TECHNISCHE
UNIVERSITÄT
DRESDEN

Auslese und Signalverarbeitung eines Photodetektors mit einer FPGA-Auslesekarte

Bachelor-Arbeit
zur Erlangung des Hochschulgrades
Bachelor of Science
im Bachelor-Studiengang Physik

vorgelegt von

Nick Fritzsche
geboren am 03.03.1994 in Dorsten

Institut für Kern- und Teilchenphysik
Fachrichtung Physik
Fakultät Mathematik und Naturwissenschaften
Technische Universität Dresden
2017

Eingereicht am 31. Mai 2017

1. Gutachter: Prof. Dr. Arno Straessner
2. Gutachter: PD. Dr. Jürgen Henniger

Kurzdarstellung

Mit auf Silizium-Photomultipliern basierenden Photodetektoren können Photonensignale effizient und mit hoher Auflösung aufgenommen werden. Ein Anwendungsbereich ist unter Verwendung von Szinitillationsmedien die Untersuchung von radioaktiven Strahlungsquellen. Neben der analogen Signaldarstellung ist häufig eine Digitalisierung der Pulse von Interesse.
In der vorliegenden Arbeit wird ein Photonenpuls direkt mit einer Leuchtdiode erzeugt und von einem Multi-Pixel-Photon-Counter ausgelesen. Im Zentrum steht die Entwicklung einer Firmware zur digitalen Signalverarbeitung. Aufgrund der schnellen Datenverarbeitung und hohen Flexibilität wird ein System-on-a-Chip mit einer FPGA-Karte verwendet. Ziel ist es, die ideale Pulsform, sowie die Energiedeposition zu bestimmen, wozu in der Hardwarebeschreibungssprache VHDL Module zum Finden von Extrema und Formen der Signale erstellt werden.
Des Weiteren werden die Triggersignale durch eine PLL-Konstruktion auf dem Board geringfügig variiert. Mit Verschiebungen innerhalb der Taktlänge der ADC-Uhr, wird die Auflösung erhöht. Außerdem wird es durch Implementation von digitalen Filtern, hier einem Optimalfilter, ermöglicht die Lage eines Pulses unbekannter Phase zu rekonstruieren. Die Auswertung wird für Einzelpulsmessungen durchgeführt.

Abstract

Using silicon photomultipliers for photo detectors, photon signals can be recorded efficiently and with high resolution. Adding scintillating media, radioactive sources can be observed and analysed. Beside the analogue visualisation of the signals, digitalising the pulses is a core interest.
For this study the photon pulses are taken directly from a light-emitting diode and detected with a multi pixel photon counter. The focus is on developing a firmware for digital signal processing. Due to the quick data handling and the high flexibility a system-on-a-chip with a FPGA-board is selected. It is aimed to determine the ideal pulse shape and the energy deposition. For this reason modules, written in the hardware description language VHDL, are developed to find extrema and signal shapes.
Furthermore the trigger signal is generated and varied slightly via a PLL-construction on the board. The displacements are shorter than a clock cycle of the ADC-clock, so the measured samples can increase the resolution. Beside this digital filters, optimal filters in this case, are implemented and can be used to reconstruct pulses of unknown phase. The analysis is done for single pulse measurements.

Inhaltsverzeichnis

1 Einleitung

In der Teilchen- und Strahlungsphysik nimmt im Arbeitsfeld der Detektoren die Auslese und Signalverarbeitung eine entscheidende Rolle ein. Eine schnelle Verarbeitung von Eingangsdaten im Nanosekunden-Bereich kann mit der Verwendung von integrierten Schaltungen und Logikauslesekarten verwirklicht werden. Insbesondere Field Programmable Gate Arrays (FPGAs) eignen sich dank der Möglichkeit der mehrfachen und raschen Rekonfiguration zur Entwicklung von Auslesefirmware. Um große Datenmengen verarbeiten zu können und eine hohe Präzision bei der Digitalisierung zu erhalten, ist die Entwicklung effizienter Analysealgorithmen bedeutsam.
In der vorliegenden Arbeit wird ein Multi-Pixel-Photon-Counter (MPPC) ausgelesen. Auf diese Einleitung folgend wird im zweiten Abschnitt die Signalerzeugung mit einer Leuchtdiode (LED) und die Pulsdetektion, sowie Signalverstärkung auf Grundlage von Siliziumphotomultipliern (SiPM) beschrieben. Das Signal wird mittels einer FPGA-Auslesekarte untersucht. Von Interesse ist die Pulsform, die Amplitude der Energiedeposition, sowie deren zeitliche Lage. Eine diesbezügliche Auswertung wird durch das Implementieren von digitalen Filtern erreicht, welche im dritten Abschnitt behandelt werden. Die zur Verarbeitung verwendete Hard- und Software, sowie die erstellten Firmware-Module werden im vierten Abschnitt vorgestellt. Im Besonderen wird auf das Triggern der LED in unterschiedlichen Phasenlagen innerhalb eines Takts eingegangen. Der fünfte Abschnitt behandelt die durchgeführten Messungen, wozu zunächst der experimentelle Aufbau, sowie der Datenfluss vorgestellt wird. Es wird eine ausführliche Untersuchung des Triggersignals durchgeführt und darauf folgend die Aufnahme und Analyse der LED-Pulse durchgeführt. Der Idealpuls wird bestimmt, Filterkoeffizienten berechnet und zum Abschluss wird die Phasenlage aus den gefilterten Signalen für Einzelpulse bestimmt. Die Ergebnisse werden im sechsten Abschnitt zusammengestellt und ein Ausblick zur Erweiterung der Versuchsanordnung für weitere Experimente wird gegeben.

2 Optoelektronische Detektion

2.1 Leuchtdioden

Als Strahlungsquelle für experimentelle Untersuchungen an Photodetektoren eignen sich LEDs. Sie weisen die elektrischen Eigenschaften von Dioden auf, dass sie in Durchlassrichtung Strom nahezu ungehindert, in Sperrrichtung hingegen fast gar nicht durchlassen. [9] Sie zeichnen sich durch Lichtemission bei Stromfluss in Durchlassrichtung aus. Der Aufbau entspricht p-n-Halbleiter-Dioden, welche durch Dotierung mit Elektronen und Löchern eine Raumladungszone am Übergangsbereich erzeugt. [12] Beim Anlegen einer Spannung wird Stromfluss in eine Richtung ermöglicht. Es gilt die Driftspannung zu überwinden. Während in Sperrichtung der Potentialwall anwächst und solange die Durchbruchspannung nicht überschritten wird lediglich Minoritätenladungsträger fließen, wird in Durchlassrichtung das elektrische Feld der Raumladungszone neutralisiert und das neue äußere E-Feld ermöglicht den Stromfluss.
Die Leuchteigenschaft der LEDs hängt vom Material ab. In der Herstellung werden direkte Halbleiter verwendet, deren Lichtaussendung mit dem Bändermodell erklärbar ist. [15] Die Bandlücke zwischen Valenz- und Leitungsband ist bei direkten Halbleitern bei gleicher Wellenzahl besonders dünn. Dadurch dominiert der Übergang unter Emission eines Photons, bei dem der Impuls erhalten bleibt. Im Gegensatz dazu ändern sich bei indirekten Halbleitern Wellenzahl und somit Impuls und nicht-strahlende Reaktionen wie Rekombination treten häufiger auf. Der charakteristische Energie- und somit auch Wellenlängenbereich zum Überwinden der Bandlücke bestimmen die Farbe des Leuchten. Es können auch Reaktionen mit Exzitonen und Phononen auftreten, bei welchen weniger Lichtquanten entstehen. Dadurch wird die Strahlung niederenergetischer, was sich in der ins Rötliche wandelnden Farbgebung widerspiegelt.

Im experimentellen Aufbau wird eine SP5601 LED Driver Box verwendet. [6] Diese verfügt über einen 12 V-Gleichstrom-Eingang und ermöglicht das Anlegen einer Grundspannung zum Betreiben der LED. Die Diode kann durch einen internen Pulsgenerator in zwei verschiedenen Frequenzeinstellungen getriggert werden. Durch Umlegen eines Schalters kann in einen externen Trigger-Modus über eine mit einem Lemo-Eingang verbundene Quelle gewechselt werden. Ein weiterer Lemo-Ausgang ermöglicht die Ausgabe des Triggersignals, was z.B. zur Darstellung mit einem Oszilloskop genutzt werden kann. Ein- und Ausgänge nutzen Transistor-Transistor-Logik (TTL).

2.2 Photodetektoren

Optoelektronische Sensoren nutzen die lichtelektrischen Effekte aus, um Photonenpulse durch Wandlung in elektrische Signale zu detektieren.
Eine Möglichkeit besteht darin basierend auf dem äußeren lichtelektrischen Effekt beim Auftreffen von einfallenden Licht auf eine Photokathode Elektronen auszulösen. [10] Dabei wird jedoch meist eine Verstärkung des Signals nötig, wozu Sekundärelektronenvervielfacher verwendet werden können. In dieser Elektronenröhre werden die Photoelektronen durch ein äußeres elektrisches Feld beschleunigt und treffen auf Dynoden genannte Elektroden. Dabei werden mehrere Sekundärelektronen ausgeschlagen. Dieser Prozess wiederholt sich mehrfach, wodurch sich das elektrische Signal exponentiell verstärkt.
Eine kompaktere und robustere Möglichkeit Photonen zu detektieren ist die Nutzung von Avalanche-Photodioden (APDs), welche auf dem inneren lichtelektrischen Effekt beruhen. [19] Sie werden knapp unter der Durchbruchspannung betrieben, bei der es plötzlich zu einem starken Stromfluss kommt. Nach dem photovoltaischen Effekt entstehen Ladungsträgerpaare an der Raumladungszone, welche sich in p- und n-Schicht auftrennen und somit einen Strom in Sperrrichtung erzeugen. [17] Die Photoelektronen werden durch die anliegende Spannung in der Raumladungszone in Richtung n-Schicht beschleunigt. Durch Stoßionisation entstehen dort Sekundärladungsträger, welche wiederum mit Stoßionisation lawinenartig weitere Ladungsträger erzeugen. Ein verbreitetes Beispiel stellen Silizium-Photomultiplier (SiPM) dar, welche in dieser Multiplikationszone eine Verstärkung um einen Multiplikationsfaktor zwischen 100 und 500 erreichen. [11] Bei einem Multi Pixel Photon Counter (MPPC) werden mehrere SiPM-Pixel zusammengeschlossen, um aktive Fläche und Auflösung zu erhöhen. Auf eine Fläche von $1x1\,\mathrm{mm}^2$ passen bis zu 1000 Pixel. [21] Zu den weiteren Vorteilen zählen die geringe notwendige Schwellspannung, die hohe Effizienz der Detektion und die Unabhängigkeit der Messung von äußeren Magnetfeldern. [22]

In dieser Arbeit wird ein SP5650 Sensor Halter von CAEN mit integriertem $1x1\,\mathrm{mm}^2$ SiPM genutzt. [5] Der von der LED kommende Puls wird mit einem auf Totalreflexion basierendem Lichtwellenleiter (LWL) zum MPPC geleitet per FC-Steckverbinder unmittelbar an diesem befestigt.

