructs the board routine to transmit a well-known, accurate number of traces. In FC2.0 prototype read-out protocol the task is done by the "transmiteuntilevtno" parameter. This is necessary as the "Receive" functions of the MPIK raw ethernet framework require the expected size of the received messages as input. As soon as the server-read-out routine can conclude the trace amount, it also can deduce the data amount (as every trace has a fixed sample size during a run, also the trace header size and the status structure size are well known). In the easiest scenarios the read-out server increases stepwise the values of the "permtrgnum" and "transmiteuntilevtno" parameter to organize the read-out of the desired number of traces. But this procedure has a dangerous drawback. After the approval of the "permtrgnum" value it takes an unknown, random time until the desired number of traces is available. The server therefore has to wait an unknown, random time for the corresponding reply. This impedes event management in the server read-out routine significantly, as every outstanding operation increases the consistency check and chronological event alignment efforts on the server. See that the server must also consider the possibility that one of its transmitted commands was lost and therefore the server never will get the corresponding event reply. In such situation the server should retry the command transmission. Recap that in real experiments a single server commonly has to read-out several boards. The server must prompt the input socket buffer with a sufficient frequency – otherwise the input buffer cannot absorb all incoming ethernet packets. In these cases, an elaborated error processing that likely includes redemanding of traces would be required. With increasing complexity of the consistency checks and the alignment efforts, it is getting harder and harder to prompt the input socket buffers frequently. At some point the server is becoming overstrained and it must throttle its trace prompting frequency. Therefore, the event management of the server read-out routine should be fed with incoming messages in a way that enables a fast analysis. This enables a high trace prompting frequency.

The "concurrent" prompting of several boards enables high performance read-outs. But the above descripted procedure already implies that its implementation is challenging. In the case of a first read-out routine that primarily is used to verify the correct function of the board, an easily debug-able server read-out routine is preferred. In the scope of this thesis the server routine just needs to control a single board. A (server read-out) routine whose event management is restricted on a single request without reply is consequently sufficient. In such a scenario it is important to reduce any wait time ("listing time") as every waiting time ultimately causes a drop of the system performance. This is the reason why the parameter "endevtno" was introduced. "Endevtno" indicates the latest number of events ("frame triggers") that were accepted by the "trigger acceptance block" and is therefore equivalent to the available "evtno" status registers (of the "trigger acceptance" and the "ADC Ring Buffer" block).

The intention of "transmiteuntilevtno" and "endevtno" is clear now. But why introducing the additional parameter "startevtno"? The (board and server) routines must work totally error free. The raw ethernet transfer includes a CDC check (done by the PS implemented ethernet controller), and therefore the likelihood of a wrong value due to a transfer error is rather unlikely. Unfortunately, this does not exclude other error sources that could tamper the values of "permtrgnum", "transmiteuntilevtno" and "endevnto". As a premature increasing of "permtrgnum" can have the most serious effects and the client routine has less possibilities to verify the validness of the actual "permtrgnum" value, "startevtno" as security parameter was introduced. "Startevtno" represents the last event number which was successfully transferred to the read-out server. The client is permitted to overwrite all corresponding traces in the ADC Ring Buffer. Thus "startevtno" enables a verification of "permtrgnum". In contrast to that, the values of "endevtno" and "transmiteuntilevtno" cannot be verified with "startevtno".

```
struct traceheader{
unsigned int evtno;
unsigned int deadevents;
unsigned int pps;
unsigned int ticks;
unsigned int adc_num;
int          num_of_adcs;
unsigned int num_of_samples;
float        deadtime;
unsigned int cmd_num;
unsigned int blsum;
};
```

Figure 159 Trace header structure

The trace header content must enable an error-free read-out of the (file) stored data and furthermore provide all information which is required for the desired analysis. The FC2.0 prototype store files contain all phase "_info" structures (Figure 27) and all desired traces (including trace header) of the corresponding measurement run. The "_info" structure contain a significant amount of information. Some information can be used to check the validness of the file without enhanced conclusions.

In the current design the trace header has the fundamental parameters evtno, pps, ticks, adc_num, num_of_samples and blsum. The remaining parameters are useful, but not essential and do not have an impact in the read-out server routine. "cmd_num" contains the corresponding data out of the "retrginfo" event ring buffer (Figure 152 (page XXVII)). In the scope of this thesis, the information is not used. "pps" and "ticks" contain the accurate time information of the event. Do not mix them with the same named "pps" and "ticks" parameter of the "messen" structures (Figure 158). "adc_num" indicates the channel number (of the board) that belongs to the trace data. "num_of_samples" specifies the number of samples per trace. "blsum" is the baseline level that was determined by the baseline follower.

The lack of any random waiting time was demanded to ease the read-out management in the server routine. Consequently, the server requested "transmiteuntilevtno" value must not be bigger than the "endevtno" value of the previous client reply.

A.9 Software design: Server-Read-Out Routine

The Server-Read-Out routine uses well developed MPIK libraries and therefore contains less innovations. Its most important innovations concern the ADC nonlinearity determination. The corresponding approach and its innovations are topic of “ADC nonlinearity determination” chapters of the thesis. Innovations that consider the trace read-out performance are not implemented. Effectively the current FC2.0 server read-out routine is not eligible to control several boards concurrently and, in this respect, less developed than the FC1.0-Legend server read-out routine. To handle the massive message emergence in multi board operation the FC1.0-Legend server read-out routine uses smart message queues (inspired by ØMQ) and multi-core threading. Both, FC1.0-Legend as FC2.0 prototype, use the MPIK raw ethernet library eftb and therefore have for a single board read-out process a comparable performance maximum. The setup therefore permits a realistic verification of the possible event build performance under a realistic ethernet charge. Again: The FC2.0 read-out routine has its focus primarily on verification and secondarily on the read-out performance. The usage of the server-routine (and the additional available software stacks) must enable a verification of the firmware. This makes sense, as the thesis has the intention to determine the performance and the possibilities of the Zynq-devices, but not of its connected laptop machine (that used a default Intel CPU). The server read-out routine does manage securities copies (in previous chapters they were called history structures), that have unnecessary overhead that is merely implemented as additional debugging support. These eases debugging and verification but impedes a fast server-read-out routine. Only routines that work completely reliable are useful as debug routines. As multi-core threading and its synchronization is definitely more error-prone for coding errors, its implementation was excluded in this thesis, though the MPIK also has elaborated thread libraries.

The Server-Read-Out Routine chapter is sub split in three subchapters. In the first chapter the rough sequence of the server read-out routine in its “Trace-Read-Out” state is elaborated. The second subchapter investigates the state-machine that is responsible for the correct management of the ethernet protocol event build parameters “startevtno”, “transmiteuntilevtno” and “permtrgnum”. The last subchapter gives an introduction to the implemented analysis possibilities.

A.9.1 Read-Out Routine procedure in Measurement State

Of all measurement phases (Figure 25) the “Trace Read-out” state is the most decisive one for the final performance. That’s why the procedure in this phase is the focus of the elaboration. Figure 160 illustrates the sequence and internal dependencies of the most important steps. In the “Trace Read-Out” state also the determination of the messen_cmd request message content must be done. Its parameters “startevtno”, ”transmiteuntilevtno”, ”endevtno” and “permtrgnum” are vital for the correct control of the event building routine on the client and for the message reception block on the server. After the accomplishment of the previous “Filter & Trace Setting” Phase of the Measurement, all “Measurement” phases start with the transmission of the initialization messen_cmd request message to the client (Start position is therefore the "EfbpSend()" rectangle in Figure 160). In the initialzatin messen_cmd request message "startevtno","transmiteuntilevtno" and "permtrgnum" have a value of zero. This is important as in the next step the server routine must determine the expected reply message size. As "startevtno" and "transmiteuntilevtno" are zero the corresponding reply message only contains the messen_info structure. The size of the reply message is therefore known and reception routine in the next step ("EfbpReceive()") can consequently assert if the corresponding ethernet transfer was successful. Note that the server routine forces the compliance of a given time limit. If the client routine does not reply (with the expected (complete) reply message) in this given time range, the reception routine returns an error. Furthermore, the equivalent (request) reception step of the client routine was not permitted to expect a particular command request message. In the client routine the measurement phase always can be interrupted by command requests of another measurement phase. In contrast to that the server routine must precisely expect the structure details of its received messages and therefore assume a particular command message as historic ancestor requirement. In the next step the received reply message bears a more elaborated error check, and the history structure of the server routine is accordingly loaded. If the message contains traces, they are stored (on the hard drive) in the next step.

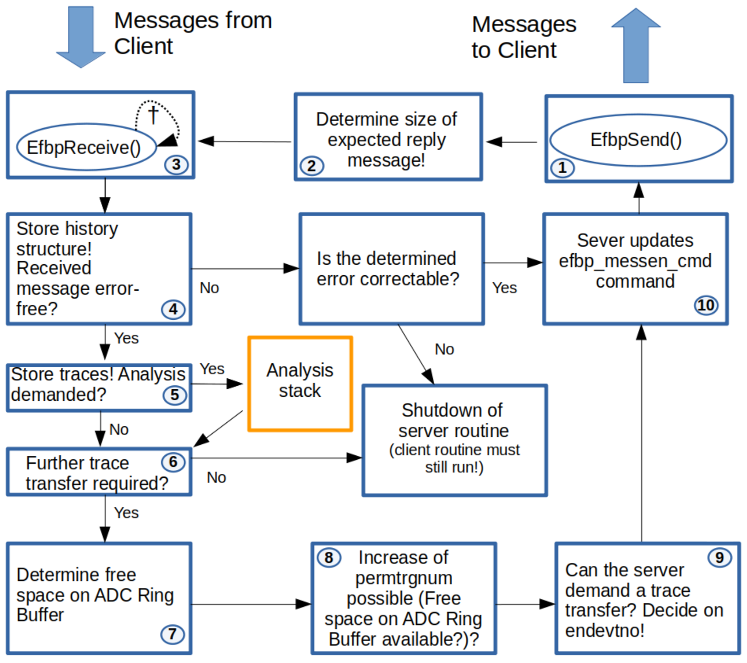

Messages from Client
Messages to Client
EfbpReceive()
†
3
Determine size of expected reply message!
2
EfbpSend()
1
Store history structure! Received message error-free?
4
No
Is the determined error correctable?
Yes
Sever updates efbp_messen_cmd command
10
Yes
No
Store traces! Analysis demanded?
5
Yes
Analysis stack
No
Shutdown of server routine (client routine must still run!)
Further trace transfer required?
6
No
Yes
Determine free space on ADC Ring Buffer
7
8
Increase of permtrgnum possible (Free space on ADC Ring Buffer available?)?
Can the server demand a trace transfer? Decide on endevtno!
9

Figure 160 Sequence of the server routine. An overview over the analysis stack is given in chapter A.9.3.

If the user selected an analysis option before the measurement routine was started, the server routine starts its analysis chain on it. In the next step the server routine resolves if it must acquire further events. The user can insert the desired number of events when he prompts the server-read-out routine. In debug and analysis mode flexible measurement termination and progress steps are also possible. The extensive debug &

analysis chain will be discussed in chapter A.9.3. The determination of the “permtrgnum”, “startevtno” and “transmiteuntilevtno” value for the next request message, requires the knowledge of the free space on the ADC Ring Buffer (or on an equivalent memory buffer in HP-DMA mode) and the actual availability of trace. This is done in steps 7-9. A detailed analysis of these determinations, that are done by the “ethernet protocol management” state-machine, is done in the next chapter A.9.2 and therefore omitted in this chapter. Note that the determination strategy significantly influences the performance.

A.9.2 Trace read-out state machines

The trace read-out state machine must control the whole “Trace-Read-Out” phase. Figure 161 depicts a simplified version that does not include the initialization and outrun specific mechanism. In the initialization mainly two tasks need to be done. Firstly, the control parameters “transmiteuntilevtno”, “startevtno” and “permtrgnum” must be set to zero. Note that this command agrees with a restart of the trigger system (on the corresponding Zynq) and therefore requires some special considerations that are not covered in this chapter. These uncovered constraints are the reason why in the actual FC2.0 prototype implementation the new start of a “Trace Read-Out” phase always requires a previous operation of previous measurement phase (“Filter & Trace Settings”). This is not a final restriction, in principle a work-around exists. Secondly a successful initialization needs to know how many traces can be buffered on the Zynq-device. A server cannot know this on its own, it must be informed. Either by the user (who starts the server-read-out routine) or by the Zynq-client itself.

The function of the trace read-out state machine can be explained with four sequential steps. In the first step it is tried to determine if requested trace transfer was successful. The errorfree parameter relates to an error monitoring structure that is managed in the error checking steps of the measurement phase reception subblock (steps 4-5 in Figure 160). The error free parameter is true, when the “last” transaction was successfully accomplished. If the transaction was successful, this also means that on the ADC Ring Buffer is free space for trace buffering available. In the state machine it is the task of the parameter “freetraces” to indicate the “actually” available free trace ADC Ring Buffer space. The reverse information is maintained by the “occupiedtraces” parameter (the corresponding ADC Ring Buffer space can actual not used to buffer further traces as it has traces that was not read-out yet). The parameters “freetraces” and “occupiedtraces” must be adjusted accordingly. If everything is done properly, the state machine is designed in a way that “stateerror” parameter will never be set (even if a message transfer is completely messed). A set “stateerror” therefore is never permitted to occur. In the second step the value of “permtrgnum” is adjusted accordingly. In the implemented FC2.0 prototype server routine this step is more elaborated as implied by this state machine draft. The implemented code must also manage the “out-run” of the state machine. In some modes the last event is already determined at the beginning of the measurement, but also modes exist, there the final event is determined during the run time of the measurement. As soon as the corresponding condition is recognized, the server routine is not allowed to allocate new free buffer space (“freetrace”) anymore. The original intention of this requirement was to enable an option for a user demanded resumption of the measurement. The implementation of the “halt” strategy was successfully accomplished, but the “resumption” option was never implemented. The code snipped in Figure 161 clearly cannot handle these (soft) requirements. For the analysis in this thesis the given code snap is sufficient. In the third step the parameter “freetraces” is updated accordingly.

