Julius-Maximilians-Universität Würzburg

Bachelorarbeit

über das Thema

- COSMOPALS -

Component Simulation Model for Pulse-Shape Analysis in Positron Lifetime Spectroscopy

Autor: Michael Fischer

Prüfer: Prof. Dr.-Ing. Sergio Montenegro
Dr. habil. Torsten Staab

Betreuer: Danny Petschke

Abgabedatum: 31.07.2017

I Abstract

Diese Arbeit widmet sich der Erstellung eines computergestützten Simulationsmodells, mit welchem Hardwarekomponenten eines Photomultipliers zur gezielten Veränderung der aus der Positronen-Annihilations-Spektroskopie resultierenden Pulsform simuliert, analysiert und getestet werden können.

In den folgenden Kapiteln erfährt der Leser anfangs eine Einführung in die physikalischen Grundlagen der Positronen-Annihilations-Spektroskopie (PAS) und die Vorgänge in einem Annihilationsspektrometer. Anschließend werden die Programme *DDRS4PALS* und *COSMOPALS* vorgestellt.

Das Hauptaugenmerk liegt dabei auf der entwickelten Simulationssoftware. Sie erlaubt es dem Benutzer, verschiedene elektronische Bauteile auf unterschiedliche Weise miteinander zu kombinieren. Außerdem können vom Nutzer gestellte Anforderungen an Pulscharakteristika adaptiv und autonom umgesetzt werden. Abschließend wird das zugrundeliegende elektrotechnische Modell anhand realer Hardware auf einem Entwicklungsboard aufgebaut und analysiert.

II Inhaltsverzeichnis

III Abbildungsverzeichnis

IV Tabellenverzeichnis

V Algorithmen-Verzeichnis

VI Abkürzungsverzeichnis

ADC Analog Digital Converter

APD Avalanche Photodiode

API Application Programming Interface

ATS Arrival Time Spread

BMBF Bundesministerium für Bildung und Forschung

CFD Constant Fraction Discriminator

CSV Comma Separated Values

DLL Dynamic Link Library

DPALS Digital Positron Annihilation Lifetime Spectroscopy

DRS Domino Ring Sampler

FWHM Full Width at Half Maximum

GNU GNU's Not Unix

GUI Graphical User Interface

HW Hardware

IDE Integrated Development Environment

IRF Instrument Response Function

ISO International Organization for Standardization

JDK Java Development Kit

LE Disc Leading Edge Discriminator

LT Linear Technology

MCA Multichannel Analyzer

MELT Maximum Entropy for LifeTime Analysis

MinGW Minimalist GNU for Windows

MOSFET Metal–Oxide–Semiconductor Field-Effect Transistor

PAS Positronen-Annihilations-Spektroskopie

PALS Positronen-Annihilations-Lebensdauer-Spektroskopie

PHA Pulse-Height Analysis

PHS Pulse-Height Distribution

PMT Photomultiplier Tube

STL Standard Template Library

SW Software

TAC Time-to-Amplitude Converter

TTS Transit Time Spread

USB Universal Serial Bus

XML Extensible Markup Language

1 Einleitung

„Kristalle sind feste Körper mit dreidimensional periodischer Anordnung der räumlichen Bausteine[1] “.

Dass diese Körper dabei perfekt gleichmäßig strukturiert sind, ist eine Idealvorstellung. Abweichungen von dieser Idealvorstellung resultieren in Unregelmäßigkeiten im Atomgitter und werden anhand ihrer Geometrie oder Dimensionalität allgemein in verschiedene Typen klassifiziert. Die in dieser Arbeit betrachteten Unregelmäßigkeiten beschränken sich dabei auf atomare Fehlstellen mit Größenordnungen von wenigen Atomdurchmessern, die sogenannten Punktdefekte.

Defekte sind im Alltag dabei ständig präsent. Sie begrenzen beispielsweise die Schadstoffemission moderner Verbrennungsmotoren durch chemische Reaktionen im Katalysator des Abgassystems. In elektrischen Schaltkreisen ermöglichen sie durch Substratdotierungen in Metall-Oxid-Halbleiter-Feldeffekttransistoren (MOSFETs) platzsparende und energieeffiziente Chips, welche die Grundlage unserer heutigen Computer- und Informationstechnologie darstellen.

Die Konzentration von Defekten in Werkstoffen beeinflusst dabei wesentliche Materialeigenschaften - unter anderem auch die Festigkeit. So kann ein wiederholt belastetes Bauteil aufgrund von Defektakkumulation versagen, obwohl die Belastung noch weit diesseits der Elastizitätsgrenze liegt. Gerade bei sicherheitskritischen Bauteilen ist es deshalb wichtig, die Anzahl solcher Fehlstellen (und somit, in enger Korrelation, die Güte des Materials) mit höchster Genauigkeit zu bestimmen, um gefährlichem - und wie im Falle des ICE-Unfalls Eschede sogar tödlichem - Materialversagen vorzubeugen [HO13].

Erfassbar ist die Defektdichte in Festkörpern unter anderem mit dem Verfahren der Positronen-Annihilations-Spektroskopie (PAS), einer nicht-destruktiven Messmethode, welche die bei der Annihilation von Positronen und Elektronen entstehende elektromagnetische Strahlung nutzt. Per PAS können sowohl Fehlstellen als auch deren Agglomerationen (z.B. nach Strahlenschäden durch hochenergetische Teilchenstrahlung im Weltall) detektiert werden.

Die Positronen-Annihilations-Spektroskopie ist anderen Methoden (wie bspw. dem Wöhler-Zugversuch) in Genauigkeit, Zeitaufwand und Energiebedarf weit überlegen. Mit der überwiegenden Digitalisierung der Messstrecke wurden die Ungenauigkeiten und Fehleranfälligkeiten analoger Geräte zudem weitgehend reduziert. Es ist jedoch nach wie vor mühsam, einzelne Komponenten der Spektrometerelektronik zu verändern, um die Aus-

[1] Quelle: Hugo Strunz, Mineralogische Tabellen: [Str58]

gangsresultate gezielt beeinflussen zu können.

In dieser Arbeit soll deshalb eine Simulationssoftware entwickelt werden, mit deren Hilfe die Vorgänge in einer der essentiellen Komponenten eines Annihilationsspektrometers, dem Photomultiplier, auf verschiedene Weise simulierbar und analysierbar gemacht werden sollen. Die folgenden Abschnitte beschreiben zunächst die physikalischen Grundlagen von Materialdefekten und Paarannihilationen in Kristallgittern und geben einen Überblick über das Messverfahren und die damit erreichbaren Möglichkeiten. Die anschließenden Kapitel beschäftigen sich mit verschiedenen Arten der Pulssimulation: Einerseits werden die vom Spektrometeraufbau erzeugten Lebensdauerspektren anhand einer Log-Normal-Verteilung simuliert. Zum anderen wird ein zunächst theoretisches, mathematisches Modell entworfen, welches als Ersatzschaltbild für einen Photomultiplier im Signalweg des Spektrometers dient. Anhand verschiedener elektrotechnischer Schwingkreise werden die Auswirkungen von Änderungen der elektronischen Komponenten im Modell auf die Lebensdauerpulsgeneration durch die Photomultiplier untersucht.

Anschließend wird versucht, das theoretische Modell durch eine elektrische Schaltung in die Praxis zu überführen. Dabei werden die Ansätze aus Arbeit [Tan08] aufgegriffen und erweitert. Die grundlegenden Hardwarekomponenten eines Photomultipliers sollen auf diese Weise zuerst berechnet und nachfolgend modelliert und simuliert werden. Abschließend werden die mit Soft- und Hardware erzielten Ergebnisse verglichen und analytisch diskutiert.

2 Physikalische Grundlagen

2.1 Defekte in Festkörpern

Festkörper werden anhand ihrer atomaren oder ionischen Bestandteile charakterisiert. Bei deren Aufbau werden im wesentlichen zwei Arten unterschieden: Zum einen gibt es den regellosen, amorphen Aufbau, wie er typischerweise bei Gläsern zu finden ist. Dem gegenüber steht der strukturierte, kristalline Aufbau, bei dem sich die Atomanordnung periodisch wiederholt. Nach dem Prinzip der Energieminimierung folgt, dass alle Elemente mit hauptsächlich ungerichteten Metall- oder Ionenbindungen kristallin aufgebaut sind, wobei die Kristallgitter dabei beliebig komplexe Formen annehmen können. Die Struktur des Kristalls beeinflusst wesentliche Werkstoffeigenschaften, wie beispielsweise Schmelztemperatur oder Zug- und Druckfestigkeit [Boh95].

Gitterfehler sind, wie eingangs bereits erwähnt, verschiedendimensionale Abweichungen von der idealen Kristallstruktur. Man unterscheidet im Wesentlichen zwischen nulldimensionalen Punktdefekten, eindimensionalen Liniendefekten, zweidimensionalen Grenzflächendefekten und dreidimensionalen Volumendefekten. Die „Nulldimensionalität“ von Punktdefekten resultiert dabei aus der Tatsache, dass ihnen Punktform durch die Ausdehnung eines einzelnen Atoms zugeschrieben wird. Man gliedert sie zu Zwischengitteratomen, Fremdatomen (substitutionell oder interstitiell) und Leerstellen, die hier im Folgenden betrachtet werden sollen. Punktdefekte sind dem Material dabei inhärent, sie entstehen bei der Kristallbildung selbst oder werden nachträglich, beispielsweise durch Radioaktivität oder Temperaturänderung, erzeugt [Boh95].

Als Leerstelle bezeichnet man einen freien Gitterplatz, der in einem regulären Gitter besetzt wäre. Die Anzahl der Leerstellen N_V in einem Kristall steigt mit der Temperatur und kann für eine bestimmte Materialmenge mit insgesamt N Gitterplätzen unter Gleichgewichtsbedingungen im Allgemeinen durch die nachfolgende Gleichung beschrieben werden.

$$N_V = N\, exp\,(-\frac{Q_V}{k\,T}) \tag{2.1}$$

Dabei bezeichnet T die Temperatur, k die Boltzmann-Konstante ($1,38 \times 10^{-23}$ JK^{-1}) und Q_V die Energie, die zur Erzeugung einer Leerstelle benötigt wird. Es besteht also ein exponentieller Zusammenhang zwischen der Temperatur und der Leerstellenanzahl. Während Reinstmetalle heutzutage mit einer Fremdatomkonzentration von weniger als 10^{-7} pro Atom erhältlich sind, liegt das Verhältnis von Leerstellen zu Gitterplätzen $\frac{N_V}{N}$ unterhalb der Schmelztemperatur bei den meisten Materialien im Bereich 10^{-4} [CR12].

2.2 Positronen und -lebensdauerspektroskopie

Positronen (e^+) sind die Antiteilchen zu Elektronen, von welchen Sie sich im Wesentlichen durch ihre positive Ladung, ihr magnetisches Moment und ihre kurze Lebensdauer unterscheiden. Positronen entstehen auf dem Anwendungsgebiet der Annihilationsspektroskopie durch radioaktiven β^+-Zerfall, außerdem bei hochenergetischen Proton-Proton-Reaktionen und beim Zerfall positiver Myonen aus kosmischer Strahlung.

Protonenreiche Nuklide zerfallen über den β^+-Prozess. Dabei wird ein Kernproton in ein Neutrino überführt, was zur Emission eines Positrons und eines Elektronen-Neutrinos führt. Aufgrund der Verringerung der Kernladungszahl um 1 geht das zerfallende Element im Zuge des Prozesses in seinen Vorgänger im Periodensystem über. Als Positronenquelle wird im Laboraufbau das künstliche Isotop ^{22}Na verwendet, da es bei akzeptabler Halbwertszeit eine gute Positronenausbeute verspricht [SW92].

Bei der Zerfallsreaktion geht also Natrium unter Aussendung eines Positrons und eines Elektronen-Neutrinos zu Neon über, was durch folgende Reaktionsgleichung beschrieben werden kann:

$$^{22}Na \longrightarrow {}^{22}Ne + \gamma + \beta^+ + \nu_e \tag{2.2}$$

Das zusätzlich in der Gleichung enthaltene γ-Quant resultiert aus der Tatsache, dass das bei der Zerfallsreaktion entstandene Neon-Isotop sich anfangs in einem angeregten Zustand befindet und nach 3.7ps unter Aussendung eines γ-Quants mit einer Energie von 1274 keV in den Grundzustand relaxiert. Dieses Quant wird auch als Start-Quant bezeichnet, da es als Trigger zum Start der Lebenszeitmessung im Spektrometer benutzt wird. Wie Abbildung 2.1 zeigt, ist auch eine direkte Umwandlung des Natrium-Isotops in den Grundzustand möglich. Dies ist statistisch jedoch weniger wahrscheinlich [KRL99].

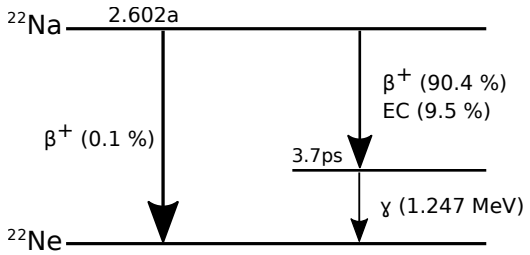

Abb. 2.1: Zerfallsschema des ^{22}Na-Isotops unter β^+-Zerfall [KRL99].

Nach Emission des Positrons wird dieses entweder elastisch rückgestreut oder tritt in das Probensubstrat ein, wo es innerhalb weniger Picosekunden durch unelastische Stoßprozesse seine kinetische Energie verliert und thermisches Gleichgewicht erreicht. Nach diesem Vorgang diffundiert das Positron durch das Kristallgitter, bis es aufgrund des

defektinhärenten Elektronegativitätsgradienten in einer Leerstelle gefangen wird. Dort annihiliert es unter Aussendung zweier fast entgegengesetzt gerichteter γ-Quanten der Energie 511 keV, welche das Stop-Signal für die Spektrometerelektronik darstellen, mit einem lokalen Elektron. Die Distanz, die dabei im Substrat zurückgelegt wird, bestimmt die Anzahl der auf Leerstellen getesteten Atome. Sie liegt üblicherweise im Bereich von einigen hundert Nanometern [KRL99].

Die Positronenlebensdauer τ steht in reziproker Beziehung zur Annihilationsrate λ, welche folgendem Zusammenhang genügt:

$$\lambda = \frac{1}{\tau} = \pi r_0^2 c \int |\psi^+(\boldsymbol{r})|^2 n_-(\boldsymbol{r}) \gamma \, d(\boldsymbol{r}) \tag{2.3}$$

Dabei bezeichnet r_0 den Elektronenradius, $\boldsymbol{r}$ den Positronenvektor, c die Vakuumlichtgeschwindigkeit und γ eine Korrelationsfunktion, die den durch die Coulombkraft zwischen Positron und Elektron bedingten Anstieg in der lokalen Elektronendichte beschreibt [KRL99]. Gängige Werte für Positronenlebensdauern bewegen sich je nach verwendetem Material in Größenordnungen von einigen Picosekunden (Metalle, metallische Legierungen) bis hin zu über einhundert Nanosekunden (Gläser, Elastomere) [RN79].

Mit der möglichst genauen Erfassung solcher Lebensdauern beschäftigt sich die Positronen-Annihilations-Lebensdauer-Spektroskopie (PALS). Dazu werden geeignete Detektoren und entsprechende Spektrometeraufbauten benötigt, welche im folgenden Abschnitt kurz thematisiert werden sollen.

2.3 Photomultiplier und Spektrometeraufbau

Die Lebensdauer eines Positrons wird anhand der Zeitdifferenz zwischen der Detektion des bei der Positronenentstehung erzeugten 1274 keV Start-Quants und der Registrierung eines der beiden aus der Annihilation stammenden 511 keV Stop-Quanten bestimmt. Da diese hochenergetischen Quanten mit wenigen Picometern sehr kurze Wellenlängen besitzen, ist ein Einzelphotonennachweis mit der benötigten Zeit- und Energieauflösung ohne weiteres nicht möglich. Man bedient sich deshalb Szintillatorkristallen und Photomultipliern, die das anfänglich schwache Signal verstärken und so eine messbare Spannung erzeugen.

Photomultiplier

Ein Photomultiplier besteht im Wesentlichen aus einer Photokathode und einer nachfolgenden Widerstandskaskade in einem evakuierten Glaskolben. Nach Auftreffen der Photonen auf die Photokathode lösen sich durch photoelektrische Effekte Elektronen aus

deren Oberfläche. Durch ein anliegendes elektrisches Feld beschleunigt, treffen diese auf Dynoden, aus denen mit einem Sekundäremissionsverhältnis von $\delta = 3...10$ weitere Elektronen herausgeschlagen werden. Die Dynodenspannung wird dabei in Abhängigkeit der Widerstandskaskade vom Eintrittsfenster weggehend vergrößert und liegt meist im Bereich von wenigen kV [HP07].

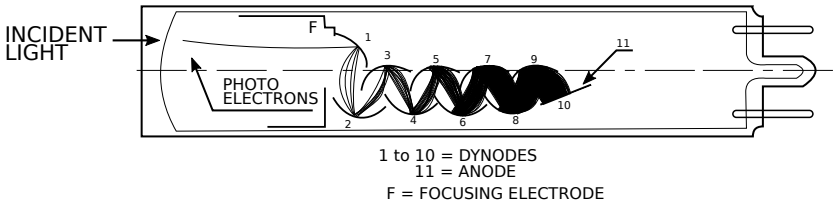

Abb. 2.2: Ein linear-fokussierter Photomultiplier mit 11 Stufen [HP07].

