Entwicklung eines Echtzeit-Strahlprofil-Monitoring-Systems für das COMPASS-II Experiment

Christoph Michalski

Fakultät für Mathematik und Physik
Albert-Ludwigs-Universität Freiburg

Entwicklung eines Echtzeit-Strahlprofil-Monitoring-Systems für das COMPASS-II Experiment

Diplomarbeit

vorgelegt
von

Christoph Michalski

Physikalisches Institut
Albert-Ludwigs-Universität
Freiburg

März 2013

Inhaltsverzeichnis

1. Einleitung

Seit Menschengedenken findet die Suche nach den kleinsten Bausteinen der Materie statt. Schon am Ende des 19. Jahrhundert konnte gezeigt werden, dass Atome nicht unteilbar sind, was neue Fragen nach den kleinsten Materiebausteinen aufwarf. Wesentlich dazu beigetragen haben Experimente von E. Goldstein und J. Thomson. Goldstein gelang es 1886, Atomen Elektronen zu entreißen, womit auf die Existenz von geladenen Teilchen innerhalb nach außen hin neutraler Atome geschlossen werden konnte [1]. Etwa zehn Jahre später konnte Thomson die negativ geladenen Elektronen mit der Kathodenstrahlröhre nachweisen [2]. Zum damaligen Zeitpunkt herrschte jedoch noch Unklarheit über die Massenverteilung innerhalb der Atome. Im Jahr 1911 konnte E. Rutherford durch Streuversuche mit Helium-Kernen an dünnen Goldfolien die damals vorherrschende Vermutung widerlegen, die Bestandteile der Atome seien „wie Rosinen in einem Kuchenteig“ verteilt. Diese besitzen einen massiven Kern, umgeben von einer fast masselosen Hülle, der Elektronenhülle [3].

Heute wissen wir, dass die sichtbare Materie des Universums größtenteils aus Elektronen und Nukleonen[1] besteht. Elektronen $e^{\pm}$ sind Elementarteilchen und gehören der Gruppe der Leptonen an. Weitere Vertreter sind Myonen $\mu^{\pm}$ und Tauonen $\tau^{\pm}$, die sich von den Elektronen nur durch ihre Massen unterscheiden. Sie wechselwirken elektromagnetisch und schwach. Zusammen mit den schwach wechselwirkenden Neutrinos ν_e, ν_μ, ν_τ beziehungsweise deren Antiteilchen werden die Leptonen in drei Generationen, (l, ν_l) mit $\mathrm{l} = \{e, \mu, \tau\}$, zusammengefasst.

Im Gegensatz dazu gehören Nukleonen zur Gruppe der Hadronen, welche einen Zusammenschluss von mehreren Elementarteilchen darstellen, die der starken Wechselwirkung unterworfen sind. Diese lassen sich in zwei Untergruppen, Mesonen und Baryonen, unterteilen. Während Mesonen aus einem Quark-Antiquark-Paar aufgebaut sind und ganzzahligen Spin tragen, bestehen Baryonen aus drei Quarks. Die Addition der einzelnen Quark-Spins resultiert im Falle der Baryonen in einen halbzahligen Spin.

[1] Nukleonen bezeichnen die massiven Bausteine des Atomkerns, also Protonen und Neutronen.

Messungen des anomalen magnetischen Moments des Nukleons [4, 5], waren erste Hinweise auf eine Substruktur desselben. Heute ist bekannt, dass neben dem Nukleon noch viele andere Hadronen existieren. Diese Erkenntnis hat man vor allem Experimenten mit Teilchenbeschleunigern und Messungen der Höhenstrahlung zu verdanken [6]. Die Beschreibung der Wechselwirkungen zwischen bekannten Elementarteilchen und deren Einordnung in Teilchenfamilien ist unter dem Begriff „Standardmodell“ bekannt.

Unabhängig voneinander postulierten George Zweig und Murray Gell-Mann 1964, dass Hadronen ihrerseits aus noch kleineren Teilchen bestehen, welche „Quarks“ genannt wurden [7, 8]. Dadurch war eine Schematisierung des hadronischen „Teilchenzoos“ möglich, ähnlich der Einordnung chemischer Elemente in das Periodensystem. Ende der 1960er Jahre erfolgten erstmals Messungen zur Untersuchung von Nukleonen mittels tief-inelastischer Elektron-Nukleon-Streuung, womit die Existenz punktförmiger Teilchen innerhalb von Hadronen bestätigt werden konnte. Diese Teilchen wurden Partonen genannt.

Heute ist bekannt, dass das Nukleon aus drei „Valenzquarks“ und mehreren Gluonen und Seequarks aufgebaut ist. Gluonen sind im Gegensatz zu Quarks elektrisch neutral. Sie bilden die Austauschteilchen der starken Wechselwirkung und koppeln an alle Teilchen die sogenannte Farbladung tragen. Dazu gehören sowohl Quarks als auch Gluonen selbst [9]. Valenzquarks bestimmen die Quantenzahlen des Hadrons, wie beispielsweise die elektrische Ladung oder die Parität. Seequarks sind virtuelle Quark-Antiquark-Paare die durch Gluonen gebildet werden. Durch ihre Erzeugung und Vernichtung kommt es innerhalb des Hadrons zu Vakuumfluktuationen. Diese Paare bilden einen „See“ um das Valenzquark in dessen unmittelbarer Umgebung der erwähnte Prozess stattfindet. Zwar tragen Seequarks nicht zu den Quantenzahlen des Hadrons bei, jedoch liefern sie einen Beitrag zu dessen Spin.

Mit dem EMC[2]-Experiment am CERN[3] konnte Ende der 1980 er Jahre widerlegt werden, dass sich der Nukleonspin nur aus den Spinbeiträgen seiner Quarks und Antiquarks zusammensetzt, da deren Spinbeiträge zu gering sind [10, 11]. Dies wurde als sogenannte „Spinkrise“ bekannt. Auch der Spinbeitrag der Gluonen reicht nicht aus, um das „Spinpuzzle“ zu lösen, was präzise Messungen der Gluonpolarisation am COMPASS[4]-Spektrometer ergaben. Bis heute ist die genaue Zusammensetzung der Spinstruktur des Nukleons noch unverstanden.

Eine Lösung des Spinpuzzles könnte das Konzept der Parametrisierung der hadronischen Wechselwirkung durch generalisierte Parton-Verteilungsfunktionen darstellen. Generalisierte Parton-Verteilungsfunktionen können über exklusive Prozesse, beispielsweise der tief-virtuellen Compton-Streuung (DVCS[5]), vermessen werden. Damit könnte eine Bestimmung totaler Drehimpulse von Quarks und Gluonen möglich sein.

Eine Beschreibung der theoretischen Grundlagen zur Untersuchung der Spinstruktur des Nukleons erfolgt in Kapitel 2. Das COMPASS-II Experiment ist ein „Fixed-

[2]EMC = European Muon Collaboration
[3]CERN = Conseil Européen de la Recherche Nucleaire
[4]COMPASS = Common Muon and Proton Apparatus for Structure and Spectroscopy
[5]DVCS = Deep Inelastic Compton Scattering

Target-Experiment“ und besteht aus mehreren Detektorkomponenten. Der Aufbau des Spektrometers zur Vermessung exklusiver Prozesse wird in Kapitel 3 erläutert.

Für die Verarbeitung der Detektorsignale werden teilweise sehr hohe Ansprüche an die Auslese-Elektronik gestellt. Dafür werden oftmals FPGAs[6] [12] eingesetzt. Dabei handelt es sich um integrierte Schaltkreise, in denen eine spezifische Funktion einprogrammiert werden kann. Vor allem ihre freie Rekonfigurierbarkeit macht FPGAs für den Einsatz in Teilchenphysik-Experimenten interessant, da in kürzester Zeit eine Zuschneidung auf sehr spezielle Aufgaben erfolgen kann. Im Rahmen dieser Diplomarbeit wird eine FPGA-Firmware zur Detektorauslese entwickelt, die auf einem Xilinx Virtex-5 FPGA [13] implementiert wird. Dieser befindet sich auf der Hauptplatine des in Freiburg entwickelten GANDALF[7]-Moduls, welches für die Auslese und Echtzeitanalyse von Detektorsignalen entwickelt wurde. Einen Überblick der vielseitigen Einsatzmöglichkeiten des Moduls befindet sich in Kapitel 4.

Die FPGA-Firmware beinhaltet einen 96-Kanal-Scaler, welcher digitale Signale bis über 500 MHz fehlerfrei zählen kann. Wie in Kapitel 5 näher erläutert, basiert der Mechanismus auf dem Johnson-Code, der einen fehlerfreien Zähl- und Auslesevorgang ermöglicht. Außerdem enthält das FPGA-Design die Funktion, sehr präzise Zeitintervalle besser als 100 ps zu vermessen, womit eine Strahlanalyse auch auf sehr kleinen Zeitabschnitten erfolgen kann. Diese Aufgabe übernimmt ein Time-to-Digital Converter (TDC).

Beide Komponenten werden kombiniert und wie in Kapitel 6 erklärt auf einem einzigen Xilinx Virtex-5 FPGA implementiert. Für die Strahlanalyse wird ein Pseudo-Randomtrigger benötigt, dessen Logik entwickelt und ebenfalls in die FPGA-Firmware integriert wird. Sowohl Scaler als auch TDC müssen in das COMPASS Datennahme-System integriert werden. Unabhängig davon ist eine Auslese der zufällig getriggerten Daten für das Beam-Monitoring-System vorgesehen.

Ziel dieser Arbeit ist die Entwicklung eines Systems zur voll automatisierten Strahlanalyse in Echtzeit. Damit soll eine Überwachung und Kontrolle verschiedener Strahlattribute, beispielsweise der Strahlrate ermöglicht werden. Das System soll vor allem dazu in der Lage sein, Systematiken im Strahl aufzudecken, um so die Strahlqualität beziehungsweise die Güte des Teilchenstrahls beurteilen zu können.

In Kapitel 7 werden erste Messungen des Teilchenstrahls vorgestellt. Das FPGA-Design kam erstmals im Rahmen der DVCS-Messungen im Herbst 2012 zur Auslese eines Detektors mit szintillierenden Fasern (FI02) zum Einsatz. Hierbei handelt es sich um einen Detektor der sich vor dem Target befindet. Damit werden Daten für die Darstellung verschiedener Strahlattribute gewonnen. Mit einer Analyse werden während des Experiments in Echtzeit Graphen und Histogramme erzeugt. Dazu gehören beispielsweise das Strahlprofil und der Strahlschwerpunkt beziehungsweise die Strahlrate in Abhängigkeit der Zeit. Die dazugehörigen Daten werden mit der Scaler-Komponente der FPGA-Firmware gewonnen. Die TDC-Komponente ermöglicht außerdem eine Kreuzkorrelation der Zeitmarken der auf dem Eingangssignal gemessenen Hits, womit die zeitliche Zufälligkeit der Strahlteilchen beziehungsweise Systematik auf kleinen Zeitabschnitten beurteilt werden kann.

[6] FPGA = Field Programmable Gate Array
[7] GANDALF = Generic Advanced Numerical Device for Analytic and Logic Functions

2. Physikalische Grundlagen

In diesem Kapitel wird ein kurzer Abriss der theoretischen Grundlagen und Konzepte zur Untersuchung der Spinstruktur des Nukleons dargelegt. Dies ist Forschungsgebiet des COMPASS-II Experiments, an dem Streuexperimente mit hochenergetischen Myonen durchgeführt werden. Grundlegend für die Erforschung der Spinstruktur des Nukleons sind die generalisierten Parton-Verteilungsfunktionen (GPD[1]), die über tief-inelastische Lepton-Nukleon-Streuung vermessen werden.

Diese Beschreibungen machen es möglich, Gesamtdrehimpulsbeiträge der einzelnen Konstituenten, also Gluonen und Quarks, des Nukleons direkt zu bestimmen und somit Aufschluss über die Substruktur des Nukleons zu erhalten. Es wird die tief-virtuelle Compton-Streuung als vielversprechende Methode zur Untersuchung der Substruktur des Nukleons vorgestellt. Die wesentlichen Konzepte dieser Theorien werden in diesem Kapitel kurz erläutert.