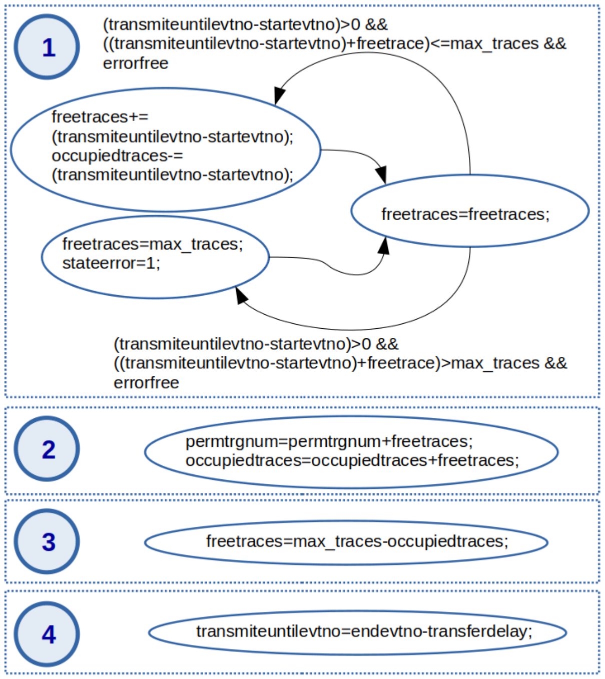

1
(transmiteuntilevtno-startevtno)>0 &&
((transmiteuntilevtno-startevtno)+freetrace)<=max_traces &&
errorfree
freetraces+=
(transmiteuntilevtno-startevtno);
occupiedtraces-=
(transmiteuntilevtno-startevtno);
freetraces=freetraces;
freetraces=max_traces;
stateerror=1;
(transmiteuntilevtno-startevtno)>0 &&
((transmiteuntilevtno-startevtno)+freetrace)>max_traces &&
errorfree
2
permtrgnum=permtrgnum+freetraces;
occupiedtraces=occupiedtraces+freetraces;
3
freetraces=max_traces-occupiedtraces;
4
transmiteuntilevtno=endevtno-transferdelay;

Figure 161 Trace read-out state machine implemented in read-outserver

In the fourth step the “transmiteuntilevtno” parameter is determined. The roughly drafted approach in chapter 5.2 implies a simple procedure like $transmiteuntilevtno = endevtno$. In contrast to that the relation is $transmiteuntilevtno = endevtno - transferdelay$. The user selected value of “transferdelay” always must be smaller as the value of “max_traces”. The parameter “transferdelay” was introduced as a fundamental requirement for an error-free read-out of the retrginfo event ring buffer. In the scope of this thesis the “retrginfo” information is not used. In the thesis an assumed value of zero for “transferdelay” is always sufficient. Note that the given code snipped of “step 4” will cause an erratic (“infinite”) out-run behavior if the selected “transferdelay” value is bigger than zero. The real implemented code can also handle such situations.

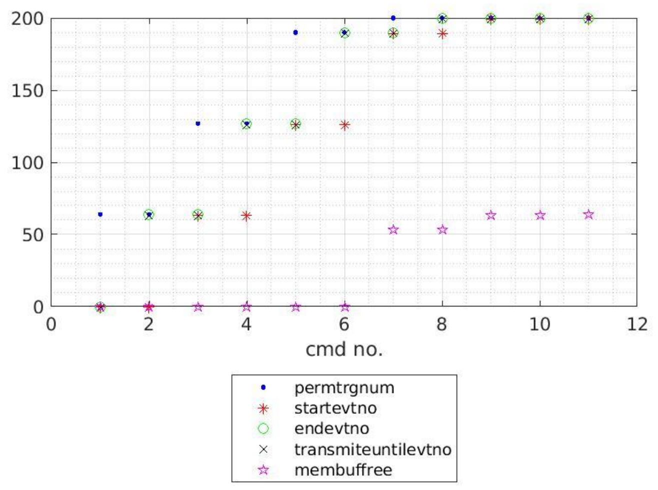


Figure 162 Simulated control trend on basis of the implemented "trace read-out state machines". The simulation was done with MATLAB routine "Checking_FlashCam2_0_Protocol_v2.m". In the simulation the server had to demand 200 events and afterwards halt the measurement. The chosen value of "max_traces" was 64 (a feasible value for a HPGe-detector measurement with the FC2.0 prototype) and "transferdelay" had a value of 0. From command number 7 the snipped code of Figure 161 is not eligible to determine the plot values as the simulation considers the implemented "out-run" of the state machine.

The depicted simulation in Figure 162 uses a trace read-out state machine that agrees with the implemented version. The x-axis relates to the number of transmitted requests (containing "messen_cmd" structures with the listed legend parameters). The y-axis obtains the value of the corresponding control parameter. In the simulation the read-out server routine had to claim 200 events. Until command no. 6 the control parameter trend agrees with the simplified state machine of Figure 161, afterwards already the out-run behavior is dominating. The "endevtno" values belong like the remaining control parameters to the (server) request message. As a server has no direct access to the endevtno (status) register of the firmware, its value in the request message is taken from the last valid reply message (by the server). In the simulation immediately after the "unlocking" of the ADC Ring Buffer space (by accordingly increasing the permtrgnum control register value), the ADC Ring Buffer is occupied with fresh new traces. Therefore, the reply message (whose values are not shown in Figure 162) parameter endevtno always has the same/equal value like the requested permtrgnum parameter. In real life such a fast reaction is mostly impossible. A real read-out of 200 traces (with comparable settings) would require more than the simulation indicated 9 request messages. The simulation results are important for the future discussion as it clearly shows that the implemented read-out protocol needs at least two messages to prompt a trace amount of the bufferable "max_traces".

The most severe deficit of the suggested trace read-out state machines is their lack in ethernet efficiency management. It always tries to demand the maximum number of available traces (tagged by endevtno). In some situations, this leads to transfers that just use a minimal fraction of the possible packet payload. A feasible workaround with the given structures is not always possible. A deliberated reduction of the demanded traces can be used to gain payload efficiency. If trace size is in the range of the designated ethernet packet size (or bigger) the reduction approach fails. Furthermore, the protocol only allows transfer of complete traces, but it does not permit the transfer of partial traces. Enabling partial trace transfer could help to increase the ethernet payload efficiency independent of the given trace length. As in the FC2.0 prototype not all firmware features (“headeracp”) support partial trace transfer, a client routine implementation of this approach may require in some modes significant (additional) workarounds that negatively influence event building performance.

The protocol behavior in Figure 162 affects a “delayed” permtrgnum control register increase. In principle the client routine can self-dependent decide on the endevnto status register (and the already available init_settings_info structure) if it can increase the permtrgnum control register value. Currently the read-out server must do this decision, and this leads to a time delay of a message transfer. In chapter 5.2 the main cause for the current procedure was the danger of a failed ethernet transmission. In case of a failed ethernet transmission the server must demand the corresponding events at least twice. This requires that the event data is still available and consequently it was decided that an early self-deployed increase of permtrgnum is not possible. The capability to retransmit previous events is a hard must. But it is not correct that this excludes a self-deployed increase of permtrgnum control register. It is correct that a self-deployed permtrgnum increase likely will overwrite the events data in the FPGA-RAM Buffer Rings (and memory). But if the routine is proper designed the old trace (reply message) data is still the only available in Cache (“(User Space) Ethernet buffer”).

A.9.3 Overview read-out server analysis chain

The server analysis chain uses MPIK libraries that are already implemented in several experiments. The signal analysis uses for pulse shaping and the FFT operations extensively the “sample” library (Developer: Thomas Kihm (MPIK, Heidelberg) a. Marco Salathe (MPIK, Heidelberg)). The sample library offers essential and enhanced DSP functions. Though same named functions in MATLAB and its available toolboxes commonly exist, their implementation differs significantly. Pulse shaping in this thesis always relates to the sample library. In order to plot the analysis data, the MPIK has a various range of toolchains. Mostly they base on Root, Phyton or GNU plot. The final plots that are shown in this thesis were generated by MATLAB routines. (Fast) Debugging and system monitoring does not require the elaborated plotting capabilities of MATLAB nor GNU plot. It requires a fast and high performant illustration of the (trace) data. A lightweight X-Window system is a feasible solution. The y2dplot library bases on the X-window “X11” library and is optimized for implementation in DAQ system debugging task (Library developer: Thomas

Kihm (MPIK, Heidelberg)). This chapter is a about the analysis possibilities of the server-read-out routine. The trace storage file API was developed in a way that enables a simple read-in other routines. A pure C server analysis routine and several MATLAB analysis routines that are capable to open and read-in such trace storage files exist.

The typical analysis steps in the server-read-out routine are depicted in Figure 163. Step 1 translates the available integer sample traces (from the reply messages) into double traces that can be used by the sample library. For the default 16bit sample firmware operation mode the corresponding procedure is trivial. In non-default 18bit modes, that are essential for special verification jobs but also can be used for feasible measurements, the correct double transformation must consider the firmware implemented bit distribution over several ACP-CMA channels (the implemented design is superficially suggested in chapter A.4.3). As the 18bit options are not used in published measurements, the implementation of this procedure is not discussed.

In some designated analyses it is important that the pulse edge has constant position in the trace and the traces show no pile up (no pulse interferences). If a trace violates this criterion, it must be rejected in corresponding analysis. The trace rejection is the (optional) task of step 2. If enabled, the traces are only rejected in the analysis. The traces are still part of the (optional) trace storage file. In step 2 the server does not analyze the virtual trace data. It uses the corresponding status word out of "retrg_info" event ring buffer that is also part of the FC2.0 event header (although it was not included in Figure 29). The "retrg_info" event ring buffer gains its information out of the internal states of the retrigger logic subblocks. As the input triggers of the retrigger logic must be triggered by the pulse detection block (of the FC2.0 firmware), the settings of the pulse detection block are decisive for the success of this rejection strategy. Another potential problem of the suggested approach is caused by the limitations of the implemented retrigger logic. The implemented retrigger logic will fail in some reapproval situations of previously occupied buffers (Figure 148 (page XXII)). In such situations the "retrg_info" content can lead the analysis code of step 2 to a false acceptance of the trace. But this also means that errors only can occur when the event rate is close to the maximum event rate or the pulse detection block is not operated accordingly. If the settings are proper chosen and the event rate considers the maximum limit with sufficient spare, an error will likely not show up. The "retrg_info" buffer is a feature of FC2.0 prototype and was not implemented in FC1.0. Like the 18bit option, the code belonging to step 2 is not used in this thesis.

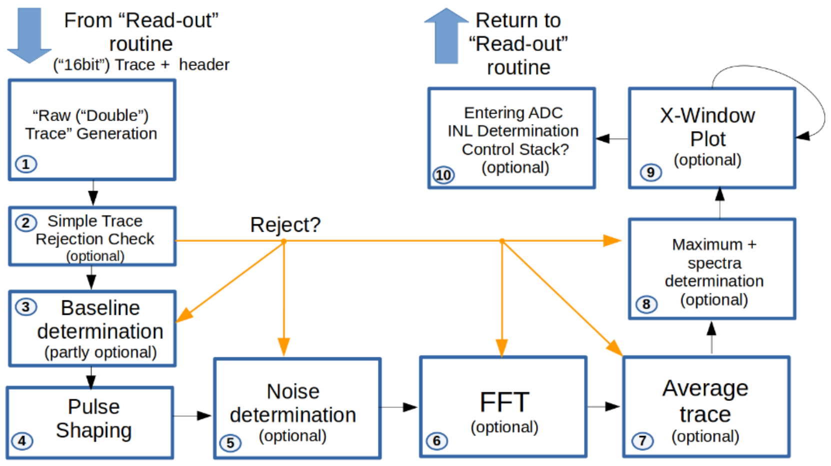

From “Read-out” routine
(“16bit”) Trace + header
Return to “Read-out” routine
“Raw (“Double”) Trace” Generation
1
Entering ADC INL Determination Control Stack? (optional)
10
X-Window Plot (optional)
9
2 Simple Trace Rejection Check (optional)
Reject?
Maximum + spectra determination (optional)
8
3 Baseline determination (partly optional)
Pulse Shaping
4
Noise determination (optional)
5
FFT (optional)
6
Average trace (optional)
7

Figure 163 Analysis stacks internal steps

Sample rate and trace length commonly determine the analysis possibilities. The effective FC2.0 prototype sample rate depends on the selected “srprop” down-sample value (chapter A.2). The FC2.0 prototype can down-sample until to 2^{10} clock cycles, this means that (in default firmware setting) its lowest sample frequency is around 200 kHz (instead of the default frequency of 200 MHz). A trace with a length of 65536 samples consequently enables a FFT bin width of roughly between 6 Hz-6 kHz or a time range between 0.3-300 ms. The analysis routine automatically considers the chosen “srprop” value. Changing “srprop” is also a simple possibility to verify the determined noise values of the server-read-out routine and its trend can provide information about possible setup errors.