Die Elektronenanzahl steigt dabei exponentiell mit der Anzahl der Dynoden: Hat ein Photomultiplier ein Sekundäremissionsverhältnis δ und n Dynodenstufen, so ergibt sich die Anzahl der an der Anode abfließenden Elektronen zu δ^n. Die ausgehende Spannung ist proportional zur Energie des eingehenden Quants. Sie ähnelt in ihrer Form einer Normalverteilung, wie Abbildung 2.3 exemplarisch zeigt. Das Vorzeichen der Spannung ist dabei nicht von Bedeutung.

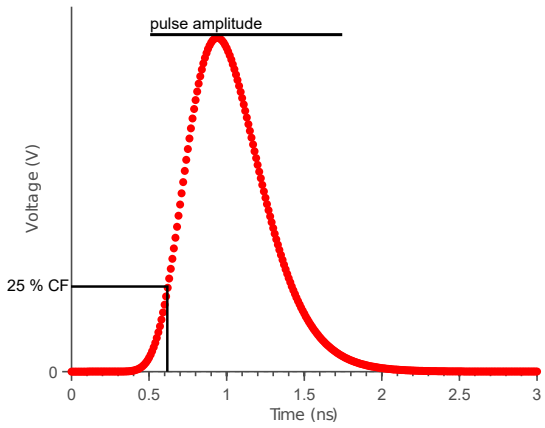

Abb. 2.3: Schematische Darstellung eines Annihilationspulses mit CF-Intervall 25%

Bei den im Laboraufbau verwendeten Photomultipliern handelt es sich um XP2020-Geräte der Firma Philips mit 12 linear angeordneten Verstärkerstufen, plan-konkavem Eintrittsfenster und einer halbtransparenten, bialkalischen Photokathode. Die Geräte zeichnen sich durch wenig Hintergrundrauschen, schnelle Reaktionszeiten, gute Linearität und einen maximalen Verstärkungsfaktor von ca. 2×10^8 aus [Phi07].

Spektrometeraufbau

Frühe Annihilationsspektrometer arbeiteten überwiegend analog, während der Großteil der damals verwendeten Hardware heute durch spezielle Software ersetzt wird. Anforderungen an moderne Geräte sind mitunter große Bandbreiten und schnelle Sampleraten, da die generierten Lebensdauerpulse im Nanosekundenbereich liegen. Abbildung 2.4 zeigt ein Blockschaltbild eines klassischen Messaufbaus (links) und des hier verwendeten Laboraufbaus (rechts).

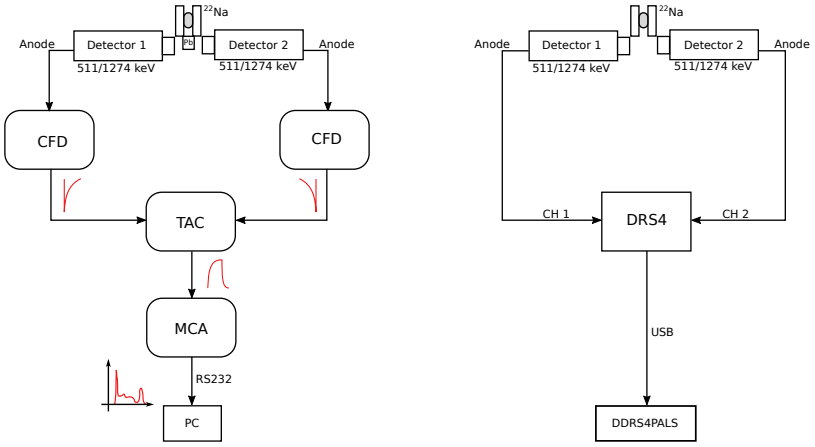

Abb. 2.4: Doppeldetektoraufbau und Laboraufbau mit DRS4-Board.

Die im klassischen Aufbau verwendeten Constant Fraction Discriminators (CFDs) ermöglichen es, exakte Zeitmarken aus breiten, „verwaschenen“ Szintillatorsignalen zu erhalten, indem stets bei einem bestimmten Bruchteil der Maximalamplitude ausgelöst wird. Möglich wird dies durch die Tatsache, dass die Szintillatorpulse zwar variable Pulshöhen, aber konstante Anstiegszeiten besitzen. Durch Addition eines Bruchteils des Originalsignals und einer invertierten, um $t_{delay} < t_{rise}$ zeitverzögerten, Replik erzeugt der CFD eine alternierende Spannungskurve. Der erste Nulldurchgang dieser Spannungskurve mit positiver Tangentensteigung ist ein konstanter Bruchteil des ursprünglichen Szintillatorpulses. So können trotz der Tatsache, dass die Anstiegszeiten des Szintillatorsignals meist über der gewünschten Zeitauflösung liegen, exakte Zeitpunkte bestimmt werden. CFDs werden deshalb als Trigger für Pulse, die durch γ-Quanten mit einer Energie über einem bestimmten Treshold verursacht wurden, verwendet (meist rund $400keV$). Sie ermöglichen zudem die Vermeidung von Messverfälschung durch zufällig eingefallene Photonen und (rück-) gestreute Quanten sowie die Unterscheidung von Start- und Stopsignalen [Pag61].

Die normierten Ausgangspulse der CFDs werden anschließend in den nachgeschalteten Time-to-Amplitude Converter (TAC) geführt. Dort wird, beginnend mit dem Startsignal, ein Kondensator durch eine Konstantstromquelle aufgeladen, was zu einem linearen Anstieg der Kondensatorspannung führt. Nach Eingang des Stopsignals wird der Kon-

densator entladen. Es entsteht also ein direkter Zusammenhang zwischen Ladezeit (und respektive dem Zeitintervall zwischen Start- und Stopdetektion) und Spannungsamplitude [Kno10].

Der durch die Kondensatorentladung erzeugte Puls wird abschließend von einem Multichannel Analyzer (MCA) aufbereitet und von einem reaktionsschnellen ADC digitalisiert. Spektroskopieanwendungen nutzen dazu meist ein Verfahren namens Pulse-Height Analysis (PHA), welches die Pulse anhand ihrer Amplitude charakterisiert und zählt. Der durch den ADC errechnete Gleitkommawert wird einer Speicheradresse zugewiesen, die mit einem bestimmen Kanal des MCA korreliert. Werden nun die Anzahl der registrierten Amplituden über die einzelnen Kanäle aufgetragen, entsteht ein Pulshöhenspektrum. Da die Photomultiplierspannung der Energie der eingehenden Quanten proportional ist, liefert das Pulshöhenspektrum direkte Informationen über die Energieverteilung der Annihilationsereignisse [BB63].

Der hier benutzte Laboraufbau ist hingegen nicht auf externe Hardware angewiesen, da die digitalisierten Pulse der Photomultiplier direkt in das DRS4-Board gespeist werden. Ein wesentlicher Vorteil dieser Anordnung ist, dass sich die Instrument Response Function (IRF) nun auf die Einflüsse des Photomultpliersystems beschränkt und dass die Kalibrierung nicht, wie beim klassischen Aufbau, Stunden oder Tage dauert, sondern in wenigen Minuten vollzogen werden kann.

Das DRS4-Board besitzt eine analoge Bandbreite von 700MHz und eine Schreibgeschwindigkeit von 5.12×10^9 samples per second. Die maximale Zählrate liegt bei ca. 500 Frames pro Sekunde, wobei ein Frame 4×1024 gemessene Werte repräsentiert. Pro Triggerevent werden vier, je 200ns lange, Spannungskurven ausgegeben, die ein interner ADC zu Gleitkommazahlen umrechnet und per USB an den angeschlossenen PC sendet, wo sie von dem Programm *DDRS4PALS* weiterverarbeitet werden [PSS14].

2.4 Lebensdauerspektren

Für das Lebensdauerspektrum einer Positronenquelle gilt allgemein:

$$N(t) = \sum_{i=1}^{k+1} \frac{I_i}{\tau_i} \, exp\left(-\frac{t}{\tau_i}\right) \tag{2.4}$$

Dabei bezeichnen τ und I die individuellen Lebensdauern und Intensitäten. k gibt die Anzahl der im Material enthaltenen Defekttypen an, wobei in praktischen Anwendungen ohne Einschränkungen an die Annihiliationsparameter maximal drei Komponenten auflösbar sind. Die Lebensdauern zeichnen sich im resultierenden Spektrum bei logarith-

mischer Auftragung als Geraden ab (vgl. Abb. 2.5).

Ferner wird angenommen, dass sich ein reales Lebensdauerspektrum zusammen mit den erhaltenen Mess- oder Simulationswerten aus einer Faltung verschiedener Einfluss- und Störterme ergibt, die wiederum als normalverteile Gauß-Funktion approximiert werden können. Störterme sind beispielsweise Unsicherheiten in der Zeitauflösung der Szintillatoren, der Photomultiplier und des DRS4-Boards oder dem Ankunftszeitpunkt der Positronen (ATS), die Intensität des Hintergrundrauschens oder die Wahrscheinlichkeit für eine Zufallsdetektion [RN79]. Zusammengefasst bilden sie die Störfunktion:

$$G(t) = \frac{1}{\sqrt{\pi}\sigma}\, exp\left(-\left(\frac{t-t_0}{\sigma}\right)^2\right) \tag{2.5}$$

Somit ergibt sich für das reale Lebensdauerspektrum folgende Beziehung:

$$I(t) = \int_{-\infty}^{+\infty} G(t-t')N(t')\,dt' \tag{2.6}$$

Abbildung 2.5 verdeutlicht am Beispiel eines Siliziumkristalls, wie die Annihilationslinien in einem Lebensdauerspektrum in der Regel verteilt sind: Die einzelnen Lebensdauern τ_1 und τ_2 treten auf der logarithmischen Ordinate als Geraden auf, die Auflösungsfunktion links des Amplitudenmaximums ähnelt einer Normalverteilung.

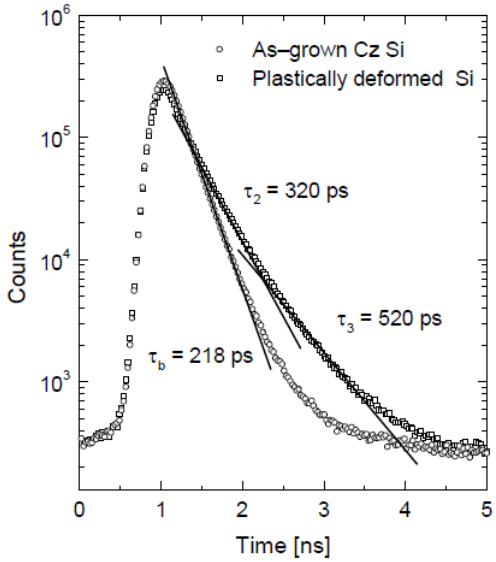

Abb. 2.5: Spektrallinien eines Czochralski-Siliziumkristalls [KRL99].

3 Softwareseitige Simulation

Der in Kapitel 2 vorgestellte Messaufbau soll nun obsolet gemacht werden. Die folgenden Abschnitte widmen sich der Frage, inwiefern man real erzeugte Lebensdauerpulse durch softwaregestützte Simulationstechniken nachbilden kann. Hierzu wird ein bereits existentes Pulsformsimulationsverfahren vorgestellt, welches sich einer Verteilungsfunktion bedient. Außerdem wird ein neues Verfahren präsentiert, welches die Pulsform mittels Variation der elektronischen Komponenten in einer Photomultiplierschaltung beeinflusst.

3.1 Bisherige Simulationsmethodik

Bisher entwickelte Softwarepakete beschäftigen sich weniger mit der Simulation von Pulsformen als vielmehr mit deren Analyse. So existieren bereits zahlreiche Programme wie Pals*fit* und Maximum Entropy for LifeTime Analysis (MELT), die mit verschiedenen Fit-Routinen, wie etwa dem Levenberg-Marquardt-Algorithmus oder der Maximum-Entropy-Methode, Parameter wie Lebensdauer oder Intensität aus Annihilationsspektren extrahieren.
Im Vergleich dazu ist die Anzahl der Simulationsprogramme, die sich mit der Erzeugung von Lebensdauerpulsen beschäftigen, wesentlich geringer. Ein Beispiel hierfür ist die von S. Eichler entwickelte Software *lt_sim*: Das Programm simuliert Lebensdauerspektren anhand einer Monte-Carlo-Simulation; die verschiedenen Lebensdauerkomponenten können dabei wahlweise aus diskreten Werten oder aus einer Gaußverteilung bestehen. Die Auflösungsfunktion kann sowohl durch Summation mehrerer Normalverteilungen, als auch durch Poisson-Hintergrundrauschen oder ein Mehrexponentenmodell simuliert werden [KRA].

3.2 DDRS4PALS

Ein kombiniertes Simulations- und Analysetool ist die von D. Petschke entwickelte Software *DDRS4PALS*, welche je nach Anwendungssituation sowohl zur Datenaufnahme und Analyse von Spektroskopiemessungen, als auch zur Simulation von Lebensdauerspektren genutzt werden kann. Die Software wurde dabei eigens für das hier verwendete DRS4-Board entworfen. Im Simulationsmodus werden Lebensdauerpulse durch logarithmische Normalverteilungen dargestellt. Dazu wird die Definition der Wahrscheinlichkeitsdichte einer Log-Normalverteilung

$$f(t) = \begin{cases} \frac{1}{\sqrt{2\pi}\sigma t}\, exp\left(-\frac{(ln(t)-\mu)^2}{2\sigma^2}\right) & \text{für } x > 0 \\ 0 & \text{für } x \leqslant 0 \end{cases} \tag{3.1}$$

verwendet, wobei $\sigma \in \mathbb{R}^+$ ein Formparameter und $\mu \in \mathbb{R}$ ein Skalenparameter ist [Sch01]. Ferner wird angenommen, dass jedes Pulshöhenspektrum - unabhängig vom verwendeten Szintillator - durch eine Linearkombination zweier Gauß-Glocken darstellbar ist. Besonders für Plastikszintillatoren, beispielsweise Fabrikate der auf Polyvinyltoluen basierenden Baureihe BC422, treffen diese Annahmen sehr gut zu [CI16].

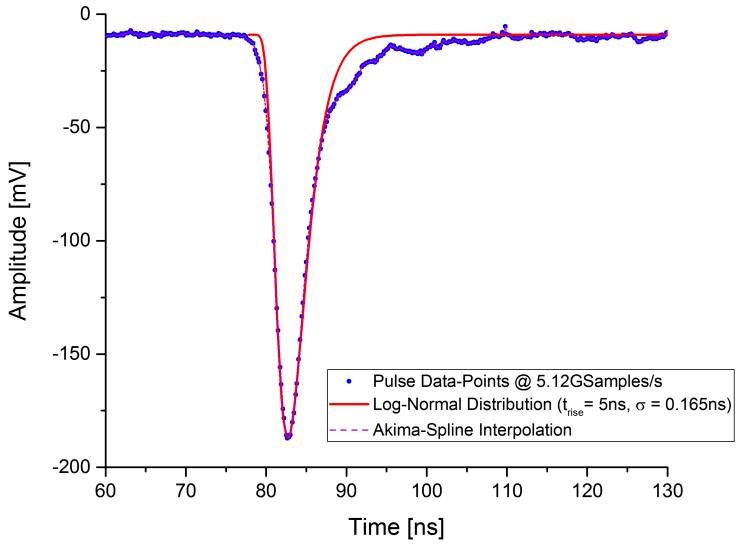

Abb. 3.1: Nachbildung eines realen, hier blau dargestellten Pulses. Der Puls wurde mit der in Abschnitt 2.3 beschriebenen Messanordnung unter Verwendung des DRS4-Boards aufgenommen. Die rote Kurve zeigt eine mit *DDRS4PALS* simulierte Log-Normalverteilung mit den Parametern $\mu \approx 75$ und $\sigma = 0.165ns$ [Pet17].

Wie Abbildung 3.1 zeigt, erreicht die Log-Normal Simulation eine größtenteils gute Passung zwischen simulierten und realen Pulswerten. Insbesondere auf der steigenden Flanke stimmen die simulierten Werte nahezu perfekt mit den aus der realen Messumgebung erhaltenen Daten überein. Kurz nach dem Verlassen und vor dem Erreichen der Horizontalen sind jedoch signifikante Diskrepanzen zwischen den Pulsdaten und der Verteilung erkennbar. Die Abweichung zu Beginn der Anstiegsflanke wird höchstwahrscheinlich durch eine im Messaufbau enthaltene, in der Auflösungsfunktion jedoch nicht berücksichtigte Ungenauigkeit verursacht. Der bis ca. -25mV vorhandene Unterschied verzögert den Beginn der Anstiegsflanke dabei um einige hundert Picosekunden, was bei niedrigen CFD-Levels (hier unter ca. 15%, respektive 30mV) folglich auch zu einer Verschiebung des Auslösezeitpunkts

der Constant Fraction Discrimatoren führt. Er beeinflusst damit die vom TAC erzeugte Ausgangsspannung, wodurch sich auch das resultierende Pulshöhenspektrum ändert (vgl. Abschnitt 2.3). Die nachfolgende, größere Abweichung liegt dabei im Intervall zwischen 85 und 100 Nanosekunden und beträgt mehrere zehn Millivolt. Sie ist dank des in Abschnitt 2.3 beschriebenen Messprinzips allerdings nicht von Bedeutung, da die CFDs bereits während der steigenden Flanke auslösen.