[1] GPD = Generalized Parton Distributions

2.1 Der Spin des Nukleons

Der Spin ist eine vektorwertige Größe und lässt sich semiklassisch als intrinsischer Drehimpuls von Teilchen verstehen, welcher eigentlich eine quantenmechanische Observable ist und in Einheiten von $\hbar$ (Planck-Konstante) angegeben wird. Aus Ableitungen der Drehimpuls-Eigenzustände $|jm\rangle$[2] im Hilbertraum ergibt sich neben ganzzahligen Werten für j auch die Möglichkeit, halbzahlige Werte, also halbzahlige Drehimpulse, anzunehmen. Der Bahndrehimpuls-Operator führt jedoch nur zu ganzzahligen Werten für j.

Dies stellt nicht nur eine mathematische Möglichkeit dar, sondern sie ist auch von physikalischer Relevanz. In der Tat existieren intrinsische Drehimpulse von Teilchen, welche auch durch diese Lösungen beschrieben werden können [14].

Das Nukleon wird von Quarks beziehungsweise Antiquarks (Spin $\frac{1}{2}\,\hbar$) und Gluonen (Spin $1\,\hbar$) gebildet. Der Gesamtspin des Nukleons kann als Addition der totalen Drehimpulse J_q und J_g der Quarks und Gluonen geschrieben werden [15]:

$$\frac{J}{\hbar} = \frac{1}{2} = J_q + J_g. \tag{2.1}$$

Damit ergibt sich für den Gesamtspin:

$$\frac{S}{\hbar} = \frac{1}{2} = \frac{1}{2}\Delta\Sigma + \Delta G + L_q + L_g, \tag{2.2}$$

wobei $\Delta\Sigma$ der Beitrag der Quarks und Antiquarks, ΔG der Beitrag der Gluonen und L_q, L_g die Drehimpulsbeiträge der Nukleonkonstituenten sind.

Nun ist aus früheren Experimenten bekannt, dass der Beitrag der Quarks und Antiquarks $\Delta\Sigma \approx 0,3$ und der der Gluonen $|\Delta G| \approx 0,2 - 0,3$ ist [16, 17]. Dabei ist man sich über die Größe der Gesamtdrehimpulse J_q und J_g noch weitestgehend im Unklaren.

Ein möglicher Zugang zu den Größen J_q und J_g, bieten die GPDs (siehe Abschnitt 2.4). Sie werden im COMPASS-II Experiment unter anderem über DVCS vermessen. Durch die Summenregel von X. Ji [18] lässt sich ein Zusammenhang zwischen den Gesamtdrehimpulsen J_f[3] der Nukleonkonstituenten und den GPDs H^f beziehungsweise E^f herstellen (siehe Abschnitt 2.4.2):

$$J_f = \frac{1}{2}\lim_{t\to 0}\int_{-1}^{1} dx\; x\left[H^f(x,\xi,t) + E^f(x,\xi,t)\right]. \tag{2.3}$$

[2] Dabei bezeichnet j die dem Betrag eines Drehimpulses $\sqrt{\vec{L}^2}$ zugeordnete Quantenzahl und m die der dritten Komponente L_3.

[3] f bezeichnet den Quarkflavour $= u, \bar{u}, d, \bar{d}, s, \bar{s}$

2.2 Tief-inelastische Lepton-Nukleon-Streuung

Die innere Zusammensetzung des Nukleons lässt sich über tief-inelastische Lepton-Nukleon-Streuung (DIS[4]) untersuchen, wobei ein einfliegendes leptonisches Teilchen l an einem Konstituentenquark des Nukleons gestreut wird. Hierbei wird ein virtuelles Photon γ^* ausgetauscht (siehe Abbildung 2.1). Um Aufschluss über die inneren Wechselwirkungspotenziale zu bekommen, müssen die dabei auftretenden Viererimpulsüberträge

$$q^2 = (P_l - P_{l'})^2 = 2m_l^2c^2 - 2\left(\frac{E_l E_{l'}}{c^2} - \vec{p_l} \cdot \vec{p_{l'}}\right) = -Q^2, \tag{2.4}$$

die durch virtuelle Photonen vermittelt werden, sehr groß sein. Die Variablen werden in Tabelle 2.1 definiert.

Das Nukleon kann in einen angeregten Zustand umgewandelt oder durch Hadronisierung in mehrere stark wechselwirkende Teilchen aufgebrochen werden:

$$l + N \to l' + X. \tag{2.5}$$

Man unterscheidet hierbei zwischen inklusiven, semi-inklusiven und exklusiven Endzuständen. Im Vergleich zur inklusiven Streuung, werden bei der semi-inklusiven und exklusiven Streuung außer dem Lepton l noch mindestens ein Hadron beziehungsweise alle Teilchen im Endzustand nachgewiesen.

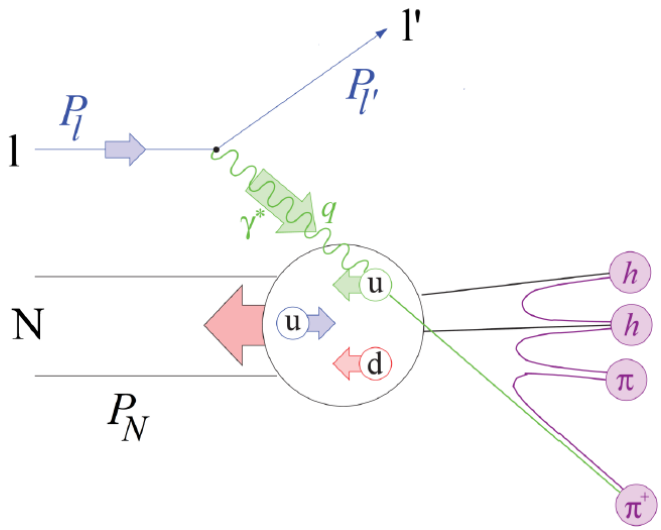

Abbildung 2.1: *Semi-inklusive tief-inelastischen Streuung [19].*

[4]DIS = Deep Inelastic Scattering

Tabelle 2.1: *Kinematische Variablen der tief-inelastische Streuung [20].*

Variable	**Beschreibung**
$P_l = (E_l, \vec{p_l})$	Viererimpuls des einlaufenden Leptons
$P_{l'} = (E_{l'}, \vec{p_{l'}})$	Viererimpuls des auslaufenden Leptons
$P_N = (E_N, \vec{p_N}) \stackrel{lab}{=} (M, 0, 0, 0)$	Viererimpuls des Protons
$q = P_l - P_{l'}$	Viererimpuls des virtuellen Photons
$\nu = \frac{P_N \cdot q}{M} \stackrel{lab}{=} E_l - E_{l'}$	Energieverlust des gestreuten Leptons
$y = \frac{P_N \cdot q}{P_N \cdot P_l} = \frac{\nu}{E_l}$	Relativer Energieverlust des gestreuten Leptons
$Q^2 = -q^2 \stackrel{lab}{\approx} 4E_l E_{l'} \cdot \sin^2 \frac{\vartheta}{2}$	negatives Quadrat des Viererimpulses des virtuellen Photons
$x_{Bj} = \frac{Q^2}{2P_N q} = \frac{Q^2}{2M\nu}$	Bjorken-Skalenvariable

Die Skalenvariable x_{Bj} kann im sogenannten „Infinite Momentum Frame“ (IMF) als Bruchteil des von einem Parton getragenen Nukleon-Viererimpulses interpretiert werden, bevor das virtuelle Photon absorbiert wird [21]. Das IMF bezeichnet ein sehr schnell bewegtes Bezugssystem in dem die transversalen Impulse und die Ruhemassen der Konstituenten vernachlässigt werden. Die Wechselwirkungen des einfliegenden Leptons mit einzelnen Konstituenten des Nukleons kann als elastisch angesehen werden, sofern es im Nukleon währenddessen zu keinen internen Wechselwirkungen kommt.

2.3 Die Parton-Verteilungsfunktionen

Der DIS-Prozess kann im sogenannten Bjorken-Limit mit

$$Q^2, \nu \to \infty, \tag{2.6}$$

in einen weichen hadronischen und einen harten leptonischen Anteil aufgeteilt werden [22, 23]:

$$\frac{d^2\sigma}{d\Omega dE_{l'}} = \frac{\alpha_{\text{em}}^2}{Q^4} \frac{E_{l'}}{E_l} L_{\mu\nu} W^{\mu\nu}, \tag{2.7}$$

wobei $L_{\mu\nu}$ den leptonische Tensor, der die Streuung des virtuellen Photons an einem Quark beschreibt und $W^{\mu\nu}$ den hadronischen Tensor bezeichnet, der die interne Struktur des Nukleons beschreibt. α_{em} bezeichnet die Feinstrukturkonstante, welche die Stärke der elektromagnetischen Kopplung angibt.

Im Gegensatz zum leptonische Tensor $L_{\mu\nu}$, der in pertubativer Quantenelektrodynamik (pQED) berechnet werden kann, ist für den hadronischen Tensor nur eine Parametrisierung durch Parton-Verteilungsfunktionen (PDFs[5]) möglich, die experimentell bestimmt werden müssen. Das Tensorprodukt lässt sich mit folgender Relation noch weiter separieren:

$$L_{\mu\nu} W^{\mu\nu} = \left[L_{\mu\nu}^{(S)}(x_{Bj})\ W^{\mu\nu(S)}(x_{Bj}) - L_{\mu\nu}^{(A)}(x_{Bj},\ \vec{s})\ W^{\mu\nu(A)}(x_{Bj},\ \vec{S}) \right]. \tag{2.8}$$

[5]PDF = Parton Distribution Functions

Bei einem mit (S) markierten Tensor handelt es sich um einen symmetrischen, spinunabhängigen Anteil. Dieser kann durch dimensionslose Strukturfunktionen $F_1(x_{Bj})$ und $F_2(x_{Bj})$, welche die Streuung an unpolariserten Nukleonen beschreiben, ausgedrückt werden. Bei einem mit (A) markierten Tensor handelt es sich um einen antisymmetrischen, spinabhängigen Anteil. Dieser kann durch die Strukturfunktionen $g_1(x_{Bj})$ und $g_2(x_{Bj})$ ausgedrückt werden, die die Streuung an polariserten Nukleonen beschreiben [24]. Die Größen $\vec{s}$, $\vec{S}$ bezeichnen den Lepton- beziehungsweise Nukleonspin.

2.3.1 Unpolarisierte Parton-Verteilungsfunktionen

Die Strukturfunktionen F_1 und F_2 können durch unpolarisierte Parton-Verteilungsfunktionen ausgedrückt werden, wobei über alle Quarkflavour f summiert wird [21]:

$$F_1(x_{Bj}) = \frac{1}{2}\sum_f e_f^2 q_f(x_{Bj}), \qquad F_2(x_{Bj}) = x_{Bj}\sum_f e_f^2 q_f(x_{Bj}). \tag{2.9}$$

Dabei bezeichnet e_f die elektrische Ladung der Quarks[6], sowie $q_f(x_{Bj})dx_{Bj}$ die Wahrscheinlichkeit, ein Quark mit Flavour f und Impulsbruchteil im Intervall $[x_{Bj}, x_{Bj} + dx_{Bj}]$ zu finden. Der Zusammenhang für F_1 ergibt sich aus F_2 und der sogenannten Callan-Gross-Beziehung, die für Dirac-Teilchen mit Spin $\hbar/2$ gilt [21],

$$2x_{Bj}F_1(x_{Bj}) = F_2(x_{Bj}). \tag{2.10}$$

2.3.2 Die longitudinal polarisierten Parton-Verteilungsfunktionen

PDFs erlauben Rückschlüsse auf die Helizitätsverteilung der Quarks. Bei tiefinelastischer Lepton-Nukleon-Streuung kann das virtuelle Photon wegen der Erhaltung der Helizität nur dann absorbiert werden, wenn die Helizität des streuenden Quarks entgegengesetzt ist (siehe Abbildung 2.2). Man bekommt somit einen Einblick in die Verteilung, indem longitudinal polarisierte Leptonen an einem zur Bewegungsrichtung dieser Leptonen parallel beziehungsweise antiparallel polarisierten Target gestreut werden.

Die Helizitätsverteilungsfunktionen erhält man mittels Subtraktion der Funktionen $q_f^{\rightleftarrows}(x_{Bj})$ von $q_f^{\rightrightarrows}(x_{Bj})$:

$$\Delta q_f(x_{Bj}) = q_f^{\rightrightarrows}(x_{Bj}) - q_f^{\rightleftarrows}(x_{Bj}), \tag{2.11}$$

Dabei geben $q_f^{\rightrightarrows}(x_{Bj})$ beziehungsweise $q_f^{\rightleftarrows}(x_{Bj})$ die Wahrscheinlichkeiten dafür an, ein Quark mit gleicher beziehungsweise entgegengesetzter Helizität, bezogen auf das Nukleon, zu finden.