Pulse shaping (step 4) and average trace analysis (step 7) is mostly related to traces that were triggered by accepted pulses. FFT and baseline analysis is sometimes useful for traces that should not have any interference with a pulse. The firmware also offers the possibility to induce a pulse detection trigger by CPU invocation without any real pulse riding on the signal. The “simultrg” property of the “messen_cmd” structure informs the client CPU to induce such a trigger. In firmware the implemented CPU trigger induction unit is independent of the pulse detection block. By coincidence the CPU can induce a trigger when the signal trend is charged with a pulse. The analysis routine therefore cannot certainly rely on a pulse-free trace. An analysis of the “retrg_info” value is also not useful in this case, as the information only permits temporal conclusions on the (actually) connect trigger inputs of the retrigger logic. In all analysis steps after the pulse shaping (step 4) its results can be used to reject a trace with an eventually charged pulse. The usefulness of this procedure also depends on the chosen pulse shaping approach.

In the setup of the thesis, it is mostly unnecessary to do an elaborated baseline determination (step 3) as the baseline value of the baseline follower is already implemented in the trace header. To verify the given baseline follower values alternative baseline determination approaches were also implemented. In one approach the baseline value is determined by a simple mean value determination of a selected trace region.

In case of a real HPGe-detector spectrum measurement the proper operation of the trace rejection (step 2) can be a requirement.

The pulse shaping step (step 4) offers the common shaping approaches in HPGe-detector spectroscopy. Besides the pole-zero compensation, it has a subsequential gaussian, trapezoidal and cusp shaping available. Consider that the shaping times are restricted by the available trace length. The pulse shaping comprises several steps. The analysis functions of steps 5, 6 and 7 also have (optional) access to these intermediated steps. If the user does not include a shaping argument (when invoking the server read-out routine), the server read-out routine tries to derive the "best" settings from the corresponding firmware settings. The chosen settings are also conserved in the result files. It is explicitly not recommended to devolve this task to the routine. Experience shows that a user can find more appropriate settings.

In the implemented code "noise determination" (step 5) simply is done by standard derivation determination of the selected trace region of interest. The selected region must be free of pulse charges.

An average trace (step 7) is the combined mean trend of several (accepted) traces. Like the FFT step, the average step has access to all relevant results of the pulse shaping. Averaging combined with proper "srprop" and "tsel" settings can enable a fast generation of high-resolution trends. In HPGe-spectroscopy average traces of accepted pulses are not useful. Their major use is in debugging and FADC specification. The latter especially considers INL/DNL determination (by the "FC1.0 procedure").

"y2dplot" function (out of the y2dplot library) is a sober plot possibility to get black and white plots (1bit mono) of a given (numerical) trends. The input arrays (whose figure trend will be plotted) must be generated in a y2dplot API compatible way. The y2dplot function has various (continuous) zoom-in and plot style features that are selectable/adjustable via prompted "hot key" usage during the run time of the routine. After its invocation (and window update with the new trend) in the routine, it waits for an infinite time for user keyboard input. If this is not considered, an unintended halt in run can be the consequence and stop the originally designated measurement. Therefore, the invocation of the y2dplot function always can be suppressed by a corresponding input argument at the invocation time of the server-read-out routine. The repatriated arrow in Figure 163 is a try to pay attention to this critical behavior. >

The developed INL/DNL determination routines require additional instrumentation control possibilities. The internal procedure of the "ADC INL Determination Control Stack" depends on the selected method. The "GERDA" approach procedure is not selectable in the Read-Out server routine. The "GERDA" procedure is done under the supervision of a bash script that controls the participated instrument and the FC2.0 prototype. The latter it controls via the Server-Read-Out routine.

A.10 Speed 2: Detailed operation performance discussion

The arithmetic elaboration results of the speed2 routine are summarized in Table 11.

Table 28 Arithmetic performance according to speed2. Speed2 is a single core thread that does not use AVX/NEON instructions. Recap that the listed values are from Zynq devices with a speed grade of -1. Other speed grades might provide a better performance. The used Raspberry Pi 4 has a quad core ARM Cortex A-72 CPU with a processor frequency of 1.5 GHz. The ACAP devices of Xilinx use a dual core ARM Cortex A-72 CPU with a processor frequency of 2.0 GHz.

MIZZI MOPS type	Zynq-7k	Zynq-Ultra+	Raspberry Pi4	Laptop
int	398.972	1331.987	1517.971	2177.029
short int	372.018	1310.323	1508.977	2296.541
double	117.527	205.643	412.565	1203.165
float	105.448	186.678	439.380	1084.195

The speed2 double arithmetic performance of the Zynq devices is significantly lower than the performance of compare CPUs. Note that speed 2 does not determine the possible maximum arithmetic performance. The possible maximum performance may be much higher, but this implicitly requires the usage of NEON instructions and efficient parallel core processing. This is not in all applications possible. [19] and especially [20] evaluate the possible maximum arithmetic performance. In HPGe spectroscopy double is the preferred data type. NEON uses 128bit register, in the best case it therefore can operate two double entries in parallel.

Table 29 DSP functions out of Mizzis DSample Library. Note that the given values of the "arithmetic instructions per input sample" is a rough lower limit guess that in practice will be significantly exceeded and only relates to the pure signal processing arithemtic (the arithmetic operations for the value loading/storing are not considered). In the functions "SmoothSamples" and "LRAverageSamplesFilter" the precise arithmetic instruction number strongly depends on the shaping condition ("integration time").

Function	Arithmetic instructions per input sample	Filter style
MultiplyAddSamples	2	FIR
DifferenceOfSamples	1	FIR
SmoothSamples	At least 4	FIR
LRAverageSamplesFilter	At least 3	IIR
InverseHighPassSamplesFilters	4	IIR
InverseLowPassSamplesFilter	3	FIR
"Trace Windowing in FC"	1	FIR
Extremumsearch via "?:" operation	No arithmetic DSP operation and therefore not part of speed2!!!	No FIR/IIR filter!
Buffercast (short int into double)	No arithmetic DSP operation and therefore not part of speed2!!!	No FIR/IIR filter!

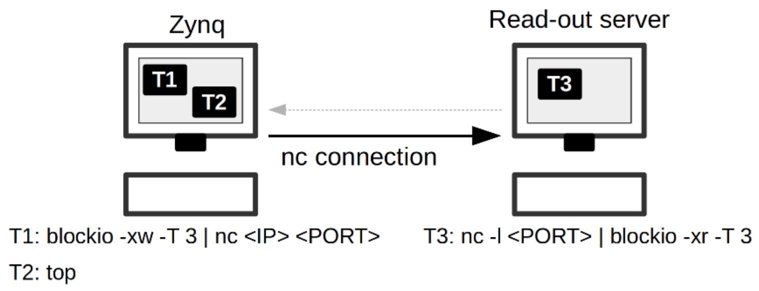
In the FC framework the fastest HPGe signal processing merely does a baseline compensation (via DifferenceOfSamples()), a pole zero compensation (via InverseLowPassSamplesFilter()) and a Gaussiann Pulse Shaping (1×MultiplyAddSamples(),1×”Windowing”,1×LRAverageeSamplesFilter()) and therefore needs at least 10 arithmetic operations per input sample. The rough maximum trace analysis frequency of such a shaping can be obtained by equation XY.

$$f_{analysis} \approx \frac{P_{Mizzi\_MOPS} \cdot \eta_{CPU\_Available}}{A_{Op} \cdot len_{trace}}$$

The Zynq-7k therefore has a theoretical (single core) analysis performance of ~23.5k traces (with 512samples) per second. In this analysis it was assumed that a buffer type cast and an extremum search are not necessary. Furthermore, in practice the APU of a Zynq is also charged with event building tasks and therefore never 100% of the CPU performance can be used for the trace analysis. A more “realistic” guess with a CPU availability of 40% ($\eta_{CPU\_Available}$=0.40) and higher average operation amount of 16 operations per input sample (A_{Op}=16) leads to ~5,7k traces per second. For some experiments this performance may be sufficient. Note that in experiments all channels of a device must share this performance. Moreover, this simple analysis chain is only possible, as the firmware already provides the baseline value. The suggested simple shaping is in practice not really appealing, as the firmware (“tsel” in the pulse detection block (Figure 130) (not discussed in this thesis)) already offers a similar (integer) shaping approach without CPU involvement. In real life a user is likely interested in enhanced pulse shaping and analysis methods. The performance would drop further.

A.11 Setup Zynq netcat performance evaluation

The focus of this setup is a rough determination of the genuine ethernet performance and costs. The actual ethernet performance of an application depends on memory and CPU dissipation of the routine. Additional memory accesses (besides the absolutely required ethernet buffer transfers) and CPU workloads deteriorate the ethernet performance. An unfavorable routine therefore can achieve a bandwidth that is far below the specified genuine ethernet performance. In order to assess the ethernet performance of the application it is therefore useful to know the (rough) trend of “genuine” performance. For this reason, a Server/Client Routines could be written that solely transfers an infinite data stream from the Zynq to the Read-Out “server”. As on both systems nc (“netcat”) is available for ethernet data transmission the development of the routines would consume time without any significant reward. Netcat is a modest utility whose efforts are centered in ethernet data transfer (connection and listen) tasks. As nc can relay any standard input stream via ethernet and forward received ethernet packets as standard output it is sufficient to operate an (TCP/UDP based) simple genuine ethernet transfer between Zynq and Read-Out “Server”. In the simplest genuine ethernet check setup nc on the transmitter side (“Zynq”) could be feed by the zero device and on the receiver side nc is connected to the null device. Running “top” on a parallel terminal shows the actual (approximated) CPU time consumption of every running process. This simple setup would run a “genuine” TCP ethernet transfer, but it is not capable to output the actual ethernet performance trend. Neither nc has any ethernt performance output, nor an indirect conclusion is possible, as also the outputted data amount of the zero device is also unknown. MPIK has therefore its “blockio” util (Developer: Thomas Kihm) that like the zero/null device can be used as data source and sink. But in contrast to the zero/null device it also tracks the amount of data and frequently outputs the determined actual bandwidth of its input/output streams. The resulting setup is illustrated in Figure 164.



```
T1: blockio -xw -T 3 | nc <IP> <PORT>
T2: top
T3: nc -l <PORT> | blockio -xr -T 3
```

Figure 164 Setup “genuine” ethernet performance determination via nc. The block rectangles represent actively running terminals that are connected to the corresponding machine. The terminals running bash command is shown below the symbolized systems. <IP> is the IP(v4) address of the read-out server, <PORT> is the corresponding port number of the listen “nc util” on the read-out server. “nc” knows TCP and UDP ports. The chosen setup uses the TCP port setting. Note that in Linux Jumbo Frames are not enabled by default and not all device drivers know Jumbo Frames. A nc ethernet Jumbo Packet transfer therefore requires an initial Jumbo Frame approval at the used network interface (via ifconfig <INTERFACE_NAME> mtu <DESIRED_JUMBO_PACKET SIZE>).

A.12 Results Zynq netcat performance evaluation

The performance results are listed in Table 30. Memorize that the PS 1Gb/s ethernet of both devices is promoted as 1Gb/s solution. The Zynq-7k setup merely has an ethernet performance of 38MB/s, the Zynq-Ultra+ setup confirms the supposed effective performance of 120MB/s.

Table 30 nc PS ethernet performance on Zynq devices . Xilinx does PS-PL ethernet performance test with iperf (instead of nc). Xilinx published its PS-PL-ethernet tests in [42]. This PS-PL ethernet tests can use the more performant MGT pins of the PL. Xilinx continuously updates its published information. In the past Xilinx also published the bandwidth of "modified" ethernet setups (for various settings and implementations) (for instance in [43]). Unfortunately, this detailed information is currently (2022/07/09) not updated anymore. Important for us, is to note that even the implemented Linux Kernel has influence on the performance (Xilinx still discusses the kernel influence on the latest version of [42]).

	CPU single core consumption	**Transfer rate**
Zynq-7k (without Jumbo Frames - unpatched) (mtu=1500)	~50% PS-PL investigated by Xilinx: ~93% (TCP in [42]) ~63% (UDP in [42])	38.9MB/s (σ=4.4MB/s) PS-PL investigated by Xilinx: 81.8MB/s (TCP in [42]) 60.8MB/s (UDP in [42]
Zynq-7k (with Jumbo Frames – patched – TX only!) (done by Thomas Kihm (MPIK, Heidelberg))	~90% (???) -Value according to the reminiscence of Thomas Kihm -	~90MB/s (???) -Value according to the reminiscence of Thomas Kihm -
Zynq-UltraScale+ (mtu=9000)	~18%	118.4MB/s (σ=3.6MB/s)

The TCP ethernet performance of the Zynq-7k setup requires the support of roughly 50% of the available (single core) CPU performance. Such a fraction is critical as the CPU in DAQ systems is mainly intended for the event building process. As the pure memory read performance of the Zynq-7k is rather poor (less than 250MB), the event building options in the Zynq-7k are severely limited. Consequently, it is expected, that a (client) routine that extensively uses ethernet and event building has a very limited spare for HPGE pulse shaping analysis tasks. "nc" uses TCP that tries automatically to determine the best possible packet size. In practice the remaining system spare (for memory accesses) can decisively influence the best possible packet size. As the ethernet performance increases with the packet size, the TCP link is seeking to operate with the largest possible packet sizes and therefore is continuously risking losing some packets. Packet losses can affect unfavorable memory access behavior. In systems with little spare, the TCP stream control can lead to a high ethernet packet drop rate. A modest TCP ethernet performance is the consequence. As nc in Z7k devices causes a high CPU consumption, its relatively large fluctuations in the ethernet performance could be caused by unfavorable TCP stream control behavior.