Kennzeichnend für *DDRS4PALS* ist dabei die adaptive Konfiguration der Messumgebung: Durch die Berücksichtigung verschiedener externer und interner Einflussparameter, die in einem XML-InputFile durch spezifische Tags gekennzeichnet sind, wird ein exaktes Nachbilden der Geräteauflösungsfunktion (IRF) und der Störfunktion $G(t)$ (vgl. Abschnitt 2.4) ermöglicht. Der Endnutzer kann diese Parameter an das jeweils verwendete Laborsetup anpassen und das Programm so ideal konfigurieren. *DDRS4PALS* ist deshalb unabhängig von der Bauart eines Photodetektorsystems und sowohl mit Photomultipliern als auch mit Avalanche-Photodioden (APDs) kompatibel.

Die wichtigsten Tags und ihre Attribute lauten:

- Electronics of Setup
  - TTS-A, TTS-B — Teilchenlaufzeit im Detektor
  - Sweep — Auflösungsfunktion des MCA
  - Number of cells — Zellen pro MCA-Kanal
  - Timing Resolution — Zeitauflösung der Elektronik des DRS4-Boards
  - ATS — Unterschiedl. Ankunftszeitpunkt in den Detektoren durch bspw. unterschiedliche Kabellängen
- Pulse-Height Distribution (PHS)
  - μ_A, μ_B — Verteilungsparameter
  - σ_A, σ_B
- Pulse-Shape Definition
  - RiseTime — Anstiegszeit des Pulses
  - PulseWidth — Halbwertsbreite des Pulses
- Lifetime-Distribution
  - LifeTime$_i$ — jeweilige Lebensdauer
  - LifeTime$_i$ enabled — Anzahl der zu bestimmenden Lebensdauern
  - Intensity of LifeTime$_i$ — Intensität der Lebensdauerkomponente
  - Background Intensity — Intensität des Hintergrundrauschens
  - Coincidence Intensity — Geräteverhalten, messbar bspw. bei (zeitgleicher) Detektion zweier entgegengerichteter 511 keV Stop-Quanten

Die Bezeichnungen A und B werden dabei jeweils als Identifikator für Start- und Stopdetektor benutzt. RiseTime bezeichnet die Anstiegszeit, PulseWidth die Halbwertsbreite FWHM, die zweite charakteristische Pulsgröße, die im Allgemeinen durch FWHM $= 2\sigma_s\sqrt{2\ln(2)}$ beschrieben wird. Dabei bezeichnet σ_s die Standardabweichung der zugrundeliegenden Verteilung [OK07].

DDRS4PALS ist dabei konform zur C/C++ API nach dem ISO-Standard von 2011. Die modulare Gestaltung und die Verwendung des Speicherklassenattributs `_declspec (dllexport)` garantieren zudem eine einfache Anbindung an andere C/C++-Programme, sowie Portabilität in verwandte Programmiersprachen. *DDRS4PALS* verwendet zur Pulserzeugung die Worker-Klasse `DPALSPulseGenerator`, welche später auch von *COSMOPALS* genutzt werden soll. Die einfache Einbindung dieser Klasse in Drittsoftware wird durch die Nutzung von shared Libraries und die Verwendung des Design-Patterns `opaque_pointer` ermöglicht.

DDRS4PALS implementiert den `opaque_pointer` als `unique_ptr<Template T> d_ptr`, was mit mehreren Vorteilen verbunden ist: Zum einen bleibt die Binärkompatibilität bei Hinzufügen neuer privater Klassenmember erhalten, zum anderen ist die konkrete Implementierung hinter dem `d_ptr` verborgen und so für Endanwender nicht sichtbar. Sie kann ohne erneute Kompilierung der darauf basierenden Module geändert werden [Wol06].

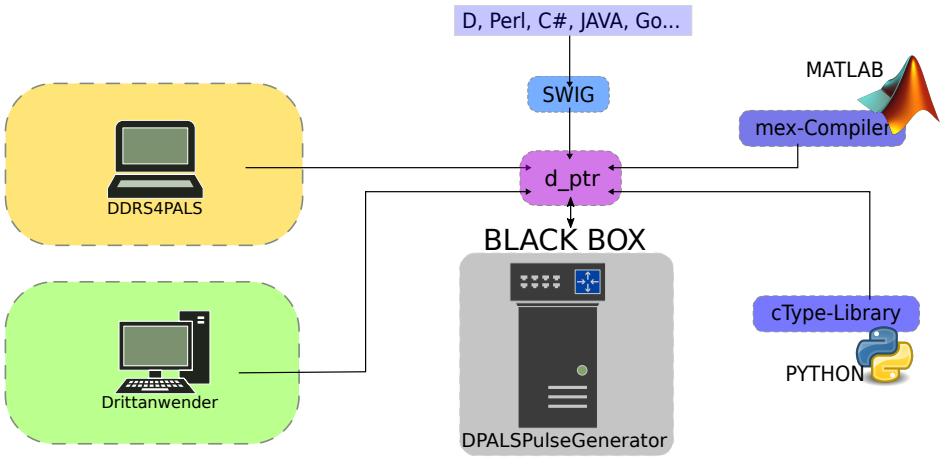

Abb. 3.2: Zugriff auf die zentrale, verlinkte Bibliothek durch verschiedene Endandwender. DPALSPulseGenerator stellt sowohl die microsoftspezifischen, dynamischen .dll- als auch die von Linux oder Unix-Systemen verwendeten, statischen .a - Bibliotheken zur Verfügung. Über verschiedene Compiler können auch andere Programmiersprachen eingebunden werden.

Abbildung 3.3 zeigt den direkten Vergleich zwischen einem künstlich simulierten und einem realen Lebensdauerspektrum einer Aluminiumprobe. Hierzu wurde das durch den Messaufbau gewonnene Pulshöhenspektrum von *DDRS4PALS* analysiert, um die für die

Pulssimulation benötigten Parameter zu extrahieren. Mit diesen Parametern wurden anschließend Lebensdauerpulse simuliert, welche wiederum in den Spektrometeraufbau eingespeist wurden. So konnte ein ganzes simuliertes Spektrum gewonnen werden. Die beiden erhaltenen, normalisierten Spektren sind in der nachfolgenden Graphik dargestellt. Analog zu Abb. 3.1 wird auch hier die steigende Flanke gut approximiert und die Abweichungen sind zu Beginn der Anstiegsflanke und gegen Ende der fallenden Flanke am größten.

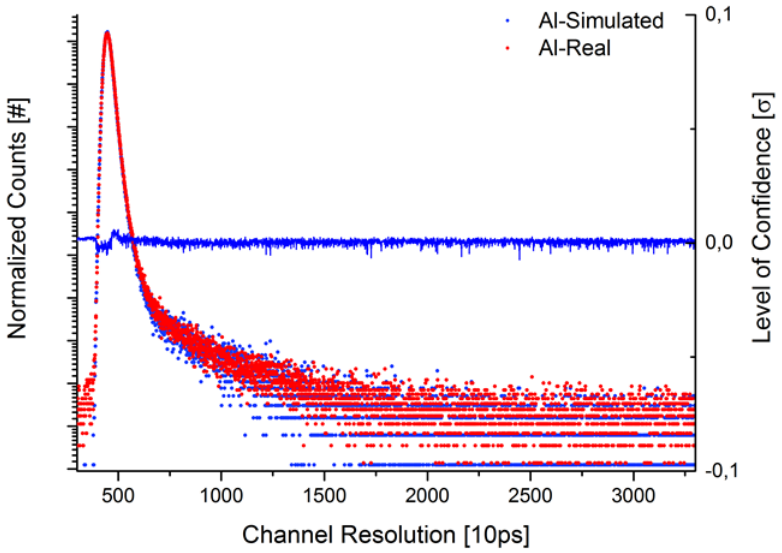

Abb. 3.3: Gegenüberstellung des realen und des simulierten Spektrums. Die roten Punkte symbolisieren das real erzeugte Spektrum, die blauen Punkte stehen für die simulierten Spektralwerte. Das rechts aufgetragene Konfidenzlevel σ ist durchgehend nahe bei null, der Ausreißer bei ca. 5000 Picosekunden ist höchstwahrscheinlich auf einen Einflussfaktor des DRS4-Boards zurückzuführen, der sich nicht durch eine Gaußverteilung beschreiben lässt [Pet17].

Die Simulationsmethodik hinter *DDRS4PALS* arbeitet dabei sehr genau: Im März 2017 wurden Simulationsergebnisse erreicht, die von tatsächlich gemessenen Geräteungenauigkeiten um nur ca. 0.5 Picosekunden abwichen. Dies kann genutzt werden, um bereits vor Aufbau des Messapparats zu untersuchen, wie Änderungen der Setup-Parameter (beispielsweise Samplegeschwindigkeit, Kurvenglättung, etc.) die IRF beeinflusst. Momentan liegt dem Bundesministerium für Bildung und Forschung (BMBF) ein Antrag auf Aufnahme in das VIP+ Förderprogramm der Bundesregierung vor, welcher bei positivem Bescheid weitere Forschungsmöglichkeiten im Bereich der Pulssimulation ermöglicht.

4 COSMOPALS

Die im Folgenden vorgestellte Software *COSMOPALS* verfolgt das Ziel, die Ansätze aus *DDRS4PALS* aufzugreifen und zu erweitern. Während das in Kapitel 3 beschriebene Programm ein direktes Einlesen der systemcharakterisierenden Geräte- und Einflussparameter des Laborsetups ermöglicht, geht *COSMOSPALS* auf die verwendete Detektorstruktur ein und erlaubt dem Nutzer, die Hardwarekomponenten im pulserzeugenden Photomultipliersystem selbst zu konfigurieren. Ziel ist es, ein ideal aufeinander abgestimmtes Setup simulieren zu können, bei dem die resultierende Pulsform durch Änderung der verwendeten Bauteile bewusst verändert und angepasst werden kann. Dazu wird zuerst die elektrotechnische und anschließend die mathematische Modellierung eines solchen Systems vorgestellt, während nachfolgend auf die konkrete Implementierung und die damit verbundenen Herausforderungen eingegangen wird.

4.1 Pulssimulation anhand elektronischer Komponenten

Bereits in Abbildung 2.3 wurde exemplarisch der Ausgangspuls eines Photomultipliers vorgestellt. Ein solcher Puls soll nun unter Aufgriff der Ergebnisse aus Arbeit [Tan08] modelliert werden.

4.1.1 Elektrotechnische Modellierung

Als Spannungsquelle für die Photomultiplierschaltung wird in Arbeit [Tan08] ein linear ansteigender Puls mit einer Periodendauer von mehreren hundert Picosekunden verwendet. Durch empirische Tests wurde im Zuge der Entwicklung von *COSMOPALS* jedoch klar, dass nicht die lineare Anstiegszeit, sondern der exponentielle Abfall des Eingangspulses für die charakteristische Pulsform verantwortlich ist. Deshalb wird für die hier skizzierte Modellierung ein PT1-Glied in Form eines RC-Glieds verwendet, welches eine gepulste Gleichspannung als Eingangsspannung erhält. Sie wird in den folgenden Abschnitten als Rechteckspannung approximiert, die durch periodisches Öffnen eines Schalters zu einer (idealen) Gleichspannungsquelle erzeugt wird.

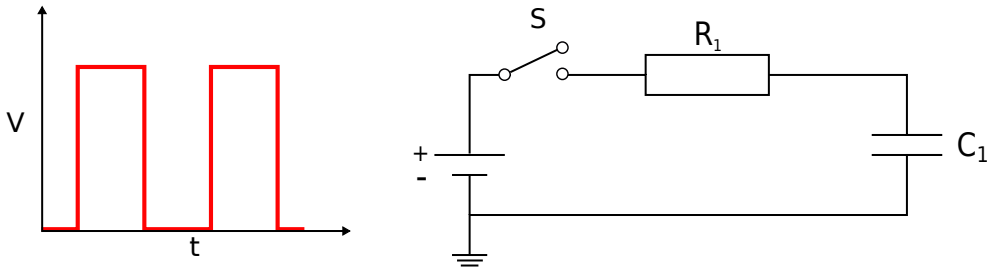

Abb. 4.1: RC-Glied mit Rechteckspannung zur Modellierung des Eingangspulses.

Ein RC-Glied wird dabei durch seine spezifische Zeitkonstante τ charakterisiert, die sich aus Multiplikation der Kondensatorkapazität C mit dem ohmschen Widerstand R ergibt. Ist der Schalter S geschlossen, liegt Gleichspannung an und der Kondensator wird über den Widerstand aufgeladen. Beim Öffnen des Schalters fällt die Spannung am Kondensator schlagartig ab und der Entladevorgang beginnt. Das RC-Glied verformt die eingehende Rechteckspannung dabei zu einer Kurve, die einem exponentiellen Zusammenhang folgt.

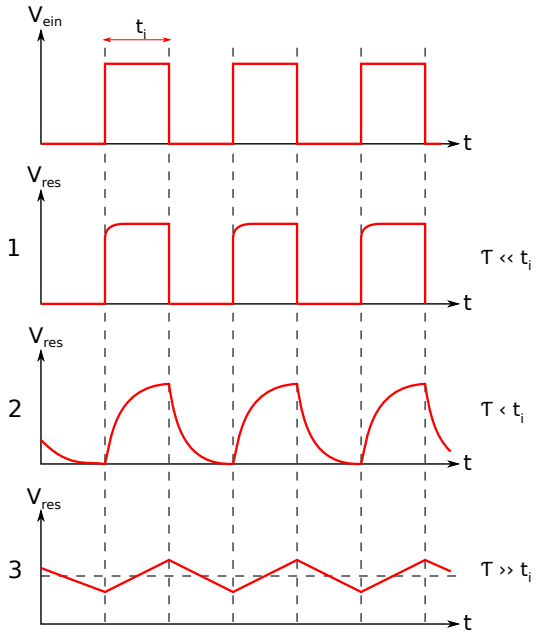

Abb. 4.2: Zusammenhang zwischen Verformung der Rechteckspannung und der Zeitkonstanten τ. Nach [BS97].

Die Zeitkonstante τ beschreibt diesen Zusammenhang durch das Verhältnis zwischen Lade- und Entladezyklen und ist somit charakteristisch für das Systemverhalten. Abbildung 4.2 zeigt, wie verschiedene Werte für τ das Ausgangssignal beeinflussen: Ist τ sehr viel kleiner als die Pulsdauer t_i, bleibt die Eingangsspannung nahezu unverändert; die Verformung ist umso stärker, je größer das Verhältnis von τ zu t_i ist. Da, wie eingangs erwähnt, ein exponentiell abfallender Puls die besten Resultate verspricht, ist in einem Zyklus der Dauer $t_{cycle} = 2 \times t_i$ nur der Entladevorgang von Bedeutung. Die Impulsdauer kann also beliebig gewählt werden, solange sie groß genug ist, um den Kondensator voll aufzuladen. Die Zeitkonstante wird deshalb ähnlich wie in Abb. 4.2, Fall 2, gewählt. Der daraus resultierende Lade- und Entladevorgang des Kondensators ist in Abb. 4.3 graphisch dargestellt.

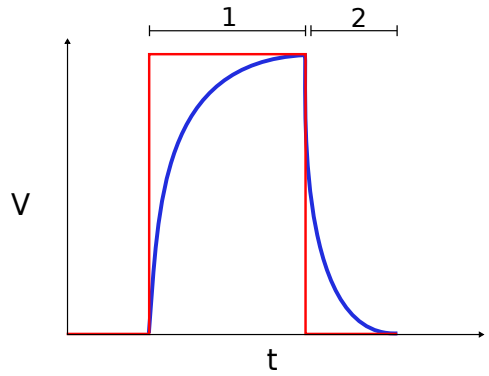

Abb. 4.3: Lade- und Entladezyklus des Kondensators im RC-Glied während einer Periode. Die fallende Kurve nimmt exponentiell ab.

Für den weiteren Verlauf soll nun die Kondensatorentladung (Abb. 4.3: Abschnitt 2) separiert werden. Dazu wird die periodisch unterbrochene Eingangsspannung mit einem NAND-Logikgatter gekoppelt. Der zweite Input (B) des Gatters wird mit der konstanten Eingangsspannung U_0 beschaltet, sodass immer dann `true` ausgegeben wird, wenn der Schalter geöffnet ist, also auf Input A keine Spannung anliegt. Das Signal des NAND-Gatters wird anschließend benutzt, um ein optoisoliertes Halbleiterrelais anzusteuern, welches mittels eines Schalters die Verbindung zwischen RC- und RLC-Glied schließt. So kann der Entladevorgang des Kondensators abgespalten und einzeln an den nachfolgenden RLC-Kreis weitergegeben werden.

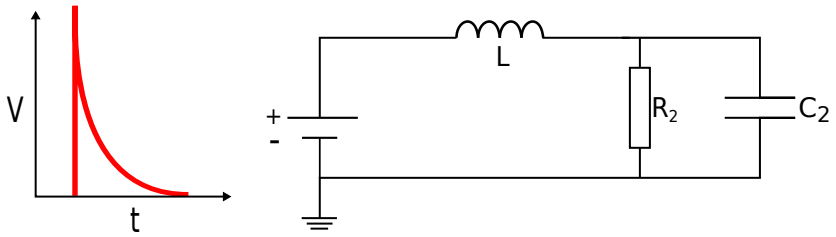

Abb. 4.4: RLC-Glied mit Eingangspulsform.