3 Signalverarbeitung

3.1 Digitale Filter

Im Gegensatz zu analogen Shaper-Schaltungen, die elektrische Signale in Amplitude, Frequenz oder Phasenlage meist unter Verwendung von Widerständen, Spulen und Kapazitäten oder aktiven Operationsverstärkern formen, verändern digitale Filter bereits digitalisierte Signale. Implementiert in Logikbausteinen oder digitalen Signalprozessoren stellt das Filter eine mathematische Funktion dar, welche auf das Eingangssignal wirkt und ein Ausgangssignal der gleichen physikalischen Größe zurückgibt. [13] Auf Kosten von theoretischen Anforderungen, Diskretisierungs- und Geschwindigkeitsverlusten können dank hoher Störimmunität im Vergleich zu analogen Filtern auch umfangreiche Verarbeitungsprozesse möglich gemacht werden. Außerdem sind Alterung und Bauteiltoleranzen weniger problematisch, was Stabilität, Reproduzierbarkeit und Vorhersagbarkeit der Signalverarbeitung verbessert und den Entwurf von Systemen beschleunigt. Gemeinsam mit der Flexibilität durch Wiederverwendbarkeit universeller programmierter Bausteine für andere Systeme, stellt dies einen großen Vorteil in der Entwicklung von Systemen dar. [18]

Im Folgenden werden ausschließlich lineare passive Filter verwendet, die nur die mathematischen Methoden der Addition und Multiplikation mit Konstanten verwenden. Diese lassen sich mit der Zustandsdifferentialgleichung

$$y(n) = \sum_{i=0}^{k} d_i x(n-i) - \sum_{i=1}^{k} g_i y(n-i) \tag{3.1}$$

beschreiben, wobei die erste Summe den nicht rekursiven, vom Eingangssignal abhängigen Anteil beschreibt, während der zweite rekursive Term vom Ausgangssignal abhängt. Für Transversalfilter (FIR), die durch einen Abbruch des gefilterten Signals nach endlich vielen Schritten charakterisiert werden, verschwinden die rekursiven Koeffizienten g_i, womit sich die nur noch aus der ersten Summe bestehende Transfergleichung ergibt.

$$y(n) = \sum_{i=0}^{k} d_i x(n-i) \tag{3.2}$$

3.2 Optimalfilter

Zur Bestimmung von Amplitude und Startzeit eines Signals können Optimalfilter verwendet werden. Im Folgenden wird ein Verfahren von W.E. Cleland und E.G. Stern beschriebenen, welches ursprünglich für Ionisationskalorimeter bei hohen Luminositäten entwickelt und zum Beispiel für das ATLAS-Experiment am CERN angewendet wurde. [8]
Hierbei werden die Transferfunktionen durch die Linearkombinationen

$$\begin{aligned} u &= \sum_{i=0}^{N-1} a_i x_i \\ v &= \sum_{i=0}^{N-1} b_i x_i \end{aligned} \tag{3.3}$$

mit den Filterkoeffizienten genannten Gewichtungsfaktoren a_i und b_i eingeführt. Das Eingangssignal x_i wird über die Amplitude A und die Pulsform $g(t)$, relativ zur Startzeit τ charakterisiert.

$$x_i = Ag(t_i - \tau) \tag{3.4}$$

Die Taylor-Entwicklung der Funktion $g(t)$ lautet

$$g(t_i - \tau) = g(t_i) - \tau \dot{g}(t_i) + ... \tag{3.5}$$

wobei $\dot{g}$ die Ableitung nach der Zeit beschreibt. Unter Vernachlässigung höherer Ableitungen und Beachtung des elektronischen Rauschens n_i mit dem Mittelwert $\langle n_i \rangle = 0$ schreibt sich das Eingangssignal

$$x(i) = Ag_i - A\tau \dot{g}_i + n_i. \tag{3.6}$$

Somit ergibt sich für die Filterfunktionen

$$\begin{aligned} u &= \sum_{i=0}^{N-1} a_i (Ag_i - A\tau \dot{g}_i + n_i) \\ v &= \sum_{i=0}^{N-1} b_i (Ag_i - A\tau \dot{g}_i + n_i) \end{aligned} \tag{3.7}$$

Die Filterkoeffizienten sollen so gewählt werden, dass die Filter im Scharmittel die Amplitude A und das Produkt $A\tau$ ausgeben. [8]

$$\begin{aligned} A = \langle u \rangle &= \sum_{i=0}^{N-1} (A a_i g_i - A\tau a_i \dot{g}_i + \langle n_i \rangle) \\ A\tau = \langle v \rangle &= \sum_{i=0}^{N-1} (A b_i g_i - A\tau b_i \dot{g}_i + \langle n_i \rangle) \end{aligned} \tag{3.8}$$

Es werden die Varianzen der Funktionen u und v unter Einführung der Rauschautokorrelationsfunktion $R_{ij} = \langle n_i n_j \rangle$ zum Zeitpunkt $t_i - t_j$ berechnet.

$$\begin{aligned} Var(u) &= \langle (\sum_{i=0}^{N-1} a_i (A g_i - A\tau \dot{g}_i + n_i) - \langle u \rangle)^2 \rangle \\ &= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} a_i a_j \langle n_i n_j \rangle \\ &= \sum_{ij} a_i a_j R_{ij} \\ \text{und analog } Var(v) &= \sum_{ij} b_i b_j R_{ij} \end{aligned} \tag{3.9}$$

Der Einfluss des Rauschens soll klein gehalten werden, wozu die Varianzen minimiert werden. Hierzu wird die Methode der Lagrange-Multiplikatoren ($\lambda, \kappa, \mu,$ und ρ) genutzt. Aus 3.8 folgen die Randbedingungen

$$\begin{aligned} \sum_i a_i g_i = 1 \quad \sum_i a_i \dot{g}_i = 0 \\ \sum_i b_i g_i = 0 \quad \sum_i b_i \dot{g}_i = -1 \end{aligned} \tag{3.10}$$

mit welchen sich die zu minimierenden Funktionen wie folgt ergeben:

$$\begin{aligned} I_u &= \sum_{ij} a_i a_j R_{ij} - \lambda (\sum_i a_i g_i - 1) - \kappa \sum_i a_i \dot{g}_i \\ I_v &= \sum_{ij} b_i b_j R_{ij} - \mu \sum_i b_i g_i - \rho (\sum_i b_i \dot{g}_i + 1) \end{aligned} \tag{3.11}$$

Die partiellen Ableitungen nach den variierenden Koeffizienten werden gebildet und mit Null gleichgesetzt.

$$\begin{aligned} \frac{\partial I_u}{\partial a_i} &= \sum_j a_j R_{ij} - \lambda g_i - \kappa \dot{g}_i = 0 \\ \frac{\partial I_v}{\partial b_i} &= \sum_j b_j R_{ij} - \mu g_i - \rho \dot{g}_i = 0 \end{aligned} \tag{3.12}$$

Somit lauten die Koeffizienten in Matrixschreibweise mit $\boldsymbol{a} = (a_0, ..., a_{N-1})^T$, $\boldsymbol{b} = (b_0, ..., b_{N-1})^T$ und $\boldsymbol{g} = (g_0, ..., g_{N-1})^T$

$$\begin{aligned} \boldsymbol{a} &= \lambda \boldsymbol{R}^{-1}\boldsymbol{g} + \kappa \boldsymbol{R}^{-1}\dot{\boldsymbol{g}} \\ \boldsymbol{b} &= \mu \boldsymbol{R}^{-1}\boldsymbol{g} + \rho \boldsymbol{R}^{-1}\dot{\boldsymbol{g}} \end{aligned} \tag{3.13}$$

mit den aus den Randbedingungen gewonnenen Lagrange-Multiplikatoren

$$\lambda = \frac{Q_2}{\Delta} \qquad \kappa = -\frac{Q_3}{\Delta} \qquad \mu = \frac{Q_3}{\Delta} \qquad \rho = -\frac{Q_1}{\Delta} \tag{3.14}$$

$$\begin{aligned} &\text{wobei } Q_1 = \boldsymbol{g}^T\boldsymbol{R}^{-1}\boldsymbol{g} \qquad Q_2 = \dot{\boldsymbol{g}}^T\boldsymbol{R}^{-1}\dot{\boldsymbol{g}} \quad Q_3 = \boldsymbol{g}^T\boldsymbol{R}^{-1}\dot{\boldsymbol{g}} = \dot{\boldsymbol{g}}\boldsymbol{R}^{-1}\boldsymbol{g} \\ &\text{und } \Delta = Q_1Q_2 - Q_3^2 \end{aligned} \tag{3.15}$$

4 System-on-a-Chip mit FPGA

4.1 Hardware

Zur weiteren Signalverarbeitung wird ein System-on-a-Chip (SoC) verwendet. Mit diesem sollen die Eingangspulse digitalisiert werden und die relevanten Informationen selektiert und gespeichert werden.
Im Zentrum steht ein FPGA-Chip der Marke Altera. Mit diesem lassen sich parallel und in hoher Geschwindigkeit Operationen an programmierbaren Logikblöcken durchführen. Signale können auf Pins gelegt werden, welche verschiedene Hardware-Elemente betreiben. Hierzu zählen Schalter, Druckknöpfe und LEDs, des Weiteren aber auch Anschlüsse nach außen, wie z.B. Anschlüsse nach den Video Graphics Array (VGA) und Sub-Miniature-A (SMA) Standards.
FPGAs können mehrfach beschrieben werden und dies in geringer Konfigurationszeit. Die Beschreibung erfolgt in einer Gerätbeschreibungssprache. In diesem Experiment wird die verbreitete Very High Speed Integrated Circuit Hardware Description Language (VHDL) verwendet. In dieser können die Logikblöcke benannt, eingestellt und durch Signale miteinander verbunden werden.
Auf dem SoC befindet sich ein 50 MHz-Oszillator, welcher Rechteckpulse mit einem Duty Cycle von 50% erzeugt. Er wird als Taktgeber für die Schaltung verwendet. Des Weiteren ist ein dedizierter Speicherbereich festgelegt, welcher als Dual Port Memory verwendet werden kann. Mittels einer JTAG-Schnittstelle kann auf diesen im laufenden Betrieb zugegriffen werden.

4.2 Software

Die Entwicklung von Schaltungen und Modulen in VHDL wird in einem Texteditor vorgenommen. Die Anweisungen werden von der Quartus II Software von Altera interpretiert. [20] Hierzu zählen Analyse und Synthese, Kompilierung, sowie das Durchlaufen eines Fitters und eines Assemblers.
Zum Anlegen und Nutzen der Speicherbereiche wird das Quartus-interne Tool Qsys genutzt. Dieses beruht auf einer grafische Benutzeroberfläche, mit welcher Master-Slave-Zuweisungen getätigt, Taktgeber angeschlossen und Reset-Funktionen zugeordnet werden können. Außerdem lässt sich jeweils Speichergröße, -adresse, und -blockbreite festlegen. Lese- und Schreibrechte werden zugewiesen. Auch andere Intellectual Properties (IPs) können genutzt werden.

So werden hier für die später erläuterten Phasenregelschleifen (PLLs) Frequenz und Phasenlage der Uhren eingestellt. Nach Abschluss der Bearbeitung kann eine VHDL-Datei generiert werden, welche unter dem Top-Layer in der Modulhierarchie eingefügt wird. Die ein- und ausgehenden Signale werden von der Software benannt und können nachgelesen werden, um sie in anderen Modulen zu verbinden.
Eine weitere interne Software ist Signal Tap, welches die JTAG-Schnittstelle ausnutzt, um Signale zwischenzuspeichern und zu visualisieren. Die Signale werden in Echtzeit ausgegeben und können in verschiedenen Binärdarstellungen oder auch als Liniendiagramme angezeigt werden. Die Daten von einzelnen Aufnahmen können exportiert werden.
Die JTAG-Schnittstelle wird auch von der System Console genutzt. [2] In dieser kann über eine Kommandozeile in der Tool Command Language (Tcl) auf die Speicherbereiche zugegriffen werden. Hierzu können auch in externen Skripten definierte Prozeduren geladen und angewendet werden.