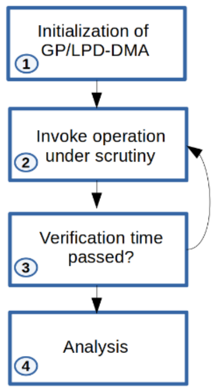
For the device driver of the Zynq-7k PS ethernet controller a patch exists that enables transmission of Jumbo Frames. The drawback of this patch is that it is not capable to enable an efficient reception of Jumbo Frames. The IEEE 802.3 ethernet standard expects that in both directions the same ethernet packet sizes are used. If an inefficient reception with a high ethernet packet drop is acceptable, the patch may be a possible solution. First pre-evaluations at MPIK indicated a significant increase in the ethernet performance, but also in the

CPU consumption. An implementation of the patch in FC2.0 is therefore not intended. As in the pretest not all ethernet controller settings were checked, the performance of the pretest may not be the maximum that is possible. Note that the checked Z-7020 device is of speedgrade -1 and therefore has a CPU base frequency of 667 MHz.

In the setup the ethernet elaboration was focused on PS solutions that do not use the EMIO interface to access the gigabit transceivers of the PL (in Table 30 denoted as PS-PL ethernet). These approaches were not elaborated as they are strongly device specific (MGT pins required). Xilinx also offers an MAC IP that enables an implementation of a more performant ethernet controller into the PL. Using the Xilinx MAC IP in FC2.0 is not intended as it is not freely available.

A.13 Setup for the GP/LPD-IO unit performance evaluation

The GP/LP-DMA framework is in FC2.0 used to set and put the CPU bank registers of the CPU block. Depending on the chosen event building mode, its performance determines the event building bandwidth.


Initialization of GP/LPD-DMA
1
Invoke operation under scrutiny
2
Verification time passed?
3
Analysis
4
Figure 165

The procedure in the elaboration routine is illustrated in Figure 165. In the first step of all routines the DMAs (including the corresponding firmware blocks) must be initialized. In the second step the operation under test must be invoked. All elaborated operations and their specifics are listed in Table 31. All (read/write) functions of the CPU block return/issue immediately the demanded data. A "waiting" function like in the ACP-CMA framework is not required. All transfers are cache coherent. As soon as the demanded measurement time is over, the routine outputs the performance analysis.

Table 31 Investigated GP/LPD-IO unit operations/functions

Operation	**Comment**
read	In cases that the "CPU bank" CDC enabling components are not required, the CPU could directly address the control/status registers (via corresponding load/store assembly commands). No management of the CDC components is required! In these cases, the read and write operations in C do not require the invocation of special functions. In the event building of FC2.0 these direct accesses are NOT available!
write	
GetCpuReg(adr)	The GetCpuReg(adr) function puts the register value of the CPU bank register with the given address adr. It also regulates the CDC bypassing via the CDC components of the CPU bank. The function has a significant overhead in comparison with its virtual read command.
SetCpuReg(adr,value)	The SetCpuReg(adr) function sets the register value of the CPU bank register with the given address adr. Functional counterpart of GetCpuReg(adr)
GetCpuRegs(adr, n, *buffer)	Like GetCpuReg(adr) but as it can retrieve n subsequent registers, its overhead fraction is significantly less. Efficient "GP/LPD-IO unit" trace header building routines base on this function. Its implementation is exclusive to the event header read out via the GP/LPD-IO unit. The

Operation	Comment
	“headerintrace” option of the ACP-CMA does not need it, as it retrieves the event header data via the ACP port.
Best Trace Header Build	The retrieving performance of the event header information of a single specific event strongly depends on the chosen “address condensation” strategy (chapter A.3). This benchmark assumes the implementation of the best possible approach in FC2.0 prototype. It uses slightly abridged FC trace header that comprises the complete event information (that is time consumption decisive) and copies it on its destination in the (user space) ethernet buffer. It does not transfer trace data and is therefore a benchmark for the highest possibly event header building rate via the GP/LPD-IO unit. The procedure is not comparable with the best procedure in FC1.0 that enables much higher bandwidths.
Actual Trace Header Build	In the actual FC2.0 prototype design the retrieving of the event header information cannot use the “Best Trace Header Build” approach. This benchmark uses the current procedure of the GP/LPD-IO unit trace header build in the FC2.0 Zynq client routine.

A.14 Results of the GP/LPD-IO unit performance evaluation

The results of the measurement are summarized in Table 32. To verify the measured "CpuReg" function performances, their calculated performance is included in brakets. In order to enable a compare with FC1.0 the table was extended by the performance of the FC1.0 CPU bank framework. The main difference between the CPU bank block in FC1.0 and FC2.0 is their implemented DMA. In the Zynq devices the DMA bases on ARM AXI4 compliant GP/LPD ports. FC1.0 uses the "Fast Simplex Link" (FSL) bus of the Microblaze as connection to the CPU bank registers. The FSL bus can offer a direct access to the CPU register of the Microblaze and is therefore architectural different from the PS/PL interface bus systems of the Zynq devices.

*Table 32 Bare GP/LPD-IO and effective performance in the CPU block framework. The FSL-DMA in FC1.0 is used for the data transfer between the CPU bank registers and the implemented Microblaze (default CPU base frequency in FC1.0 is 100 MHz). The CPU bank in FC2.0 is a modified CPU bank of FC1.0 whose decisive main distinguishment is the implemented DMA. The function GetCpuReg(adr) returns the value of the bank register with address adr, the GetCpuRegs(adr,n,*buffer) function returns the value of n subsequent bank registers. The given Microblaze functions relate to task which include CDC. The ADC Ring Buffer read-out via the FSL-DMA in FC1.0 is "CDC management" free (CDC is done in ADC Ring Buffer (without software support)) and therefore feasibly identical to the pure read/write performance.*

Operation	**Zynq-7k**	**Zynq-UltraScale+**	**FSL-DMA (FC1.0)**
read	30.3MB/s	20.3MB/s	400MB/s
write	23.8MB/s	126.5MB/s	400MB/s
GetCpuReg(adr)	6.3MB/s *[6.29MB/s]*	8.7MB/s *[13.71MB/s]*	~23MB/s
SetCpuReg(adr,value)	3.4MB/s *[3.43MB/s]*	20.2MB/s *[10.34MB/s]*	~23MB/s
GetCpuRegs(adr,6,*buffer)	14.2MB/s *[14.23MB/s]* (~590k (abridged) FC2.0 event header/s)	15.6MB/s *[17.91MB/s]* (~650k (abridged) FC2.0 event header/s)	~40MB/s
Best Trace Header Build	8.3MB/s	12.8MB/s	Different design!
Act. Trace Header Build	5.9MB/s	11.7MB/s	Different design!

The GP/LPD port is promoted by Xilinx as port with a low latency and bandwidth. The results confirm Xilinx's assumptions. The "CpuReg" functions are solely consisting out of GP/LPD read/write accesses to the CPU bank registers (first/second row in Table 32). In Table 32 the heron calculated "CpuReg" function performances are added in brackets as additional entry. In case of the Zynq-7k devices the calculated bandwidth of the "CpuReg" functions agrees with the measured performance. In the Zynq-UltraScale+ devices the calculated and measured performance differ without a clear trend. The ZU+ devices have an interconnect design with significant number of data rooting and quality of service options. In the ZU+ devices the derivation between calculation and measurement results is a hint that the data flow through the interconnects depends on the command history. In contrast, the "CpuReg" functions assume a constant bandwidth. To ensure the proper CDC functionality in ZU+ devices the functions were extended by spare read/write commands. A FC2.0 framework that is solely focused on Zynq-7k devices therefore could further improve the bandwidth by removing the unnecessary spare read/write commands. In FC2.0 the most efficient way to retrieve the event information values (buffered in the corresponding event information ring buffers) via the GP/LPD ports is by the invocation of a single SetCpuReg() function (to select the information of the desired event number) and hereon subsequent invocation of the GetCpuRegs() function (to fetch the corresponding 6 information words). The performance of such a GetCpuRegs(adr,6,*buffer) data read is approximately similar for both devices. This efficient access is only possible, if a compatible address condensation strategy is used in the corresponding CPU bank registers.

The FSL-CPU bank interaction in FC1.0 utilizes the advantage the direct CPU register access of the FSL bus. The time consumption of the “CpuReg” functions in FC1.0 is given by the function invocation (and the corresponding rebound to the PC departure address), the CDC management and the data access that comprises further $8 \cdot n + 2$ clock cycles (n represents the number of retrieved event information words). Although the Microblaze in FC1.0 is only driven by a 100 MHz clock, its 1:1 FSL-CPU register clock interaction boosts its IO performance. A compare of the IO operation/functions (in Table 32) between FC1.0 and FC2.0 therefore results in a deplorable Zynq GP/LPD performance despite its CPU core and bus frequencies are much higher. The in FC1.0 implemented Microblaze (without an FPU unit) is solely optimized for IO tasks and would be beaten dramatically in a speed2 arithmetic performance tests[27].

In FC1.0 the dense coverage of the Spartan-6 device and the fabric performance limit the feasible Microblaze core frequency to 100 MHz. New device generations should enable an even more performant FSL bus. Xilinx promotes their Virtex UltraScale+ devices with a maximum Microblaze core frequency of 682 MHz [26 S. 382]. In case of the Zynq-UltraScale+ device ZU9EG with the highest speed grade and a typical Microblaze design (not frequency optimized), Xilinx states a maximum CPU frequency of 428 MHz [26 S. 388]. A transfer of the old FC1.0 design to the newer devices therefore should enable an increase of the Microblaze core frequency to 200 MHz. The performance of such a design would improve accordingly.

[27] The available memory in FC1.0 is also totally insufficient for a Microblaze IO performance elaboration with the speed2 routine.

A.15 Results of event building performance evaluation

The discussion of the results is separated in two subchapters. In subchapter A.15.1 the performance results with Z7k devices are discussed. Subchapter A.15.2 finalizes the performance elaboration of ZU+ devices.

A.15.1 Results with the Znyq-7k

To catch a glimpse of the central ideas it is important to understand that the client routine only can provide a sufficient performance, in cases the FPGA internal RAM is efficiently used. If the client routine gets a read-out request that only considers a single, specific channel of the board, the client routine cannot use the majority of the FPGA internal RAM (this RAM is assigned to the other ADC channels). Hence in such a “Single” channel mode the poorest payload bandwidth performance of all modes is expected. In the ideal case, the received read out server request demands the same event number of all available channels. Such “(Global) Multi Channel” modes have an increased trace buffer capacity and can consequently offer a higher bandwidth. Furthermore, it also increases the possible event building efficiency. In FC a “Global Multi Channel” mode that guarantees the availability of the same trace number for all available channels is possible. The usefulness of the implemented trigger styles (“Single Channel”, “Independent Multi Channel” and “Global Multi Channel” (the latter with Major trigger options)) strongly depends on the desired application. In the Trace-Read-Out the “Single Channel” mode[28] and the “Global Multi Channel” mode represent setups whose Trace-Read-Out can fairly (“Single Channel”)/perfectly (“Global Multi Channel”) operate the available ADC Ring Buffers in the FC2.0 prototype design. It is expected that the performance of all Trace-Read-Out styles is between these “extremum” examples. The performance of the “Global Multi Channel” mode was already presented in chapter 6.5.3 and consequently is omitted in a large fraction of this chapter.

Figure 166 and Figure 167 show the FC2.0 prototype performance in “Single Channel” mode. The setup is still identical to “event building performance” setup of chapter 6.5.3. The difference is the investigated trigger mode. Figure 166 illustrates the payload bandwidth performance, Figure 167 the corresponding event rate. The meaning of “headerintrace” and “headeracp” was explained in chapter A.4.3. The “headerintrace” approach surpasses “headeracp” as soon as the latter achieves its event buffer capacity limit. In Single Channel mode the “headeracp” read-out method can buffer until 128 events (in Multi Channel mode it can buffer until 512 events). In the shown setup this means that the “headerintrace” mode will outperform the “headeracp” mode in settings that demand a read out of traces smaller than 1024 samples.

[28] Remember that even in the „Single Channel Mode” it is assumed that several channels are implemented in firmware. But only one of these channels is controlled by a server request message, the remaining ones are disabled (“single channel is actively operated”). Consequently, only the ADC Ring Buffer and Event Information Ring buffers of the active channel can be used to buffer events.