Das RLC-Glied soll nun den bereits erzeugten Puls an den in Abbildung 2.3 Vorgestellten anpassen. Durch den Eingangsimpuls angestoßen, führt der RLC-Reihenschwingkreis eine harmonische Eigenschwingung aus, die mit fortschreitender Zeit aufgrund der Änderung der, durch die Spule erzeugten, Spannung abnimmt. Zudem wird ein Teil der elektrischen Energie am Widerstand R_2, unter anderem durch Wärmekonvektion, an die Umgebung abgegeben. Dieser Effekt wird in der Realität durch dissipative Prozesse in den Kupferleitungen, parasitäre Kapazitäten und Nicht-Idealität der Bauelemente noch verstärkt, was im Folgenden zunächst allerdings nicht beachtet werden soll. Das RLC-Glied kann daher wie ein idealer, gedämpfter harmonischer Oszillator behandelt werden.

Nachdem die Versorgungsspannung „eingeschaltet“, also der durch die Kondensator-

entladung entstandene Puls vom Halbleiterrelais durchgeschaltet, wurde, erzeugt die Spule eine der Eingangsspannung entgegenwirkende Induktionsspannung. Der Spulenstrom kann sich dabei nicht sprungartig ändern, sondern wird durch die Induktionsspannung graduell erhöht. Da der Strom der an einem Leiter anliegenden Spannung proportional ist, verursacht der Eingang des durch die Kondensatorentladung erzeugten Pulses und die damit verbundenen Induktionsprozesse also eine „Abflachung“ der anfangs sehr steilen Pulsform. Diese wird in Abhängigkeit von R_2, C_2 und L entweder schwach oder kritisch gedämpft, oder überdämpft. Durch Abgriff der an C_2 anliegenden Kondensatorspannung entsteht die in Abb. 2.3 gezeigte Pulsform [Ada79].

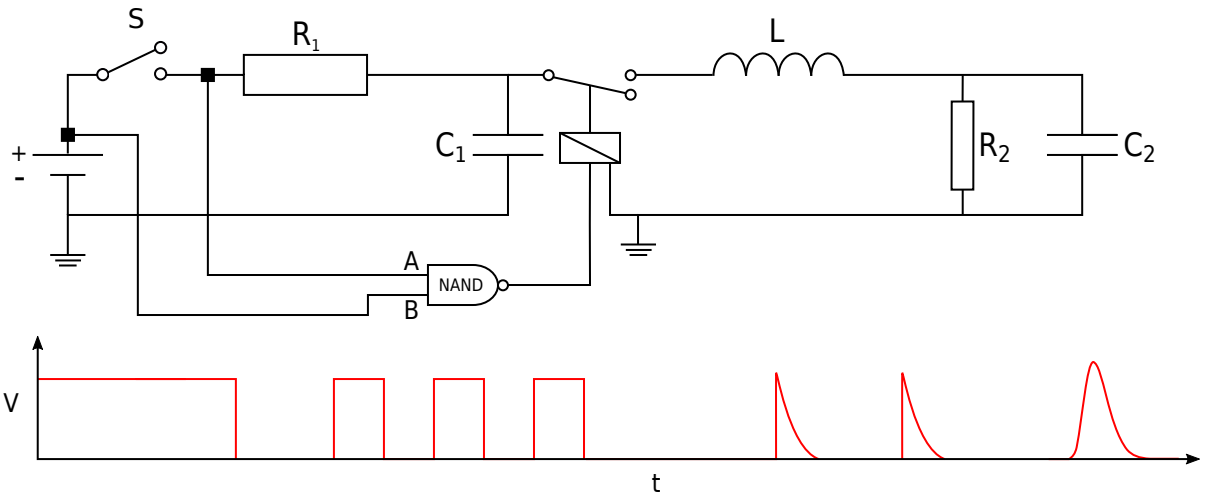

Abb. 4.5: Gesamtansicht des Simulationsmodells mit exemplarischem Spannungsverlauf. In den nachfolgenden Abschnitten wird vereinfachend von einem einmaligen Anstoßvorgang ausgegangen, der sich später beliebig vervielfachen lässt.

4.1.2 Mathematische Modellierung des Systemverhaltens

Kennt man die charakteristischen Differentialgleichungen eines Übertragungssystems, kann der Zusammenhang zwischen Ein- und Ausgangsspannung mit beliebigen Bauteilkombinationen leicht beschrieben werden. Diese Gleichung und die genaue mathematische Beschreibung des zuvor erstellten Modells sollen nun ermittelt werden. Anzumerken ist, dass die in der obigen Abbildung dargestellten, unterbrochenen Verbindungslinien das Trennrelais symbolisieren. Da dieses allein dazu dient, die beiden Schaltungen zur richtigen Zeit zu verbinden, leistet es keinen Beitrag zur Übertragungsfunktion und wird folglich bei der mathematischen Modellierung nicht benötigt. In Abb. 4.6 wurde es deshalb, analog zum NAND-Gatter, vereinfachend ausgespart. Ferner kann die Zeitabhängigkeit der Eingangsspannung U_{ein} hier vernachlässigt werden, da sie aufgrund der vom Schalter S erzeugten Rechteckspannung nur den konstanten Wert U_0 oder null annehmen kann. Wenn dabei

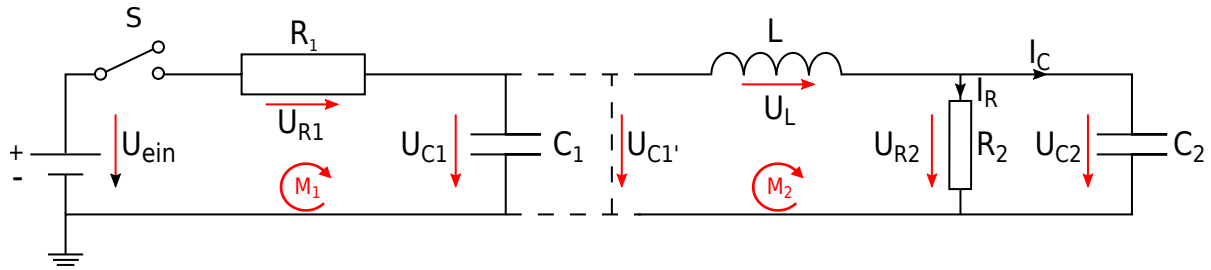

Abb. 4.6: Modellierung der Gesamtschaltung nach den Kirchhoff'schen Regeln.

noch von einer symmetrischen Rechteckspannung der Periode $P = t_{low} + t_{high}$ ausgegangen wird, ergibt sich U_{ein} also zu $U_{ein} = U_0\,\varepsilon(t)$, mit

$$\varepsilon(t) = \begin{cases} 0 & \text{für } t < 0.5(2k-1)P \quad \text{mit } k \in \mathbb{N}_+ \\ 1 & \text{für } t \geqslant 0.5(2k-1)P \quad \text{mit } k \in \mathbb{N}_+ \end{cases}$$

Zunächst soll die Kondensatorspannung U_{C_1} bestimmt werden. Dazu wird das RC-Glied gesondert betrachtet. In der Masche 1 gilt nach der Kirchhoff'schen Maschenregel folgende Beziehung:

$$U_{R_1} + U_{C_1}(t) - U_{ein} \;=\; IR_1 + U_{C_1}(t) - U_{ein} \;=\; 0$$

Mit den allgemeinen Zusammenhängen $I = \dfrac{\mathrm{d}Q}{\mathrm{d}t}$ und $Q = CU$ ergibt sich:

$$R\,C\,\dot{U}_{C_1}(t) \;+\; U_{C_1}(t) \;-\; U_{ein} \;=\; 0$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\,U_{C_1}(t) \;=\; \frac{U_{ein} - U_{C_1}(t)}{RC_1} \;=\; \frac{U_{ein}}{RC_1} - \frac{1}{RC_1}\,U_{C_1}(t)$$

Diese lineare inhomogene Differentialgleichung erster Ordnung lässt sich durch Variation der Konstanten lösen, wodurch sich unter Zuhilfenahme der beim Start des Entladevorgangs geltenden Anfangsbedingungen $t := 0$, $U_{C_1} = U_{max} = U_0$ und $U_{ein} = 0$ die spezielle Lösung für die Kondensatorspannung U_{C_1} ergibt:

$$U_{C_1}(t) \;=\; U_0\,e^{-\dfrac{t}{RC_1}} \tag{4.1}$$

U_0 beschreibt dabei die maximal auf den Kondensator aufbringbare Spannung, also die Amplitude der anfänglichen Rechteckspannung U_{ein}.

Für den Aufladevorgang des Kondensators gelten andere Anfangsbedingungen; da er nach Abschnitt 4.1.1 bei passender Wahl von τ jedoch keinen Einfluss auf das Verhalten des Gesamtsystems hat, wird er hier nicht weiter betrachtet. Gleichung 4.1 gilt also nur für

die Spannung bei Kondensatorentladung. In Abb. 4.6 ist sie deshalb als $U_{C_1'}$ eingetragen; da $U_{C_1'}$ ferner jedoch die Eingangsspannung des nachgeschalteten RLC-Glieds ist, wird im Folgenden der Übersicht halber synonym die Bezeichnung $U_e(t)$ verwendet.

Nun soll die zur Pulserzeugung genutzte Ausgangsspannung der Schaltung, also die am Kondensator C_2 anliegende Potentialdifferenz, berechnet werden. Dazu wird das RLC-Glied einzeln betrachtet. Die Abhängigkeit von der vorhergehenden Schaltung ergibt sich durch die oben errechnete Spannung $U_e(t)$. Mit der Maschenregel folgt für M2:

$$U_L \; + \; U_{R_2} \; = \; U_e(t)$$

Da die beiden Komponenten R_2 und C_2 parallel geschaltet sind, muss $U_{R_2} = U_{C_2}$ gelten. Mit der Beziehung $U_L(t) = L \, \dfrac{\mathrm{d}}{\mathrm{d}t} \, I(t)$ ergibt sich:

$$L \, \frac{\mathrm{d}}{\mathrm{d}t} \, I(t) \; + \; U_{C_2}(t) \; = \; U_e(t) \; = \; U_0 \, e^{-\dfrac{t}{RC_1}} \tag{4.2}$$

Für den Gesamtstrom I gilt: $I_{ges} = I_R + I_C$. Aus der Beziehung $Q = C_2 \, U_{C_2}$ und der oben bereits erwähnten Abhängigkeit zwischen Ladung und Strom $\dot{Q} = I$ sowie dem Ohmschen Gesetz folgt somit:

$$\begin{aligned} I_C \; &= \; C_2 \dot{U}_{C_2}(t) \\ I(t) \; &= \; \frac{1}{R_2} \, U_{C_2}(t) \; + \; C_2 \, \frac{\mathrm{d}}{\mathrm{d}t} \, U_{C_2}(t) \end{aligned} \tag{4.3}$$

Die beiden Differentialgleichungen 4.2 und 4.3 bilden ein Differentialgleichungssystem, welches das Verhalten des RLC-Glieds beschreibt. Dabei steht (I) für Gl. 4.2 und (II) für Gl. 4.3. Da das abzubildende System kausal - also nur für $t \geq 0$ definiert - ist, lassen sich beide Gleichungen mittels Laplace-Transformation in den Bildbereich überführen, wo sie sich zu algebraischen Gleichungen mit analytischer Lösbarkeit vereinfachen. Man erhält die Gleichungen:

$$\begin{aligned} \mathcal{L}\{(\mathrm{I})\}: \quad & L \, sI(s) \; + \; U_{C_2}(s) \; = \; U_0 \, \frac{1}{s + \dfrac{1}{RC_1}} \\ \mathcal{L}\{(\mathrm{II})\}: \quad & I(s) \; - \; (sC_2 + \frac{1}{R_2}) \, U_{C_2}(s) \; = \; 0 \end{aligned}$$

Löst man den zweiten dieser beiden Terme nach $I(s)$ auf und setzt in die erste Gleichung ein, erhält man nach einiger Umformung den Ausdruck für die Kondensatorspannung U_{C_2} im Bildbereich:

$$U_{C_2}(s) \;=\; \frac{U_0}{\left(s+\frac{1}{RC_1}\right)\left[LC_2\left(s^2+\frac{1}{R_2C_2}s+\frac{1}{Ls}\right)\right]} \tag{4.4}$$

Mit der Kennkreisfrequenz des elektrischen Schwingkreises $\omega_0 = \sqrt{\frac{1}{LC_2}}$ und der Abklingkonstante $\delta = \frac{1}{2R_2C_2}$ sowie den Zeitkonstanten τ_1 und τ_2 der Übertragungsglieder ergibt sich eine übersichtlichere Form [Ada79]. Dabei stellt der erste Faktor die Einflüsse des RC-Glieds dar, der nachfolgende Faktor beschreibt das RLC-Glied.

$$U_{C_2}(s) \;=\; \frac{U_0}{s+\frac{1}{\tau_1}}\;\frac{{\omega_0}^2}{s^2 \;+\; 2\delta s \;+\; {\omega_0}^2} \tag{4.5}$$

Um die gesuchte Lösung der ursprünglichen Differentialgleichungen 4.2 und 4.3 zu erhalten, soll die Kondensatorspannung nun mittels der inversen Laplace-Transformation in den Zeitbereich rücktransformiert werden. Zur Rücktransformation muss Gl. 4.5 anhand der Polstellen von $U_{C_2}(s)$ partialbruchzerlegt werden. Da in der umgeformten Darstellung ein quadratischer Ausdruck der Form $ax^2 + bx + c$ auftritt, ist durch obige Darstellung und anhand der Diskriminanten leicht ersichtlich, dass die Pole bei $s_1 = -\frac{1}{\tau_1}$ und $s_{2,3} = -\delta \pm \sqrt{\delta^2 - \omega_0}$ liegen. Je nach Polstelle kann die Ausgangsspannung einem von drei Fällen zugeordnet werden, die sich jeweils durch den Dämpfungsgrad $\vartheta = \dfrac{\delta}{{\omega_0}^2}$ unterscheiden [WU07].

1. Fall - Aperiodischer Grenzfall: $\vartheta = 1$, und folglich $\delta^2 = {\omega_0}^2$.
Im aperiodischen Grenzfall ergibt sich durch den Wegfall der Diskriminanten die reelle Polstelle zweiter Ordnung $s_{2,3} = -\delta$, was zu folgender Partialbruchzerlegung führt:

$$U_{C_2}(s) \;=\; \frac{U_0\,{\omega_0}^2}{\left(s \;+\; \dfrac{1}{\tau_1}\right)(s+\delta)^2} \;=\; \frac{A}{s+\tau_1^{-1}} \;+\; \frac{B}{s+\delta} \;+\; \frac{C}{(s+\delta)^2}$$

Aus Umformung und Lösung des resultierenden, linearen Gleichungssystems ergeben sich durch Koeffizientenvergleich die Konstanten A, B und C. Die spezielle Lösung im Bildbereich errechnet sich zu:

$$U_{C_2}(s) \;=\; \frac{U_0\,\omega_0^2}{\left(\delta - \frac{1}{\tau_1}\right)^2}\;\left[\frac{1}{s+\tau_1^{-1}} - \frac{1}{s+\delta} - \frac{\delta - \tau_1^{-1}}{(s+\delta)^2}\right]$$

Nach Rücktransformation in den Zeitbereich folgt für die Kondensatorspannung U_{C_2} im aperiodischen Grenzfall:

$$U_{C_2}(t)_1 \;=\; \frac{U_0\,\omega_0^2}{\left(\delta - \frac{1}{\tau_1}\right)^2}\;\left\{e^{-\frac{t}{\tau_1}} - e^{-\delta t}\left[t(\delta - \frac{1}{\tau_1}) + 1\right]\right\} \tag{4.6}$$

2. Fall - Periodischer Fall: $\vartheta < 1$, und folglich $\delta^2 < \omega_0^2$.
Die Diskriminante nimmt im periodischen Schwingungsfall aufgrund des negativen Radikanden Werte im Komplexen an. Mit der Eigenkreisfrequenz $\omega = \sqrt{\omega_0^2 - \delta^2}$ ergeben sich die beiden komplex konjugierten Polstellen $s_{2,3} = -\delta \pm i\omega$. Die resultierende Partialbruchzerlegung

$$U_{C_2}(s) \;=\; \frac{U_0\,{\omega_0}^2}{\left(s \;+\; \frac{1}{\tau_1}\right)(s+\delta+i\omega)(s+\delta-i\omega)} \;=\; \frac{A\big((s+\delta)^2+\omega^2\big) + (s+\frac{1}{\tau_1})(Bs+C)}{(s+\frac{1}{\tau_1})\big((s+\delta)^2+\omega^2\big)}$$

führt nach Koeffizientenvergleich analog zu Fall 1 auf die spezielle Lösung im Bildbereich:

$$U_{C_2}(s) \;=\; \frac{U_0\,\omega_0^2}{\left(\delta - \frac{1}{\tau_1}\right)^2 + \omega^2}\;\left[\frac{1}{s+\frac{1}{\tau_1}} - \frac{s+2\delta-\tau_1^{-1}}{(s+\delta)^2+\omega^2}\right]$$

Die Ausgangsspannung des RLC-Glieds ergibt sich im periodischen Fall also zu:

$$U_{C_2}(t)_2 \;=\; \frac{U_0\,{\omega_0}^2}{\left(\delta - \frac{1}{\tau_1}\right)^2 + \omega^2}\;\left\{e^{-\frac{t}{\tau_1}} - e^{-\delta t}\left[\frac{(\delta - \frac{1}{\tau_1})\sin(\omega t) + \omega\cos(\omega t)}{\omega}\right]\right\} \tag{4.7}$$

3. Fall - Aperiodischer Fall: $\vartheta > 1$, und folglich $\delta^2 > \omega_0^2$.
Der nun positive Radikand erzeugt zwei reelle, verschiedene Polstellen $s_{2,3} = -\delta \pm$

$\sqrt{\delta^2 - \omega_0^2}$. Mit der Hilfsgröße $a^2 = \delta^2 - \omega_0^2$ erfolgt die Partialbruchzerlegung zu:

$$U_{C_2}(s) = \frac{U_0\,{\omega_0}^2}{\left(\frac{1}{s+\tau_1}\right)\left((s+\delta)^2 - \delta^2 + \omega_0^2\right)} = \frac{A\big((s+\delta)^2 - a^2\big) + (\delta + \frac{1}{\tau_1})(Bs\ +\ C)}{\left(s + \frac{1}{\tau_1}\right)\left((s+\delta)^2 - a^2\right)}$$

Analog zu Fall 1 und 2 ergibt sich die spezielle Lösung im Bildbereich:

$$U_{C_2}(s) = \frac{U_0\,{\omega_0}^2}{\left(\delta - \frac{1}{\tau_1}\right)^2 - a^2}\left[\frac{1}{s + \frac{1}{\tau_1}} - \frac{s + 2\delta - \tau_1^{-1}}{(s+\delta)^2 - a^2}\right]$$

In der abschließenden Gleichung zur Beschreibung der Kondensatorspannung im aperiodischen Fall finden sich aufgrund des Vorzeichenunterschieds statt der trigonometrischen Funktionen nun die entsprechenden Hyperbelfunktionen:

$$U_{C_2}(t)_3 = \frac{U_0\,{\omega_0}^2}{\left(\delta - \frac{1}{\tau_1}\right)^2 - a^2}\left\{e^{-\frac{t}{\tau_1}} - e^{-\delta t}\left[\frac{(\delta - \frac{1}{\tau_1})\sinh(at) + a\cosh(at)}{a}\right]\right\} \tag{4.8}$$

Somit wären alle möglichen Fälle behandelt und die Ausgangsspannung der Schaltung kann für alle möglichen Bauteilkombinationen errechnet werden.