[6] $e_f = \pm\frac{1}{3}e, \pm\frac{2}{3}e$

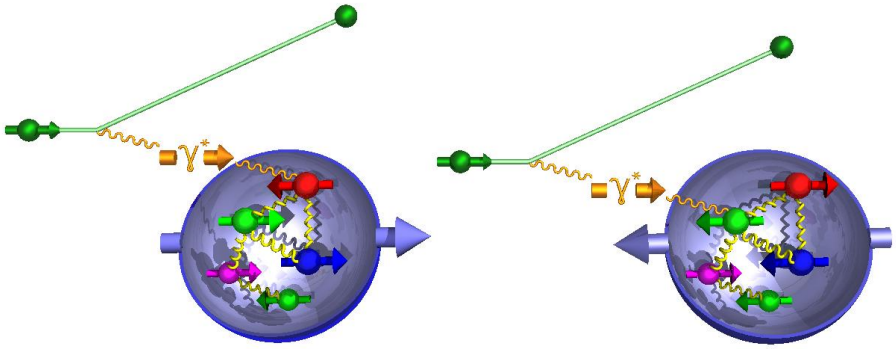

Abbildung 2.2: *Streuung longitudinal polarisierter Leptonen an polarisierten Protonen. Im linken Bild sind Lepton und Proton gleich polarisiert. Rechts sind diese entgegengesetzt polarisiert [25].*

Wenn nun über alle Quarkflavour summiert und über x_{Bj} integriert wird, erhält man den Helizitätsbeitrag der Quarks und der Antiquarks ($\Delta\Sigma$ in Gleichung 2.12). Eine Darstellung der helizitätsabhängigen Strukturfunktion g_1 (siehe Gleichung 2.13) ist analog zu den Gleichungen in 2.9 mittels Gleichung 2.11 möglich. Eine entsprechende Darstellung für g_2 durch PDFs im Partonmodell existiert nicht.

$$\Delta\Sigma = \sum_f \int_0^1 \Delta q_f(x_{Bj})\, dx_{Bj}, \tag{2.12}$$

$$g_1(x_{Bj}) = \frac{1}{2} \sum_f e_f^2\ \Delta q_f(x_{Bj}). \tag{2.13}$$

2.4 Generalisierte Parton-Verteilungen

Generalisierte Parton-Verteilungen (GPDs) wurden in theoretischen Arbeiten von Müller [26], Radyushkin [27] und Ji [18] eingeführt. Für DVCS (siehe Abschnitt 2.5) kann die Faktorisierung in einen harten und einen weichen Anteil, wie im Kapitel 2.2 beschrieben, angewendet werden. DVCS ist ein Prozess bei dem die wechselwirkenden Nukleonkonstituenten nicht zerstört werden, das heißt das Nukleon bleibt intakt. Will man ein vollständiges Bild der dreidimensionalen Struktur des Nukleon erhalten, so ist dies nur über solche Prozesse möglich, da dann Wechselwirkungen der Konstituenten des Nukleons untereinander vermessen werden können.

Durch die Einführung generalisierter Parton-Verteilungen ist eine dreidimensionale Vereinigung über sowohl longitudinale Impulse als auch transversale Positionen der Partonen möglich. Dies stellt bis heute die einzige Möglichkeit dar, Informationen über die Gesamtdrehimpulsbeiträge der Quarks und Gluonen zum Spin des Nukleons zu erhalten.

2.4.1 Kinematische Variablen

Im Folgenden werden weitere zur Beschreibung der GPDs wichtige Variablen erläutert [28]:

$$t = (p - p')^2 = (q - q')^2 := -\Delta^2, \tag{2.14}$$

$$\xi = x_{Bj} \frac{1 + \frac{\Delta^2}{2Q^2}}{2 - x_{Bj} + x_{Bj} \frac{\Delta^2}{Q^2}} \approx \frac{x_{Bj}}{2 - x_{Bj}}. \tag{2.15}$$

Hierbei ist t die Mandelstam-Variable, die den Viererimpuls-Übertrag auf das Nukleon angibt. q bezeichnet den Viererimpuls des virtuellen Photons γ^* sowie q' den des auslaufenden Photons γ (verlgeiche Abbildung 2.3). ξ ist die sogenannte Skewness[7]-Variable, die von x_{Bj} abhängig ist.

Das Quark besitzt nach der Emission eines reellen Photons einen um 2ξ veränderten Viererimpuls ($p = x + \xi$, $p' = x - \xi$). Die Größe ξ kann als Bruchteil des Impulsübertrags bezüglich der charakteristischen Richtung des unendlichen Impulses (IMF) betrachtet werden. $\xi = 0$ bedeutet, dass kein longitudinaler Impuls übertragen wird. Für $\xi \neq 0$ hat der Impulsübertrag auch eine nichtverschwindende Komponente parallel zum Impuls des virtuellen Photons.

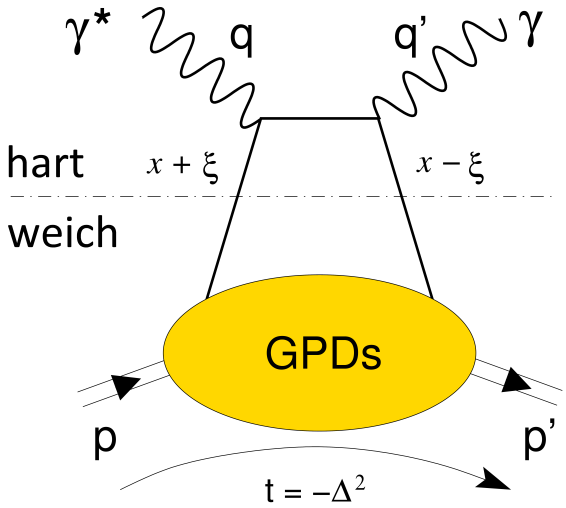

Abbildung 2.3: *Handbag-Diagramm der tief-virtuellen Compton-Streuung [29]. Die Beschreibung des weichen Anteils erfolgt durch GPDs.*

Wie bereits erwähnt ist für die Beschreibung des weichen hadronischen Anteils des Streuprozesses eine Parametrisierung durch Parton-Verteilungsfunktionen möglich. Im Folgenden werden vier GPDs (H, $\tilde{H}$, E, $\tilde{E}$) eingeführt, die von den kinematischen Variablen t, ξ und x abhängig sind und sich bezüglich der Quark-Helizitätserhaltung und der Nukleon-Helizitätserhaltung unterscheiden. Dabei sind H und $\tilde{H}$ Nukleon-Helizität-erhaltende und E beziehungsweise $\tilde{E}$ Nukleon-Helizität-umkehrende GPDs. Die Abhängigkeit von der Quark-Helizität wird in Tabelle 2.2 durch (˜) unterschieden.

[7] engl.: Schiefe

Tabelle 2.2: *Übersicht über die vier GPDs H, $\tilde{H}$, E und $\tilde{E}$.*

	Nukleon-Helizität **erhaltend**	Nukleon-Helizität **umkehrend**
Quark-Helizitäts **unabhängig**	$H^{q,g}$	$E^{q,g}$
Quark-Helizitäts **abhängig**	$\tilde{H}^{q,g}$	$\tilde{E}^{q,g}$

2.4.2 Zusammenhang der GPDs mit bekannten Verteilungen

Im Grenzfall lassen sich GPDs auf PDFs zurückführen. Diesen Spezialfall nennt man „Vorwärtslimit“ und es gelten folgende Relationen:

$$t = 0 \quad \text{und} \quad \xi = 0. \tag{2.16}$$

In diesem Grenzfall sind Vierer-Impuls und Helizität des Nukleons im Anfangs- und Endzustand gleich. Es ist möglich, einen Zusammenhang zwischen den GPDs und den schon bekannten PDFs (vergleiche Abschnitt 2.3) herzustellen [30]:

$$\text{für } x > 0: \quad H^f(x,0,0) = q_f(x), \quad \tilde{H}^f(x,0,0) = \Delta q_f(x), \tag{2.17}$$

$$\text{für } x < 0: \quad H^f(x,0,0) = -\bar{q}_f(-x), \quad \tilde{H}^f(x,0,0) = \Delta \bar{q}_f(-x). \tag{2.18}$$

Für $x > 0$ werden die GPDs H^f und $\tilde{H^f}$ als Quark- beziehungsweise für $x < 0$ als Antiquark-Verteilungsfunktionen interpretiert. In Abbildung 2.4 ist die GPD $H^u(x, \xi, t = 0)$ in Abhängigkeit von x und ξ dargestellt.

Desweiteren besteht neben den Beziehungen der GPDs zu den PDFs ein Zusammenhang zwischen den ersten Momenten der GPDs und den elastischen Formfaktoren des Nukleons [18]:

$$\begin{aligned} \sum_f z_f \int_{-1}^{+1} dx H^f(x,\ \xi,\ t) &= F_1(t), \qquad \sum_f z_f \int_{-1}^{+1} dx \tilde{H}^f(x,\ \xi,\ t) = G_A(t), \\ \sum_f z_f \int_{-1}^{+1} dx E^f(x,\ \xi,\ t) &= F_2(t), \qquad \sum_f z_f \int_{-1}^{+1} dx \tilde{E}^f(x,\ \xi,\ t) = H_A(t), \end{aligned} \tag{2.19}$$

wobei $F_1(t)$, $F_2(t)$, $G_A(t)$ und $H_A(t)$ die elastischen Dirac-, Pauli, Axial- und Pseudo-Skalaren Formfaktoren bezeichnen. Die zweiten Momente der GPDs führen auf den gesamten Drehimpuls der Quarks. Dieser Zusammenhang ist als Jis Summenregel bekannt (siehe Gleichung 2.3).

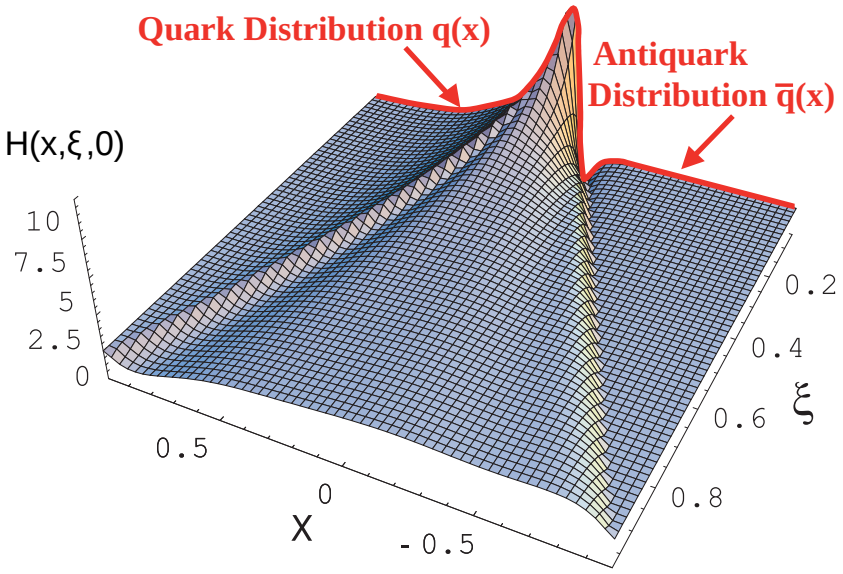

Abbildung 2.4: *Simulation der GPD $H(x, \xi, t = 0)$ für das u-Quark [31]. Die rote Linie kennzeichnet den Verlauf der unpolarisierten PDFs für Quarks ($x > 0$) und Antiquarks ($x < 0$). Der Bereich $|x| \geq \xi$ ist über DVCS zugänglich.*

2.4.3 Stoßparameterabhängige Parton-Verteilungen

Für $\xi = 0$ kann eine besonders einfache physikalische Interpretation der GPDs als Wahrscheinlichkeitsdichte angegeben werden. Damit kommt es nur zu transversalen Impulsüberträgen:

$$t = -\Delta^2 = -\Delta_L^2 - \Delta_\perp^2 = -\Delta_\perp^2. \tag{2.20}$$

Die Fouriertransformierte der GPD $H(x, 0, -\Delta_\perp^2)$ beschreibt die räumliche Verteilung der Partonen bei einem transversalen Abstand $|\vec{b}_\perp|$ vom Impulsschwerpunkt des Nukleons in Abhängigkeit des Impulsbruchteils x [32]:

$$q_f(x, \vec{b}_\perp) = \int \frac{d^2\Delta_\perp^2}{(2\pi)^2}\, e^{-i\Delta_\perp \cdot \vec{b}_\perp} H^f(x, 0, -\Delta_\perp^2). \tag{2.21}$$

Integriert man zudem über $b_\perp$, würde man die gewöhnlichen PDFs erhalten. Mit der stoßparameterabhängigen Funktion $q_f(x, \vec{b}_\perp)$ lässt sich wie in Abbildung 2.5 veranschaulicht, eine „Tomografie“ des Nukleons erstellen.