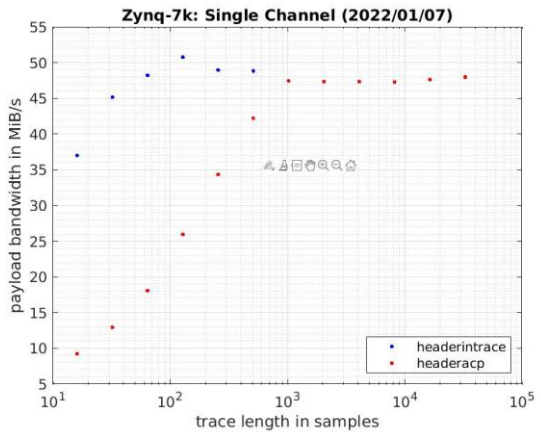


Figure 166 Zynq-7k: Payload bandwidth in Single Channel Mode

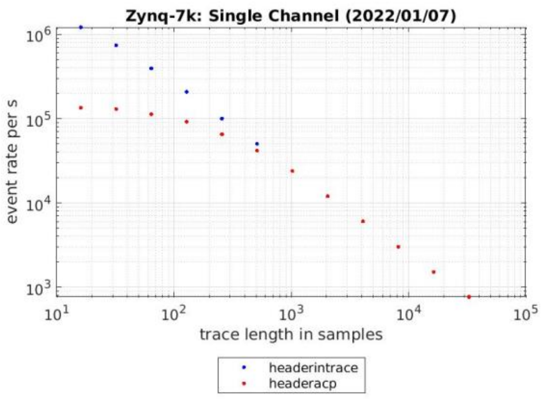


Figure 167Zynq-7k: Event rate in Single Channel Mode

The Single Channel mode has a maximum payload bandwidth of 50 MB/s. The low bandwidth is not surprising as preliminary studies of the Z7k ethernet performance even implied that a maximum performance of only 40 MB/s is supposable (chapter A.12). The high event rate in "headerintrac" mode is interesting for experiments that must bear extreme event rate requirements (like the suggested "Natsirt" in chapter 6.8). As the pulse detection firm block also has an energy signal that can be forwarded to the ADC Ring buffers as trace data (via "tsel" selection), the FC2.0 prototype DAQ can generate a spectrum even if the transmitted traces violate the introduced length requirements (the shaping of the energy signal is not necessary anymore, it is sufficient to transmite a tiny ("energy") sample number around the energy signal peak). Another option that currently is not implemented (and would require a more developed "Trigger Acceptance Control" Block with eligible pile up rejection skills), is the forwarding of the pulse determined energy as trace header content. In such an approach the ethernet transfer of the trace data would be not necessary and there this approach could handle the highest event rates. It should be emphasized that for such applications the FC2.0 DAQ is currently not intended and therefore the development of such a firmware is unlikely.

A.15.2 Results with the Zynq-UltraScale+

Figure 168 presents the performance trends in Single Channel mode for the FC2.0 prototype design with the current ACP-CMA. The performance trends of a FC2.0 prototype with the (device driver and firmware) enhanced 128bit ACP-CMA design of Figure 43 is illustrated in Figure 168. In chapter 6.5.2 it was shown that the enhanced ACP-CMA has a bandwidth of 2.41 GB/s (in place of 0.75 GB/s).

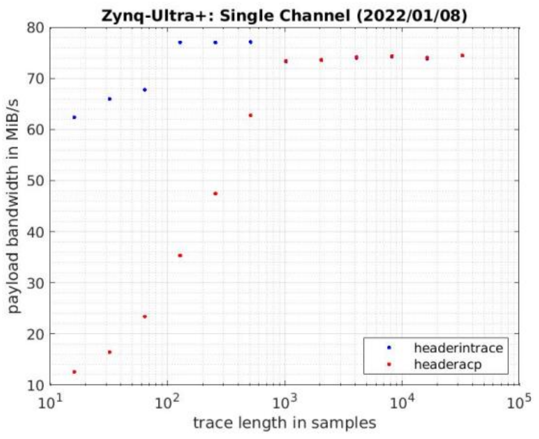


Figure 168 Zynq-Ultra+: Payload bandwidth in Single Channel Mode. In this setup a default ACP-CMA with an operation frequency of 200 MHz was used. The bare performance of such an ACP-CMA can be seen in Figure 39.

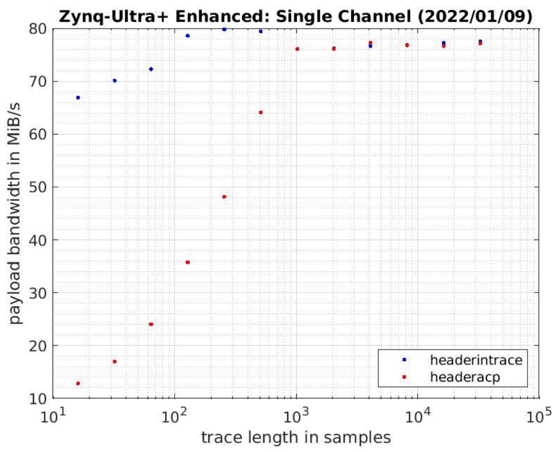


Figure 169"Enhanced" Zynq-Ultra+: Payload bandwidth in Single Channel Mode. In this setup the optimized ACP-CMA (128bit data width/no interconnect & new device driver) (with an operation frequency of 200 MHz) was used. The bare performance of such an enhanced ACP-CMA can be seen in Figure 43.

The enhanced designs performance is just slightly more performant as default FC2.0 prototype design. This can be explained by the high bandwidth disparity between ethernet and ACP-CMA. Bulk of time, the client routine is preoccupied with ethernet transmission. As the current ACP-CMA already has a much higher bandwidth than the ethernet, an accelerated ACP-CMA just slightly can increase the ethernet transmission fraction of the job. Consequently, a slight increase of the total performance is expected. The shown read-out performances (of this chapter and chapter 6.5.3) are not the total maximum performances of a possible setup. The main reasons are imperfections in the server-read-out routine and in the read-out protocol. For instance, the server-read-out routine does not determine the best possible ethernet packet size and consequently losses a significant amount of bandwidth performance. Other possible improvements of the read-out routines and read-out protocol additionally exist. Note that the current setup was intended for a single board read-out and in such a design the assumed flaws have a fatal effect that prevents the proper operation of the setup. Omitting the implementation of the assumed "bottleneck" points enabled a design that is debug able with reasonable efforts. This is especially important in the firmware development epoch of this thesis.

The current read-out server routine also has settings that empower a glimpse to the real maximum performance of FC2.0 on Zynq devices. Figure 170 illustrates the possible payload bandwidth in dependence on the used ethernet packet size (in x-axis label called “efbp packet size”). In the corresponding scrutiny the “headeracp” approach with the same settings like in Figure 168 was used. In case of a selected trace length of 4096 samples (red trend), a simple modification of the ethernet packet size would be sufficient to cross the desired minimum payload bandwidth. In all previously presented ZU+ payload bandwidth results an ethernet packet size of 8100 was chosen by the read-out server routine. Figure 170 implies that Jumbo Frame sizes are not a proper selection. This conclusion is affected by the used setup (“network interface of a poor laptop”) and imperfections in the read-out procedure and hence is not a general truth. In DAQ arrays that based on ethernet read-out, Jumbo Frames are an essential part of an applicable solution. A big issue in the current server read-out routine are the time perils due to ethernet packet loss. In the “single” board measurement setup a packet loss always results in a draconic bandwidth loss. In settings that enable the maximum read-out speed of the server routine, the likelihood of a packet loss is high. Mostly the packet loss is the consequence of insufficient reception buffer space. An ideal server read-out routine reacts by throttling of the transmitted read-out commands. The current FC2.0 prototype server read-out routine misses this capability and consequently it creates packet losses in cases where it could show supreme performance. Figure 171 and Figure 172 depict performance results that were obtained read-out server settings that enable the highest read-out performance of the used laptop (as server). The Zynq- device settings in these measurements still represent the same realistic experimental settings as before, but the server routine does not permanently store the traces (“payload”) anymore.

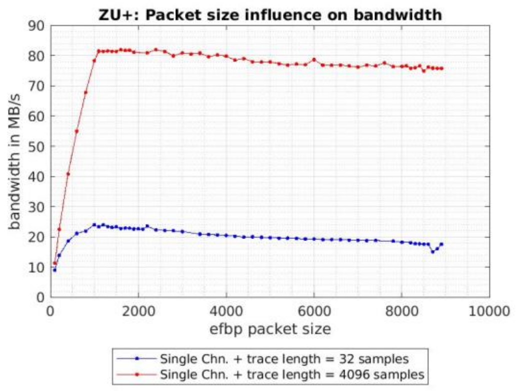


Figure 170 ZU+: Packet size influence on bandwidth

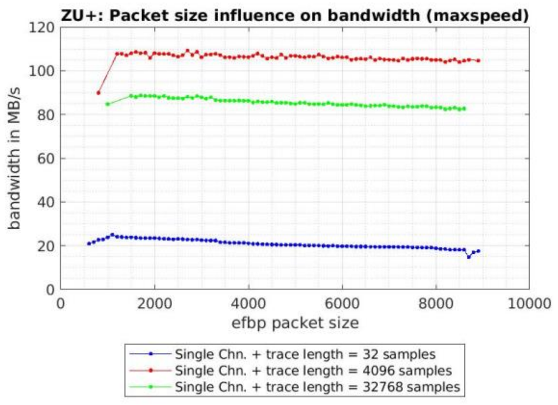


Figure 171Maxspeed measurements with the ZU+ in "Single Channel" mode. Uses still the same firmware as in chapter 6.5.3, but now does not permanently store the (ethernet transferred) event data.

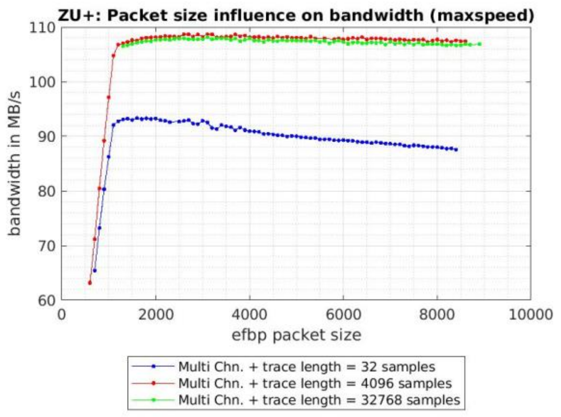


Figure 172 Maxspeed measurements with the ZU+ in "MultiChannel" mode. Uses still the same firmware as in chapter 6.5.3, but now does not permanently store the (ethernet transferred) event data.

A.16 Setup of the PB-5 INL determination approach

In the thesis the focus was on INL determination approaches that are not dependent on the precision of their implemented pulser(s). The decision to focus on such INL determination setups was mainly a consequence of the collected experiences with the first developed “PB-5” setup[29]. The “PB-5” setup is a deviation of the GERDA method where the DMM is not implemented. In the setup a potential nonlinear behavior of the PB-5 precision pulser (Berkeley Nucleonics Corporation, San Rafael-CA) is not correctable and hence the approach belongs to the “classical” INL determination setups. The “PB-5” pulser is used in HPGe-spectroscopy as precision pulser and has a typical precision of roughly 16 bit on its selectable signal output voltage range (amongst other output voltage ranges: 0-10V,0-1V,0-0.1V,0-0.01V (assumes an high-impedance input)) [33]. The pulse amplitude is finely selectable in step sizes in the range of the pulser precision and offers several “exponential” pulse shapes. The used “PB-5” pulser was a loan of the GERDA collaboration and is not nonstop available at MPIK. The listed price of a PB-5 pulser is significant and therefore the suggested PB-5 setup is not a general solution that can be applied in any workshop. A schematic of the “PB-5” setup is illustrated in Figure 173.

In the “Single Region PB-5” procedure the baseline level is during the complete measurement unchanged. To cover all (accessible) ADC codes of the ADC (“full range of ADC”), the measurement routine only increases the pulse amplitude accordingly.

In the GERDA setup the mean ADC code value of an issued DC level is used to determine the ADC nonlinearity. The GERDA setup therefore must shift the DC level over the complete signal range of the ADC. Remember that the GERDA setup must accurately know the issued DC level. Determining the accurate DC value is the task of the implemented DMM. The INL determination (in the GERDA approach) uses the measured “mean ADC code”/”accurate DC voltage” as input. In the PB5 setup the mean ADC code is replaced by the determined amplitude height (given as “max value” of the shaped “energy” signal (PB-5 setup therefore requires a (modified) signal chain that is similar to Figure 70)). The PB-5 setup has no DMM and therefore the accurate DC level is replaced by the (remotely) selected pulse amplitude heights. Otherwise, the INL calculation of the “(Single region) PB5” procedure is completely identical to the GERDA procedure. The PB5 setup has the advantage that the time consumptive DMM measurement is missing, but due to the lack of an additional “noise circuit filter” (like the “adder circuit” in the GERDA

[29] The „PB-5“ is the (alleged) state of the art approach to measure the suggested exponential pulse approach. In the old days (at MPIK) the briefly mentioned “mercury relay & passive filter shaping” approach was used. In this approach the possible resolution depends on the quality of the used capacitor. Air capacitors (allegedly) fulfill the permittivity requirements. Unfortunately, the low switching frequency in such setups can be an issue in nowadays ADCs that require several ten thousands of measurements. In [38] it was tried to adapt the approach – instead of using a mercury relay to generate the voltage pulse, a “default” signal generator is used. Ultimately the adapted approach was limited by the used capacitors (and not be the nonlinearity properties of the signal generator) – but the final procedure required a lot of additional considerations and still could not reliably used to determine 16bit ADCs.

setup) its ADC input signal (complete signal bandwidth of the FADC card input!!!) also has a higher noise charge. Effectively the chosen measurement time - that is also in the PB5 approach strongly dependent on the targeted effective noise level (<<0.125LSB) - of the PB5 approach is just “slightly” less than the measurement of the GERDA procedure. The PB5 approach can influence the effective noise level by the number of accepted pulses (per selected pulse amplitude height) and by the chosen shaping parameters.