4.3 Firmware

4.3.1 Triggerpulsgenerator

Zum Triggern der LED sollen später Signale in verschiedenen Phasenlagen verwendet werden. Deshalb soll ein externes Triggersignal genutzt werden. An dessen Pulsform sind Anforderungen gestellt: Zum einen muss das Signal eine Mindestbreite überschreiten, um überhaupt einen Effekt bei der Leuchtdiode zu erzeugen. Außerdem gilt es den Abstand zwischen den Triggerpulsen groß genug zu wählen, dass der Ausläufer des LED-Pulses sich nicht mit dem nächsten Puls überlagert und die Wärmeentwicklung begrenzt wird. Hierfür wurde ein in der Abbildung 7.1 im Anhang einsehbares VHDL-Modul entwickelt, welches den Taktgeber auf dem Board ausnutzt. Es werden zwei Zählregister eingeführt, von denen eines das Taktgebersignal, das andere das um eine vorgegebene Taktweite verschobene Signal speichert. Als Triggersignal wird die Differenz einer einzelnen Bitposition beider Register gewählt. Die Länge des Triggersignals hängt von der eingestellten Weite ab. Der Abstand zwischen zwei Triggerpulsen sinkt bei steigender Bitposition um jeweils eine Zweierpotenz.

4.3.2 Phasenregelschleifen

Um bei den Messungen möglichst viele Stellen abtasten zu können, gilt es die Schaltungsblöcke mit hoher Frequenz zu betreiben. Um eine höhere Frequenz, als die der internen 50 MHz-Oszillatoren zu erreichen, werden PLLs verwendet. [1] Diese basieren auf spannungsabhängigen Oszillatoren (VCOs), welche abhängig von dem aus einem Loop-Filter kommendem Strom mit höherer oder niedriger Frequenz arbeiten. Der Strom für den Loop Filter wird durch eine Ladungspumpe erzeugt, welche wiederum das Signal eines Phasenfrequenzdetektors (PFD) nutzt,

der das einkommende Taktgebersignal verarbeitet.
Im Versuchsaufbau soll die Genauigkeit der Pulslagenbestimmung untersucht werden. Hierzu werden mit Funktionsblöcken in Qsys über eine PLL 16 verschiedene Phasenlagen in $22,5\,°$-Schritten erzeugt. Für diese wird jeweils 150 MHz eingestellt, wodurch die Pulse alle $6,67\,\text{ns}$ abgetastet werden können. Die Taktlänge wird nicht niedriger gewählt, da die maximale Frequenz erreicht ist, mit welcher der Analog-Digital-Konverter (ADC) betrieben werden kann. Die phasenverschobenen Uhren werden im Top-Modul mit vier Kippschalter auf dem FPGA-Board verbunden, sodass je nach Einstellung der Schalter ohne Neukompilierung des Quartus-Projekts eine beliebige Phasenlage eingestellt werden kann.

4.3.3 Transversalfilter

Das aus dem Detektor kommende Signal wird im ADC digitalisiert, dahinter aufgeteilt und von jeweils einem Filter mit endlicher Impulsantwort moduliert. Die beiden Filter funktionieren nach dem identischen Prinzip. Für die Filtertiefe fünf wird die Summe über dem Produkt der einzelnen Signale multipliziert mit den Filterkoeffizienten gebildet. Die Filtertiefe ist unter Abwägung von Ressourcenaufwand und Verringerung der Standardabweichung gewählt [14] Es lassen sich unterschiedliche Filterkoeffizienten für beide Filter wählen. Diese können mit der System Console im laufenden Betrieb an einen Speicherplatz übergeben werden. Eine in VHDL geschriebene Einlesefunktion verbindet zwei Druckknöpfe auf dem FPGA mit den Lesesignalen der beiden Koeffizientenspeicherplätze, sodass die Filterkoeffizienten per Knopfdruck aktualisiert werden können.

4.3.4 Maximumfinder

Von den gefilterten Pulsen sollen die Maxima bestimmt werden, welche bei geeigneter Koeffizientenwahl der Energiedeposition E bzw. dem Produkt aus Energie und Phasenlage $E\tau$ entsprechen. Hierzu wird ein Maximumfinder-Modul geschrieben, welches die Pulshöhe eines Eingangssamples mit denen der umliegenden Samples vergleicht. Liegt die Pulshöhe oberhalb der jeweils zwei Nachbarn zeitlich vor und nach dem Sample und sind außerdem die ersten Nachbarn höher als die zweiten Nachbarn, so wird das Mittlere als Maximum identifiziert. Außerdem wird durch das Abgleichen mit vorgegebenen Schwellwerten überprüft, dass der Wert oberhalb des elektronischen Rauschens und unterhalb des Sättigungswerts liegt. Um die Vergleichsoperationen durchführen zu können, wird ein Schieberegister angelegt. Dieses ist ein Array von 32 Standard Logic Vektoren, welche zeitlich aufeinanderfolgende Samples enthalten. Mit jedem Taktgeberzyklus werden die Samples um eine Position in Richtung des kleineren Indizes weitergeschoben. In jedem Uhrzyklus wird der fünfte Wert des Arrays auf ein Maximum überprüft. Dadurch ist zum einen gewährleistet, dass die zeitlich folgenden Samples bereits abgespeichert sind, zum anderen lässt sich bei Maximumerkennung der gesamte Puls

abspeichern. So wird in einem Speicherbereich lediglich der Maximalwert gespeichert, in einem weiteren Speicherbereich darüber hinaus eine Reihe von 32 Samples. Dies geschieht, indem die 32 16-Bit Samples als Teile eines 512-Byte-Vektors interpretiert werden, dem eine Adresse im Speicher zugeordnet wird (siehe Abbildung 7.4 im Anhang). Somit ist es möglich ohne Totzeit innerhalb eines Taktes den gesamten Puls in den Speicher zu schreiben. Des Weiteren wird für die bei der Auswertung irrelevanten Bereiche zwischen den Pulsen kein Speicherplatz verwendet.

5 Messungen

5.1 Experimenteller Aufbau und Datenfluss

Der experimentelle Aufbau wird dem Datenfluss folgend erklärt. Dieser ist in Abbildung 5.1 visualisiert. Zunächst wird mit dem auf dem FPGA implementierten Triggerpulsgenerator ein Rechtecksignal erzeugt. Für die weitere Übertragung wird der Pin zur horizontalen Synchronisation vom VGA-Ausgang des SoC gewählt, da das Signal hier nicht differenziert wird, wie es zum Beispiel bei den SMA-Schnittstellen mit Übertragern geschieht. Der Puls wird in den Lemo-Trigger-Eingang der LED-Treiberbox geführt, wo ein Lichtpuls ausgelöst wird. Dieser wird per LWL zu den SiPM-Pixel des MPPC geführt. Der Detektor wird dabei mit einer Bias-Spannung von 70 V betrieben, welcher von der Spannungs- und Verstärkungskontrolle (Power Supply and Amplification Unit, PSAU) zur Verfügung gestellt wird. Als TTL-Signal wird das Detektorsignal aus dem Lemo-Ausgang über eine SMA-Schnittstelle auf das Data Conversion Board des SoC in den ADC geführt. Das digitalisierte Signal wird dann vom FPGA mit dem E- bzw. $E\tau$-Filter geformt. Die zugehörigen Maximum-Finder bestimmen anschließend die relevanten Puls-Extrema und geben ausschließlich diese und die umgebenden Puls-Samples in den Speicher weiter. Aus diesem können per JTAG-Interface die Daten über eine USB-Schnittstelle auf die Festplatte des Versuchs-Computers übertragen werden. In anderer Richtung kann über die JTAG-Schnittstelle auch in den Speicherbereich geschrieben werden, um etwa Filterkoeffizienten zu ändern.

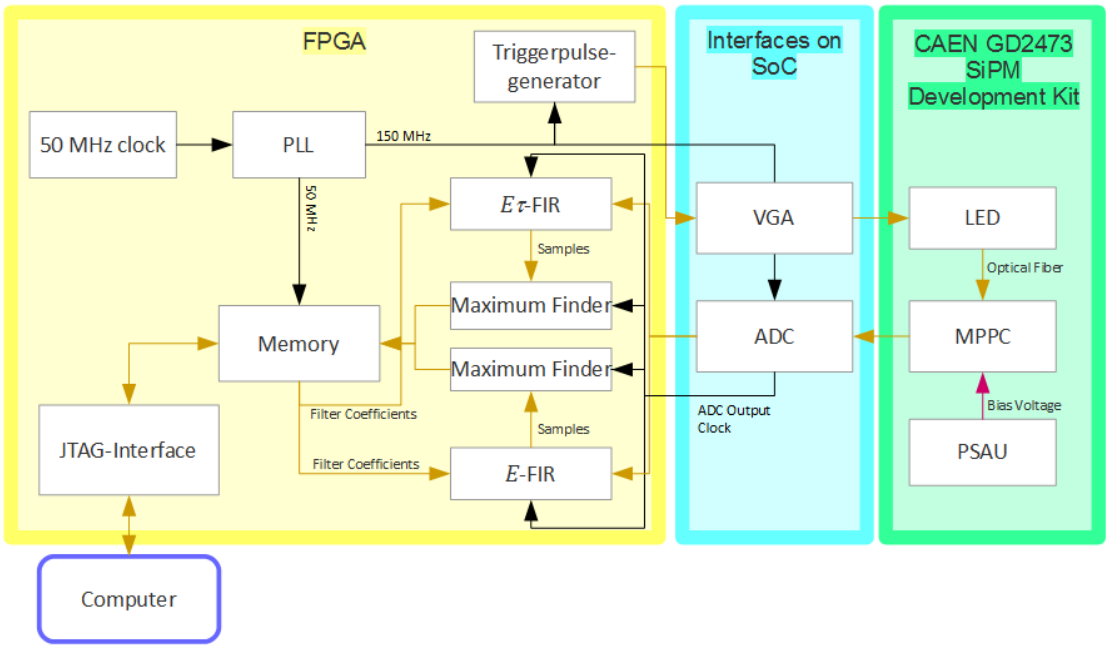

Abbildung 5.1: Datenfluss im experimentellen Aufbau, aufgeteilt in Module der integrierten Schaltung auf dem FPGA (links), den Schnittstellen des SoC (mitte) und den zum CAEN GD2473 SiPM Development Kit gehörigen Geräten (rechts). Schwarze Pfeile beschreiben Taktgebersignale, dunkelgelbe Pfeile den beobachteten Puls.

5.2 Untersuchung des Triggersignals

Zur Justage des Triggerpulses wird zunächst das interne Triggersignal der LED-Treiberbox mit einem Oszilloskop betrachtet. Es zeigt sich ein Rechteckpuls mit sehr geringem Duty Cycle. Bei einem angelegten Lastwiderstand von $R = 50\,\Omega$ wird auf dem Plateau eine mittlere Amplitude von $A = 3,369\,\mathrm{V}$ bei einer Standardabweichung von $\sigma = 0,036\,\mathrm{V}$ gemessen. Das Signal wiederholt sich periodisch, wobei die Frequenz mit einer Stellschraube aus einem Bereich von 500 Hz bis 500 MHz gewählt werden kann. Die LED schaltet sich bei hohen Frequenzen zeitweise selbstständig aus, vermutlich als Schutz bei hohen Temperaturen. Die für die weitere Versuchsdurchführung gewählte Frequenz von $f = 84\,\mathrm{kHz}$ ist niedrig genug, um ein Ausschalten zu verhindern, und hoch genug, dass eine Überlagerung des Schweifs eines Pulses mit dem Folgenden bei einem entsprechenden Abstand von $d = 12\,\mu\mathrm{s}$ ausgeschlossen werden kann. Die Breite eines Triggerpulses beträgt 125 nm.