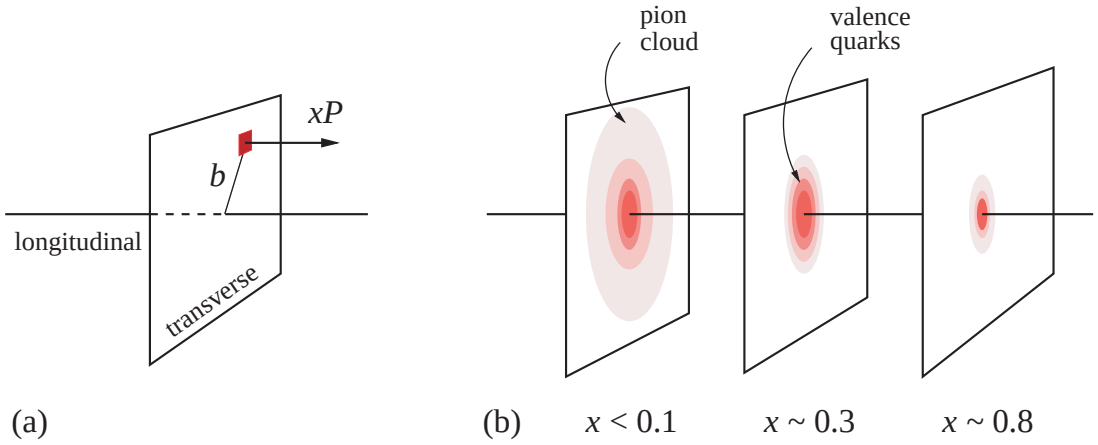

Abbildung 2.5: *Nukleon-Tomografie [33]: (a) Die Fouriertransformierte der GPD $H(x, 0, -\Delta_\perp^2)$ beschreibt die b-Verteilung der Partonen mit Longitudinalimpuls xP bei konstantem x. Hierbei ist $b = |\vec{b}_\perp|$ der Abstand der Partonen zum Impulsschwerpunkt in der Transversalebene. (b) Für verschiedene Impulsbruchteile x ist die Ortsverteilung der Partonen dargestellt. Die Verteilung wird bei kleinen x von Seequarks und Gluonen, sowie bei $x \approx 0,3$ von Valenzquarks dominiert. Für $x \to 1$ wird die Verteilung nur durch das aktiv wechselwirkende Parton bestimmt.*

2.5 Tief-virtuelle Compton-Streuung

Die Tief-virtuelle Compton-Streuung bietet eine vielversprechende Methode, Informationen über die innere Struktur des Nukleons zu erhalten. Hierbei wird ein Lepton l an einem Nukleon gestreut. Das Nukleon, welches dabei unversehrt bleibt, emittiert ein reelles Photon γ:

$$l + N \to l' + N' + \gamma. \tag{2.22}$$

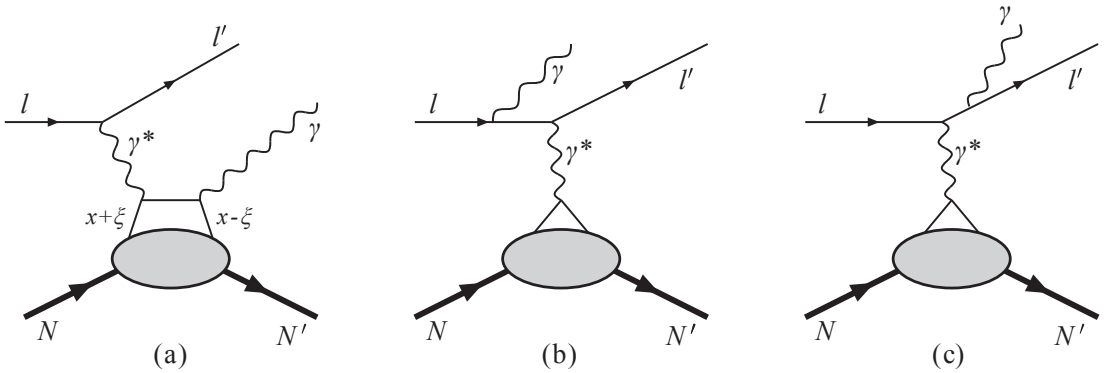

Abbildung 2.6: *Leptoproduktion reeller Photonen durch Prozesse führender Ordnung. (a) DVCS, (b) und (c) Bethe-Heitler-Prozess [19].*

In Abbildung 2.6 (a) wird der DVCS-Prozess dargestellt. Ein Quark mit longitudinalem Impulsbruchteil $x + \xi$ absorbiert ein virtuelles Photon γ^* mit Virtualität

$Q^2 = -q^2$ (vergleiche Tabelle 2.1) und emittiert ein reelles Photon γ. Da der in Abbildung 2.6 (b) beziehungsweise (c) dargestellte Bethe-Heitler-Prozess (BH-Prozess) den gleichen Anfangs- und Endzustand wie der DVCS-Prozess aufweist, interferieren die Amplituden beider Prozesse miteinander. Somit lässt sich der differentielle Wirkungsquerschnitt schreiben als [19]:

$$\frac{d\sigma(lN \to l'N'\gamma)}{dxdQ^2d|t|d\phi} \propto |\tau_{BH}|^2 + |\tau_{DVCS}|^2 + \underbrace{\tau_{DVCS}\tau^*_{BH} + \tau^*_{DVCS}\tau_{BH}}_{I}, \tag{2.23}$$

wobei ϕ den azimutalen Winkel zwischen der durch die Richtung des ein- beziehungsweise auslaufenden Leptons aufgespannten Streuebene und der durch das virtuelle und reelle Photon aufgespannten Produktionsebene darstellt. Die Streuamplitude des jeweiligen Prozesses ist mit τ bezeichnet. Wie in Abbildung 2.6 ersichtlich ist der Bethe-Heitler-Prozess rein elektromagnetisch. Dieser Prozess kann in der Quantenelektrodynamik berechnet werden, sofern man die elektromagnetischen Formfaktoren kennt. Diese können jedoch vermessen werden. Aus dem Bethe-Heitler-Prozess kann für die Bestimmung der GPDs keinerlei Information herangezogen werden. Im Gegensatz dazu kann das aus einem DVCS-Prozess entstandene reelle Photon γ zur Bestimmung der GPDs benutzt werden, da es durch die Wechselwirkung mit dem Nukleon entsteht. Der Interferenzterm I kann herangezogen werden, um Informationen über den Real- und den Imaginärteil der DVCS-Amplitude zu erhalten.

Die Zusammensetzung des Wirkungsquerschnitts ist stark vom kinematischen Bereich abhängig. Für $x_{Bj} > 0,03$ setzt sich dieser hauptsächlich aus der DVCS-Amplitude zusammen:

$$\frac{d\sigma(lN \to l'N'\gamma)}{dxdQ^2d|t|d\phi} \propto |\tau_{DVCS}|^2, \tag{2.24}$$

wohingegen der BH-Prozess für sehr kleine Werte von x_{Bj} dominant ist (siehe Abbildung 2.7).

Durch die Messung der Strahl-Ladungs-Asymmetrie, (BCA[8]) ist eine Möglichkeit gegeben, DVCS-Prozesse zu untersuchen. Der von den BH-Prozessen stammende Beitrag ist unabhängig von der Ladung des einfliegenden Leptons, jedoch nicht der von den DVCS-Prozessen stammende, welcher sein Vorzeichen wechselt. Dadurch lässt sich der Realteil der DVCS-Amplitude durch Subtraktion der jeweils gemessenen Wirkungsquerschnitte für unterschiedlich geladene Leptonen extrahieren.

$$\frac{d\sigma(l^+N \to l^{+'}N'\gamma)}{dxdQ^2d|t|d\phi} - \frac{d\sigma(l^-N \to l^{-'}N'\gamma)}{dxdQ^2d|t|d\phi} \propto \tau_{BH}\, Re(\tau_{DVCS}). \tag{2.25}$$

τ_{BH} ist nur von den elastischen Nukleonformfaktoren abhängig. Sie ist somit eine wohlbekannte Größe. Misst man nun mit Strahlung entgegengesetzter Helizität und gleicher Ladung, lässt sich der Imaginärteil der DVCS-Amplitude ermitteln:

$$\frac{d\sigma(l^\uparrow N \to l^{\uparrow'}N'\gamma)}{dxdQ^2d|t|d\phi} - \frac{d\sigma(l^\downarrow N \to l^{\downarrow'}N'\gamma)}{dxdQ^2d|t|d\phi} \propto Im(\tau_{DVCS}). \tag{2.26}$$

[8]BCA = Beam Charge Asymmetry

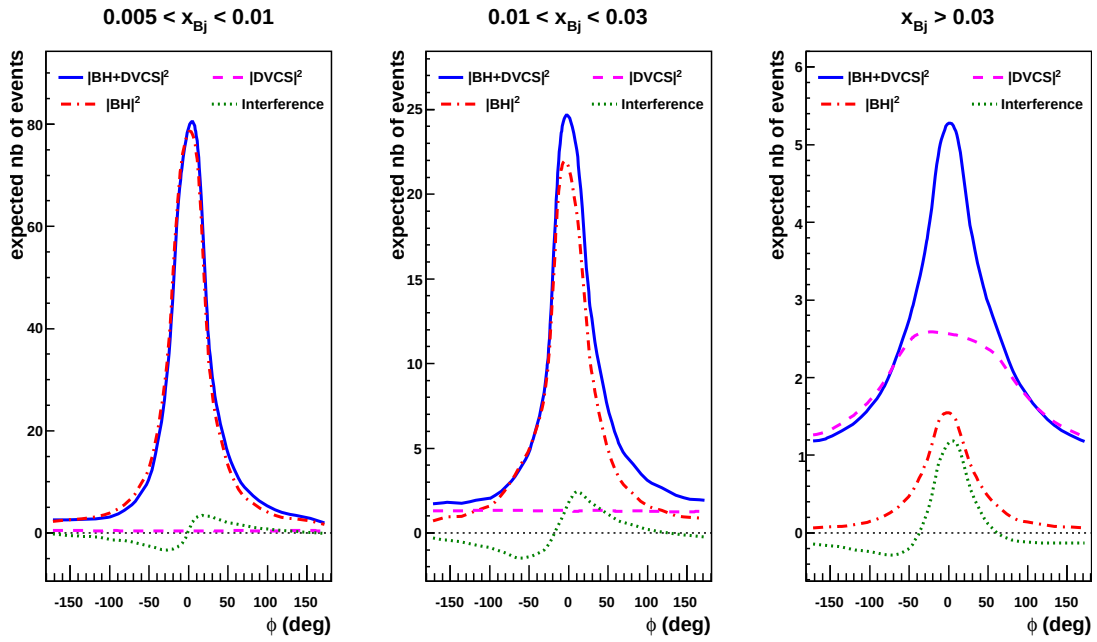

Abbildung 2.7: *Ergebnisse aus einer Monte-Carlo-Simulation. Die Wechselwirkungsbeiträge der BH- und DVCS-Prozesse für die Reaktion $\mu p \to \mu p \gamma$ sind in Abhängigkeit des Winkels ϕ für verschiedene x_{Bj} aufgetragen [29].*

Da der Myonstrahl im COMPASS-II Experiment durch den maximal Paritätsverletzenden Zerfall von geladenen Pionen $\pi^{\pm}$ und Kaonen $K^{\pm}$ zustande kommt, ist dieser natürlicherweise schon polarisiert. Durch die BCA-Messung gewinnt man mit Hilfe folgender Relationen Informationen über die GPD $\mathcal{H}$:

$$\mathcal{S} = \frac{d\sigma(l^{+\downarrow}N \to l^{+\downarrow'}N'\gamma)}{dxdQ^2d|t|d\phi} + \frac{d\sigma(l^{-\uparrow}N \to l^{-\uparrow'}N'\gamma)}{dxdQ^2d|t|d\phi} \propto Im(F_1\mathcal{H})\sin(\phi), \tag{2.27}$$

$$\mathcal{D} = \frac{d\sigma(l^{+\downarrow}N \to l^{+\downarrow'}N'\gamma)}{dxdQ^2d|t|d\phi} - \frac{d\sigma(l^{-\uparrow}N \to l^{-\uparrow'}N'\gamma)}{dxdQ^2d|t|d\phi} \propto Re(F_1\mathcal{H})\cos(\phi). \tag{2.28}$$

Hierbei ist $\mathcal{S}$ die Summe und $\mathcal{D}$ die Differenz der gemessenen Wirkungsquerschnitte.