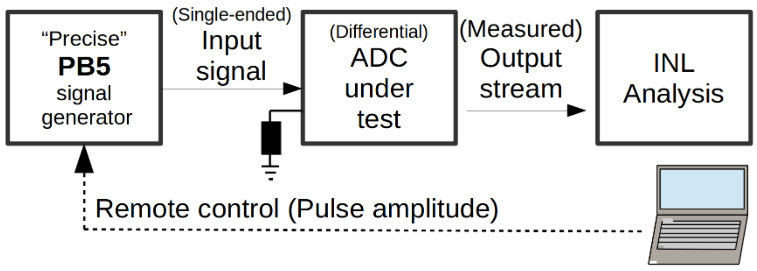


Figure 173 Single Region PB-5 setup for INL determination. The FADC (evaluation) board “DC2266A” was in the PB-5 setup also the main device under investigation. As it has a differential input and the PB-5 has a non-differential output signal, the remaining input was terminated by a 50Ω termination.

The “Single PB-5” setup has peculiarities that will be discussed in the result chapter A.17. One of the determined issues was that the ADCs input full range (1.2V) does not match to the available output ranges of the PB5 and hence ultimately a (pulser) resolution loss is the consequence. This was critical as the (assumed) pulsers resolution is in the magnitude of the supposed ADC nonlinearities. Hence it was decided to process the complete full range of the ADC in “several sub” measurement campaigns (Figure 174). In this procedure the complete full range of the ADC is for instance divided into 8 subregions (which may partly overlap each other). The start position of the subregions is controlled by the issued DC voltage level of the SDG6022X. Consequently, the total measurement must change the issued DC level in 8 times. All subregions combined comprise the complete full range of the ADC again. Every region is still scanned by increasing the amplitude height according – during the scanning (in the sub region measurement) the issued DC offset level must be unchanged.

To understand the resolution advantage of the suggested procedure, the attenuation circuit must be considered. In this “several regions” setup the selected full range of the PB-5 output signal can be chosen much smaller than in the original procedure. This reduces the absolute nonlinearity error of the PB-5 output signal. An improvement factor of 8 could be achieved by the suggested example split into 8 regions. The PB-5 effective signal error on the final result therefore would be in the rough magnitude of a LSB of 19 bit ADC and consequently much smaller than a LSB of the 16bit ADC under investigation. In the proceeded real measurements, the full range of the ADC was split in ten subregions which partly overlap each other (can be seen in the result chapter Figure 178). The expected resolution improvement is only slightly affected by this measure.

The challenge in the “several range PB-5” procedure is the correct combining of the regions in the final INL calculation. This procedure faces an issue that is already known in a similar characteristic by the “FC1.0 INL” procedure. Using the biased baseline level as calculation reference will unavoidably (dramatically) degenerate the from it calculate INL trend.

In the beginning determining the correct combination procedure caused issues. Again, simulations of the procedure with a perfect pulser as signal source were required to determine the best procedure.

The concatenation of regions is done on basis of their DNL trend. Like in the “Single Region PB-5” approach the measured data of a region relates to a INL trend. The determined INL trend of the region is then transferred into the DNL trend of the region. In the first simulation approaches the results of all DNL region trends subsequently were combined in a total DNL trend without any ADC nonlinearity bias compensation. In cases where regions had an overlap the corresponding mean DNL value (of every related ADC code) was used in the total DNL trend. The determined “total” INL trends of this procedure still had a significant deviation from the given INL trends. In the result discussion in chapter A.17 this procedure therefore will be called “poor simulation approach”.

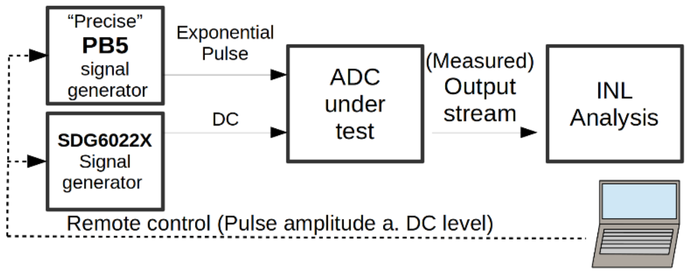


Figure 174 Several Regions PB-5 setup for INL determination . The issued DC level is constant during the measurement of a region. Every region has its specific DC level. During the measurement of a region only the pulse amplitude is changed.

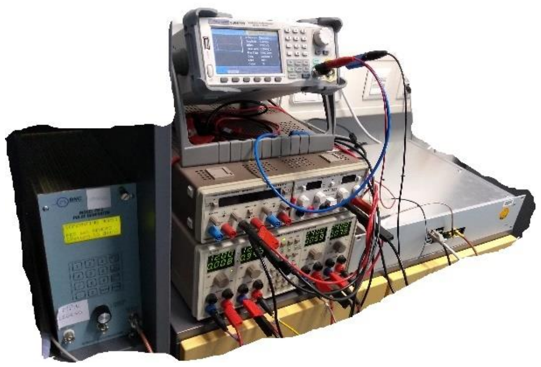


Figure 175 Several Regions PB-5 setup with FC2.0 prototype. The PB- 5 is visible on the left side of the figure. It needs an external (NIC compatible) power supply (at the bottom in the middle) to be operated. The SDG022X can be seen on the top of the power supplies.

In the previous approach it is not considered that every single region INL trend is specifically biased by the ADC nonlinearities. This means that the INL trend is defined by gain and offset parameters that are the result of the regions nonlinearity trend. The “best simulation” procedure tries to find the best full range gain value for all regions (the pulse height of the PB-5 output signal should be independent of the region). To do so, it is permitted to correct every biased baseline level (“offset”) by a region-specific value. A successful determination of the global gain and the region offset correction parameter was done, when in all regions a selected PB-5 output signal pulse amplitude of X results in a minimum fluctuation between the determined region pulse heights. The adaption of this (optimization) procedure uses the MATLAB internal function *fminsearch*. The so optimized INL trends then subsequently pass the same procedure like in the "poor simulation approach".

A.17 Results of the PB5 INL determination approach

The "Single Region" PB-5 setup proved more challenging than initially expected. Due to its high noise level a direct extraction of the pulse amplitude ("energy value") from the unshaped raw ADC data stream was not feasible. Hence it was decided to implement an additional (upstream) moving average filter. Like in the "FC1.0" procedure moving average filtering smooths the DNL trend. The determined INL trends of the first measurements strongly varied and their magnitude was also not credible. Thus, it was decided to investigate the influence of the PB-5 pulsers shape settings on the determined INL trend. A switch of the PB-5 pulsers pure "exponential (tail) pulse" shape into a "flat top pulse" shape resulted in INL trends that better agreed with the expected INL trend. A precise analysis of the available data concluded in the assumption that the determined INL trend also depends on the flat top region that is used to extract (via "Max" Search) the energy value ("pulse amplitude") of the pulse signal. Instance INL trends are shown in Figure 176 and Figure 177. In both figures the same data sets are used, they only differ in the used flat top region of the pulse. Especially the INL trend of ADC codes below 45000 are prone to the select flat top region.

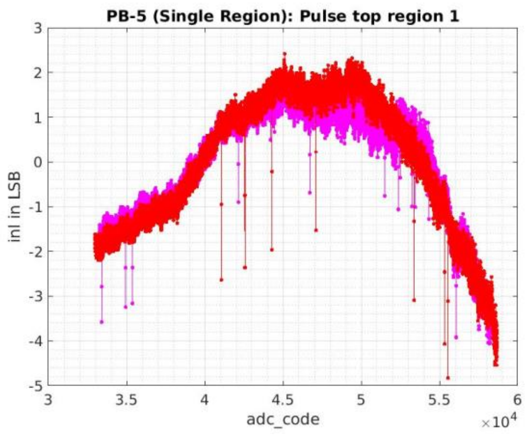


Figure 176 Determined Single Region PB-5 INL trend for flat top region 1. The red and the magenta trend line were measured in two identical (subsequent) measurements. The trend outliers are likely caused by a not adapted change of the (routine controlled) pulse amplitude.

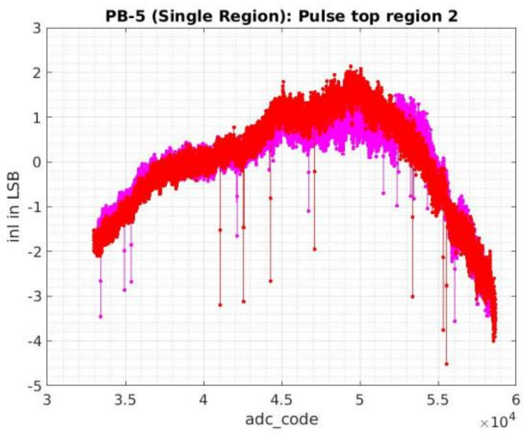


Figure 177 Determined Single Region PB-5 INL trend for flat top region 2. The used trace data set is identical to the data set of Figure 176. The only difference is the flat top region that is used for the maximum extraction.

The region dependence can be construed as pulser misbehavior or as dynamic nonlinearity effect. To mitigate a potential pulser misbehavior the "Several Regions PB-5" procedure was developed. The (unoptimized) INL trends of the subregions of a complete instance measurement are shown in Figure 178.

It is clearly to see that the 10 selected regions have a high overlap. This was done as it was initially supposed that a high overlap is profitable for a well behaving region concatenation. Figure 180 illustrates the corresponding (real measurement) results of the “poor” procedure. Its trend has the expected double peak behavior and is hence quite appealing. Note that in the illustrated INL trend the final gain correction (by a straight line fit) was not done and therefore the presented trend is not complete yet. Figure 179 illustrates the resulting INL trend with the “best” procedure (again without the final gain correction). The trend is derived (again) from the measured region data sets of Figure 178, but its global trend clearly differs. Its INL magnitude agrees much better to the results of the Majorana and the “FC1.0 INL via DNL” procedure, but it also lacks the expected double peak behavior. Whether Figure 179 really shows the real INL trend is therefore still unclear. In the simulations of the “best” procedure, the simulated Pulser did not have any Nonlinearity. The real PB-5 pulser has nonlinearities in its output signal. This could be a potential burden for the “best” procedure.

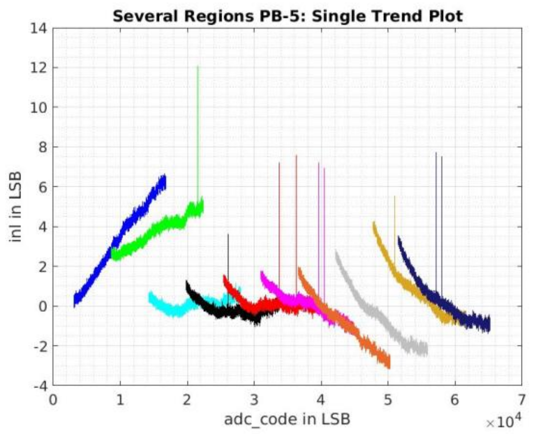


Figure 178 Several regions PB-5 procedure: INL trends of the subregions. The illustrated INL trends are without any gain and offset correction (done by linear fit subtraction).

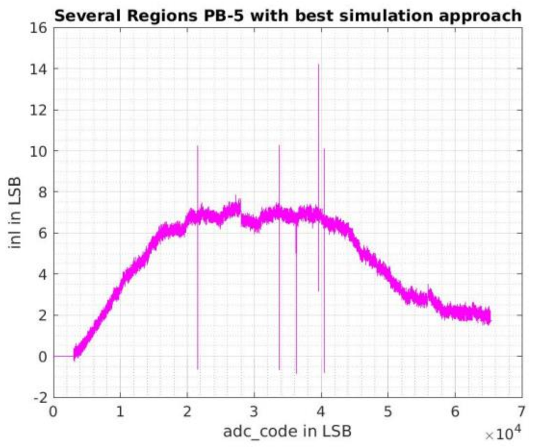


Figure 179 Several regions PB-5 procedure done with the region concatenation approach that provided the best simulation results . In this procedure it is tried to compensate the ADC nonlinearity bias on the baseline level.

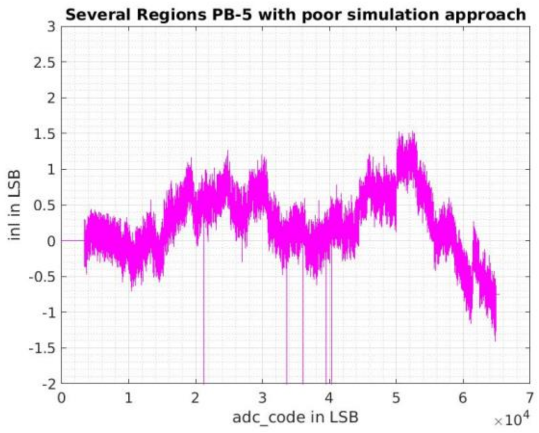


Figure 180 Several regions PB-5 procedure done with the first DNL procedure try to concatenate the subregions in a complete trend. As the title implies, simulations showed that this procedure returns results that are poorer than the results of the procedure of Figure 179.

A.18 Signal processing chain in HPGe-detector experiments

Before in the middle of the 90s the first feasible experimental setups with FPGAs/DSPs cropped up, the experimental possibilities were limited by the available pulse shaping and coincidence logic circuits. Operational amplifiers were besides resistors and capacitors the basis of analog shapers. Consequently, pulse shaping was suffering of the corresponding component frailties. For instance, especially the flaws of real operational amplifiers impeded the common implementation of the sought-after trapezoidal shapers. Further challenging points in analog coincidence experiments were the unit synchronization and the limited delay time of analog setups. Furthermore, expensive fast memory and a limited analog-to-digital conversion bandwidth yielded to experimental setups were just the finally required values were digitized. Though many analysis capabilities already in analog setups existed, the connected efforts (and dimensions) frequently rejected their implementation. In today's experiments the event rejection (cut) possibilities are significant in the final analysis. Modern ADCs analog-to-digital conversion limits can be negligible, and ASICs/FPGAs enable a signal buffering for reasonable costs. So, the requisites for modern coincidence analysis is given. The permanent storage of the accepted trace and its precise time information enabled an ensuing analysis which can consider almost all kind of physical correlations.