Das extern erzeugte Signal wird zum Vergleich auf einen weiteren Kanal des Oszilloskops gelegt. Um ein erfolgreiches Triggern der LED sicher zu stellen, werden die Parameter zum Einstellen von Pulsbreite und -abstand im VHDL-Modul des Triggerpulsgenerators so gewählt, dass das Signal dem Internen möglichst ähnlich sieht. Eine Gegenüberstellung ist in Abbildung 5.2 zu sehen. Es stellt sich heraus, dass der externe Puls die gleiche Höhe wie der Interne aufweist. Somit eignet sich das Signal zur Interpretation mit TTL, welches im Bereich von 2 V bis 5 V als 1 interpretiert wird und zwischen 0 V und 0,4 V als 0. [3]

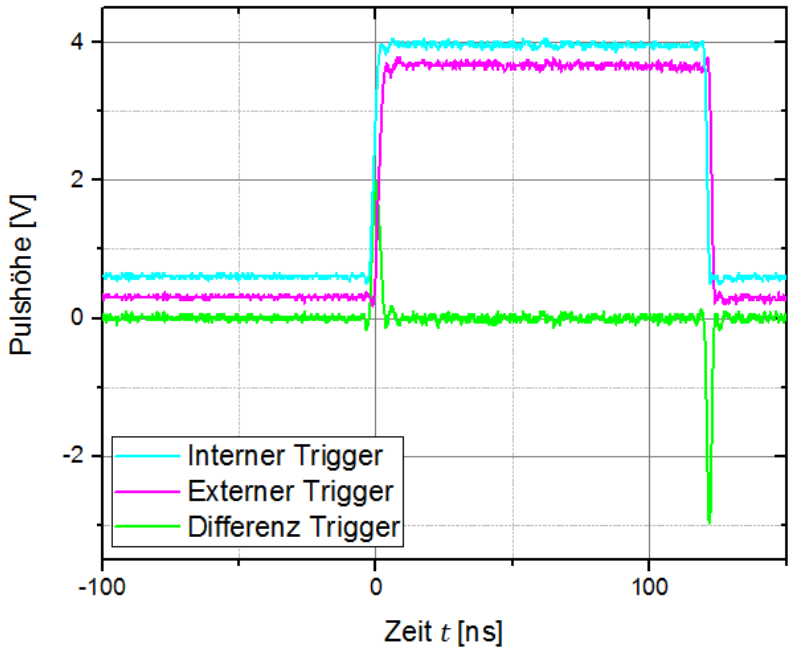

Abbildung 5.2: Das interne Triggersignal der LED-Treiberbox und das auf dem FPGA generierte externe Triggersignal in der Höhe versetzt aus dem Tektronix DPO 7104 Digital Phosphor Oscilloscope. Die Differenz der Pulshöhen liegt um 0. Es zeigt sich, dass die Signale bis auf einen geringfügigen zeitlichen Versatz der steigenden und fallenden Flanke als gleich angenommen werden können.

Im Folgenden wird das externe Triggersignal auf den Eingang der LED-Treiberbox gelegt und das Detektor-Signal mit dem Oszilloskop dargestellt. In der Bedienungssoftware der PSAU wird eine Biasspannung von 70 V eingestellt. Abbildung 5.3 zeigt, dass Amplitude und Pulsform mit dem intern getriggerten Signal übereinstimmen.

Der Einfluss der Breite des Triggersignals auf den LED-Puls wird untersucht. Dabei sollen Änderungen im Triggerpulsgeneratormodul zur Variation der Pulsbreite umgangen werden, was Neukompilierungen notwendig machen würde und nur für ausgewählte Einstellungen Ergebnisse lieferte. Deshalb wird das Triggersignal an einen Gate- und Delay-Generator angeschlossen und für diese Untersuchung als Gatesignal betrachtet. Mit einer Stellschraube lässt sich die Pulsweite kontinuierlich von der kleinstmöglichen bis zu sehr hohen Einstellung ändern. Das Ausgangssignal wird zusammen mit dem ursprünglichen Triggerpuls und dem vom MPPC detektierten LED-Signal auf dem Oszilloskop ausgegeben. Abbildung 5.4 zeigt die Signale bei kleinstmöglicher, sowie einer sehr hohen Weite. Abbildung 5.5 zeigt für die gewählten Pulsweiten aus Abbildung 5.4, dass das LED-Signal für beide Einstellungen unverändert bleibt. Beim Durchlaufen aller möglichen Weiten zeigt sich, dass das LED-Signal unabhängig davon ist. Es wird also ein Lichtpuls erzwungen, allerdings nicht die Form und Amplitude verändert, weshalb die Pulsweitenwahl nicht weiter beachtet wird.

Es wird ein phasenverschobenes Triggersignal aus der PLL auf das Oszilloskop gelegt. Am Beispiel von 180 ° wird der Versatz zum 0°-Signal grafisch ermittelt. Bei den mit 150 MHz

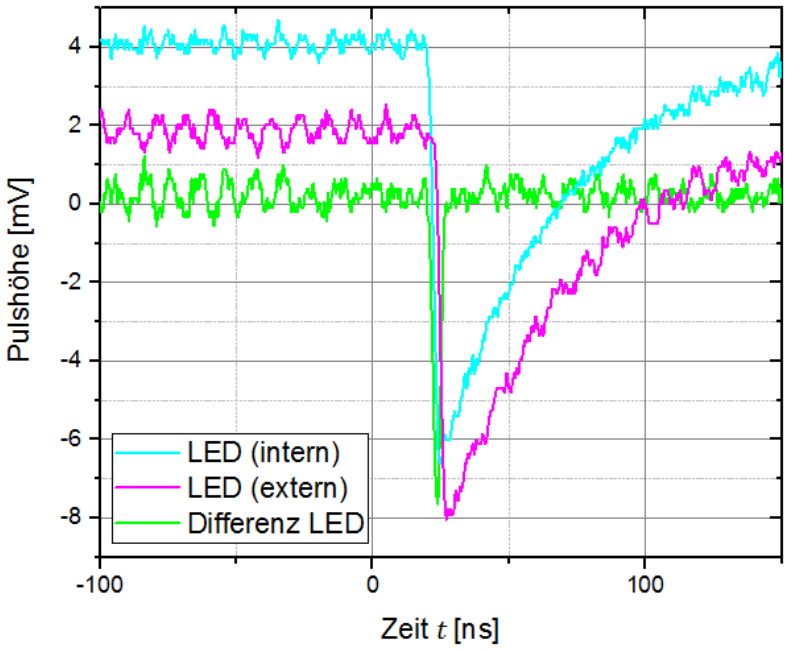

Abbildung 5.3: Die intern und extern jeweils auf die steigenden Flanken getriggerten LED-Signale werden in der Höhe versetzt übereinandergelegt. Die Differenz der Pulshöhen liegt um 0. Amplitude und Pulsform erweisen sich als gleich.

betriebenen PLL-Uhren ist zu erwarten, dass der Abstand der halben Taktlänge von $3,3\,\mathrm{ns}$ entspricht. Abbildung 5.6 lässt erkennen, dass der 180 °-Puls um etwa 5 ns verschoben ist, was oberhalb der Erwartung liegt. Wie in der analogen Darstellung in Abbildung 5.7 zu sehen ist, weißt der LED-Puls eine steile steigende Flanke auf. Um den Puls nach der Digitalisierung durch den ADC rekonstruieren zu könnnen ist es wichtig, dass mindestens ein Sample-Punkt auf der steigenden Flanke liegt. Der Anstieg muss sich folglich mindestens über eine Taktdauer von $6,7\,\mathrm{ns}$ erstrecken. Die grafisch ermittelte Flankenbreite von $7,0\,\mathrm{ns}$ zwischen 0% und 100% der Pulshöhe liegt knapp über diesem Minimalwert, was die Samplefrequenz von 150 MHz als hinreichend bestätigt. Für die übliche Wahl von 10% und 90% liegt die Anstiegszeit bedingt durch die geringe Steigung innerhalb der ersten 10% mit $3,0\,\mathrm{ns}$ jedoch nicht einmal bei der halben Taktzeit. Daher wird nach der Digitalisierung für einige Phasenlagen lediglich ein geringfügiger Unterschied des Samples vor dem Maximum zum Grundniveau erwartet. Die Fallzeit von 90% bis 10% der Pulshöhe wird zu 83 ns bestimmt, was mehr als 12 Taktzyklen entspricht, weshalb zur Rekonstruktion des Schwanzes viele Samplepunkte hinter dem Maximum gespeichert werden müssen.

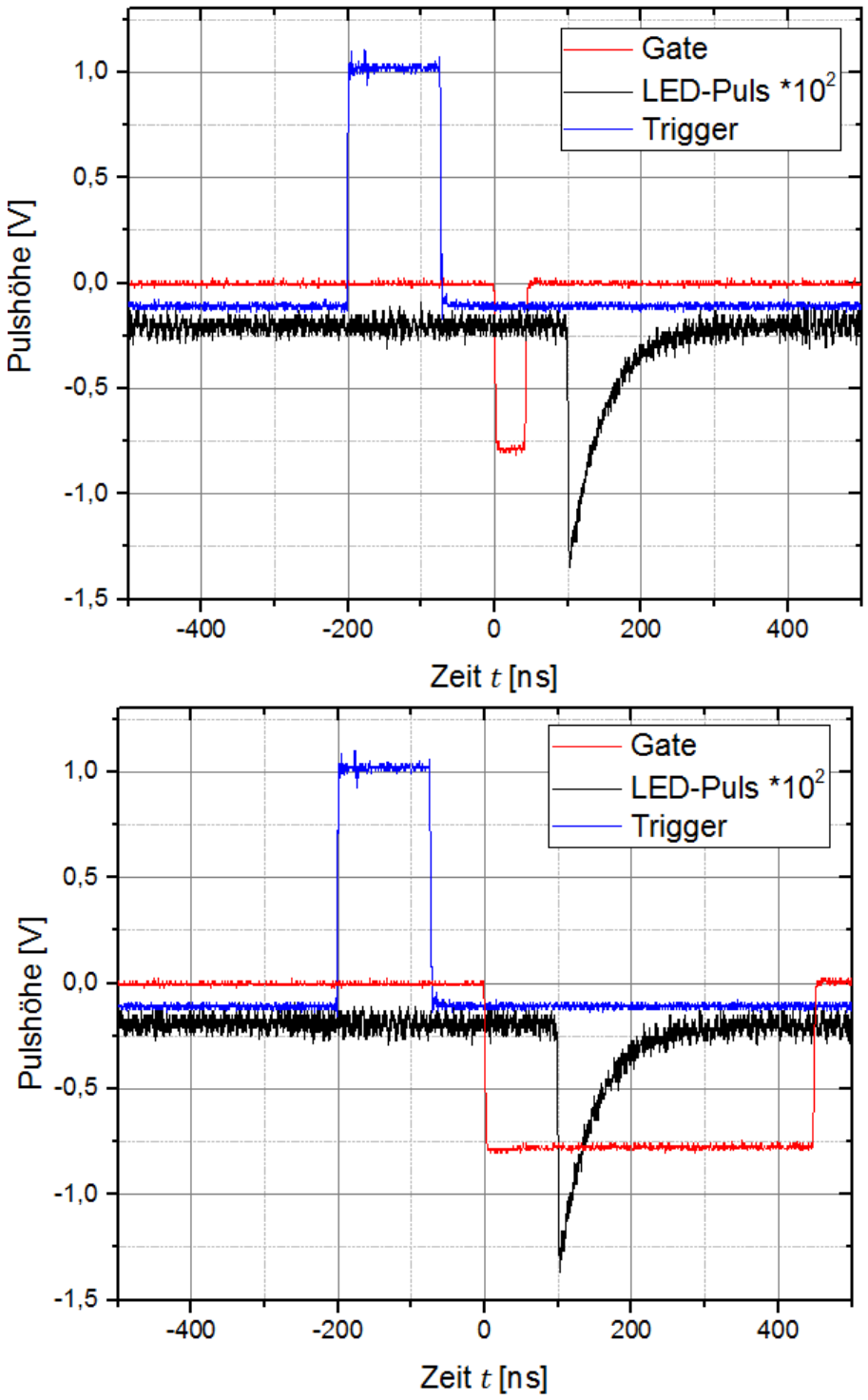

Abbildung 5.4: Triggerpuls, LED-Puls und Ausgangssignal des Ortec GG8010 octal gate and delay generator ausgegeben mit dem Tektronix DPO 7104 Digital Phosphor Oscilloscope. Oben ist die kleinstmögliche, unten eine sehr hohe Pulsweite gewählt. Der LED-Puls bleibt von der Veränderung unberührt.