Da negative Induktivitäten, Widerstände und Kondensatoren physikalisch nicht möglich sind, können die Bauteile $L, R_{1/2}$ und $C_{1/2}$ keine negativen Werte annehmen. Für den Fall $R_{1/2} = 0$ oder $C_{1/2} = 0$ folgt eine Division durch null, da dann die Zeitkonstanten $\tau_{1/2}$ ebenfalls zu null werden. In diesem Fall könnte das Bauteil jedoch äquivalent aus der Schaltung entfernt werden, was weitere Überlegungen hinfällig macht, da der Stromkreis dann kurzgeschlossen wird. Mit der Überlegung, dass L, $R_{1,2}$ und $C_{1,2}$ also nur positive Werte annehmen können, folgt, dass auch δ, ω_0, τ_1, τ_2 und a echt größer null sein müssen. Offensichtlich besitzen alle vorkommenden Exponentialterme negative Vorzeichen. Zusammen mit der Bedingung, dass in einem kausalen System $t \geq 0$ gelten muss, erhält man für alle drei Fälle:

$$\lim_{t\to\infty} U_{C_2}(t)_1 \ =\ \lim_{t\to\infty} U_{C_2}(t)_2 \ =\ \lim_{t\to\infty} U_{C_2}(t)_3 \ =\ 0 \tag{4.9}$$

Die Schwingung des angestoßenen RLC-Glieds klingt also unabhängig von der gewählten

Bauteildimensionierung nach dem Einschalt- oder Anstoßvorgang für große Zeiten ab, was bereits durch die Annahmen in Abschnitt 4.1.1 postuliert und hier nun bestätigt wurde.

Zusammenfassend lässt sich sagen, dass die Bauteilwahl den zeitlichen Verlauf der Ausgangsspannung maßgeblich beeinflusst. Signifikante Unterschiede entstehen im besonderen durch Variation der Zeitkonstanten τ_1 und τ_2. Durch die gewählte Modellierung lassen sich Elektronikkomponenten zu Gruppen zusammenfassen, die durch die durchgeführte Fallunterscheidung dann einem bestimmten Ausgangssignal zugeordnet werden können. Der Dimensionierung dieser Komponenten widmet sich der folgende Abschnitt.

4.1.3 Selektion der Hardwarekomponenten

Die für die Schaltung benötigten Komponenten wurden im Zuge der Entwicklung von *COSMOPALS* empirisch mithilfe des Netzwerksynthesetools LTspice XVII ermittelt. Als Grundlage dienten erneut die Ansätze aus Arbeit [Tan08].

Zunächst soll der aus dem RC-Glied resultierende Spannungspuls dimensioniert werden. Nach Abbildung 4.1 hängt der fallende Puls offensichtlich nur von den Bauteilkomponenten R_1 und C_1 ab. Zusammen mit dem Zeitverhalten folgt die bereits beschriebene Gleichung 4.1:

$$U_{C_1}(t) \;=\; U_0\, e^{-\dfrac{t}{R_1 C_1}}$$

Aus obigem Zusammenhang ist leicht ersichtlich, dass der Einfluss der fortschreitenden Zeit t auf die Kondensatorspannung mit der Zunahme der Zeitkonstanten $\tau_1 = R_1 C_1$ kontinuierlich abnimmt. Dies korreliert mit dem allgemeinen Funktionsprinzip eines Kondensators: Desto geringer die verfügbare Kondensatorkapazität ist, desto weniger Ladung kann auf den Kondensator aufgebracht werden, und desto schneller ist der Entladevorgang nach Spannungswegfall beendet. Der multiplikative Proportionalitätsfaktor U_0 beeinflusst das Abklingen des Pulses dabei nicht, sondern fungiert lediglich als Skalierungsfaktor für die Spannung und ist beliebig anpassbar. Er wird im Zuge der Dimensionierung deshalb nicht weiter beachtet, sondern beliebig, aber fest gewählt.

Da nach Arbeit [Tan08] Pulsdauern im mittleren Picosekundenbereich gefordert sind, muss die Zeitkonstante entsprechend klein gewählt werden. Zunächst soll allerdings der Begriff der Pulsdauer definiert werden, da die obige Gleichung dank dem Exponentialterm im nicht-trivialen Fall $U_0 \neq 0$ selbst für große Zeiten nur Werte produziert, die asymptotisch gegen null gehen, den Wert null jedoch nie erreichen. Dies resultiert aus der Idealisierung des Entladungsvorgangs und repräsentiert eine vollständige Entladung des Kondensators nach unendlich langer Zeit. In realen Kondensatorschaltungen werden auftretende Verluste meist durch einen Kondensatorinnenwiderstand approximiert, welcher obige Gleichung um einen zusätzlichen Term erweitert und zu einer endlichen Entladezeit führt. Vereinfachend wird hier deshalb ein bestimmtes Maß eingeführt, unterhalb welchem die Kondensatorspannung als null betrachtet wird. Diese Grenze wurde auf ein tausendstel Volt festgelegt.

Für die weiteren Betrachtungen wird nach Arbeit [Tan08] eine Pulsdauer von ca. 500 ps angenommen. Mit den beiden zum Entladezeitpunkt t_0 geltenden Anfangsbedingungen $U_{C_1}(t_0) = U_0$ und $U_{C_1}(t_0 + 500ps) \approx 0^+ (:= 0.001)$ folgt nach kurzer Rechnung:

$$\tau_1 = \frac{500 \times 10^{-12} s}{-\ln\left(\frac{0.001V}{5V}\right)} = 5.87 * 10^{-11} s$$

Die Zeitkonstante τ_1 muss also in der Größenordnung von ca. 10^{-11} s liegen. Für den Faktor U_0 wurden konstant 5V verwendet, da dies die von einem USB-Netzteil bereitgestellte Versorgungsspannung approximiert, welche später auch zum Bau der realen Schaltung verwendet werden soll. Das angestrebte Verhältnis von Widerstand zu Kondensatorkapazität kann dabei durch verschiedene Kombinationen von Bauteilen realisiert werden. Für die später angedachte praktische Umsetzung sollen jedoch bereits hier handelsübliche Bauteilwerte benutzt werden. Da Kondensatoren gut mit geringer Kapazität gefertigt werden können, kann die Zeitkonstante trotz des kleinen Exponenten ohne Probleme realisiert werden. Nachfolgend wird deshalb ein Kondensator mit der festen Kapazität 0.1nF verwendet.

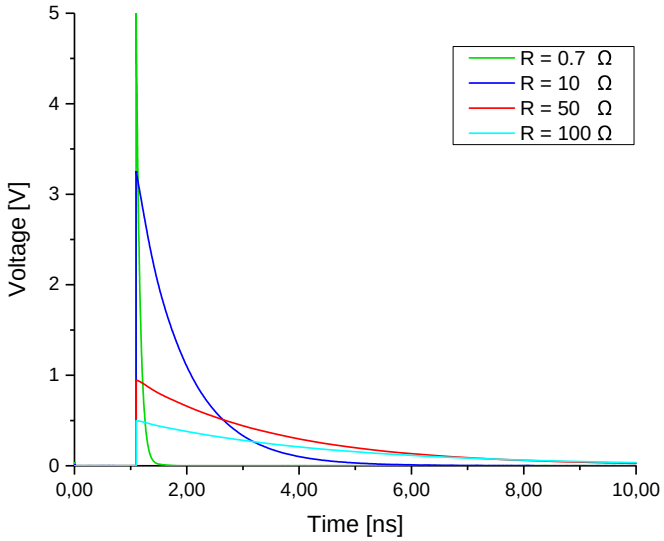

Abb. 4.7: Der Ausgangspuls des RC-Glieds für verschiedene Widerstandswerte R_1 bei konstanter Kondensatorkapazität $C_1 = 0.1$nF. Die spitze, grüne Kurve ergibt sich für $R_1 = 0.7\,\Omega$, nachfolgend wurden in absteigender Reihenfolge die Werte $10\,\Omega$, $50\,\Omega$ und $100\,\Omega$ aufgetragen. Die Pulse wurden aus dem kompletten Schaltkreis synthetisiert, weshalb durch die NAND-Gatterschaltung die Verzögerung von ca. 1.1ns entsteht.

Abbildung 4.7 zeigt die aus verschiedenen Widerstandswerten resultierenden Pulse. Offensichtlich beeinflusst die Wahl des Widerstands R_1 den Spannungsabfall am Kondensator stark. Mit R_1 steigt auch die Zeitkonstante τ_1, was, wie oben beschrieben, zu einer längeren Entladezeit führt: während die grüne Kurve bereits nach weniger als einer Nanosekunde wieder gegen null geht, benötigt die rote Kurve dazu mehr als die fünffache Zeit. Nachdem der durch das NAND-Gatter abgegriffene Puls an den Eingang des RLC-Glieds weitergegeben wird, stößt er eine eigenharmonische Schwingung an (vgl. Abschn. 4.1.1). Da sich besonders kurze Stöße als bestgeeignet erwiesen haben, wird der Wert für R_1 auf $0.7\,\Omega$ festgelegt.

Je nach Wahl der nachfolgenden Komponenten R_2, C_2 und L und des daraus resultierenden Dämpfungsgrades ϑ schwingt der RLC-Kreis periodisch, aperiodisch oder im aperiodischen Grenzfall - insgesamt resultieren die drei Fälle in unendlich vielen verschiedenen Bauteilkombinationen und Pulsformen. Die folgenden Graphiken zeigen deshalb die Ausgangspulsform des RLC-Glieds anhand verschiedener konkreter Werte für einzelne Bauteile und verdeutlichen die Unterschiede im jeweiligen Schwingungsverhalten.

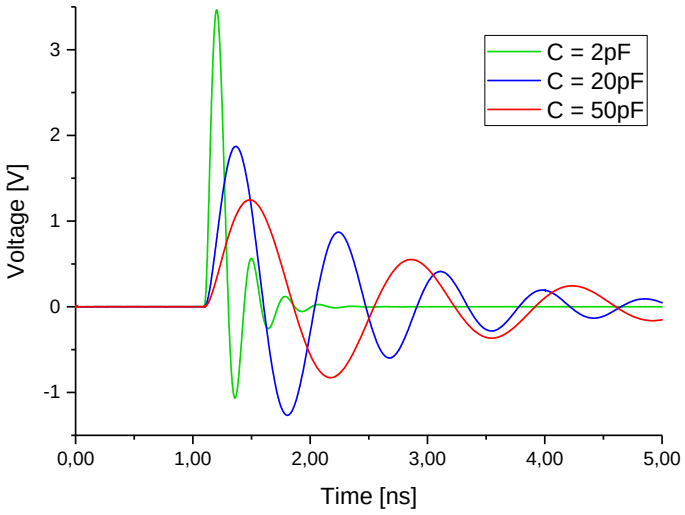

Abb. 4.8: Der Ausgangspuls des RLC-Glieds für verschiedene Kondensatorwerte bei konstantem Widerstand $R_2 = 50\,\Omega$ und konstanter Induktivität $L = 1$nH. Es wurden die resultierenden Kurven für die Kondensatorkapazitäten $C_2 = 2$pF, $C_2 = 20$pF und $C_2 = 50$pF aufgetragen. Alle drei Bauteilkombinationen schwingen periodisch.

Abbildung 4.8 verdeutlicht, wie die Ausgangsspannung im periodischen Fall durch den Dämpfungsgrad ϑ an Energie verliert. Da $\vartheta < 1$ gilt, wird die Schwingung nicht überdämpft und kann sich eine gewisse Zeit periodisch fortsetzen, wobei ϑ per definitionem mit zunehmender Kapazität abnimmt (vgl. Abschn. 4.1.2). Im Falle der periodischen Schwingung können Einhüllende entlang der lokalen Extrema definiert werden. Für die Pulssimulation ist der periodische Schwingungsfall jedoch nur bedingt geeignet, da die aus dem realen Spektrometeraufbau resultierenden Lebensdauerpulse wenig bis kein Überschwingen aufweisen.

Abb. 4.9 zeigt die resultierenden Pulsformen für verschiedene Induktivitäten. Hierbei ist anzumerken, dass die grüne Kurve mit einer Induktivität von $L = 10$nH trotz starker Dämpfung noch dem periodischen Schwingungsfall zuzuordnen ist, wie das Unterschreiten der 0V-Marke bei ca. 2,05ns zeigt. Die nachfolgenden Schaltungskonfigurationen erfüllen aufgrund der höheren Induktivität die Bedingung $\delta^2 > \omega_0^2$ und schwingen somit aperiodisch mit $\vartheta > 1$. Die einmal angestoßene Schwingung wird dabei so stark gedämpft, dass sie die 0V-Marke nicht wieder berührt, sondern sich asymptotisch annähert, wobei

auch hier die oben definierte Auflösungsgrenze von einem tausendstel Volt gilt.

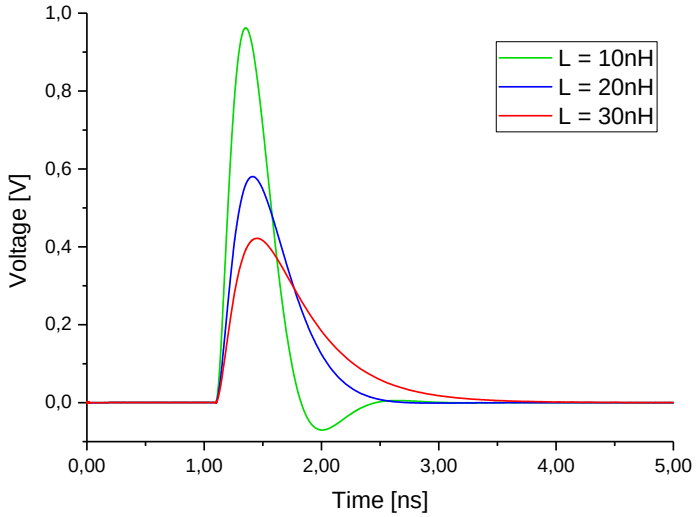

Abb. 4.9: Der Ausgangspuls des RLC-Glieds für verschiedene Induktivitäten bei konstantem Widerstand $R_2 = 50\,\Omega$ und konstanter Kondensatorkapazität $C_2 = 2.5$pF. Es wurden die resultierenden Kurven für die Induktivitäten $L = 10$nH, $L = 20$nH und $L = 30$nH aufgetragen.

Abschließend soll der aperiodische Grenzfall betrachtet werden. Kennzeichnend hierfür ist die Bedingung $\delta^2 = \omega_0^2$, die in keinem anderen Fall erfüllt werden kann. Somit gilt $\vartheta = 1$, was analog zu den bereits betrachteten Fällen durch verschiedene Bauteilkombinationen erfüllbar ist. Eine mögliche resultierende Pulsform ist in Abbildung 4.10 dargestellt.