Durch Integration beziehungsweise Analyse der Winkelabhängigkeit lassen sich Kombinationen verschiedener Quark-GPDs bestimmen. So erhält man beispielsweise mittels Analyse der Summe $\mathcal{S}$ den Imaginärteil des Compton-Formfaktors $\mathcal{H}$. Integriert man über den Winkel ϕ, erhält man Aufschluss über die transversale Ausdehnung des Nukleons. Der Differenzterm $\mathcal{D}$ enthält den Realteil des Compton-Formfaktors $\mathcal{H}$. Dieser ist in führender Ordnung durch eine Faltung der GPD H mit einer die harte Quark-Photon-Wechselwirkung $\gamma^* q$ beschreibende Funktion gegeben.

3. Das COMPASS-II Experiment

Das COMPASS-II Experiment befindet sich auf dem nördlichen CERN-Gelände in der Nähe von Genf. Es ist ein Fixed-Target-Experiment zur Erforschung der Nukleon-Spinstruktur und der Hadronspektroskopie, bei dem hochenergetische Myonen beziehungsweise Hadronen auf ruhende Nukleonen treffen. Es zeichnet sich dabei durch seine hohe Flexibilität bezüglich Einbindung seiner Spektrometerkomponenten aus. Der Aufbau des COMPASS-II Experiments besteht aus drei Teilen. Der erste Bereich befindet sich vor dem Target. Dort werden Impulse und Lage der einlaufenden Teilchen mittels Spurdetektoren vermessen. Der zweite Bereich, dort wo sich das Target befindet, kann als Wechselwirkungszone bezeichnet werden. Daran anschließend befindet sich das COMPASS-II-Spektrometer, in welchem auslaufende Teilchen vermessen werden. Das Spektrometer ist zweistufig aufgebaut und besteht aus einem Bereich zum Nachweis von Teilchen, die unter großen Winkeln bis 180 mrad gestreut werden und aus einem Bereich in dem auslaufende Teilchen bis 30 mrad nachgewiesen werden können.

Im Folgenden werden sowohl die wichtigsten Komponenten des Spektrometers, die Führung des Teilchenstrahls, als auch die Datennahme am COMPASS-II Experiment vorgestellt. Für eine detailliertere Beschreibung des Experiments sei auf [34] verwiesen.

3.1 Der Strahl

Das COMPASS-II Experiment ist für unterschiedliche Strahlkonfigurationen ausgelegt. So können wahlweise positive beziehungsweise negative Myonen, Pionen oder Elektronen als Strahlteilchen verwendet werden. Die Beschleunigung der benötigten Protonen erfolgt durch das SPS[1], worin Protonen bis zu 450 GeV/c beschleunigt werden. Die Protonen werden dann teilweise ausgekoppelt und in Richtung COMPASS-II Experiment (siehe Abbildung 3.1) in die M2-Beamline geleitet. Dort treffen sie auf ein Beryllium-Target („primary target T6“) mit einer wählbaren Dicke von bis zu 60 cm. Diese kann je nach gewünschter sekundären Strahlintensität verändert werden.

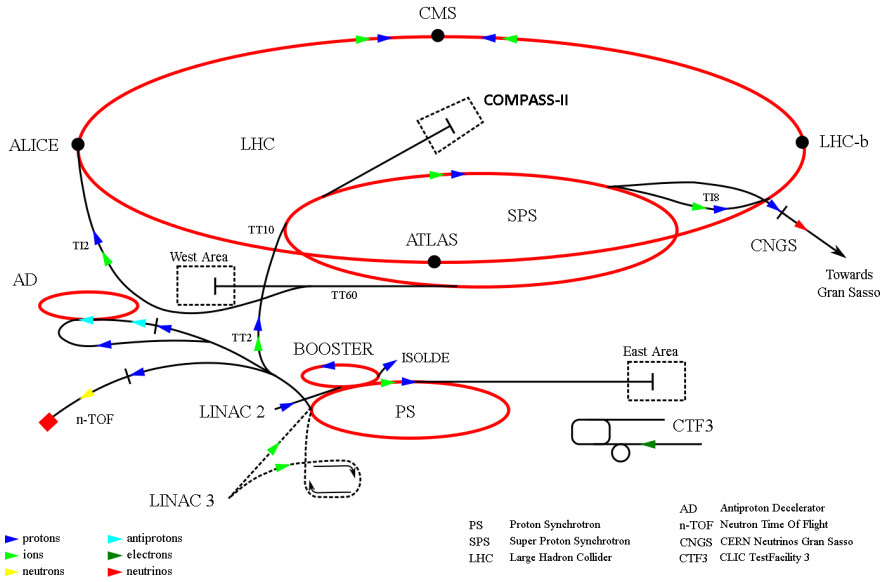

Abbildung 3.1: *Übersicht über den Beschleunigerkomplex am CERN [35].*

Für die DVCS-Messung (vergleiche Abschnitt 2.5) sind polarisierte Myonen $\mu^{\pm}$ nötig. Diese erhält man größtenteils durch den paritätsverletzenden schwachen Zerfall von geladenen Kaonen $K^{\pm}$ und Pionen $\pi^{\pm}$, die nach dem Aufprall der Protonen auf das Beryllium-Target entstehen:

$$K^+ \to \mu^+ + \nu_\mu, \quad \pi^+ \to \mu^+ + \nu_\mu, \tag{3.1}$$
$$K^- \to \mu^- + \bar{\nu}_\mu, \quad \pi^- \to \mu^- + \bar{\nu}_\mu. \tag{3.2}$$

Der Zerfallsprozess und die Impulsselektion findet in einem 600 m langen Tunnel statt, bevor der Myonstrahl auf das „eigentliche“ Target beziehungsweise die Wechselwirkungszone des COMPASS-II Experiments trifft. Davor wird der Teilchentrahl durch mehrere Quadrupol- und Dipolmagnete fokusiert und so abgelenkt, das die Myonen möglichst mittig auf das Target treffen. Aufgrund der Paritätsverletzung

[1]SPS = Super Proton Synchrotron

obiger Zerfälle sind die Myonen natürlicherweise polarisiert, wobei der Polarisationsgrad χ_μ vom Impulsverhältnis der Myon- (p_μ) und Hadronimpulse (p_{Had}) abhängig ist.

$$\chi_\mu = f\left(\frac{p_\mu}{p_{Had}}\right) \tag{3.3}$$

Die Auskopplung der Protonen aus dem SPS erfolgt periodisch mit abwechselnden „Onspill"- und „Offspill"-Phasen, wobei die aktive Onspill-Phase, in der Teilchen am Experiment ankommen etwa $9,6\,\mathrm{s}$, die inaktive Offspill-Phase circa $4 \times 9,6\,\mathrm{s} = 38,4\,\mathrm{s}$ dauert, sodass eine komplette Periode ungefähr $5 \times 9,6\,\mathrm{s} = 48\,\mathrm{s}$ beträgt. Die Dauer der Onspill-Phase ist unter anderem auch von der Anzahl der Experimente abhängig, an welche beschleunigte Protonen vom SPS verteilt werden. In der aktiven Phase kommen in etwa $4 \times 10^7\,\mu/s$ am COMPASS-II Experiment an, wobei nicht alle Myonen genügend fokusiert werden können. Diese werden als sogenannte Halo-Myonen über ein Vetosystem bei der Triggererzeugung ausgeschlossen (vergleiche Abschnitt 3.4).

In Abbildung 3.2 ist die Strahlführung dargestellt. Die Impulse der Myonen werden mittels mehrerer BMS[2] (BM01 - BM06) vermessen. Die Berechnung der Myonimpulse ergibt sich aus den Krümmungsradien der Spuren, die jeweils vor und hinter den Dipolmagneten vermessen werden.

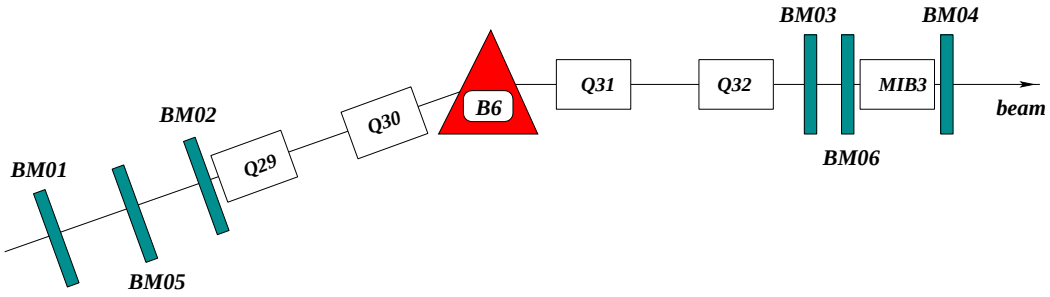

Abbildung 3.2: *Führung des Teilchenstrahls in die COMPASS-Halle [34]. Um Information über den Polarisationsgrad χ_μ zu erhalten ist eine Impulsmessung der Myonen notwendig. Dies geschieht mit mehreren BMS-Hodoskopen. Die Kenntnis des Impulses ist für das Experiment wesentlich. Der Dipolmagnet B6 lenkt die Strahlteilchen in die Horizontale ab, bevor diese am Experiment ankommen.*

3.2 Das Flüssigwasserstoff-Target

Für das DVCS-Programm wird ein sehr reines Proton-Target benötigt, um die exklusive Reaktion $\mu p \to \mu p \gamma$ zu vermessen. Deshalb wird ein $2,5\,\mathrm{m}$ langes Target verwendet, welches mit flüssigem Wasserstoff (LH_2) gefüllt ist. Das Target ist für den Nachweis rückgestoßener Protonen von einem Rückstoß-Detektor (RPD[3]) umgeben. Dieser wird „CAMERA[4]" genannt und in Abschnitt 3.3.2 beschrieben. Die Länge

[2]BMS = Beam Momentum Station
[3]RPD = Recoil Proton Detector beziehungsweise Rückstoß-Proton-Detektor
[4]CAMERA = COMPASS Apparatus for Measurements of Exclusive ReActions

des Targets wurde so gewählt, dass die Luminosität des Myonstrahls, die sowohl von der Targetlänge als auch von der Strahlintensität abhängt, ungefähr $10^{32}\,cm^{-2}s^{-1}$ beträgt.

3.3 Das COMPASS-II-Spektrometer

Das COMPASS-II-Spektrometer ist zweistufig aufgebaut. Es besteht aus dem LAS[5] zum Nachweis von Teilchen mit kleinen Impulsen und großen Streuwinkeln bis ±180 mrad und dem dahinter liegenden SAS[6], wo höherenergetische Teilchen mit kleineren Streuwinkeln bis ±30 mrad nachgewiesen werden. Die beiden Teile des Experiments bestehen aus „ähnlichen“ Detektorkomponenten, wobei diese jeweils Aussparungen besitzen, sodass unter kleinen Winkeln gestreute Teilchen möglichst ungehindert auf die nachfolgenden Detektoren auftreffen können. Eine detaillierte Darstellung des COMPASS-II-Spektrometers ist in Abbildung 3.3 gezeigt. Im Folgenden werden die größten unter den Detektoren am COMPASS-II Experiment kurz erläutert.