In today's experiments commonly the signal processing is distributed in several tiers. The tiers build on each other, hence with the tier level the adjustment on the particular experimental conditions rises. In the thesis the focus is on the low levels which are mostly independent of the particular specifications and therefore in all experiments have a similar setup. A general example for a low-level signal processing chain is depicted in Figure 70 (page 104 in chapter 7.6). Note that the figure does not consider the upstream analog tier that mostly includes a charge sensitive amplifier (CSA).

The low-level signal chains (Figure 70) consists out of shaper(s), a high pass deconvolution unit (HPD) (sometimes also denoted as pole-zero cancellation unit) and an optional baseline restoration unit. The behavior of the GRETINA signal chain and its interference with ADC nonlinearities are one of the main topics in [3]. The shaper is amongst others responsible for SNR improvement. The most important shaper specifications and their interplay with the remaining signal processing units are discussed in subchapter A.18.1. The CSA deconvolution unit is motivated by the inputs need of the shapers, but also can induce significant baseline drawbacks caused by its DC sensitive misbehavior. Supplying the deconvolution with an input signal whose DC shares are sufficiently low, can be a serious challenge. If a shaper/filter system with sufficient DC/low frequency mitigation is implemented in the signal chain, the deconvolution error caused by DC effects is negligible. As closer the (low pass) deconvolution filter output shape approximates an impulse, as better the deconvolution filter was capable to mitigate negative effects of the baseline shifts. Baseline restoration can be implemented as approach to mitigate the remaining degeneration effects which could not be removed by the deconvolution filter. A cause of the remaining (low frequency) interferences are for example microphonics in amplifiers or leakage current fluctuations.

The (low level) signal processing in the server side uses the same shaping filters like the pulse detection signal processing chain in firmware. Despite to that, they differ significantly. The signal processing in firmware is done with a continuous, volatile (integer) data stream. Recursive filters are a reasonable choice. The sample flow and sample buffering must be organized in a way that during processing time the required (recursive) filter algorithm samples are available. The volatility of data stream can be a challenge in the algorithm design of the filters as the data buffer capability in firmware is limited and the sample position in the buffers is dependent on time. The signal processing of the server analysis routine can access the whole trace samples immediately. Convolution filters are an option. As soon the corresponding trace is accepted, the samples position in the trace is fixed and does not vary in time anymore. Server routines have the drawback that their possible shaping and analysis approaches are constrained by the available trace and trace header information. Besides the data availability server and firmware signal processing routines distinguish in the data type. In server routines floating point data types are default. In firmware integer operations are preferred as their implementation in firmware is frequently less hardware consumptive. Due to the differences in data type, the same filter algorithms can have a different resolution and behavior.

Common analog shaping filter systems in HPGe-spectroscopy can be approximated as a linear time-invariant system (LTI system). LTI systems have following properties:

- The (time domain) output signal of LTI systems relates to a convolution of the input signal and the systems impulse response (Equation (40))
- The (frequency domain) output signal of LTI systems relates to a multiplication of the input signal and the systems impulse response (Equation (41))
- A systems (LTI) transfer function can also be used to calculate the output signal of a given input signal (Equation (42)). The derivation of so called recursive digital filters commonly uses the corresponding LTI system as starting position. The system transfer function can also be the basis for elaborated analysis – for instance with the Nyquist stability criterion.
- The tools of the “z-transformation” can transfer continuous (“analog”) LTI systems into time discrete (“digital”) LTI systems.
- Equation (43) is the “digital” counterpart of Equation (40).

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau) \cdot h(t-\tau) d\tau \tag{40}$$

$$y(f) = x(f) \cdot h(f) \tag{41}$$

$$G_{LTI} = \frac{\mathcal{L}\{y(t)\}}{\mathcal{L}\{x(t)\}} = \frac{\mathcal{F}\{y(t)\}}{\mathcal{F}\{x(t)\}} \tag{42}$$

$$y[i] = x[i] * h[i] = \sum x[j] \cdot h[i-j] \tag{43}$$

As convolution filtering is hard to implement in FPGAs (and other DSP devices), these systems commonly use filters which based on recursive filters formula:

$$y[i] = \sum_{j=0}^{i} a_j \cdot x[i-j] + \sum_{k=0}^{n} b_k \cdot y[i-k] \tag{44}$$

A.18.1 Shaping

The development of the shaping filters in HPGe-spectroscopy can be split in two epochs. In the first epoch shaping was a solely task of analog electronics as eligible digital electronics was not available. In these times the shaping filters setup was mostly a combination of CR (low-pass filter) and RC (high-pass filter) segments. Most critical in analog shapers is the amplifier performance. Shaper issues were commonly amplifier issues. The subsequent effects of the amplifiers issues strongly depended on the selected shaper type. Besides their implementation, the shapers obviously are dissimilar in their transfer function and therefore have different resolution properties. To get a better understanding of the shapers maximum resolution, Radeka et alia developed analytical methods for their simplified noise elaboration. The shapers and their possible noise mitigation performance is listed in Table 33.

Table 33 Various pulse shapes and their SNR relative to infinite cusp. If not otherwise noted out of [44]. The textbook of Nicholson is not available currently (2021/08/17). The (summarized) results are also available in [2 S. 652]. The TZ results are (likely) not implemented in the original textbook of Nicholson and therefore some of its significant basis parameter (especially integration time) could distinguish – as in the case of a missing flat top (a=0 ("Triangular")) the equation returns a TZ SNR$_{norm}$ value which is in correspondence with the result of Nicholson, the TZ result of Radeka was regardlessly implemented.

Shaper	**"SNR"$_{norm}$**
Triangular	0.93
DL-RC	0.911
CR-(RC)^4 ("Gaussian")	0.858
Trapezoid ("TZ") **(depends on flat top proportion a – given SNR$_{norm}$ value is for a flat top which has a proportion of 0.4 (a=0.4) to its falling/rising edge ("integration time") – equation is out of [45])**	$\frac{1}{\sqrt[4]{2 \cdot \left(\frac{2}{3} + a\right)}}$ 0.83/0.93 (a=0.4/a=0.0)
CR-RC	0.736

Tough TZ shapers have not the best possible SNR value and their (analog) implemented electric circuits have their flaws, the ballistic deficit issue (a fraction of the liberated (detector) charge achieves delayed at the CSA and therefore the pulse height is not a reliable measure for the gamma ray energy anymore) sanctifies the implementation of TZ shapers. As rule of thumb, a rising HPGe crystal weight effects more charge collection time. In the analysis the degeneration caused by this effect partly can be mitigated with flat-top shapers like the TZ or alternatively with a charge drift compensation method.

Digital recursive filters can be a challenging task due to undesired behavior caused by the quantization and overflow of discrete values. Moving average filter are a popular exception. A pseudo-gaussian shaper is the result of three convoluted moving average shaper (with equal window width) and therefore accessible via a fast (multiple) recursive filter implementation [46]. A TZ shaper is the result of a convoluted moving average shaper and a truncated ramp [47] and therefore the exclusive usage of moving average filters is in this case not possible. In contrast to that a rounded top (pseudo) TZ shaper is constructable via exclusive usage of moving average shapers (3 convolutions of moving average shaper whereas two different window widths must be used).

An activated filter, which was not implemented in the signal line before its activation, needs time to bias its components into the corresponding baseline level "equilibrium". Its output values before it achieves the equilibrium state are in transient oscillation state and therefore corrupted. The Gaussian and the TZ shaper have in common that the duration of this state takes approximately as many samples as their selected shaping time (integration time). This means that an (approximately) error free determination of the pulse height is possible, if the signal trace (with the accepted pulse) has at least $n_{shape} = \frac{t_{shape}}{T_S}$ recorded samples before its peak summit. Furthermore, Gaussian and TZ shaper have in common, that an input signal consisting out of an (perfect Dirac) impulse has an output signal which achieves its maximum value after the selected filter shaping time was passed (this is a (side) assumption in the derivations of the corresponding recursive equations via convolution). As the maximum value offers the best SNR proportion, the read-out trace conclusively needs at least a duration of two shaping times (one for the equilibrium state and one for the amplitude determination). As in real runs the charge arrives at the shapers input with some temporal distribution, the trace length should be at least extended by the flat top duration (of an appropriate selected TZ filter setting). This "minimum trace length" would be sufficient for analysis routines which can use the complete event header of FC-Germanium. The FC-Germanium approach has in the trace header the corresponding baseline level included. As it is not possible to determine a reliable baseline level value out of the reasoned "minimum trace length" (of 2 shaping times and 1 flat top time), determined spectrums which were determined only on basis of this traces cannot offer the best resolution.

A.18.2 Pole-zero-compensation (PZ) and its DC issue

The output of the CSA is an exponential pulse. The task of the PZ filter is to transfer the output pulse of the CSA into the desired input shape of the shaper. In this thesis this was an impulse, but that needs not to be the case. In other publications the assumed input of the shapers is a Heaviside ("step") function. In order to transform a CSA output signal into an impulse, a low pass deconvolution (LPD) is required. A step (out of the CSA output signal) requires a high pass deconvolution (HPD). Both deconvolutions have in common that their result depends on the baseline level – a variation in the baseline level negatively affects their performance. In case of the HPD a baseline level different from zero affects a continuously ascent whose slope dependents on the baseline level. A detailed discussion of this effect is done for instance in [48]. The HPD baseline misbehavior is more eye appealing than the LPD baseline flaws. As shapers that require a HPD filter include differential shaping (or have a subsequent differential shaper (DL)), such systems effectively behave like systems which are build up with an LPD. The LPD/HPD misbehavior are therefore effectively the same. To eliminate baseline fluctuation effects, which distorts the systems resolution (as the determined pulse height will fluctuate), it is in both deconvolution styles necessary to generate an input signal which is baseline free. In a system with a stable baseline the negative effects of baseline restoration are existent, but they are not the main contributor (the error it causes is mostly around the same value) and therefore it can be acceptable not to consider them in analysis. In systems with a fluctuating baseline and high resolution demands the baseline free input in the PZ filters is a hard requirement in order to get an acceptable resolution.

Server-side low level signal analysis in FlashCam uses the "baseline" free PZ filtering approach. In FC-Germanium the baseline level value is part of the trace header and therefore the Server Routine just has to subtract this value from the trace in order to gain a baseline free input signal for the PZ shaper input. The FlashCam-Germanium signal processing determines the baseline trace header value via an implemented baseline follower.

A.18.3 Baseline reconstruction

In the previous chapter A.18.2 the importance of the baseline in high resolution analysis was alluded. It was also listed that FC Germanium uses a baseline follower to determine the exact baseline value. The most famous digital baseline compensation approaches are likely the (old-fashioned) trigger-controlled bypassing of a high pass circuit (a digital implementation of this approach is done in GRETINA [3]) or subtraction of the assumed baseline value like in [49]. The latter approach therefore has some agreement with FlashCam-Germanium, the former is principial different and close to the suggestions of [2]/ [1]. These approaches have in common that their baseline value is effectively corrected/determined when the signal is not charged with an (detected) event. For instance, in situation with a high event rate it is possible that the actualization process of the "determined" baseline process is not fast enough anymore and the setup thus uses superseded values which can make the situation even worse than without baseline compensation. The usefulness of this (time-variant) baseline restoration approaches therefore strongly depends on the application – a general recommendation is not fruitful. In contrast to this designs FlashCam-Germanium uses a baseline follower that continuously determines the baseline level independent of the trigger situation. The design of the baseline follower is still (2022/06/28) under disclosure even though it is already used in LEGEND-200.

In chapter A.18.2 it was already noted, that in the FlashCam-Germanium Low-Level Server analysis routine every trace has its corresponding baseline level in the trace header. Determining the baseline level on basis of a given trace is not as obvious as it seems, as the trace is charged with noise and possibly with tail runouts of previous events (who might be unknown to the analysis routine). The suggested approach of [34] avoids the issues of undetected tail runouts. As in this approach the noise is still an issue, is worth to consider the consequences in more detail.

A.19 Example DAQ designs on FC2.0 prototype basis

The following DAQs are example DAQs that never were operated neither synthesized. They are examples whose intention is to discuss specific advantages and disadvantages of the discussed DMAs and read-out styles.

A.19.1 First experiment: HPGe-array DAQ

In this state of the art experiment the targeted cost per HPGe-detector channel is below 500€. To achieve this target the FPGA should process 6 or more ADC channels of 16 bits with 100 MHz sampling rate. Every accepted trace covers a time of 160 µs. The DAQ must be capable to process at least 1000 events/s in the calibration runs. During physical runs much lower event rates (<1 event/s) are common. The listed requirements are similar to the requirements in GERDA and LEGEND double beta decay experiments.

Trace length considerations: A time span of 160 µs requires 16000 samples in a 100 MHz clock domain. As designs whose trace length relates to power of 2 permits a time closure friendly control logic design, a destination trace length of 16384 samples is determined. In a 50/200 MHz clock domain 8192/32768 samples would be required.

Total Buffer and Ethernet link considerations: In most of the run time an event rate (far) below 1 Hz is expected (ethernet payload bandwidth of only ~32 kB/s). Just in the calibration (every week) an event rate larger as 1k events/s is demanded (payload bandwidth of ~32 MB/s or greater). An event rate of 10k events/s would require a payload bandwidth of 320 MB/s (~2.6 Gb/s) and hence would require a 3 Gb/s ethernet link. The PS ethernet link has a theoretical maximum event rate of approximately 4k events/s (~0.6 kHz per channel). This bandwidth must be shared by all channels of the device. A 6 kHz operation (= 1 kHz per channel) is therefore not possible with the PS 1 Gb/s link. As the calibration can be done piecewise per channel this is not a fatal issue.