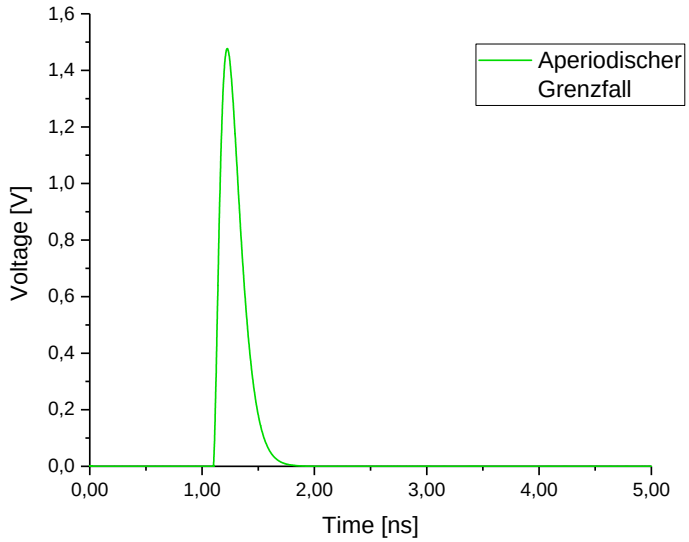

Abb. 4.10: Der Ausgangspuls des RLC-Glieds für den aperiodischen Grenzfall. Die verwendete Bauteilkombination besteht aus $R_2 = 30\,\Omega$, $C_2 = 1$pF und $L = 3.6$nH.

Zusammenfassend lässt sich sagen, dass die aus dem aperiodischen Schwingungsfall und dem aperiodischen Grenzfall resultierenden Pulsformen der in Abb. 3.1 gezeigten Pulsdarstellung durch eine Log-Normal-Verteilung bereits sehr nahe kommen. Abbildung 4.11

stellt beide Pulse vergleichend gegenüber. Der simulierte Puls wurde durch Variation von U_0 an die Amplitude des Realpulses angepasst. Da er, wie bereits beschrieben, auf einer elektrischen Entladung basiert, wird das Amplitudenmaximum bereits wenige Nanosekunden nach Wegfall der Versorgungsspannung erreicht. Zum Erreichen einer Vergleichbarkeit zwischen den beiden Pulsen wurden die Simulationsergebnisse deshalb translatorisch in der Zeit verschoben, wogegen die Pulsform invariant ist.

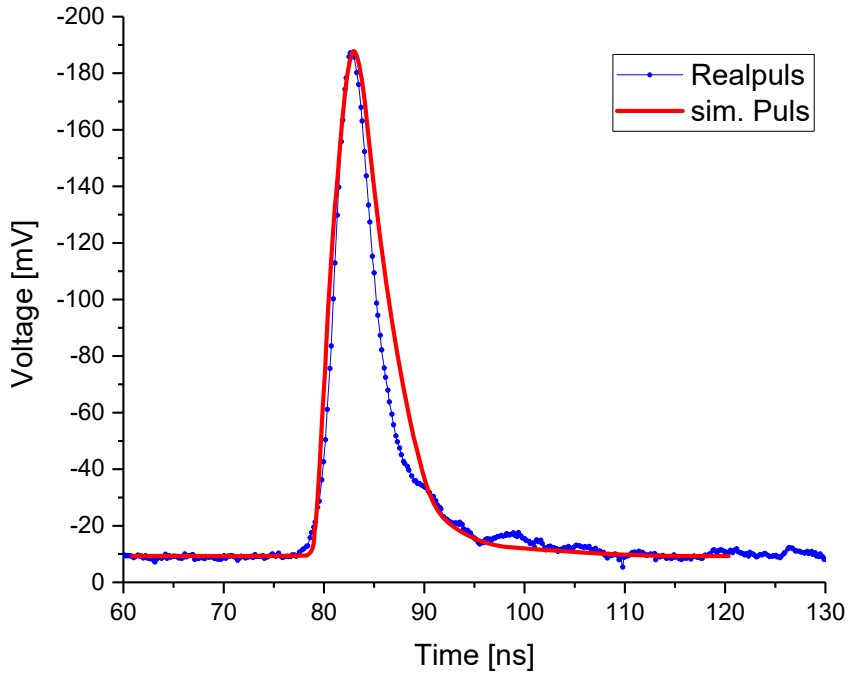

Abb. 4.11: Direkter Vergleich zwischen simuliertem und realem Puls. Obwohl besonders an der fallenden Flanke noch deutliche Abweichungen bestehen, ist die grundsätzliche Ähnlichkeit gegeben und die Simulationsmethodik somit verifiziert.

Im Folgenden soll die eben durchgeführte Dimensionierung verfeinert und das theoretische Modell in Software implementiert werden. Zudem soll ein dynamisch adaptives Simulationsverfahren entwickelt werden, welches vom Benutzer spezifizierte Vorgaben autonom umsetzt.

4.2 Pulssimulation mittels Software

Die hier entwickelte Software *COSMOPALS* soll die besprochenen Lebensdauerpulse unter Anwendung der elektrotechnischen Modellierung aus den vorherigen Abschnitten simulieren. Zudem sollen vom Benutzer spezifizierte Anforderungen eigenständig umgesetzt werden. Ziel und Motivation von *COSMOPALS* ist es, Einflüsse von Elektronikkomponenten im Spektrometeraufbau analysieren und simulieren zu können. Dabei sollen zwei verschiedene Endversionen entstehen: Zum einen soll eine eigenständige Stand-Alone-Version mit den nachfolgend erläuterten Grundfunktionen erhältlich sein, zum anderen soll *COSMOPALS* modular in das in Abschnitt 3 vorgestellte Softwarepaket *DDRS4PALS* eingebunden werden. Das Anwenderdiagramm 3.2 erweitert sich somit zu Abbildung 4.12.

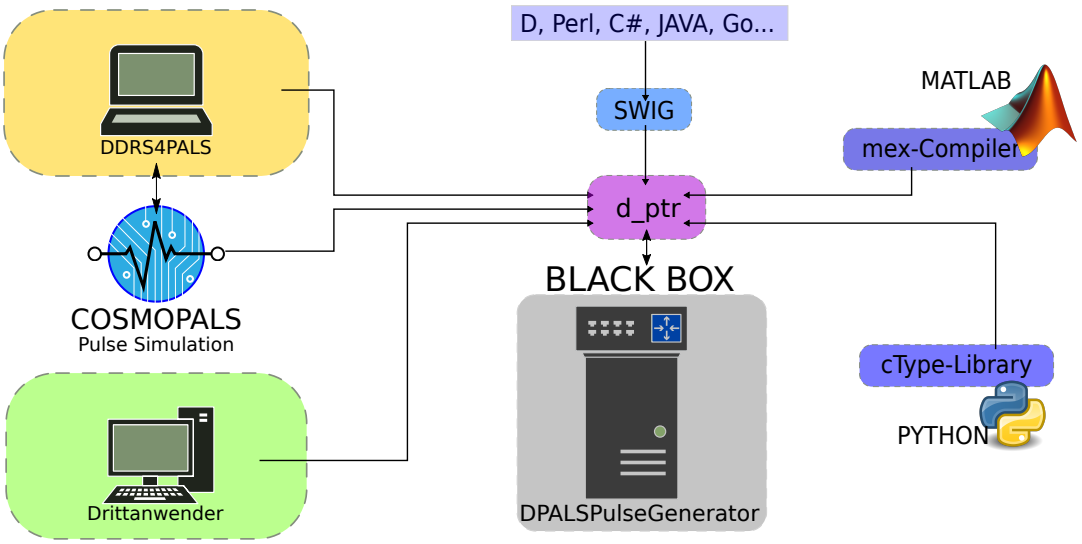

Abb. 4.12: Die Anbindung von COSMOPALS im Netzwerkdiagramm.

Analog zu *DDRS4PALS* greift auch *COSMOPALS* über den `d_ptr` auf die Arbeiterklasse `DPALSPulseGenerator` zu. Da *COSMOPALS* mit der Cross-Platform IDE Qt Creator entwickelt wurde, benutzt die Software im Gegensatz zu *DDRS4PALS* zusätzlich Elemente und Funktionen des Qt-Frameworks. Als einzig anderes externes Framework wird QCustomPlot von E. Eichhammer zur Pulsabbildung verwendet. Auf die Verwendung weiterer externer Bibliotheken, wie beispielsweise den BOOST-Libraries, wurde bewusst verzichtet, um die Cross-Plattform-Kompatibilität zu bewahren und um den Sourcecode übersichtlich zu halten. *COSMOPALS* wurde für Windows unter Verwendung von `MinGW Ver.5.3.0` kompiliert. Abbildung 4.13 zeigt die graphische Oberfläche der Anwendung und die im Folgenden erläuterten Teilbereiche.

Abb. 4.13: Die Benutzeroberfläche von COSMOPALS wurde mit dem Design-Tool Qt Designer erstellt. Sie ist in vier verschiedene Abschnitte unterteilt, die nach Funktionalität geordnet sind.

4.2.1 Bereich 1: Hardwarekomponenten

Bereich eins besteht aus insgesamt sechs verschiedenen Schiebereglern, die dazu dienen, die Hardwarekomponenten der beiden Schaltkreise anzupassen. Die obige Optionsschaltfläche bestimmt die dabei zugrunde gelegte Schaltung, deren Elemente durch jeweils drei Regler repräsentiert werden. Wird „RC Circuit“ ausgewählt, stehen die Schieberegler für jene Bauteile, deren Indizes in den Gleichungen für die Kondensatorspannung mit 1 gekennzeichnet wurden. Zudem ändert sich der unterste Schieberegler, der bei der Auswahl „RLC Circuit“ die Spule L verändert, zum Eingabeelement für die Eingangspannung U_0 des RC-Glieds. Die momentan angewählte Schaltung ist zusätzlich auf dem gezeigten Bild zu sehen. Erwähnenswert ist, dass die Schieberegler auf einen bestimmten Wertebereich beschränkt sind, um die Anzahl an möglichen (und sinnvollen) Eingabekombinationen einzuschränken. Das einstellbare Intervall ist über dem jeweiligen Regler abzulesen.

Der aktuell gewählte Wert wird mit Positionsänderung des Schiebereglers im jeweiligen rechten Anzeigefeld aktualisiert. Die Regler sind dabei durch Ziehen mit der Maus kontinuierlich veränderbar, durch Klicken auf die Skala wird der Wert um zehn Einheiten verändert. Mit dem am unteren Rand des ersten Bereichs zu sehenden Button „Reset“ (Alt+R) werden alle Eingaben, Schalterstellungen, Oberflächenelemente und Anzeigen auf Standardwerte zurückgesetzt. Als Default wurde hier mit der in Abb. 4.13 gezeigten Bauteilkombination der periodische Schwingungsfall gewählt.

4.2.2 Bereich 2: Pulssimulation

Bereich zwei zeigt den für den Start der Pulssimulation relevanten Bereich. Mit den Schaltflächen „Simulate Pulse (SW)“ und „Simulate Pulse (HW)“ (Alt+S) können Lebensdauerpulse entweder durch Softwaresimulation (SW) oder Hardwaresimulation (HW) erzeugt werden. Nach Klick auf die entsprechende Schaltfläche wird die Simulationsroutine aktiv: Im Software-Modus wird der Puls von *DDRS4PALS* mittels Geräteparametern und logarithmischer Normalverteilung simuliert (vgl. Abschnitt 3), im Hardware-Modus entspricht die Simulation der in diesem Kapitel vorgestellten Modellierung anhand der elektrischen Schaltung.

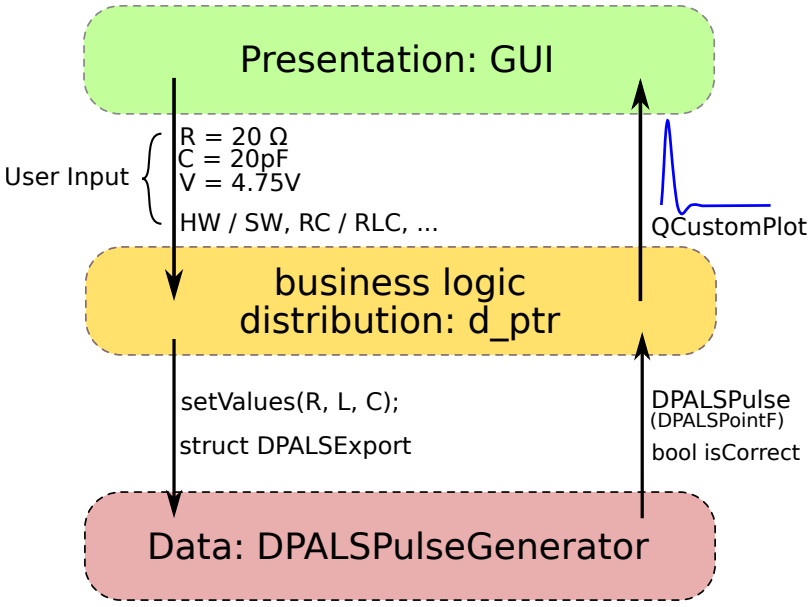

Abb. 4.14: Die 3-Layer-Architecture von COSMOPALS. Nach [Kam10].

COSMOPALS realisiert das Konzept der Dreischichtenarchitektur. Die oberste Schicht, das Presentation Layer, besteht hierbei aus der graphischen Oberfläche (GUI). Wie in den Erläuterungen zu Bereich 1 bereits beschrieben können hier die Hardwareparameter definiert werden. Alternativ sind über den Button „Edit SW params“ (Alt+P) in Bereich zwei einige der Geräte- und Verteilungsparameter der logarithmischen Simulationsmethodik (bspw. FWHM, rise time, Amplitude, etc.) einstellbar. Diese Daten erhält COSMOPALS je nach Simulationsmethode durch einen Eingabedialog oder die Schieberegler; zunächst als Absolutwerte. Nachfolgend werden sie in SI-Einheiten konvertiert, durch eigens definierte Setter über den `d_ptr`, welcher symbolisch als zweite Schicht (business/-distribution layer) dargestellt wird, in das Struct `DPALSExport DPALSDeviceInfo` eingefügt und an den `DPALSPulseGenerator` übergeben. Wird im Softwaremodus simuliert, wird ein neuer `DPALSPulseGenerator` mit den übergebenen Parametern erstellt, der den bereits Existenten ersetzten. Das verwendete Makro `DPALSExport` steht für den Microsoft-

spezifischen Speicherklassenmodifikator `__declspec(dllexport)`, welcher es dem Compiler ermöglicht, Exportdirektiven von .dll-Libraries zu Objektdateien hinzuzufügen, sodass Daten, Funktionen und Klassen direkt aus einer .dll-Datei exportiert werden können [Mic].

Die dritte, unterste Ebene besteht aus dem `DPALSPulseGenerator`. Je nach Simulationsmodus (SW/HW) werden die Methoden `emitPulse(...)` oder `emitRealPulse(...)` aufgerufen, die die mit den Simulationsparametern aktualisierten Structs als Eingabeparameter enthalten. Mit den in Abschnitt 3, bzw. 4.1.2, vorgestellten Ansätzen werden die aufeinanderfolgenden Punkte berechnet, die am Ende den Lebensdauerpuls darstellen, wobei ein Punkt als `DPALSPointF` bezeichnet wird und aus x- und y-Koordinate in Gleitkommadarstellung besteht. Nach Berechnung aller Punkte gibt die aufgerufene Routine den entstandenen `DPALSPulse`, einen Vektor aus den berechneten `DPALSPointF`s und der Maximalamplitude, zurück. Über den `d_ptr` erreicht dieser wieder die Mittelebene, wo er auf Korrektheit geprüft (dazu später mehr) und anschließend an das Presentation Layer weitergerreicht wird. QCustomPlot stellt den erhaltenen Puls anschließend in einem `QCPGraph` dar.

Ein von *COSMOPALS* erstellter `DPALSPulse` besteht standardmäßig aus 1024 Punkten, deren zeitlicher Abstand auf 10 Picosekunden festgelegt wurde. Die Software erreicht somit insgesamt eine darstellbare Pulsdauer von 10,24 Nanosekunden. Da dies bei stark oszillierendem Systemverhalten (beispielsweise für Werte von $C_2 > 30$) nicht genügt, um den kompletten Puls abzubilden, lässt sich die Pulsgröße mittels der Checkbox „Dynamic Pulse Size“ anpassen. Bei aktivem Kontrollkästchen werden Pulse mit einer Anzahl an Punkten produziert, die einem ganzzahligen Vielfachen von 1024 entspricht. Die mögliche Zeitdarstellung wird also zunächst verdoppelt, anschließend verdreifacht, et cetera. Dies wird solange fortgeführt, bis die letzten Pulspunkte unterhalb eines Tresholds von 0.01 Millivolt liegen, die Schaltung also so stark gedämpft wurde, dass keine Schwingvorgänge mehr erkennbar sind. Die dynamische Pulserzeugung sollte jedoch nur verwendet werden, wenn der Benutzer wirklich am Zeitverhalten in Intervallen von mehr als 10 Nanosekunden interessiert ist, da die Berechnungsroutine erwartungsgemäß schnell ressourcen- und zeitintensiv wird: Bereits bei einem C_2-Wert von beispielsweise 50pF werden anstelle der ursprünglichen 1024 Punkte 37.888 Datenpunkte errechnet, was 37 Inkrementierungsschritten entspricht. Meist genügt jedoch die steigende Flanke des ersten Pulsmaximums, um Aussagen über die Pulsgüte treffen zu können.

Die Checkbox „clear previous pulse“ bestimmt bei mehrfacher Simulation, ob die Ergebnisse des vorherigen Simulationslaufs erhalten bleiben sollen. Mit dem Kontrollkästchen „Show Pulse Points“kann der Benutzer die einzelnen Datenpunkte in einem Puls anzeigen lassen. Abbildung 4.15 zeigt ein Beispiel eines Pulses mit sichtbaren Pulspunkten.