- **LAS:**

  1. Der CAMERA-Detektor umhüllt das LH_2-Target und registriert rückgestoßene Protonen (vergleiche Abschnitt 3.3.2).
  2. Der Dipolmagnet SM1 steht im Anschluß an das Target und lenkt die Teilchen in der horizontalen Ebene ab. Er besitzt ein Feldintegral von 1 Tm.
  3. Mit dem RICH[7]-1 Detektor werden geladene Teilchen im kinematischen Bereich zwischen 2,5 GeV/c und 50 GeV/c identifiziert [38].

     Dieser Detektor registriert sogenannte Cherenkov-Strahlung (γ), die immer dann emmitiert wird, wenn sich ionisierende Teilchen innerhalb eines Mediums schneller als deren Phasengeschwindigkeit bewegen:

     $$v > \frac{c}{n}, \tag{3.4}$$

     wobei c die Lichtgeschwindigkeit in Vakuum und n den Brechungsindex des Mediums bezeichnet. Dies lässt sich durch ein zeitabhängiges Dipolmoment aufgrund einer asymmetrischen Ladungsverteilung innerhalb des Mediums, hervorgerufen durch das relativistische Teilchen, erklären. Die Cherenkov-Strahlung wird senkrecht zur Polarisationsebene abgestrahlt (siehe Abbildung 3.4). Die Abhängigkeit des Abstrahlungswinkel θ von der Geschwindigkeit $v = \beta c$ zeigt folgende Relation:

     $$\cos\theta_{ch} = \frac{1}{n\beta} = \frac{1}{n} \cdot \sqrt{\frac{1}{1 + \frac{m^2}{p^2}}}. \tag{3.5}$$

[5] LAS = Large Angle Spectrometer
[6] SAS = Small Angle Spectrometer
[7] RICH-Detektor = Ring Imaging CHerenkov-Detektor

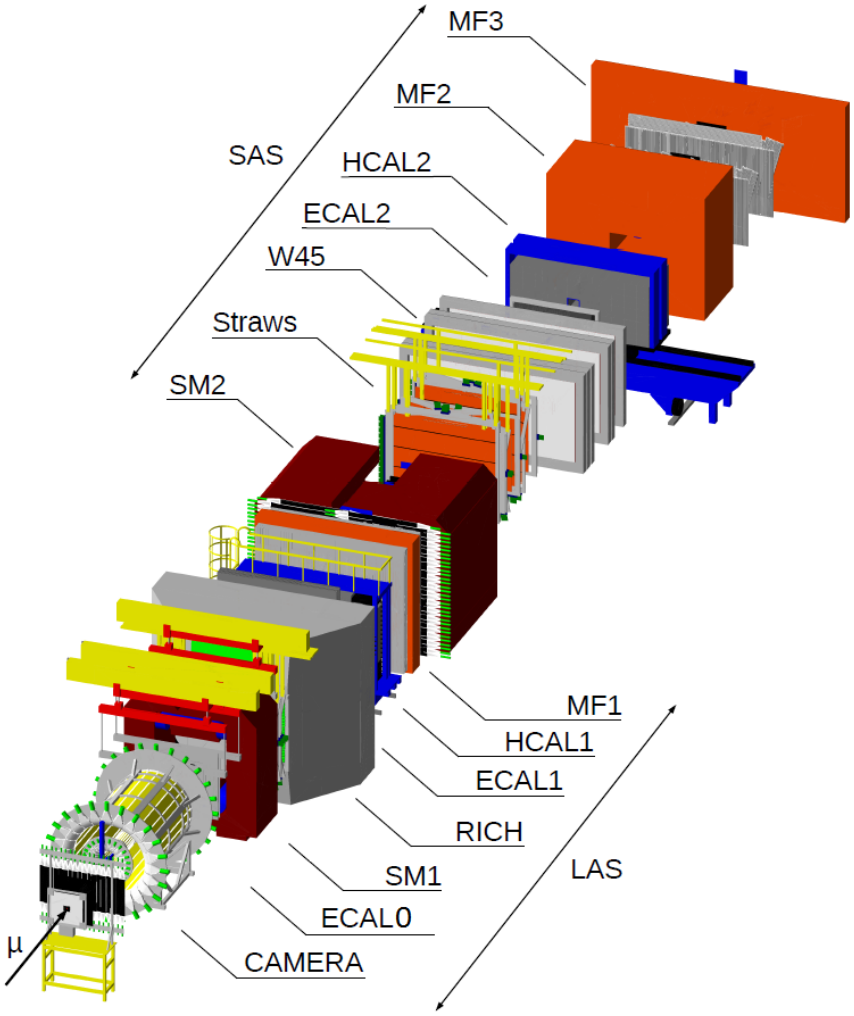

Abbildung 3.3: *Isometrische Ansicht des COMPASS-II Experiments mit der Simulationssoftware TGEANT [36, 37].*

4. Mit einem elektromagnetischen Kalorimeter wird die Energie der Elektronen $e^{\pm}$ und Photonen γ bestimmt (ECAL0 und ECAL1 in Abbildung 3.3). Elektromagnetische Kalorimeter fungieren als Schauerzähler. Durch Wechselwirkungen hochenergetischer Photonen γ und Elektronen $e^{\pm}$ werden kaskadenförmige Schauer gebildet, womit ein Nachweis bis zu einer Energie von etwa 100 MeV erfolgen kann.

5. Ein hadronisches Kalorimeter (HCAL1) bestimmt die Energie der Hadronen. Durch ein primäres hadronisches Teilchen kommt es durch eine Serie inelastischer hadronischer Wechselwirkungen ebenfalls zur Schauerbildung.

6. Der Myon-Filter MF1 schließt den ersten Teil des Experiments ab. Es wird ein 60 cm dicker Eisenabsorber verwendet. Kann man ein Teilchen sowohl vor als auch hinter einem Myon-Filter nachweisen, handelt es sich auf Grund des hohen Durchdringungsvermögens mit hoher Wahrscheinlichkeit um ein Myon. Diese wechselwirken im Vergleich zu Hadronen

nicht stark und im Vergleich zu Elektronen geht viel weniger Energie durch Bremsstrahlung verloren [39]:

$$\frac{\Delta E^{\mu}_{Br}}{\Delta E^{e}_{Br}} = \left(\frac{m_e}{m_\mu}\right)^2 \approx \frac{1}{40000}. \tag{3.6}$$

- **SAS:**

  1. Ein weiterer Dipolmagnet (SM2), steht direkt im Anschluss an MF1. Er besitzt ein Feldintegral von $4,4\,\mathrm{Tm}$.
  2. Ein elektromagnetisches Kalorimeter (ECAL2) bestimmt die Energie der Elektronen $e^{\pm}$ und Photonen γ im SAS.
  3. Ein weiteres hadronisches Kalorimeter (HCAL2) bestimmt die Energie der Hadronen im SAS.
  4. Der Myon-Filter MF2 schließt den zweiten Teil des Experiments ab. Er besitzt eine dicke Betonwand (2,40 m).
  5. Dahinter ist ein zusätzlicher Myon-Filter MF3 eingebunden. Zwischen MF2 und MF3 befinden sich weitere Spurdetektoren.

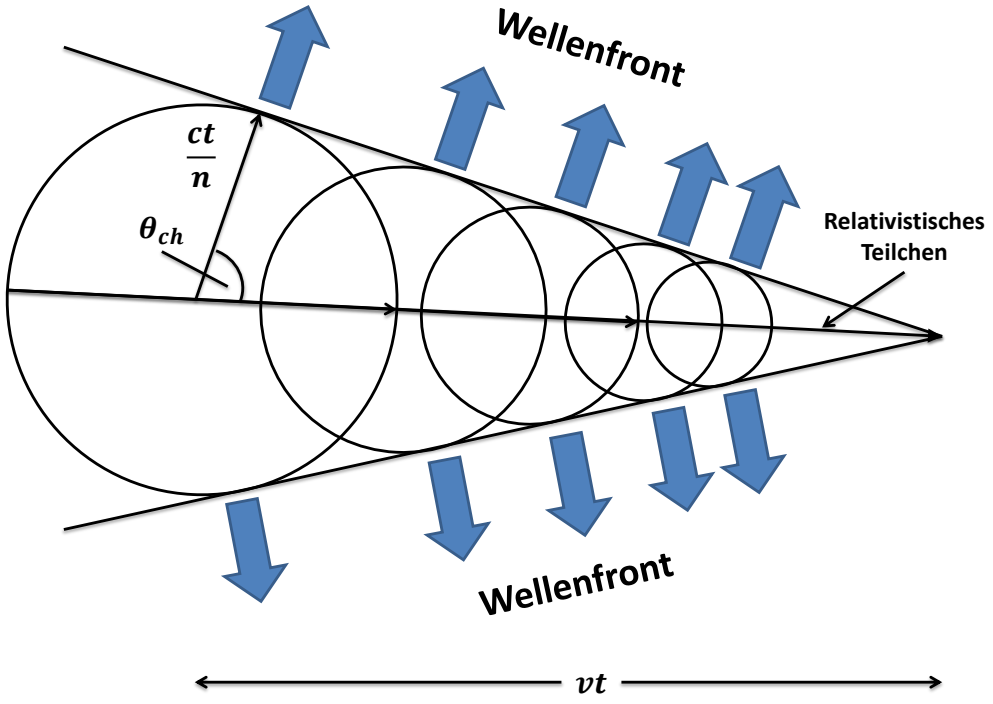

Abbildung 3.4: *Entstehung eines Cherenkov-Lichtkegels in einem Medium mit Brechungsindex n unter einem Winkel θ_{ch}, hervorgerufen durch ein ionisierendes Teilchen mit Geschwindigkeit $v = \beta c$.*

3.3.1 Spurdetektoren

Zu den oben erwähnten Detektoren kommen über das gesamte COMPASS-II-Spektrometer hinweg unterschiedliche Orts- beziehungsweise Spurdetektoren zum Einsatz. An sie werden hohe Anforderungen gestellt, da eine Vermessung der Teilchentrajektorien ohne eine präzise Ortsmessung unmöglich wäre. Im Folgenden werden die wichtigsten Spurdetektoren am COMPASS-II Experiment kurz erläutert.

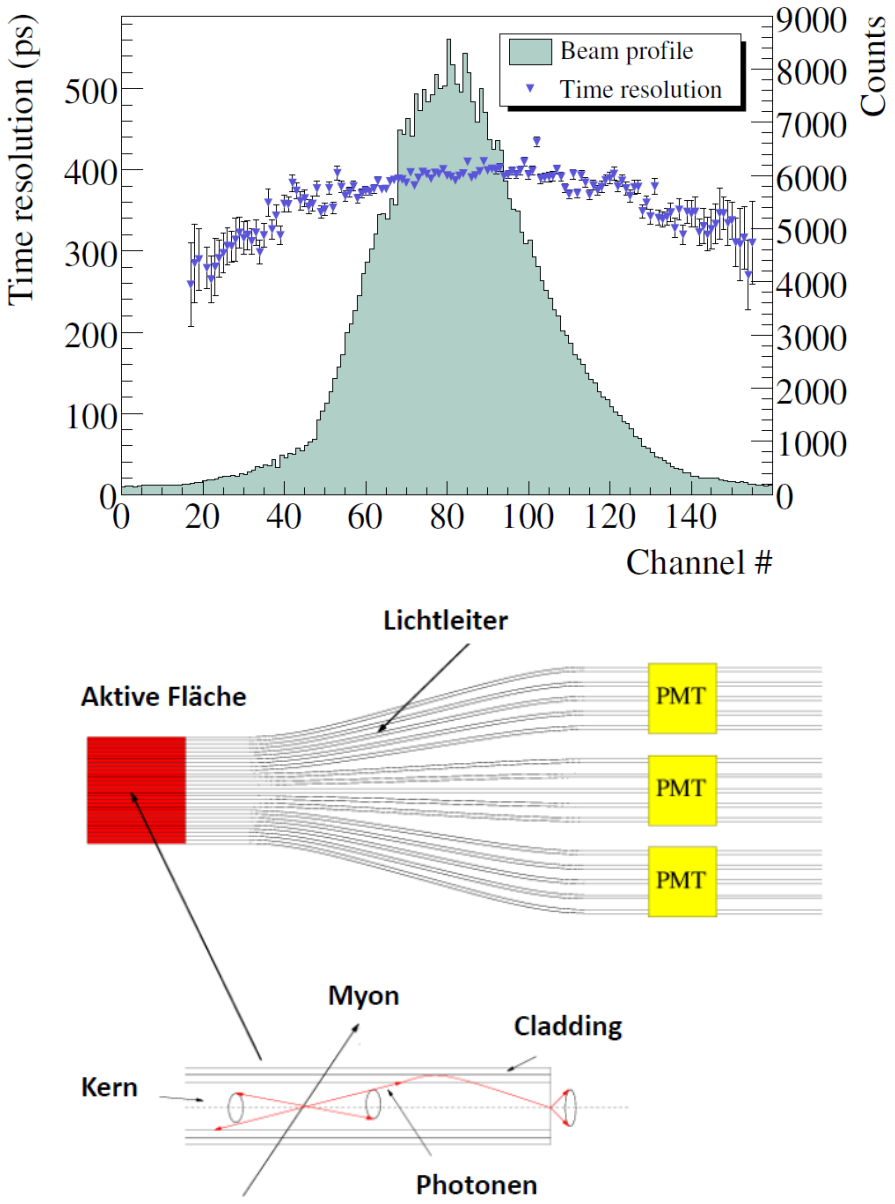

Abbildung 3.5: *Zeitauflösung einer SciFi-Ebene zusammen mit dem Strahlprofil [34] (oben). Unten ist der Aufbau eines SciFi-Hodoskops abgebildet [40]. Durch ein ionisierendes Teilchen wird Szintillatorlicht erzeugt welches durch Totalreflektion an der Grenzfläche zur äußeren Hülle (Cladding) zu den Photomultipliern (PMT) befördert wird, wo es nachgewiesen werden kann.*