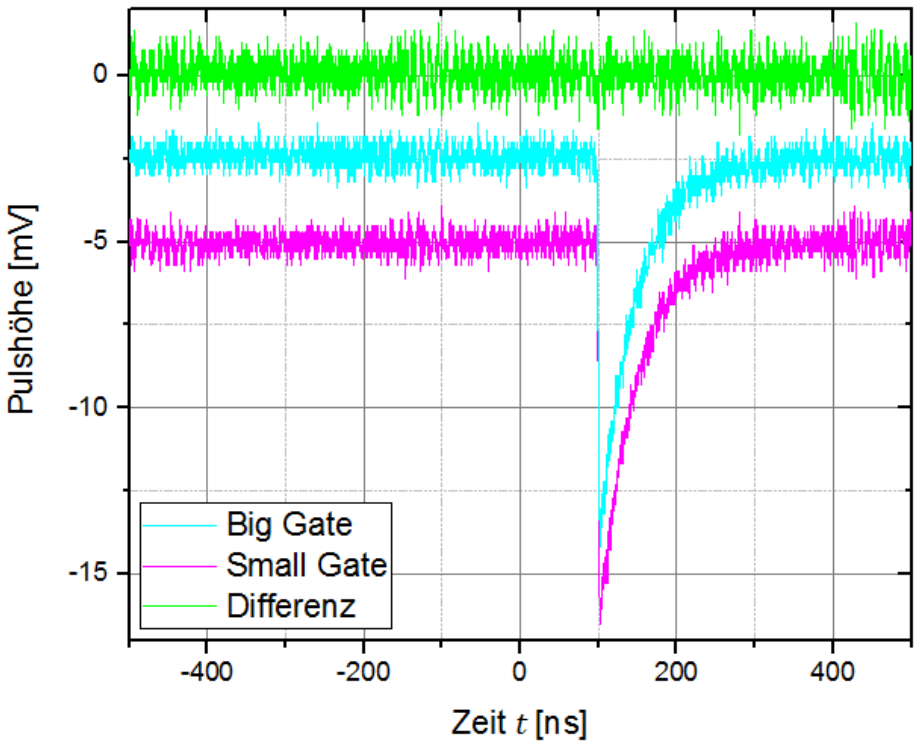

Abbildung 5.5: Die mit schmalem und breitem Triggerpuls (als „Gate“bezeichnet) aufgenommenen LED-Signale aus Abbildung 5.4 werden in der Höhe versetzt übereinandergelegt. Die Differenz der Pulshöhen liegt um 0. Amplitude und Pulsform erweisen sich als gleich.

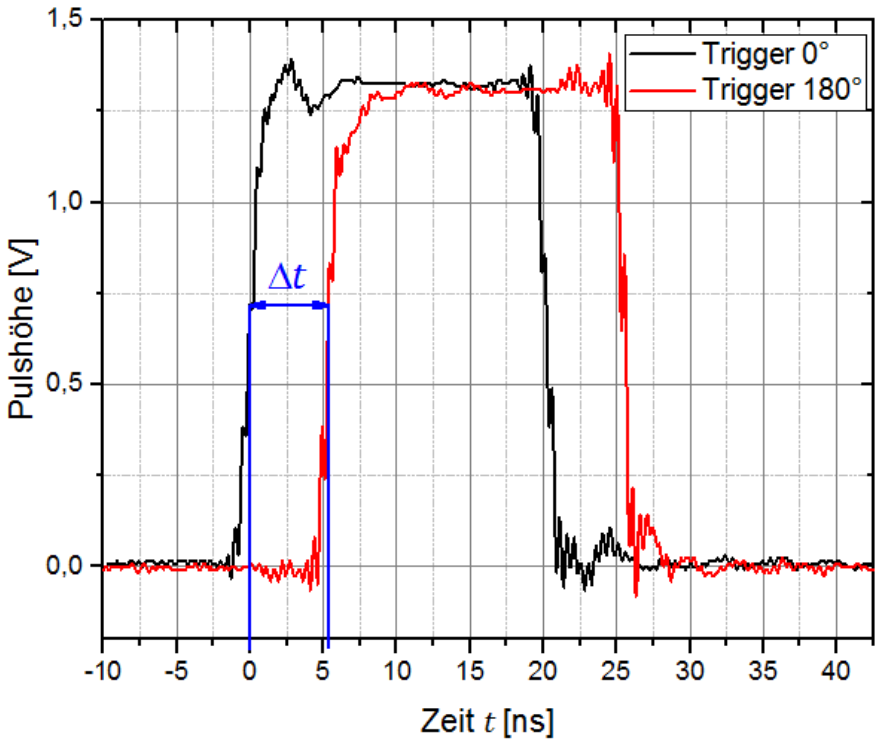

Abbildung 5.6: Die Signale zweier um die halbe Taktdauer verschobenen PLL-Uhren werden übereinander gelegt. Der Versatz Δt der steigenden bzw. fallenden Flanken spiegelt den Phasenunterschied wider.

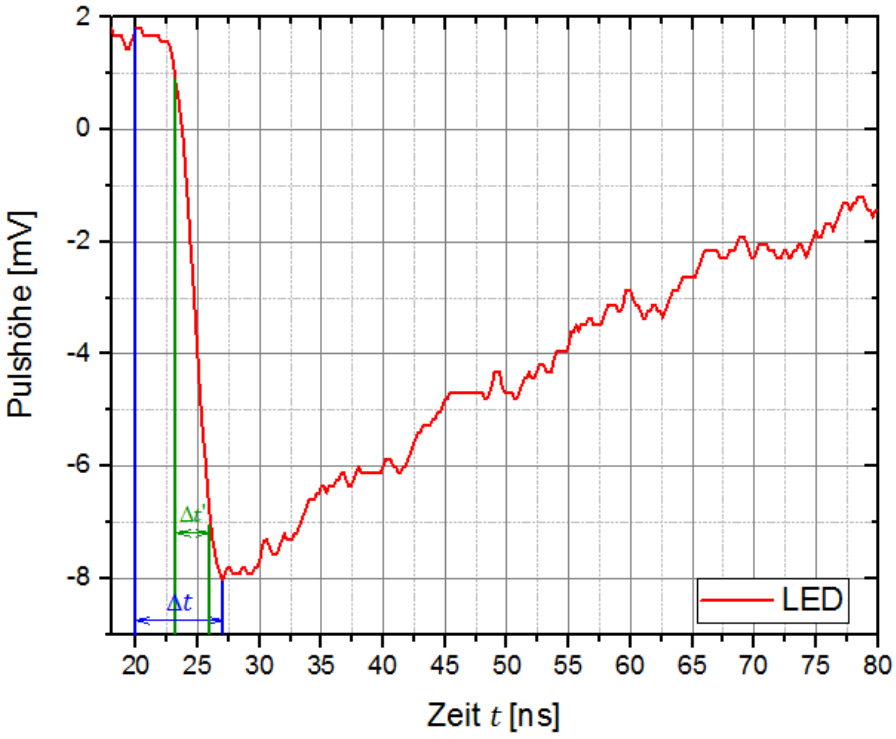

Abbildung 5.7: Analoge Darstellung eines LED-Pulses mit einem Sample-Intervall von $0,2\,\text{ns}$. Die Anstiegszeit von 0% bis 100% der Signalhöhe wird zu $\Delta t = 7,0\,\text{ns}$ bemessen, was oberhalb der Taktlänge von $6,7\,\text{ns}$ des ADCs liegt. Für eine Wahl von 10% und 90% hingegen ergibt sich $\Delta t' = 3,0\,\text{ns}$, was nicht mal einer halben Taktlänge entspricht.

5.3 Bestimmung von E, $E\tau$ und τ

Ziel ist es für die verschiedenen einstellbaren Phasenlagen jeweils die Energiedeposition E und dessen Produkt mit der Phasenlage $E\tau$ zu bestimmen. Durch Division soll hiermit auf die Phasenlage rückgeschlossen werden und die Präzision dieser Angabe festgestellt werden. Hierzu werden zunächst die Idealpulse bestimmt und damit die Filterkoeffizienten berechnet, sodass die gesuchten Werte aus den gefilterten Signalen abgelesen werden können.
Das folgende Vorgehen wird für alle sechzehn einstellbaren Phasenlagen zwischen 0° und 360° in $22,5°$-Schritten durchgeführt. Der aus der PLL kommende Triggerpuls wird an die LED-Treiberbox angeschlossen. Das Detektorausgangssignal wird in den ADC geführt, dort digitalisiert und im FPGA verarbeitet. Es erreicht die beiden Filter, für welche zunächst die Koeffizienten so gewählt sind, dass es sich um Einheitsfilter handelt und das Signal unverändert bleibt. Dann wird der Maximum-Finder auf das Signal angewendet, wobei der Puls als positives Signal behandelt und das eingehende Signal dazu invertiert wird. Alle relevanten Maxima und eine 32-Sample-Umgebung werden in ein Register gespeichert und als Ausgangssignal weitergegeben. Alle weiteren, zwischen den Pulsen liegenden und somit irrelevanten Samples werden verworfen. Die Pulse in Umgebung der Maxima werden per Knopfdruck in den zugewiesenen Speicherbereich geschrieben, bis dieser mit 256 Umgebungen vollständig gefüllt ist. In Abbildung 5.8 sind die Pulse im Speicher exemplarisch für die Phasenlage 0° dargestellt. Mit einer Lesefunktion in Tcl wird in der System Console auf den Speicherbereich

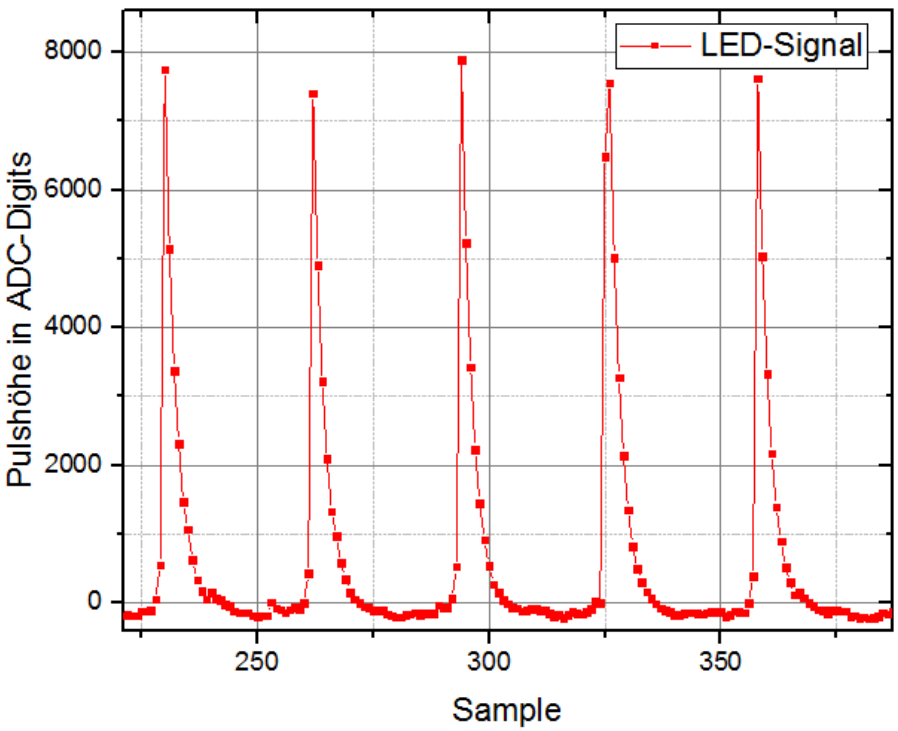

Abbildung 5.8: Digitalisierte LED-Signale aus dem Chip-Speicher, welche nach Einheitsfilter und Maximum Finder auf 32-bit Pulse zusammengeschnitten wurden.