Die übrigen Elemente in Bereich zwei sind zum einen der Button „Export as CSV“(Alt+E),

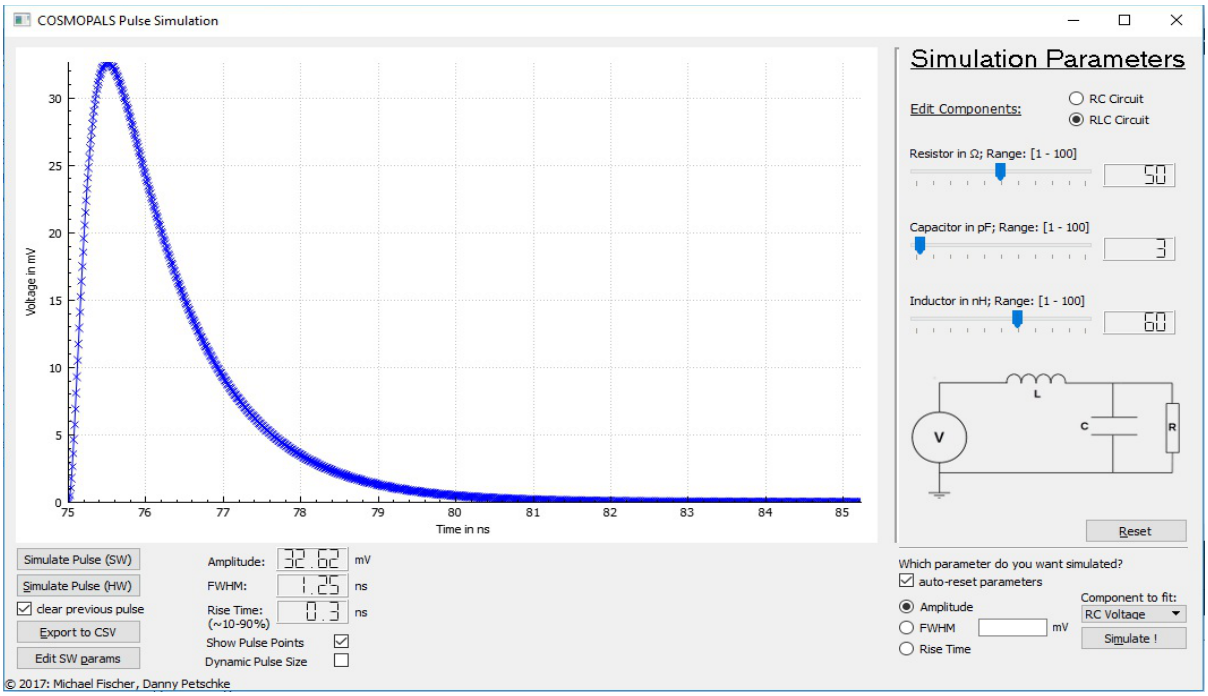

Abb. 4.15: Ein Puls mit sichtbaren Pulspunkten. Die Bauteilkombination schwingt im aperiodischen Fall. Die Pulsdaten auf der Anstiegsflanke sind trotz zeitlicher Gleichverteilung weiter voneinander entfernt als die restlichen Pulspunkte.

der die Pulspunkte und die für deren Erstellung benutzte Komponentenkombination als Textdatei ausgibt. Dabei wird die CSV-Datei mit dem jeweiligen Datum gekennzeichnet, falls im gewählten Verzeichnis noch keine Datei desselben Namens existiert. Ansonsten wird der Benutzer aufgefordert, sich zwischen der Inkrementierung des Dateinamens oder der Löschung der bereits existenten Datei zu entscheiden. Sinnvoll ist ein Export zu CSV vor allem für die Weiterverarbeitung und Visualisierung der Pulsdaten in anderen Programmen, wie beispielsweise Matlab oder Origin. Zusätzlich zu der Exportmöglichkeit stehen dem Benutzer drei untereinander angeordnete Anzeigetafeln zur Verfügung. Diese zeigen im Single-Plot Modus die jeweiligen Pulscharakteristika Amplitude, FWHM und rise time. Im Multi-Plot-Modus (ohne „clear previous pulse“) sind die Anzeigen deaktiviert, da die gezeigten Werte dann nicht mehr eindeutig zuzuordnen sind.

Die angezeigte Amplitude ergibt sich dabei direkt aus dem Maximumsattribut des dargestellten `DPALSPulse`. Die Berechnung von rise time (definiert als Zeitintervall zwischen 10% und 90% des Pulsmaximums) und FWHM wird rekursiv ausgeführt. Grund dafür ist die Tatsache, dass die Gleichungen 4.6 bis 4.8 transzendent sind und nicht explizit nach der Zeit t umgestellt werden können. Ohne diese Umformung sind exakte Zeitmarken für gegebene Funktionswerte nicht analytisch ermittelbar. Da diese Zeitpunkte $(t_{rise1}, t_{rise2}, t_{fwhm1}, t_{fwhm2})$ zur Berechnung der Anstiegszeit und Halbwertsbreite jedoch unabdingbar sind, wurde ein alternatives Verfahren erdacht. Anstelle von aufwendigen

Näherungs- oder Interpolationsalgorithmen benutzt *COSMOPALS* rekursiv angepasste Vergleichslogik, um die besten Werte zu ermitteln. Der verwendete Algorithmus ist nachfolgend in Pseudocode dargestellt. Er wird benötigt, da die Pulspunkte zwar zeitlich gleichverteilt sind, auf Teilen mit großen Amplitudenunterschieden zwischen den Pulspunkten weniger dicht beisammen liegen als auf Teilen, die sich beispielsweise langsam der Horizontalen annähern. Dies ist besonders für schnell ansteigende Pulse der Fall (vgl. Abb. 4.15). Zusammen mit der Tatsache, dass der resultierende Puls keine kontinuierliche Kurve, sondern eine Menge aus diskreten Punkten ist, führt dies dazu, dass die Stelle, an der die tatsächliche Halbwertsbreite (oder Anstiegszeit) zu finden ist, meist nicht mit einem Pulspunkt zusammenfällt. Um den exakten Wert wird deshalb ein Akzeptanzintervall gelegt, innerhalb welchem dann ein Punkt gefunden werden kann. Ein Ansatz, der dieses Problem verringern würde, wäre die dynamische Festlegung des Zeitintervalls zwischen zwei Punkten. Da dies aber für jede erdenkliche Bauteilkombination und Pulsform durchgeführt werden müsste, wird hier die einfachere, statische Schrittweite gewählt.

```
1  double FWHM = getFWHM(pulse, treshold, pulse.max/2)
2
3  getFWHM(pulse, treshold, goal) {
4    for every DPALSPointF in pulse do
5      if(pulse.y < goal+treshold && pulse.y > goal-treshold) {
6        exists = true;
7        val = pulse.x;
8      }
9    if(exists) return val;
10   else val = getFWHM(pulse, treshold+step, goal);
11 }
```

Algorithmus 4.1: Rekursive Berechnung von FWHM und rise time (stark vereinfacht).

Zu Beginn legt die Software einen Treshold von 0.1 Millivolt fest. Anschließend wird in der chronologisch geordneten, diskreten Menge von Pulspunkten überprüft, ob die geforderte Bedingung (bswp. FWHM-Berechnung: $U(t) \stackrel{!}{=} U_{max}/2$) von einem Pulspunkt innerhalb der Treshold-Grenzen erfüllt wird. Ist dies nicht der Fall, wird der Treshold in den folgenden Rekursionsläufen sequentiell erhöht, bis die Bedingung erfüllt ist und die zugehörige Zeit t dem gefundenen Punkt entnommen werden kann. Auf diese Weise erhält der Algorithmus die Start- und Entpunkte der Zeitmessung für die Halbwertsbreite oder die Anstiegszeit. Subtraktion liefert den gewünschten Parameter. Die Tatsache, dass die Genauigkeit der so errechneten Werte mit zunehmendem Treshold abnimmt, fällt aufgrund der insgesamt ausreichend dicht verteilten Pulsdatenpunkte nicht weiter ins Gewicht.

4.2.3 Bereich 3: Adaptive Pulserzeugung

Bereich drei bietet die Möglichkeit, Pulse nach vom Nutzer gesetzten Vorgaben generieren zu lassen. Dazu muss zunächst ein Simulationsparameter (Amplitude, FWHM, rise time) aus dem Optionsfeld und die anzupassende Komponente aus dem Drop-Down-Menü ausgewählt werden. Anschließend kann der gewünschte Parameterwert eingegeben und durch einen Klick auf die Schaltfläche „Simulate !“ (Alt+M) bestätigt werden. Da das resultierende Systemverhalten von allen verwendeten Bauteilen auf unterschiedliche Weise beeinflusst wird (vgl. Gl. 4.6 - 4.8), beschränkt sich der Simulationsvorgang hier auf die schrittweise Veränderung der ausgewählten Schaltungskomponente. Die anderen Bauteile werden dabei als konstant angenommen und nicht weiter verändert.

Anstelle der simplen Brute-Force-Methode, deren Rechenaufwand proportional mit der Anzahl der möglichen Eingabekombinationen wächst, benutzt *COSMOPALS* den effizienteren Bisection-Algorithmus in leicht veränderter Form. Der eigentlich für die iterative Berechnung von Polynomnullstellen gedachte Algorithmus ähnelt in seiner Funktionsweise dabei einer binären Suche, deren Rechenaufwand im schlimmsten Fall logarithmisch zur Zahl der Eingabemöglichkeiten ist [Knu71]. Um den Anforderungen von *COSMOPALS* gerecht zu werden wurde das Bisection-Verfahren leicht verändert und rekursiv implementiert. Der folgende, vereinfachte Pseudocode zeigt die Funktionsweise des Algorithmus am Beispiel der Suche nach einer gegebenen Amplitude.

```
1  double result = bisection(component, step, goal, tol, nMax, i) {
2
3    DPALSPulse pulse = getRealPulse(...);
4    if(i > nMax) return pulse;
5
6    if(pulse.max < goal) {
7      while (pulse.max < goal) {
8        component += step;
9        pulse = getRealPulse(...);
10     }
11   } else if(pulse.max != goal) {
12     while(pulse.max > goal) {
13       component -= step;
14       pulse = getRealPulse(...);
15     }
16   }
17
18   if(pulse.max > goal-tol && pulse.max < goal+tol) return pulse;
19   else pulse = bisection(component, step/2, goal, tol, nMax, i+1);
20 }
```

Algorithmus 4.2: Bisection-Algorithmus zur Anpassung selektierter Bauteile.

Der Algorithmus bekommt den zu variierenden Parameter (component), die dynamisch aus der vom Nutzer gestellten Forderung errechnete Schrittweite (step), das vom Benutzer gewünschte Simulationsergebnis (goal), die Genauigkeit der Simulation (tol) und die maximale, sowie die aktuelle Anzahl an Rekursionsaufrufen (nMax, i) als Eingabeparameter. Zu Beginn jedes Funktionsaufrufs wird ein Puls aus der aktuellen Bauteilkombination erstellt, dessen Parameter über den weiteren Verlauf des Algorithmus entscheiden. Ist die maximal zulässige Anzahl an Rekursionsaufrufen erreicht, bricht die Routine ab und der Puls wird ausgegeben. Andernfalls beginnt das Bisectionsverfahren: Im obigen Pseudocode soll die Pulsamplitude angepasst werden, der Algorithmus vergleicht also, ob der momentane Wert unter oder über der vom Benutzer definierten Spannung liegt. Die zu variierende Komponente wird solange in die entsprechende Richtung verändert, bis die Amplitude des aus dieser Veränderung resultierenden Pulses den Zielwert erreicht und passiert hat (vgl. Alg. 4.2, Z. 6-16). Anschließend wird überprüft, ob der momentane Wert den Forderungen im Rahmen des Toleranzintervalls genügt. Bei positivem Vergleichsergebnis bricht die Simulationsroutine ab und der entstandene Puls wird ausgegeben, während die dazugehörige Bauteilkombination im Struct `DPALSDeviceInfo` gespeichert wird. Andernfalls wird nach Inkrementierung der aktuellen Anzahl von Funktionsaufrufen ein erneuter Bisectionslauf gestartet, der das Verfahren mit der nun halbierten Schrittweite durchführt. Auf diese Weise wird die Amplitude selbst bei großen Schrittweiten nach bereits wenigen Durchgängen zuverlässig angenähert. Die erreichbaren Genauigkeiten des Verfahrens resultieren aus dem Systemverhalten der Schaltkreise und sind somit von der verwendeten Komponente und dem zu simulierenden Parameter abhängig. Sie können Tabelle 4.1 entnommen werden.

Die in Algorithmus 4.2 vorgestellte Bisection-Methode ist jedoch nicht ohne Weiteres auf jede Simulationskonfiguration anwendbar. Spezielle Kombinationen aus Simulationsparameter und Schaltungskomponente führen zum exakt entgegengesetzten Systemverhalten und müssen gesondert behandelt werden. So führt beispielsweise die Erhöhung von L oder C_2 nicht zu einem Anstieg der Spannung, sondern aufgrund der in Abschnitt 4.1.1 beschriebenen Beziehungen zu einem Spannungsabfall. Der Bisection-Algorithmus muss diese Kondition also erkennen und das Verfahren invers durchführen. Ferner ist anzumerken, dass nicht alle Bauteil- und Parameterkombinationen im Zusammenspiel mit allen Benutzerforderungen möglich und sinnvoll sind. Ein Beispiel für einen physikalisch nicht sinnvollen Simulationslauf ist die Kombination aus zu skalierender Eingangsspannung und geforderter Pulsweite, da die Spannung den Puls nur in vertikaler Richtung skaliert, nicht

jedoch seine Halbwertsbreite verändert. Wird vom Nutzer eine solche Kombination eingegeben, weist *COSMOPALS* mit einem Hinweisdialog auf die Nichtdurchführbarkeit der Forderung hin. Die Simulationsmethodik zeigt dabei auffallend gute Genauigkeiten: Wird als zu simulierender Parameter die Anstiegszeit ausgewählt, errechnet der Simulator innerhalb seiner möglichen Grenzen Ergebnisse, die nur maximal 0.05 Nanosekunden von den geforderten Werten abweichen. Ferner kann die Halbwertsbreite auf 0.1 Nanosekunde und die Pulsamplitude auf bis zu 1 Millivolt genau simuliert werden. Simulationseingaben, die sich unterhalb dieser Genauigkeitsgrenzen befinden, sind nicht sinnvoll und werden von *COSMOPALS* mit einem Warnhinweis abgefangen. Alle möglichen Simulationskombinationen, Ober- und Untergrenzen und Genauigkeiten sind in folgender Tabelle zusammengefasst, wobei die aus den Schaltungskomponenten resultierenden Spannungsamplituden offensichtlich von der anliegenden Eingangsspannung abhängig sind. Um auch die untere Grenze sinnvoll testen zu können, wurde für die Erstellung der nachfolgenden Tabelle eine Eingangsspannung von 0.5V gewählt. Die Genauigkeitsangaben beziehen sich jeweils auf den erfolgreichen Simulationsfall.

		Amplitude [mV]		**FWHM [ns]**		**rise time**		**Accuracy**		
		lower	upper	lower	upper	lower	upper	[mV]	[ns]	[ns]
RC	R_1	1	~ 553	0.3	6.25	0.15	0.4	1	0.1	0.05
	C_1	1	~ 553	0.3	1.03	0.18	0.19	1		
	V	1	99 999	x	x	x	x	1		
RLC	R_2	20.5	~ 253	0.4	1.55	0.14	0.21	1		
	C_2	13.59	153.3	0.33	7.53	0.12	8.14	5		
	L	5	406.6	0.21	8.01	0.08	0.32	5		

Tab. 4.1: Maximal und minimal mögliche erreichbare Werte der Simulationsumgebung. Die Spannungsversorgung hat keinen Einfluss auf die Charakteristika eines Pulses, weshalb hier keine Werte angegeben wurden. Die Genauigkeit bezieht sich auf die einzelnen Simulationsparameter.

Im Hinblick auf Tabelle 4.1 ist anzumerken, dass die angegebenen Werte die minimal und maximal erreichbaren Grenzen darstellen, nicht die Eingaben, um diese Grenzen zu erreichen. Der Operator ~ steht dabei für ungefähre Werte, die nicht exakt ermittelbar sind. Oftmals entstehen solche Resultate bei an sich unmöglichen Simulationskonfigurationen, bei denen der Simulator versucht, das beste erreichbare Ergebnis zu simulieren. Beispielsweise entsteht bei einer Amplitudensimulation mit gewünschten 400mV und Bauteil R_2 eine Simulationsausgabe von 242.6mV, da der Bisection-Algorithmus nach Erreichen der maximalen Rekursionszahl bei einem Widerstand von $R_2 = 1061\Omega$ abbricht. Gibt man für den gewünschten Wert allerdings die Maximaleingabe von 99999mV ein, errechnet der Simulator dank der größer gewählten Schrittweite nun einen Wert von 253.3mV. Der Widerstand R_2 beträgt dabei bereits über 30kΩ. Da der Eingabedialog auf fünf Ziffern

beschränkt ist, ist dies die obere Grenze des mit R_2 erreichbaren Spannungswerts.