Für die Messung des Strahls bis 3 cm um die Strahlachse kommen Detektoren zum Einsatz, die eine geringe Totzeit haben, da die Rate der Teilchen teilweise sehr hoch

(im Bereich MHz) sein kann. Dafür eignen sich sowohl SciFis[8] als auch Silizium-Streifendetektoren. Aufgrund ihrer sehr guten Zeit- und Ortsauflösung ($\approx 400\,\mathrm{ps}$, $\approx 150\,\mu\mathrm{m}$ [34]) werden SciFis vor allem zur Bestimmung des Impulses der Strahlteilchen herangezogen (siehe Abbildung 3.5 oben). Diese sind dünne Fasern aus Plastikszintillator, umgeben von einem Materialmantel mit kleinerem Brechungsindex, wodurch ein ionisierendes Teilchen über Totalreflexion nachgewiesen werden kann (siehe Abbildung 3.5 unten). Silizium-Streifendetektoren werden aufgrund ihrer guten Ortsauflösung ($\approx 10\,\mu\mathrm{m}$) für die Rekonstruktion der Trajektorien gestreuter Myonen beziehungsweise Strahlteilchen verwendet. Für Entfernungen zur Strahlachse im Bereich weniger cm bis etwa 50 cm werden hauptsächlich MicroMegas[9] und GEMs[10] verwendet. GEMs bestehen aus dünnen, durchlöcherten, metallbeschichteten Polymerfolien mit Lochabstand $\approx 140\,\mu\mathrm{m}$ und Lochdurchmesser $\approx 70\,\mu\mathrm{m}$. Durch die Verwendung mehrerer Lagen solcher Schichten wird eine hohe Gasverstärkung erzielt (siehe Abbildung 3.6). Diese Detektoren zeichnen sich unter anderem auch durch ihre geringe Totzeit aus.

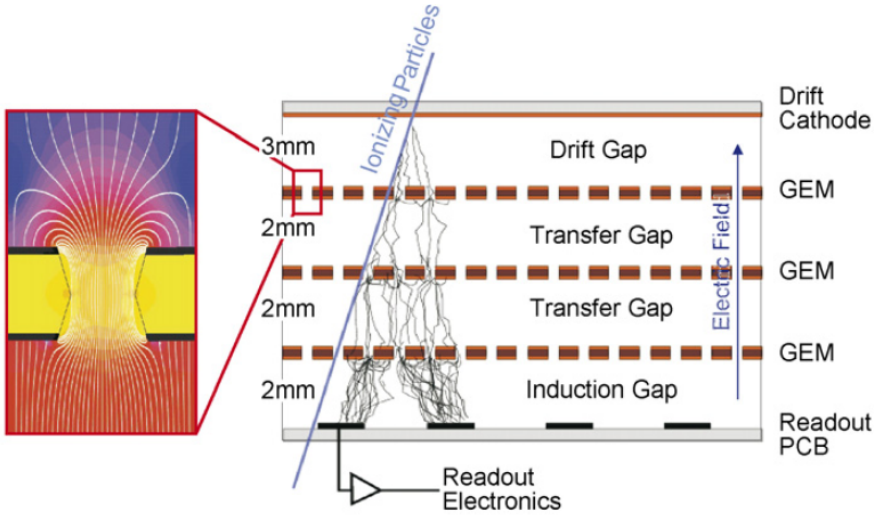

Abbildung 3.6: *Die Gasverstärkung der GEMs wird lawinenartig durch mehrere Schichten erzeugt. Diese findet innerhalb der Löcher statt. Links sind die elektrischen Feldlinien schematisch dargestellt [34].*

Für große Ablenkungswinkel, also weite Entfernungen zur Strahlachse kommen sehr großflächige Detektoren zum Einsatz. Darunter befinden sich Straws, DCs[11] und MWPCs[12], deren aktive Flächen im Bereich einiger m^2 liegen.

3.3.2 Der CAMERA-Detektor

Für das DVCS-Programm wurde der CAMERA-Detektor gebaut, der im Herbst 2012 in das Experiment eingebunden wurde. Dabei handelt es sich um einen Detektor für den Nachweis rückgestoßener Protonen, womit exklusive DVCS-Ereignisse

[8] SciFi = Szintillierende Fasern
[9] MicroMega = MICROMEsh GASeous structure
[10] GEM = Gas Electron Multiplier
[11] DC = Drift Chamber
[12] MWPC = Multi Wire Proportional Chamber

präzise detektiert werden können. Der Detektor besteht aus Szintillatorstreifen, welche an beiden Enden von PMTs ausgelesen werden. Die Szintillatorstreifen bilden zwei konzentrische Ringe – Ring A und Ring B – um das LH_2-Target (siehe Abbildung 3.8). Beim Durchflug eines Teilchens durch gegenüberliegende Streifen können Trigger erzeugt werden [41]. Jeder der Ringe enthält 24 Szintillatorstreifen, sodass jeweils 15° des Raumwinkels abdeckt werden. Die Ringe sind um die Hälfte (7,5°) gegeneinander rotiert, womit die Winkelauflösung verdoppelt wird. Die Dicke der Streifen wurde so gewählt, dass gute Messungen zur Energiedeposition und Flugzeitmessungen der Protonen duchgeführt werden können, woraus der Protonimpuls bestimmt werden kann. Dazu ist eine sehr gute Zeitauflösung essentiell. In Abbildung 3.7 beziehungsweise 3.8 befinden sich Darstellungen des CAMERA-Detektors.

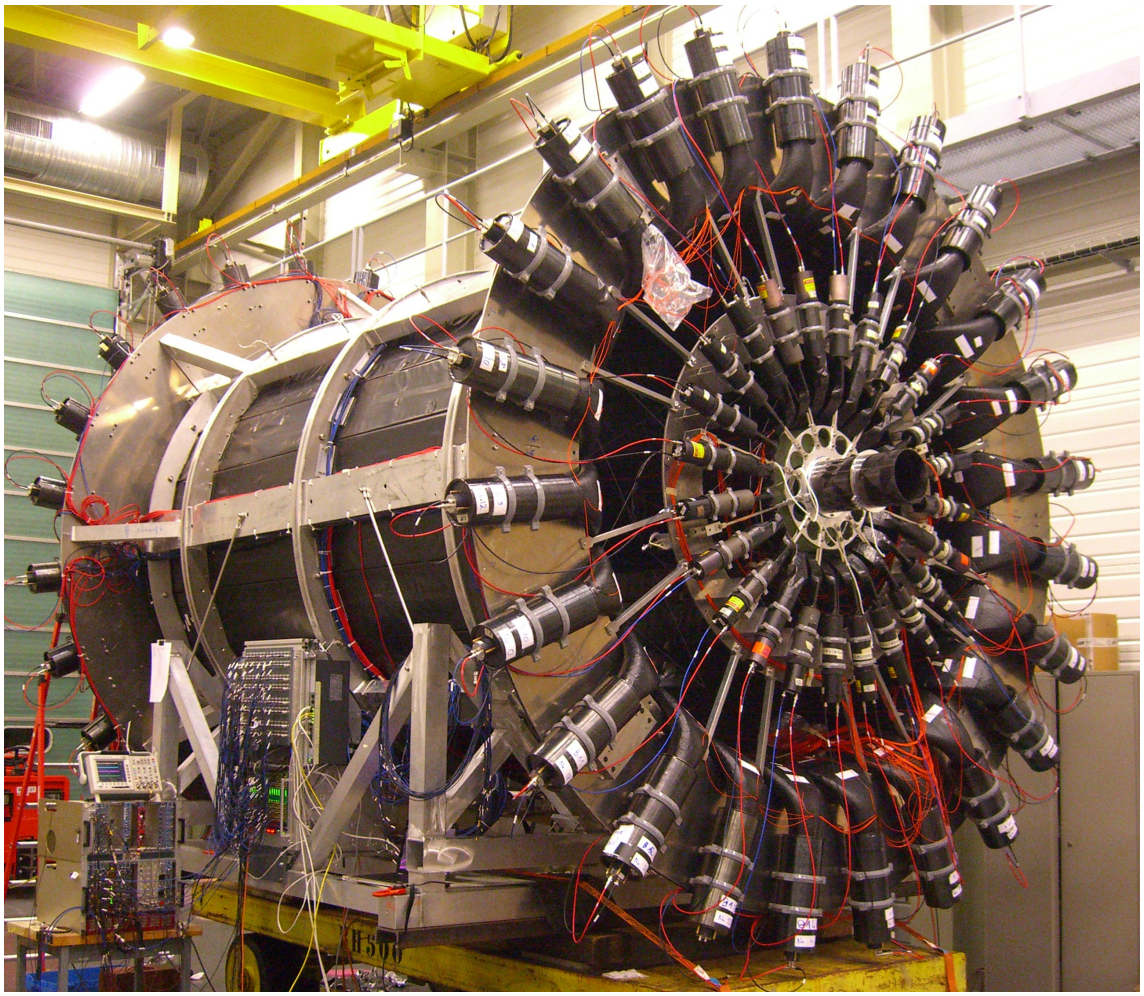

Abbildung 3.7: *Darstellung des* 4 *m langen CAMERA-Detektors nach dem Zusammenbau und gegen Ende der Testphase. Die Szintillatoren die den A-Ring bilden haben die Abmessungen* $275 \times 6,3 \times 0,4\,cm^3$ *und befinden sich* 24 *cm vom Mittelpunkt des Targets entfernt. Der B-Ring misst* $360 \times 30 \times 5\,cm^3$ *und ist etwa* 110 *cm vom Mittelpunkt des Targets entfernt.*

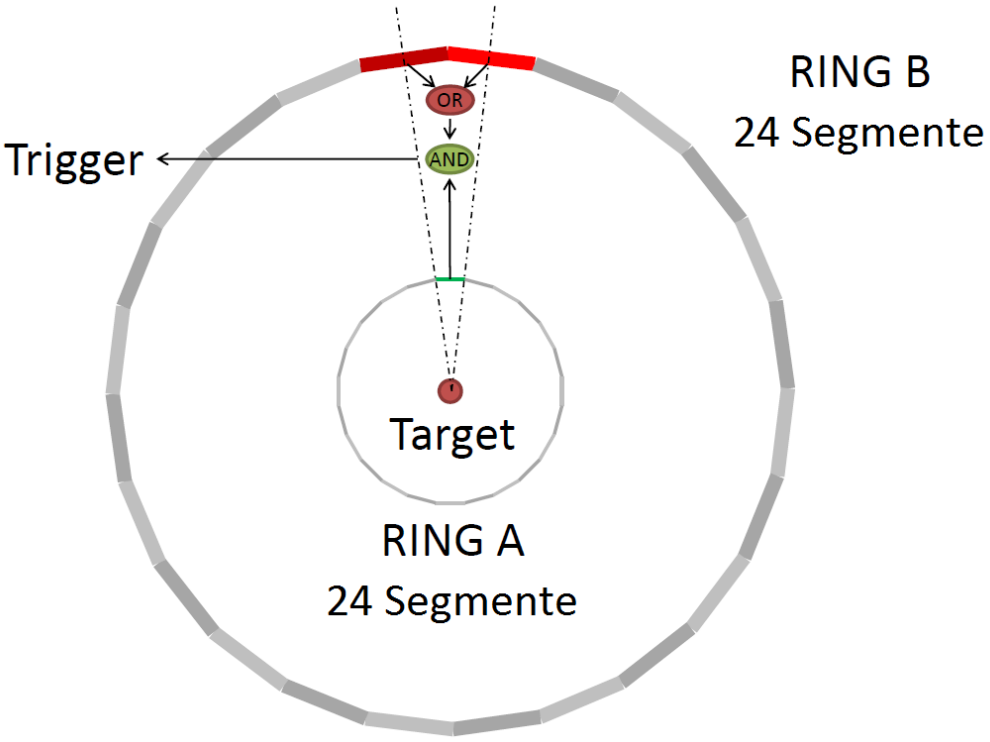

Abbildung 3.8: *Schematische Darstellung der zwei Szintillatorringe und der Triggerlogik durch Koinzidenzen gegenüberliegender Streifen [20].*