zugegriffen, die Daten als 16bit-Hexadezimalzahlen interpretiert und als solche in ein neu angelegtes Textdokument geschrieben. Die Messung wird wiederholt bis 1000 Maxima und ihre Umgebungen aufgenommen sind. Da die Filtertiefe $t = 5$ gewählt wurde, werden im Folgenden nur noch das Maximum, sowie die zwei linken und rechten Nachbarn betrachtet. Aus den 1000 Pulsen werden für jedes Sample der Mittelwert gebildet und somit ein Idealpuls für die Phasenlage bestimmt. Die Ergebnise für acht verschiedene Phasenlagen sind in Abbildung 5.9 dargestellt. Im Gegensatz zur Reihenfolge der Phasen, kann nicht vorausgesagt werden welches

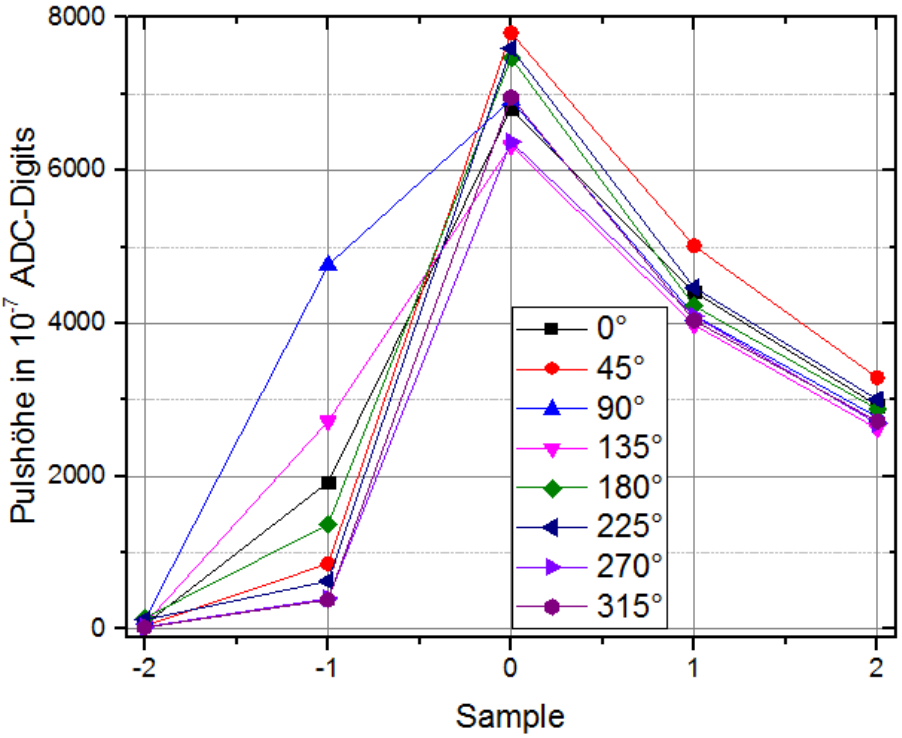

Abbildung 5.9: Idealpulsdarstellung nach Digitalisierung durch Mittelwertbildung der einzelnen Samples um das Maximum (Sample 0) herum für acht verschiedene Phasenlagen.

Sample das jeweils zeitlich Erste ist. Es ist zu erwarten, dass mit Erhöhung der Phasenlage die Sample-Amplitude an der steigenden Flanke ansteigt. Deshalb wird jeweils das Sample vor dem Maximum betrachtet und der dem analogen Signal am besten entsprechende Anstieg gesucht. Auf diese Weise werden die Samples mit 180 °-Phasenverschiebung als frühste Samples erkannt. Mit dieser Voraussetzung wird in Abbildung 5.10 der Idealpuls unter Einfluss aller acht Phasenlagen dargestellt.

Mit der erlangten Kenntnis der Pulsform g ist es nun möglich die Filterkoeffizienten nach Gleichung 3.13 zu bestimmen. Rauschen wird vorerst nicht berücksichtigt und die Autokorrelationsmatrix R deshalb als Einheitsmatrix gewählt. Die berechneten Filterkoeffizienten $\boldsymbol{a} = \sum_{i=1}^{5} a_i$ und $\boldsymbol{b} = \sum_{i=1}^{5} b_i$ sind für acht verschiedene Phasen in Tabelle 5.1 festgehalten und in Abbildung 5.11 vergleichend gegenübergestellt.

Damit diese Koeffizienten im Filtermodul der Firmware verwendet werden können, welches

Tabelle 5.1: Filterkoeffizienten für den E-Filter (oben) und den $E\tau$-Filter (unten) bei acht verschiedene Phasenlagen

Phasenlage in °	$a_1 \cdot 10^8$	$a_2 \cdot 10^8$	$a_3 \cdot 10^8$	$a_4 \cdot 10^8$	$a_5 \cdot 10^8$
0	-149,5	2046,2	8589,4	5907,4	3945,7
45	-29,4	515,2	7819,4	5368,6	3544,7
90	-26,5	671,9	8083,9	5521,6	3661,0
135	-154,0	4803,9	7306,7	4448,6	3004,2
180	-191,5	3535,8	8965,8	5896,7	3902,2
225	39,8	1224,6	8769,2	5313,7	3592,0
270	88,6	328,1	8536,3	5387,0	3615,0
315	-4,6	173,6	9598,1	6597,3	4360,6
Phasenlage in °	$b_1 \cdot 10^8$	$b_2 \cdot 10^8$	$b_3 \cdot 10^8$	$b_4 \cdot 10^8$	$b_5 \cdot 10^8$
0	-8330,3	-14902,1	-4738,1	9264,4	7024,0
45	-2897,7	-13764,8	-6683,1	8530,0	6475,8
90	-11835,3	-8386,7	1264,3	5545,7	3563,8
135	-11813,9	-13659,0	-2112,1	8709,9	6357,5
180	-5080,1	-15114,6	-5112,0	10059,0	5959,1
225	-1996,1	-14734,4	-6793,0	9568,5	6116,0
270	-1955,1	-16696,4	-8859,3	10306,9	7808,8
315	-1669,6	-15955,9	-7578,6	10274,7	6415,8

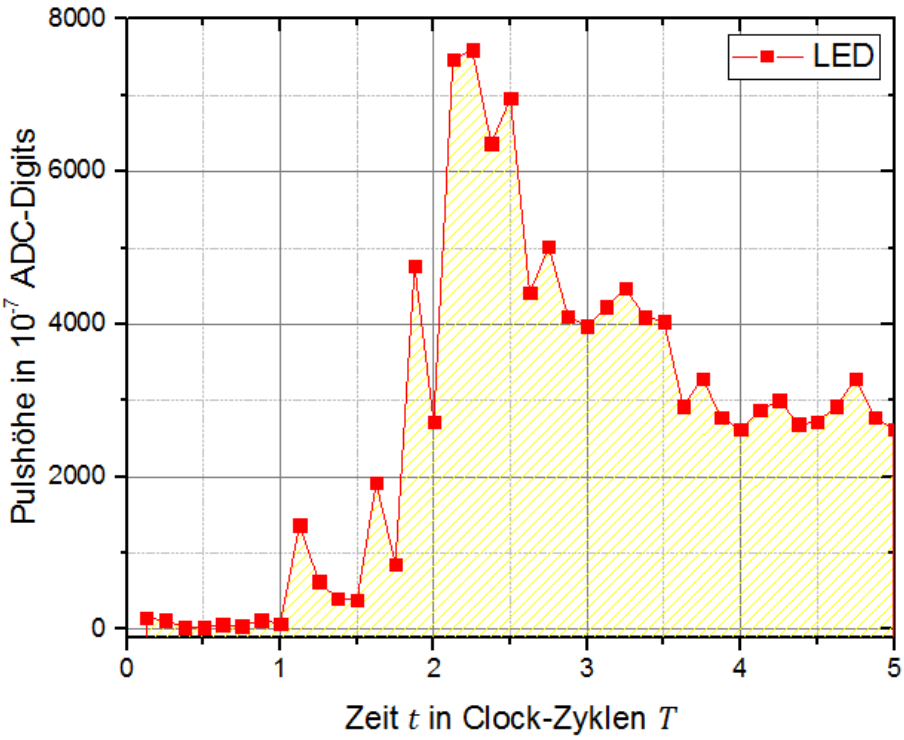

Abbildung 5.10: Idealpulsdarstellung nach Digitalisierung unter Verwendung der Samples in acht verschiedenen Phasenlagen aus Abbildung 5.9

16-bit Integer erwartet, werden sie zunächst umskaliert. Aufgrund der Normierung des Ausgangssignals

$$y = \sum_{i=1}^{5} c_i x_i, \tag{5.1}$$

wobei c für den jeweiligen Filterkoeffizienten a oder b steht, ergeben sich Koeffizienten der Größenordnungen 10^{-4} bis 10^{-8}. Für die Umwandlung in ganze Zahlen bei möglichst hoher Genauigkeit werden sie mit dem Faktor 10^8 multipliziert und gerundet. Damit liegen alle neu skalierten Koeffizienten betragsmäßig unter dem maximal darstellbaren Wert von $2^{15} = 32768$. Diese werden mit einem in der System Console ausgeführten Tcl-Skript in einen für die Filterkoeffizienten reservierten Speicherbereich des FPGAs geschrieben.
Im weiteren Versuchsverlauf werden nacheinander die verschiedenen phasenverschobenen PLL-Signale per Umlegen der vier Kippschalter auf dem Board als Triggersignale an die LED-Treiberbox angelegt. Es werden jeweils die zugehörigen Filterkoeffizienten wie oben erläutert in den Speicher geschrieben und per Knopfdruck ausgelesen und von den Filtermodulen verwendet. In der Signal-Tap-Umgebung werden die ungefilterten Signale, sowie die nach dem E- bzw. $E\tau$-Filter dargestellt. Es wird auf die steigende Flanke des PLL-Signals getriggert und für Einzelmessungen die Signale exportiert. Diese werden zur Bestimmung von E, dem zugehörigen $E\tau$ und damit τ ausgewertet. Dies ist exemplarisch für das um 90° verschobene Signal