Buffer size: The optimal event buffer size can be determined by a Monte Carlo Simulation or estimated out of the determined event building performance results (chapter 6.5.3). We use latter approach. The results (in Figure 44 and Figure 45) that belong to a trace length of 16384 samples used an ADC Ring Buffer with a size of 65536/131072 samples (Z7k/ZU+). This corresponds to 4/8 traces per ADC Ring Buffer. A similar discussion (with different points) was already done in chapter 6.1. In this chapter the discussion the ADC Ring Buffer design was realigned on the ADC Ring Buffer size in FC1.0 that comprised 32768 samples (would correspond to 2 trace sizes). In the “FPGA resource cost per device” Table 10 the FPGA resources in FC1.0 were (partly) used as benchmark. “FC1.0” in the RAM size calculations will be used as acronym for an ADC Ring Buffer size that comprises 32768 samples. We now can guess that a feasible buffer size for such trace lengths is in the range of 4...16 trace sizes (2...8 FC1.0). As the channel number also the total buffer capacity increases, it was decided that for the requested 6 channel DAQ a buffer size of 8 traces per channel is sufficient. We finally can calculate the RAM size that is required to build up the trace buffers of the requested DAQ: 6 Chn×8×0.5×FC1.0=6 Chn×8×16384 samples= 12 Mb =24FC1.0. As only a small number ("8×6×5×2") of events must be buffered, the required RAM size of the event information ring buffers is negligible and must not be considered in this discussion. Another main RAM consumer is the delay block of the firmware. A detailed derivation of its size must consider several points that were not listed in the main part of the thesis. In the case of the HPGe array experiment a delay line should have a size of approximately 8192 samples. Thus, the delay lines consume 6×0.25×FC1.0=1.5×FC1.0=768 kb of RAM. The suggested design therefore requires roughly 13 Mb FPGA-RAM. Looking into Table 9 reveals that the Trenz modules "TE0803-04-4AE11-A" (ZU4CG) and "TE0808-05-9BE21-AK" (ZU9EG) satisfy the requirements. As the total input bandwidth of all ADC channels is only 1.2 GB/s (6Chn×100MHz×2B), for the Trenz ZU+ modules a continuous HP-DMA approach could also be considered. The HP-DMA approach would be particular interesting for a similar setup where the ADCs have a sample rate of 200 MHz or that uses even longer traces. The only critical point that is remaining is the limited ethernet transfer rate (<4k events/s). This issue could be tackled be down sampling to 50 MHz (→ "srprop" in appendix chapter A.2) or using a smaller trace length.

De-facto the suggested design is quite simple, as for instance it does not require the following FC2.0 prototype features:

- "headeracp" has no performance gain in compare with the default "via GP/LPD" procedure (consequence out of Figure 46 and Figure 47)
- "pile up rejection" is only a (minor) issue in the calibration phase (even there the event rate is always below 10kHz), in the physic phase it is de-facto useless – this limits the urge to implement a pile up rejection in the DAQ firmware

A potential ADC nominee, the Ltc2107, currently costs around 185€ (2022/07/01). According to Table 9 all Trenz module nominees have a sufficient IO pin number to implement 6 Ltc2107 ADCs. A DAQ that uses the Trenz module "TE0803-04-4AE11-A" has a channel price of ~ 250€ and hence the suggested DAQ fulfills all requirements.

A.19.2 Second experiment: PMT-array DAQ

In order to decrease the background, the previous HPGe experiment requires an active background shielding by PMT array. The channel price of the PMT-DAQ is targeted to be below 200€. To achieve the goal the device should process at least 12 ADC channels whereas the 12bit ADCs have a sample frequency of 500MHz (0.75GB/s). The desired trace length of 64samples is modest and should not cause any significant issue. Simulations estimate that the average event rate will be in the range of 1k events/s with short highs during calibration (~80k events/s). In this experiment the PS ethernet of the Zynq devices is hence preferred.

Input bandwidth considerations: The demanded (detector signal) input bandwidth is 9GB/s (12×0.75GB/s). Any continuous HP-DMA design is thus invalid (remember that the determined maximum bandwidth limit of the continuous HP-DMA in the ZU+ devices is ~4GB/s (chapter 6.7.1)). The ACP-CMA is the only way to go!

Total Buffer and Ethernet link considerations: Figure 45 indicates that the desired event rates can be achieved by both Zynq devices. The "Global Multi Channel" operation mode of Figure 45 could buffer 512 trace header values in each of its event information ring buffers. All event ("header") ring buffer of all channels would only require modest 0.24Mb FPGA-RAM (512×(3+12)×4B≈0.24 Mb). The "Global Multi Channel" mode was already introduced in FC1.0 and should not be mixed with the "Channel Independent Multi Channel" mode that represents a new feature of FC2.0. The required FPGA-RAM for the ADC Ring Buffers is solely 4.5Mb (512×64samples×12Chn×12b=4.5 Mb) but as the available ADC Ring Buffer designs are not intended for an input data width of 12bit significant efficiency losses are to expect (BRAM neither UltraRAM offer an input data width of 12 bit[30]). A FPGA-RAM consumption of 6Mb (512×64samples×12Chn×16b) hence would be more accurate. As the FPGA-RAM consumption of the delay unit does not significantly contribute to the total RAM consumption in the suggested PMT-DAQ it is omitted in this estimate. The total FPGA-RAM consumption is thus roughly in the scale of 6.3Mb.

The most critical point in the Trenz module selection is not the available FPGA-RAM or the possible ethernet link, it is the available number of IO pins. In the desired DAQ design a single Znyq device must implement 12FADCs. Only the Trenz module "TE0808-05-9BE21-AK" has enough MGT pins to implement the Ultra-High-Speed ADCs by the JESD204B/ JESD204C procedure. As eligible ADCs with a JESD output data stream also typically cost more than 200€ per channel this approach is due to its costs not possible. Fortunately 500MHz ADCs with DDR-LVDS data output exist. An example would be the AD9434 (Analog Devices) for roughly 150€ per device. If the selected Zynq device costs less than 600€ the PMT-DAQ could be offered for the requested price limit. Every AD9434 requires 12 LVDS input pins of the Zynq device. In total the Zynq device must have 144 LVDS pins only for the ADC data streams. As the remaining firmware blocks also need external IO signals (and hence need IO-pins) the Trenz module "TE0720-03-1CF" is the only possible nominee. But its available FPGA-RAM is not sufficient to fulfill the "6.3Mb" requirement. Three possible loopholes exist. First of all, the PMT-array experiment could accept higher costs per channel. The PMT-DAQ than could use an ADC with JESD output and the "TE0808-05-9BE21-AK" module. Another possibility would be the implementation of a targeted JESD/DDR-LVDS ADC combination on the "TE0803-04-4AE11-A" module. One or two JESD ADCs should be sufficient to

[30]It is assumed that every channel has its own independent trigger domain – if all channels are implemented in the same trigger domain (for instance like in the "Global Multi Channel Mode") an efficient UltraRAM implementation is not an issue (72bit=6Chn×12bit).

save a sufficient number of LVDS pins. Substituting two “AD9434” ADCs by a single (two channel ADC) “AD9234-500” (also Analog Devices, ~ 286€) should do the job and still respects the cost limit. As using different ADC types in a single experiment is critical the following third loophole is likely the best choice. The requested average event rate was only 80 kHz, but the Z7k device in Figure 45 was capable to handle at least 300 kHz. A decrease of the event information ring buffer size to 256 events also decreases the event rate, but the new maximum event rate is still far away from 80 kHz. The size reduction of the buffers only slightly will influence the analysis quality of the experiment. This reduced buffer design also can be implemented in the Trenz module “TE0720-03-1CF“. As this module also has a plenty of LVDS pins, all desired 12 channels can be implemented by the “AD9434”.

Comment: The event rate and trace length restrictions of the “PMT array DAQ” were arbitrarily chosen. The instance was constructed to emphasize the most appealing advantage of the Z-7020 -CLG484 device (Z7k device on “TE0720-03-1CF“). It has plenty of pins for a cheap price. This is the basic requirement to design a DAQ with a “cheap” channel price like FC1.0. In the advent of this thesis the 7-7020 device was intended as default device for the new FC2.0 DAQ. Unfortunately, the poor memory access rate (of the CPU) and PS ethernet controller of the Z7k devices disproved the originally intended concept. A changeover to the ZU+ devices was required.

Though in the example many efforts were undertaken to lower the channel price, it still could not compete with FC1.0 costs (~70€ per PMT channel, ~280€ per HPGe channel).

A.19.3 Third experiment: Ultra-High-Event Rate DAQ

A pre-evaluation DAQ (16 bit ADC with Fs=100 MHz) determined an optimal trapezoidal shaper setting of an integration time below 1 µs. To simplify it is assumed that an optimized trace covers 200 16 bit samples. The final experiment will cover at around 1000 channels. All channels of the DAQ must together be capable to process 10^8 events per second. If like in FC the corresponding traces are forwarded to a read-out server, an event data stream of 40 GB/s would be the consequence. After 3 years of continuous data taking, the experiment will achieve its intended statistical quality. During the total measurement campaign, a total trace data amount of 4×10^6 TB must be processed.

Prelog: At such high event rates a significant event fraction will have pile up interference. It is possible to build up a read-out server system that can process and analyze an event data stream of 40 GB/s. The event building performance setup and the event building performance extrapolations of chapter 6.5.3 also imply that such a DAQ where a Zynq devices covers several channels can be build, but the designer now definitely should consider the usage of a MGT ethernet links. The disadvantages of MGT ethernet links still apply and obviously such a DAQ will have a substantial price. A loophole is therefore required. Instead of transferring the trace, a DAQ client unit (“Zynq”) could only convey the trace header and the determined peak energy.

In the pulse detection block of the FC DAQs a high precision energy signal exists. The main obstacle in this procedure is the degeneration effects of pile up on the firmware's energy signal. The pile up rejection can be done by the server (on the time stamp basis of the events that are part of the trace header) or by the firmware itself. In the latter approach this obviously has consequences on the state machine design of the "Trigger acceptance control" block (appendix chapter A.5). As a rejection in firmware significantly eases the server requirements it is the preferred option. Remember that also the server's read-out command generation times affect the performance of the total system.

Design consideration and FC2.0 prospects: The abstinence of the trace data enables a design where all the FPGA-RAM can be used to build up event information ring buffers (the determined energy gets assigned another additional event ring buffer). This also dramatically improves the buffer capacity and hence reduces the dead time. Such a DAQ easily can handle millions of events per second. The potential of such a design can be recognized by the "headerintrace" results that are (in case of the poor Z7k) published in the appendix chapter A.15 (Figure 167). The "headerintrace" approach still buffers unrequired data and therefore does not represent the possible maximum performance of the suggested procedure. For pre-evaluation reasons a "pile-up-rejection" state machine was also implemented in the firmware code. As this pre-code is not completely capable to reliably reject all pile-up cases a discussion and presentation of it was omitted in this thesis. Another disadvantage of the suggested procedure is its high specialized matching on the conditions of the "Ultra-High Event Rate" experiment. Users like DAQs whose setting can easily adopted on changing requirements (for instance trace length).

Design comparison between example DAQs and investigated FC2.0 prototype:

- The most serious difference between the prototype and the examples is the flexible trace length of the prototype. Without a new synthesis/invocation of adjusted firmware/software the read-out server routine can prompt a different trace length. This flexibility considers all significant settings of the ADC Ring Buffer/Trigger Acceptance Control combination and not just the trace length. The flexibility impedes the choice of the optimal event ring buffer/ ADC ring buffer size in the firmware. In order to counterbalance the negative effects of a fixed event ring buffer size for instance the "headerintrace" mode was developed (appendix chapter A.4.3). Unfortunately, this mode is not eligible for seamless retriggering. In principle it is possible to further decrease its (retrigger trace) dropout samples, but it is not possible to decrease the dropout to zero[31].

[31] Currently the dropout has 11 samples. The supposed degeneration effect in the most HPGe applications is likely marginal, but this assumption requires a deeper investigation. In principle ways exist that enable a (very likely) push to 0 – but the required bypassing strategies were excluded (as implementation) in appendix "trigger acceptance control" chapter 0 for various reasons.

- In the suggested example designs an ADC operation frequency was easily adaptable to 500 MHz. In the prototype design the read-in of such a data stream is possible, but the PL internal logic cannot be operated at such high frequencies. In order to handle such data streams a major rewrite of the firmware code of the pulse detection block and the ADC read-in block would be required (split of the 500 MHz stream into two 250 MHz streams). The changes in the remaining firmware blocks are expected to be less extensive, as stream doubling is mostly a (Verilog) “genvar” adaption issue of already available code.
- In the example DAQs pin mapping was not an issue as all LVDS pins could be used. In practice the pin behavior is also given by its “pin bank” assignment. As the “pin bank” operation voltage also has influence on the selectable voltage level (of the IO primitives) in a real design easily situations can arouse there not sufficient eligible IO pins are available. Consider that the FPGA also has to control the ADC chips via SPI/IC2 communication. The involved pins typically demand a (CMOS) voltage level of 3.3V and therefore contradicts the common LVDS voltage requirements. This is not limited to the FC2.0 prototype design, this is an issue of all real designs.