Tabelle 4.1 zeigt zudem, wie die verwendeten Elektronikkomponenten die Pulsparameter beeinflussen. So lassen sich mit R_1 und C_1 beispielsweise hohe Amplituden erreichen, da diese den Eingangspuls des RLC-Glieds beeinflussen. Der für die Anstiegszeit- oder Pulsweitensimulation bestgeeignete Parameter ist hingegen C_2. Besonders bei hohen gewünschten Werten von FWHM oder rise time - generell immer dann, wenn der Simulator keinen exakten Wert finden kann, sondern anstelle dessen den nach Ablauf aller Rekursionsebenen entstandenen Wert ausgibt - können aufgrund der doppelten Rekursion hohe Laufzeiten entstehen. Ebenfalls aus diesem Grund ist es nicht möglich, den Simulator Pulse mit dynamischer Größe (vgl. Abschnitt 4.2.2) erstellen zu lassen. Da diese Fälle jedoch nur einen marginalen Bruchteil aller Möglichkeiten darstellen und im realen Anwendungsfall nur selten vorkommen, stellt diese Restriktion keine wirkliche Beschränkung in der Anwendung von *COSMOPALS* dar.

Ein weiterer Spezialfall in der Pulserzeugung ergibt sich bei der Verwendung bestimmter Bauteilkombinationen: wählt man beispielsweise $C_2 = 1\text{pF}$, $C_1 = 100\text{pF}$, $R_2 = 50\Omega$, $R_1 = 1\Omega$ und $L = 10\text{nH}$, wird der in allen die Kondensatorspannung beschreibenden Gleichungen vorkommende Term $\delta - \frac{1}{\tau_1}$ zu null. Besonders fällt dies im Falle des aperiodischen Grenzfalls (vgl. Gl. 4.6) ins Gewicht, da dann eine Division durch null auftritt. Um dies zu verhindern, addiert *COSMOPALS* kleine Zahlen der Größenordnung $\times 10^{-9}$ zu kritischen Bauteilwerten hinzu. So wird die Division durch null verhindert, ohne dass der Wert allzu sehr verändert wird.

Ein bereits in Abschnitt 4.2.2 angesprochenes Problem ist die Korrektheit der von `DPALSPulseGenerator` ausgegeben Pulse. Für bestimmte Bauteilkombinationen resultieren die Terme e^{2at} und $e^{-\delta t - at}$ in einem Double-Over-, bzw. Underflow, da der Exponent die Zahl ± 308 überschreitet und der Wert vom Datentyp `double` mit 64 Bit nicht mehr dargestellt werden kann. Die Routine `emitRealPulse(...)` ist vom Rückgabetyp `bool` und prüft ab, ob die bei der Pulserzeugung entstandenen Werte die oberen oder unteren Schranken des Double-Wertebereichs überschreiten. Falls dies der Fall ist, wird `false` zurückgegeben. *COSMOPALS* füllt den Puls dann mit konstant null und kennzeichnet ihn als `badPulse`. So wird sichergestellt, dass nur korrekt erzeugte Pulse weiterverwendet und dargestellt werden. Ein Beispiel für einen `badPulse` ergibt sich mit der Schaltungskonfiguration $C_2 = 3\text{pF}$, $C_1 = 100\text{pF}$, $R_2 = 1\Omega$, $R_1 = 1\Omega$ und $L = 10\text{nH}$. Einige relevante Werte dieses Beispiels sind in Anhang B angegeben.

Das letzte Bedienelement in Bereich drei ist die Checkbox „auto-reset params“, welche dazu dient, die Hardwarekomponenten vor jedem Simulationslauf auf Standardparameter zurückzusetzen. Wird der Haken entfernt, simuliert *COSMOPALS* mit den zuvor errechneten Werten weiter, was gegebenenfalls zu unerwünschten Ergebnissen führt. Das

Rücksetzten ist deshalb standardmäßig aktiviert.

4.3 Bereich 4: Pulsdarstellung

Bereich vier widmet sich der graphischen Darstellung der aus den vorigen drei Bereichen erhaltenen Pulsdaten. Dazu wird das bereits erwähnte, frei zugängliche Framework QCustomPlot von E. Eichhammer verwendet.

Die vom `DPALSPulseGenerator` erzeugten Pulse werden zur Darstellung zu einem designierten `QCPGraph` hinzugefügt. Ein `QCPGraph` kann dabei nur einen Puls beinhalten. Zur klaren Abgrenzung von mehreren Pulsen im Multi-Plot-Modus verwendet `COSMOPALS` für die verschiedenen Graphen zufällig gewählte Farben. Das Plot-Widget bietet dem Benutzer dabei mehrere Möglichkeiten zur Interaktion: der Sichtausschnitt kann verschoben werden, der Graph kann vergrößert oder verkleinert werden und es können einzelne Graphen - beispielsweise zum Export - ausgewählt werden. Per Rechtsklick bieten sich dem Benutzer weitere Möglichkeiten zur Modifikation des Gezeigten: ist im Multi-Plot-Modus ein Puls ausgewählt, kann dieser mittels „Remove selected graph“ entfernt werden. Ferner können alle Pulse entfernt und die Achsen neu skaliert werden. Die Schaltfläche „Get Pulse Info“ bietet dem Nutzer die Möglichkeit, die im Single-Plot-Modus in den Anzeigetafeln gezeigten Pulsparameter zu erhalten. Zusätzlich wird die Hardwarekombination, die für die Pulserzeugung genutzt wurde, angegeben. Dazu benutzt *COSMOPALS* den STL-Container `std::multimap<...>`. Nötig wird dessen Verwendung aufgrund der Funktionsweise von QCustomPlot: Der vom Nutzer selektierte Puls wird intern als `QCPGraph` bereitgestellt und beinhaltet zwar die Pulsdaten, hat jedoch keine Relation zur pulserzeugenden Hardwarekombination. Diese Verbindung schafft *COSMOPALS* mit dem oben beschriebenen Speicherelement `multimap`, indem jeder Puls als Key und die zugehörige Bauteilkombination als Value gespeichert wird. Werden per Rechtsklick dann Informationen über die verwendete Hardware abgerufen, sucht *COSMOPALS* den selektierten Puls in der `multimap` und benutzt die bereitgestellte Key-Value-Verbindung zur Extraktion der Daten.

Ferner ist anzumerken, dass alle von *COSMOPALS* erzeugten Pulse standardmäßig um 75 Nanosekunden in der Zeit verschoben dargestellt werden. Dies erscheint zu Anfang kontraintuitiv, da das in Abschnitt 4.1.1 beschriebene Systemverhalten zum Zeitpunkt $t = 0$ mit der Kondensatorladung, beziehungsweise zum Zeitpunkt $t = t_0$ mit der Kondensatorentladung, beginnt. Die Verschiebung wurde gewählt, um eine bessere Vergleichbarkeit zwischen den durch die Log-Normalverteilung simulierten und den von *COSMOPALS* erzeugten Pulsen zu erreichen. Dabei wird vor der Darstellung durch QCustomPlot und der damit verbundenen Umwandlung zum `QCPGraph` der Wert 75 auf die x-Koordinaten der im darzustellenden `DPALSPulse` enthaltenen `DPALSPointF`s addiert.

Abbildung 4.16 zeigt einen software- und einen hardwaregenerierten Puls.

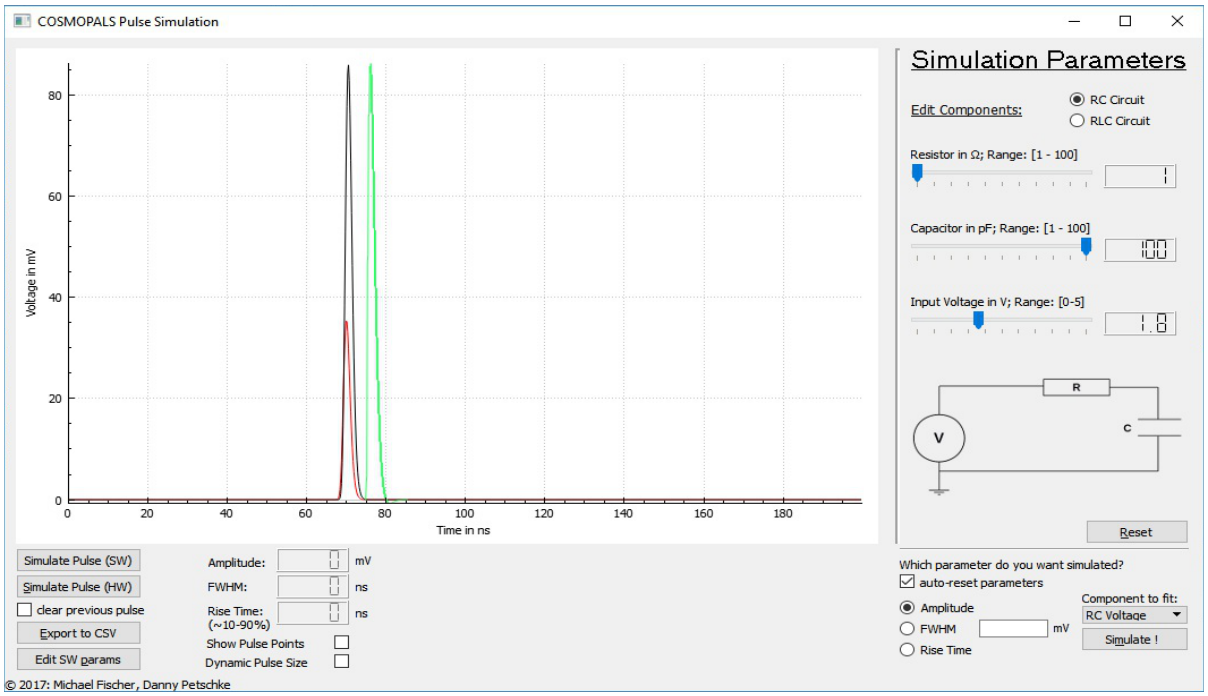

Abb. 4.16: Vergleichende Gegenüberstellung zweier softwaregenerierter Pulse (links) und eines hardwaregenerierten Pulses (rechts). Die für den Puls benutzte Bauteilkombination lautet: $R_1 = 1\Omega, C_1 = 100\text{pF}, V = 1.8\text{V}, R_2 = 7\Omega, C_2 = 73\text{pF}, L = 10\text{nH}$.

Die dargestellten Pulse sind nahezu identisch, der einzige Unterschied zeigt sich im leichten Überschwingen des Hardwarepulses nach Ende der fallenden Flanke. Der zeitliche Unterschied zwischen den Pulsen kann, wie eben beschrieben, bei den von *COSMPALS* erzeugten Pulsen beliebig angepasst werden, da diese invariant gegenüber einer Translation in der Zeit sind. Die softwaregenerierten Pulse resultieren aus den anfänglichen Standardeinstellungen des `DPALSPulseGenerators`, die im Makro `DPALSDEMO` im Sourcecode in Anhang A nachzulesen sind. Die durch *DDRS4PALS* simulierten Pulse werden mit den Einstellungen aus `DPALSDEMO` simuliert und können vom Benutzer mittels „Edit SW params“ angepasst werden (vgl. Abschnitt 4.2.2).

Die Pulssimulation mit den in diesem Kapitel vorgestellten Methoden führt also zu probaten Ergebnissen (vgl. Abb. 4.11 und Abb. 2.3). Im Folgenden soll nun versucht werden, die in diesem Kapitel theoretisch modellierten Ansätze durch eine in Hardware aufgebaute Schaltung in die Praxis zu überführen.

5 Hardwareseitige Simulation

5.1 Aufbau und Funktionsweise der realen Schaltung

bla

5.2 Messungen und Auswertungen der Ergebnisse

bla

5.3 Abweichungen von der Idealvorstellung

Wikipedia Rechteckspannung, Gibbsches RInging TODO

6 Zusammenfassung und Ausblick

ausblick

6.1 Anwendbarkeit der Methodik auf verschiedene Messeinrichtungen

6.2 Bau eines Entwicklungsboards als Gütetest von Geräten

7 Quellenverzeichnis

[Ada79] Adamowicz, T.: Handbuch der Elektronik. In: *München: Franzis-Verlag* (1979), S. 512

[BB63] Birks, LS ; Batt, AP: Use of a Multichannel Analyzer for Electron Probe Microanalysis. In: *Analytical Chemistry* 35 (1963), Nr. 7, S. 778–782

[Boh95] Bohm, Joachim: Realstruktur von Kristallen. (1995)

[BS97] Beuth, Klaus ; Schmusch, Wolfgang: *Grundschaltungen-Elektronik 3*. Vogel Verlag, 1997

[CI16] Ceramics, Saint-Gobain ; Inc., Plastics: *BC-418, BC-420, BC-422 Premium Plastic Scintillators*. 8 2016

[CR12] Callister, William D. ; Rethwisch, David G.: *Materialwissenschaften und Werkstofftechnik: Eine Einführung*. John Wiley & Sons, 2012

[Eic] Eichhammer, Emanuel: *QCustomPlot*. `http://www.qcustomplot.com`, Abruf: 24.07.2017

[HO13] Hüls, Ewald ; Oestern, Hans-Jörg: *Die ICE-Katastrophe von Eschede: Erfahrungen und Lehren Eine interdisziplinäre Analyse*. Springer-Verlag, 2013

[HP07] Hamamatsu Photonics, KK: Photomultiplier tubes: Basics and applications. In: *Edition 3a* 310 (2007)

[Kam10] Kambalyal, Channu: 3-tier architecture. In: *Retrieved On* 2 (2010)

[Kno10] Knoll, Glenn F.: *Radiation detection and measurement*. John Wiley & Sons, 2010

[Knu71] Knuth, Donald E.: Optimum binary search trees. In: *Acta informatica* 1 (1971), Nr. 1, S. 14–25

[KRA] Krause-Rehberg, R. ; Amarendra, G.: *Positron Annihilation*. `http://www.positronannihilation.net/index.htm`, Abruf: 15.07.2017

[KRL99] Krause-Rehberg, Reinhard ; Leipner, Hartmut S.: *Positron annihilation in semiconductors: defect studies*. Bd. 127. Springer Science & Business Media, 1999

[Mic] Microsoft: *Exportieren aus einer DLL mithilfe von declspec(dllexport)*. `https://msdn.microsoft.com/de-de/library/a90k134d.aspx`, Abruf: 24.07.2017

[Mus15] Mustermann, Max: *Musterbuch.* Verlag, 2015

[OK07] Olsen, Jens V. ; Kirkegaard: PALSfit: A new program for the evaluation of positron lifetime spectra. In: *physica status solidi (c)* 4 (2007), Nr. 10

[Pag61] Page, I: Constant Fraction Discriminator. In: *Signal* 3 (1961), S. 1

[Pet17] Petschke, D: A new Method for Lifetime-Simulation. (2017), 3

[Phi07] Philips: *Photomultiplier XP2020.* 8 2007

[PSS14] Petriska, M ; Sojak, S ; Slugeň, V: Positron lifetime setup based on DRS4 evaluation board. In: *Journal of Physics: Conference Series* Bd. 505 IOP Publishing, 2014, S. 012044

[RN79] R.M. Nieminen, J. L.: *Applied Physics A.* Springer-Verlag, 1979

[Sch01] Scheid, S: Die verallgemeinerte Lognormalverteilung. In: *Univ., Dortmund* (2001)

[Str58] Strunz, Hugo: Mineralogische Tabellen. In: *GFF* 80 (1958), Nr. 1, S. 131–132

[SW92] Schatz, Günter ; Weidinger, Alois: Positronenvernichtung. In: *Nukleare Festkörperphysik: Kernphysikalische Meßmethoden und ihre Anwendungen* (1992)

[Tan08] Tang, F.: *Modeling for MCP-PMT Output Signal.* Universität Chicago, 2008

[Wol06] Wolf, Jürgen: C++ von A bis Z. In: *Galileo Computing* (2006)

[WU07] Weber, Hubert ; Ulrich, Helmut: *Laplace-Transformation: Grundlagen-Fourierreihen und Fourierintegral-Anwendungen.* Springer-Verlag, 2007

Anhang

A Sourcecode

hier kommt der Code hin.

B Überlaufprotokoll

t: 0 - exp(-delta*t): 1
t: 0 - exp(2at): 1
t: 1e-11 - exp(-delta*t): 0.0357097
t: 1e-11 - exp(2at): 27.9756
t: 2e-11 - exp(-delta*t): 0.00127518
t: 2e-11 - exp(2at): 782.634
t: 3e-11 - exp(-delta*t): 4.55364e-05
t: 3e-11 - exp(2at): 21894.7
t: 4e-11 - exp(-delta*t): 1.62609e-06
t: 4e-11 - exp(2at): 612516
t: 5e-11 - exp(-delta*t): 5.80672e-08
t: 5e-11 - exp(2at): 1.71355e+07
t: 6e-11 - exp(-delta*t): 2.07356e-09
t: 6e-11 - exp(2at): 4.79376e+08
t: 7e-11 - exp(-delta*t): 7.40462e-11
t: 7e-11 - exp(2at): 1.34108e+10
(...)
t: 2.1e-09 - exp(-delta*t): 1.21644e-304
t: 2.1e-09 - exp(2at): 6.66314e+303
t: 2.11e-09 - exp(-delta*t): 4.34388e-306
t: 2.11e-09 - exp(2at): 1.86405e+305
t: 2.12e-09 - exp(-delta*t): 1.55119e-307
t: 2.12e-09 - exp(2at): 5.2148e+306
t: 2.13e-09 - exp(-delta*t): 5.53924e-309
t: 2.13e-09 - exp(2at): 1.45887e+308
t 2.14e-09 - exp(-delta*t): nan
t 2.14e-09 - exp(2at): inf

Erklärung

Hiermit versichere ich, dass ich meine Abschlussarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Datum: .....

(Unterschrift)