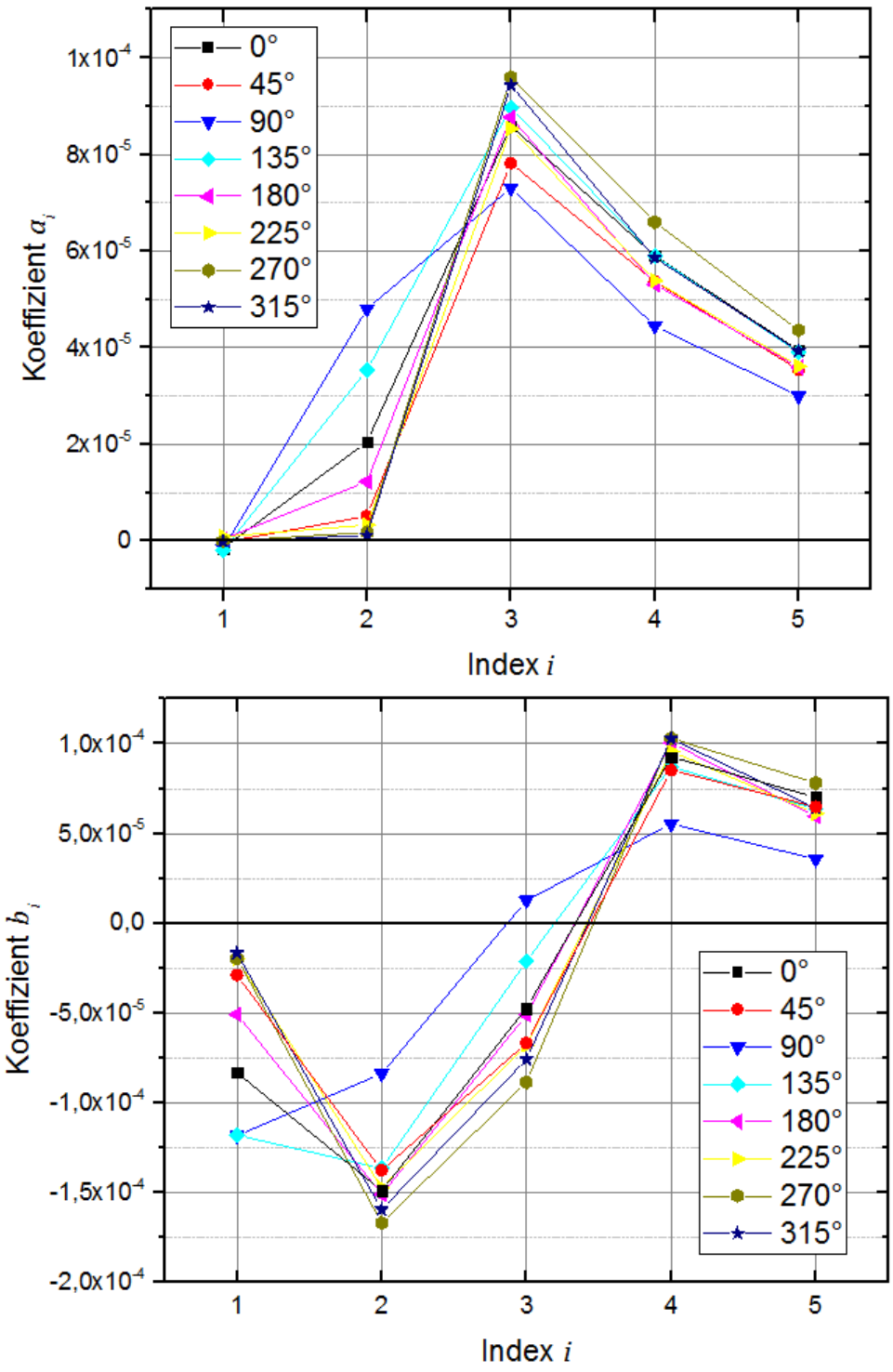

Abbildung 5.11: Die jeweils fünf Filterkoeffizienten für acht verschiedene Phasenlagen des E-Filters (oben) und des $E\tau$-Filters (unten).

in Abbildung 5.12 dargestellt. Die Energiedeposition E entspricht dem Betrag des Extremums

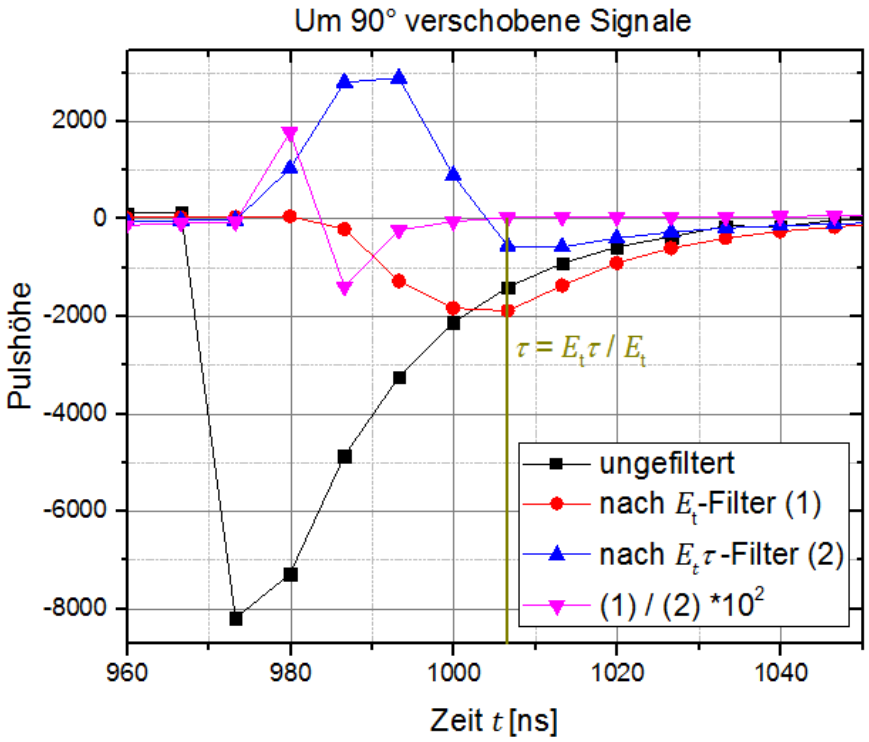

Abbildung 5.12: Bestimmung von τ aus den Signalen des $E\tau$-Filters und des E-Filters am Extremum E für das um 90° verschobene PLL-Signal.

des E-Filter-Signals. Für das gleiche Sample wird der Wert des Signals aus dem $E\tau$-Filter vermerkt und die Phasenlage τ per Division $\frac{E\tau}{E}$ bestimmt. Aufgrund des hohen Rechenaufwandes wird dieses Teilen nicht auf dem FPGA ausgeführt.
Die zeitliche Stabilität von τ wird untersucht, indem für eine Phasenlage 10 Einzelpulse nach obigem Vorgehen ausgewertet werden. Die bestimmten τ für die Phasenlage 0° werden in Abbildung 5.13 gezeigt. Die gemessenen τ variieren stark und zeigen auf, dass eine Analyse die Durchführung einer höheren Zahl an Messungen notwendig macht, um nach 3.8 das Scharmittel zu erhalten.

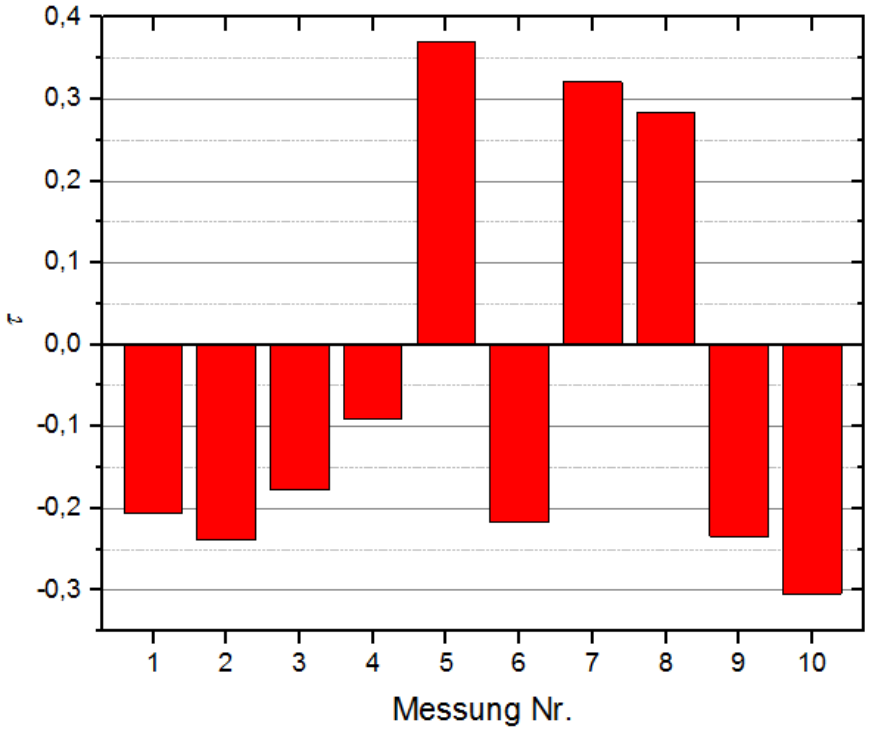

Abbildung 5.13: τ von zehn Einzelmessungen bei einer Phasenlage von 0°. Die großen Unterschiede zeigen die Notwendigkeit einer höheren Anzahl Messungen und einer Auswertung im Mittelwert auf.

6 Ergebnisaufstellung und Ausblick

Ziel der Arbeit war es eine Firmware zu entwickeln, welche die Energiedeposition und die Phasenlage eines gemessenen LED-Pulses rekonstruieren kann. Dieses wurde durch das Implementieren von Optimalfiltern auf dem FPGA-Board ermöglicht. Ein Maximum Finder erkennt die Energiedepositionen und durch das Training zweier Filter mit 1000 Pulsen konnten Filterkoeffizienten bestimmt werden, mit denen $E\tau$ und daraus die Phasenlage τ ermittelt werden kann. Für einzelne Pulse wurde diese Analyse durchgeführt und die Notwendigkeit einer Betrachtung des Scharmittels zur Bestimmung der Phasenlage aufgezeigt werden.

Die Arbeit am Versuchsaufbau kann weitergeführt werden, indem eine Überprüfung der erkannten Phasenlagen durch eine Auswertung des Mittelwerts durchgeführt wird. Hierzu gilt es neben dem bereits vorhandenen Speicherbereich für Samples des E-Filters auch einen für den $E\tau$-Filter anzulegen und die Signalverarbeitung so anzupassen, dass die Samples beider Filter zu gleichen Zeiten einander zugeordnet werden können und in für eine statistische Auswertung hinreichendem Umfang gespeichert werden.
Des Weiteren kann der Aufbau verwendet werden um radioaktive Quellen zu untersuchen, welche ein Signal in einem Szintillator auslösen, das wiederum vom MPPC erkannt wird. Hierbei auftretende Pile-Up-Effekte gilt es zu berücksichtigen.

7 Anhang

7.1 Verwendete Geräte

- LED-Treiber-Box einschließlich LED: CAEN SP5601 LED Driver, S/N 120
- Spannungsquelle: CAEN SP5600 2 channels general purpose amplifier and power supply unit, S/N 109
- SiPM: Hamamatsu MPPC S10362-11-100C, S/N 2810
- System-on-chip: Terasic SoCKit mit AD/DA data conversion card, einschließlich Altera Cyclone V FPGA-Chip 5CSXFC6D6F31FPGA
- Tektronix DPO 7104 digital phosphor oscilloscope
- Ortec GG8010 octal gate and delay generator

7.2 VHDL-Code

```vhdl
library ieee;
   use ieee.std_logic_1164.all;
   use ieee.numeric_std.all;

entity pulse_clock is
   generic (
      speed:  in  natural := 25;
      width:  in  natural := 1
   );
   port (
      clk            : in  std_logic;
      pulse_clock_out: out std_logic
   );
end pulse_clock;


-- ================== PULSE_CLOCK ============================
architecture arch_pulse of pulse_clock is
   signal prev_counter: std_logic_vector(speed downto 0);
   signal counter:      std_logic_vector(speed downto 0);
begin
   process(clk)
   begin
      if rising_edge(clk) then
         prev_counter <= std_logic_vector(unsigned(counter) - width +1);
         counter <= std_logic_vector(unsigned(counter) + 1);
      end if;
   end process;
pulse_clock_out <= counter(speed) xor prev_counter(speed);
end arch_pulse;


--=================== HEARTBEAT ======================
architecture arch_hb of pulse_clock is

  signal counter: std_logic_vector(speed downto 0);

begin

   process(clk)
   begin
      if rising_edge(clk) then
         counter <= std_logic_vector(unsigned(counter) + 1);
      end if;
   end process;
pulse_clock_out <= counter(speed);
end arch_hb;
```

Abbildung 7.1: VHDL-Modul des Pulsgenerators

```
 1 library ieee;
 2    use ieee.std_logic_1164.all;
 3    use ieee.numeric_std.all;
 4
 5 library jtag_mem_pll;
 6    use jtag_mem_pll.all;
 7
 8 library work;
 9    use work.mppc_pkg.all;
10
11 entity max_finder is
12    port (
13      clk           : in  std_logic;
14      reset         : in  std_logic;
15      mykey         : in  std_logic_vector(3 downto 0);
16      pulse_in      : in  std_logic_vector(BITLENGTH_MAX-1 downto 0);
17      write_fir     : out std_logic;
18      writedata_fir: out std_logic_vector(FIR_WRITE_SIZE-1 downto 0);
19      addr_fir      : out std_logic_vector(FIR_ADDR_SIZE-1 downto 0);
20      write_max     : out std_logic;
21      writedata_max: out std_logic_vector(MAX_WRITE_SIZE-1 downto 0);
22      addr_max      : out std_logic_vector(MAX_ADDR_SIZE-1 downto 0);
23      write_cnt     : out std_logic;
24      writedata_cnt: out std_logic_vector(CNT_WRITE_SIZE-1 downto 0);
25      addr_cnt      : out std_logic_vector(CNT_ADDR_SIZE-1 downto 0)
26    );
27 end entity;
28
```

Abbildung 7.2: VHDL-Modul des Maximum Finders (1/3)

```
29 architecture a_coef_arch of max_finder is
30
31 signal x_mem_reg : sample_t;
32
33   begin
34      process(reset, clk) is
35         variable addr_fir_v : integer := BASIS_FIR;
36         variable addr_max_v : integer := BASIS_MAX;
37         variable addr_cnt_v : integer := BASIS_CNT;
38         variable pos_v      : natural := 0;
39         variable pulse_v    : std_logic_vector(FIR_WRITE_SIZE-1 downto 0) := (others => '0');
40         variable cnt_v      : natural := 0;
41         begin
42            if(reset = '1') then
43               x_mem_reg     <= (others => (others => '0'));
44               write_max     <= '0';
45               write_fir     <= '0';
46               writedata_max <= (others => '0');
47               writedata_fir <= (others => '0');
48               addr_max      <= std_logic_vector(to_signed(basis_max,MAX_ADDR_SIZE));
49               addr_fir      <= std_logic_vector(to_signed(basis_fir,FIR_ADDR_SIZE));
50               pos_v         := 0;
51               cnt_v         := 0;
52               pulse_v       := (others => '0');
53            elsif(rising_edge(clk)) then
54               cnt_v := cnt_v+1;
55               -- adress reset to rewrite memory:
56               if mykey(0)='1' then
57                  addr_max_v := BASIS_MAX;
58                  addr_fir_v := BASIS_FIR;
59                  addr_cnt_v := BASIS_CNT;
60                  pos_v      := 0;
61                  cnt_v      := 0;
62               end if;
63               -- shift register for pulsesamples:
64               for i in SPL_NR-2 downto 0 loop
65                  x_mem_reg(i) <= x_mem_reg(i+1);
66               end loop;
67               x_mem_reg(SPL_NR-1) <= pulse_in;
68               -- maximum finder:
69               if (x_mem_reg(3) < x_mem_reg(4)) and (x_mem_reg(4) < x_mem_reg(5))
70               and (x_mem_reg(5) > x_mem_reg(6)) and (x_mem_reg(6) > x_mem_reg(7))
71               and addr_max_v < BYTE_MEM_MAX and addr_fir_v < BYTE_MEM_FIR
72               and (THRESHOLD < to_integer(signed(x_mem_reg(5))))
73               and (to_integer(signed(x_mem_reg(5))) < SATURATION) and (cnt_v > SPL_NR-1) then
```

Abbildung 7.3: VHDL-Modul des Maximum Finders (2/3)

```
68              -- maximum finder:
69              if (x_mem_reg(3) < x_mem_reg(4)) and (x_mem_reg(4) < x_mem_reg(5))
70              and (x_mem_reg(5) > x_mem_reg(6)) and (x_mem_reg(6) > x_mem_reg(7))
71              and addr_max_v < BYTE_MEM_MAX and addr_fir_v < BYTE_MEM_FIR
72              and (THRESHOLD < to_integer(signed(x_mem_reg(5))))
73              and (to_integer(signed(x_mem_reg(5))) < SATURATION) and (cnt_v > SPL_NR-1) then
74                 -- write maximum in memory:
75                 write_max     <= '1';
76                 writedata_max <= x_mem_reg(5);
77                 addr_max_v    := addr_max_v + 1;
78                 addr_max      <= std_logic_vector(to_signed(addr_max_v,MAX_ADDR_SIZE));
79                 -- write count in memory:
80                 write_cnt     <= '1';
81                 writedata_cnt <= std_logic_vector(to_signed(cnt_v,CNT_WRITE_SIZE));
82                 addr_cnt_v    := addr_cnt_v + 1;
83                 addr_cnt      <= std_logic_vector(to_signed(addr_cnt_v,CNT_ADDR_SIZE));
84                 -- write samples in memory:
85                 write_fir <= '1';
86                 for i in 0 to SPL_NR -1 loop
87                    pulse_v(pos_v+15 downto pos_v) := x_mem_reg(i);
88                    pos_v                          := pos_v+16;
89                 end loop;
90                 addr_fir_v    := addr_fir_v + 1;
91                 addr_fir      <= std_logic_vector(to_signed(addr_fir_v,FIR_ADDR_SIZE));
92                 writedata_fir <= pulse_v;
93              end if;
94           end if;   -- reset
95        end process;
96
97  end a_coef_arch;
98
```

Abbildung 7.4: VHDL-Modul des Maximum Finders (3/3)

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  library work;
6     use work.mppc_pkg.all;
7
8  entity fir is
9  port (
10    clk        : in std_logic;
11    reset      : in std_logic;
12    clk_en     : in std_logic;
13
14    coef0      : in signed((COEFF_BIT_LENGTH-1) downto 0);
15    coef1      : in signed((COEFF_BIT_LENGTH-1) downto 0);
16    coef2      : in signed((COEFF_BIT_LENGTH-1) downto 0);
17    coef3      : in signed((COEFF_BIT_LENGTH-1) downto 0);
18    coef4      : in signed((COEFF_BIT_LENGTH-1) downto 0);
19
20    pulse_out  : out signed(15 downto 0);
21    pulse_in   : in signed((BITLENGTH_ADC-1) downto 0)
22 );
23 end entity;
24
25 architecture rtl of fir is
26 signal x                        : signed(COEFF_BIT_LENGTH - 1 downto 0);
27 signal y,ya,yb,yc,yd            : signed(31 downto 0);
28
29 begin
30
31    x <= signed'(resize(pulse_in, 16));
32
33    process(reset, clk) is
34    begin
35    if( reset = '1') then
36       y  <= (others => '0');
37       ya <= (others => '0');
38       yb <= (others => '0');
39       yc <= (others => '0');
40       yd <= (others => '0');
41    elsif( rising_edge(clk)) then
42       if ( clk_en = '1') then
43          ya <= (signed'(resize(x * coef4, 32)));
44          -- ya <= x * coef4;
45          yb <= ya + x * coef3;
46          yc <= yb + x * coef2;
47          yd <= yc + x * coef1;
48          y  <= yd + x * coef0;
49       end if;
50    end if;
51
52     pulse_out    <= y(31 downto 16);
53
54   end process;
55 end rtl;
```

Abbildung 7.5: VHDL-Modul des FIR

7.3 Glossar

ADC - Analog Digital Converter (Analog-Digital-Konverter)
APD - Avalance Photo Diode
ATLAS - A Toroidal LHC ApparatuS (Teilchendetektor am Large Hadron Collider der europäischen Organisation für Kernforschung)

CERN - Conseil européen pour la recherche nucléaire (Europäische Organisation für Kernforschung)

FC - Fiber Connector (Lichtwellenleiter-Steckverbinder-Typ)
FIR - Finite impulse response filter (Transversalfilter)
FPGA - Field programmable gate array

IP - Intellectual Property (Geistiges Eigentum)

LWL - Lichtwellenleiter

PFD - Phase frequency detector (Phasenfrequenzdetektor)
PLL - phase-locked loop (Phasenregelschleife)
pn-Diode - positive negative diode (Diode basierend auf pn-Übergang)
PSAU - Power Supply and Amplification Unit

SMA - Sub-Miniature-A

Tcl - Tool command language
TTL - Transistor-Transistor-Logik

VCO - Voltage Controlled Oscillator (spannungsgesteuerter Oszillator)
VHDL - Very high speed integrated circuit hardware description language
VGA - Video Graphics Array

8 Literaturverzeichnis

[1] Altera. Phase-Locked Loop Basics, PLL. `https://www.altera.com/support/support-resources/operation-and-testing/pll-and-clock-management/pll-basics.html`. Accessed: 2017-05-19.

[2] Robert P Battaline, James R Robinson, Edward H Welbon, and Ralph J Williams. Performance monitoring through jtag 1149.1 interface, June 16 1998. US Patent 5,768,152.

[3] James L Buie. Coupling transistor logic and other circuits, November 1 1966. US Patent 3,283,170.

[4] CAEN. *Digital Pulse Processing For SiPM Kit - Preliminary User's Guide*, 12 2010. Rev. 0.

[5] CAEN. *SP5650 Sensor Holder for SP5600 SiPM Kit Data Sheet*, 11 2011. Rev. 0.

[6] CAEN. *SP5601 Led Driver - Data Sheet*, 12 2014. Rev. 3.

[7] Pong P Chu. *FPGA prototyping by VHDL examples: Xilinx Spartan-3 version.* John Wiley & Sons, 2011.

[8] WE Cleland and EG Stern. Signal processing considerations for liquid ionization calorimeters in a high rate environment. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 338(2-3):467–497, 1994.

[9] M. George Craford. Visible light-emitting diodes: Past, present, and very bright future. *MRS Bulletin*, 25(10), 2000.

[10] Albert Einstein. Über einen die erzeugung und verwandlung des lichtes betreffenden heuristischen gesichtspunkt. *Annalen der physik*, 322(6):132–148, 1905.

[11] Perkin Elmer. Avalanche photodiodes: A user's guide. *Technical information, PerkinElmer Optoelectronics*, 2006.

[12] Stefan H Goßner. Grundlagen der elektronik. *Halbleiter, Bauelemente und Schaltungen. Aachen*, 2008.

[13] Wolfgang J Hess. *Digitale Filter: eine Einführung*. Springer-Verlag, 2013.

[14] Martin Jutisz. Implementierung digitaler Filter als Modell der ATLAS-LAr-Kalorimeter Signalverarbeitung. Bachelorarbeit, TU Dresden, 2016.

[15] Charles Kittel, Jochen Matthias Gress, and Anne Lessard. *Einführung in die Festkörperphysik*, volume 14. Oldenbourg München, 1969.

[16] Hanno Krieger. *Strahlenphysik, Dosimetrie und Strahlenschutz: Band 2: Strahlungsquellen, Detektoren und klinische Dosimetrie*. Springer-Verlag, 2013.

[17] K Lehovec. The photo-voltaic effect. *Physical Review*, 74(4):463, 1948.

[18] Martin Meyer. *Signalverarbeitung: analoge und digitale Signale, Systeme und Filter*. Springer-Verlag, 2009.

[19] Hari Singh Nalwa. *Photodetectors and fiber optics*. Elsevier, 2012.

[20] II Quartus. Handbook version 13.1. *Timing Analysis Overview TimeQuest Terminology and Concepts*, 2012.

[21] Dieter Renker. Geiger-mode avalanche photodiodes, history, properties and problems. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 567(1):48–56, 2006.

[22] K Sato, K Yamamoto, K Yamamura, S Kamakura, and S Ohsuka. Application oriented development of multi-pixel photon counter (mppc). In *Nuclear Science Symposium Conference Record (NSS/MIC), 2010 IEEE*, pages 243–245. IEEE, 2010.

Erklärung

Hiermit erkläre ich, dass ich diese Arbeit im Rahmen der Betreuung am Institut für Kern- und Teilchenphysik ohne unzulässige Hilfe Dritter verfasst und alle Quellen als solche gekennzeichnet habe.

Nick Fritzsche
Dresden, Mai 2017