3.4 Das COMPASS-II-Trigger-System

Aufgrund der hohen Ereignisraten am COMPASS-II Experiment ist es unmöglich, alle Daten aufzunehmen. Dies ist auch nicht erstrebenswert, da nur die physikalisch interessanten Ereignisse aufgezeichnet werden sollen. Die Entscheidungen dafür übernimmt ein Triggersystem, welches Ereignisse herausselektiert und uninteressante verwirft. Für eine Triggerentscheidung benötigt das System in etwa $1,922\,\mu$s. Dies ist die Zeit zwischen Durchflug eines Myons durch den Detektor FI01[13] und der Generierung eines Triggers [42]. Das Trigger-Signal initialisiert den Auslesevorgang der Daten. Diese liegen auf den Front-End-Karten bereit, die direkt an den Detektoren angebracht sind. Eine schnelle Triggerentscheidung zu treffen ist gerade deshalb essentiell, da die Auslesemodule über begrenzten Speicherplatz zum zwischenspeichern der Daten verfügen.

3.4.1 Der Myontrigger

Myontrigger werden durch Vermessung der Spur eines gestreuten Myons erzeugt. Die Entscheidung wird sowohl mit Hilfe mehrerer Triggerhodoskope, als auch mit einer Messung des Energieverlusts durch Kalorimeter getroffen. Dabei werden gestreute Myonen von Halo-Myonen getrennt. Dies wird unter anderem durch ein vor dem Target installiertes Veto-Triggersystem erreicht. Wird ein Halo-Myon detektiert, so wird für eine gewisse Zeit das Auslösen eines Triggers verhindert.

[13] Hierbei handelt es sich um szintillierende Fasern vor dem Targetbereich.

Eine schematische Darstellung des Triggersystems befindet sich in Abbildung 3.9, dabei sind *H4I*, *H5I* die inneren, *H4M*, *H5M* die mittleren und *H3O*, *H4O* die äußeren Triggersysteme, die jeweils aus zwei Szintillator-Hodoskopen bestehen. Außerdem existiert noch ein sogenanntes Leitertriggersystem *H4L*, *H5L*.

Durch die vier Triggersysteme ist es möglich, Myonen in einem großen kinematischen Bereich zu detektieren. Myonen mit geringem relativen Energieverlust

$$0,2 < \frac{\Delta E}{E_0} < 0,5$$

werden vom inneren Triggersystem erfasst. Im Bereich

$$0,5 < \frac{\Delta E}{E_0} < 0,9$$

werden diese vom Leitertriggersystem registriert. Dafür muss jeweils der Ablenkungswinkel durch den Magneten gemessen werden. Mit einer geforderten unteren Schranke der Energiedeposition im hadronischen Kalorimeter werden Hintergrundereignisse ausgeschloßen.

Für Ereignisse mit $Q^2 > 0,5\,(\mathrm{GeV/c})^2$ ist der Streuwinkel des Myons groß genug um vom äußeren Triggersystem mit der sogenannten Vertical-Target-Pointing-Methode erfasst zu werden. Diese wird ausführlich in [43] beschrieben.

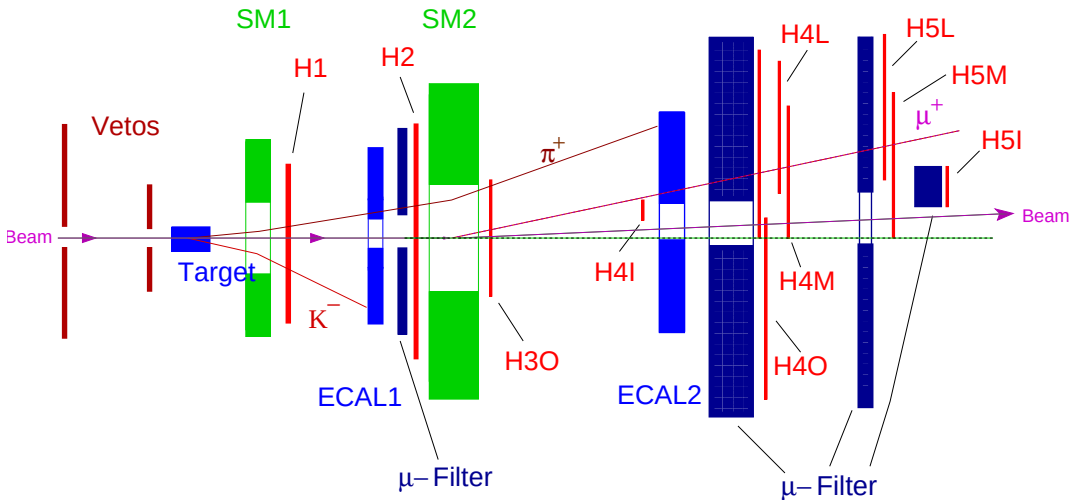

Abbildung 3.9: *Darstellung der Triggerhodoskope im COMPASS-II-Spektrometer [29].*

3.4.2 Der Protontrigger

Entscheidungen für Protontrigger werden durch den CAMERA-Detektor getroffen [41]. Die Triggergenerierung erfolgt durch eine Koinzidenzmessung, sobald ein Proton in zwei sich gegenüberliegenden Szintillatoren (Ring A und Ring B) detektiert wird (vergleiche Abbildung 3.8). Dabei ist es wichtig, Protonen zweifelsfrei zu identifizieren, um sie von anderen Teilchen wie Pionen oder δ-Elektronen auseinanderzuhalten. Dafür wird eine Messung der Flugzeit und des Energieverlusts sowohl für Ring A als auch für Ring B herangezogen, womit eine Separation in bestimmten kinematischen Bereichen möglich ist (vergleiche Abbildung 3.10).

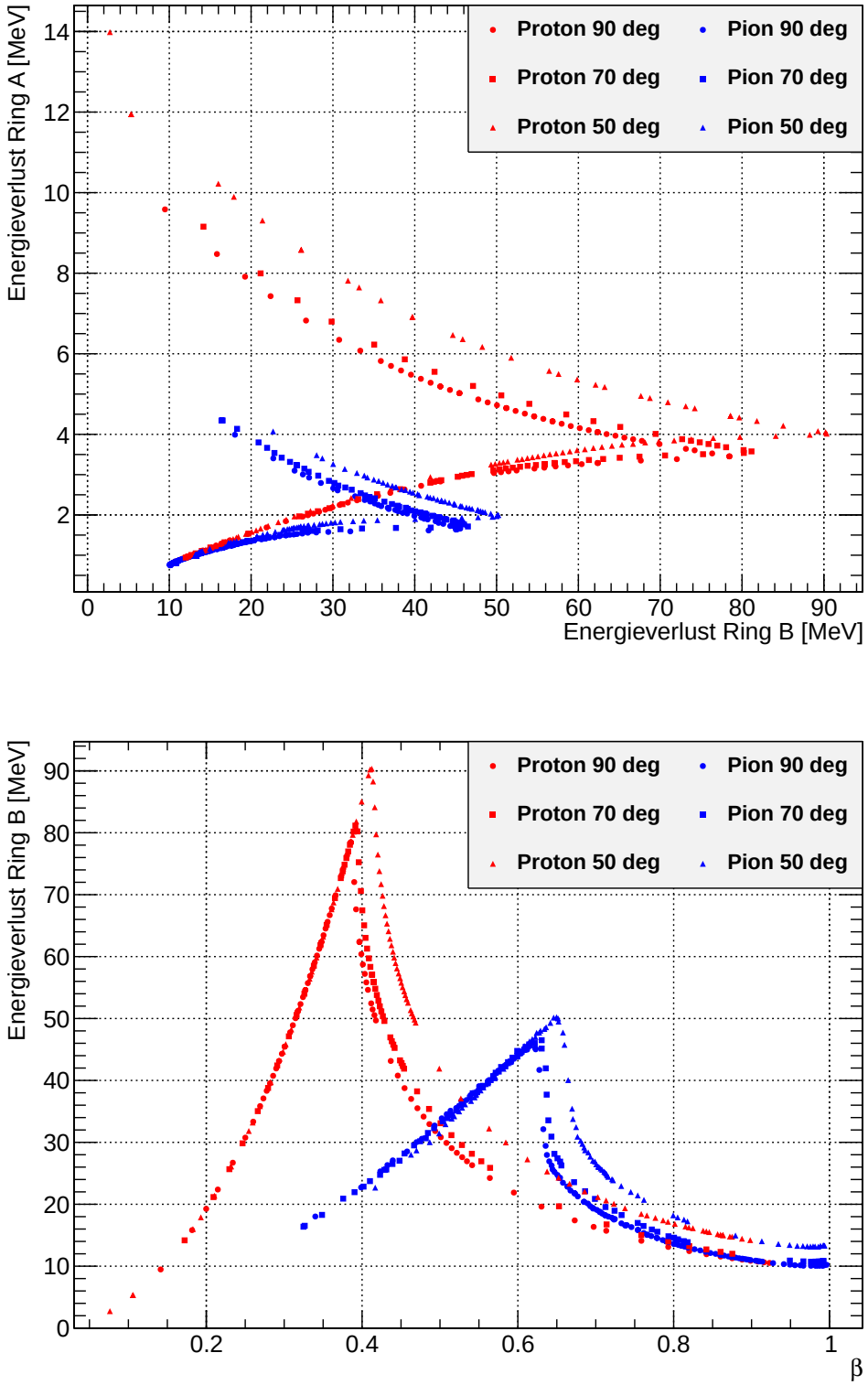

Abbildung 3.10: *Simulation mit TGEANT [36, 37]: Auftragung der Energiedeposition in Ring A für Protonen (rot) und Pionen (blau) gegen die Energiedeposition in Ring B (oben). Unten ist der Energieverlust in Ring B in Abhängigkeit von der Teilchengeschwindigkeit $\beta = v/c$ aufgetragen.*

3.5 Die Datennahme

Das Datennahmesystem (DAQ[14]) am COMPASS-II Experiment muss in der Lage sein, Datenraten bis zu mehreren Gigabyte pro Sekunde zu verarbeiten. Diese Datenrate kommt durch die große Anzahl an Detektorkanälen (etwa 250000), die teilweise sehr hohen Triggerraten (bis zu 100 kHz) und durch die große Hitrate (im Bereich MHz) in einzelnen Kanälen zustande. Eine Zusammenfassung der Datenweiterleitung von den Detektorsignalen bis zur Speicherung auf Magnetbändern über die zentrale Datenannahme des CERN ist in Abbildung 3.11 dargestellt.

Die Detektorsignale werden vor Ort mittels ADCs[15] oder TDCs[16] auf den Front-End-Karten digitalisiert und in Buffern zwischengespeichert. ADCs tasten die Amplitude des Signals ab und digitalisieren diese. Ein TDC ermittelt den Zeitpunkt einer Zustandsänderung auf einem digitalen Signal und gibt die Zeitmarke in digitaler Form aus. Viele der am COMPASS-II Experiment zum Einsatz kommenden Front-End-Karten und Module, wie das GANDALF- und das CATCH[17]-Modul wurden in Freiburg entwickelt. Dabei wurde auf eine möglichst flexible Nutzbarkeit geachtet. So lassen sich auf dem CATCH-Modul unterschiedliche Front-End-Karten wie beispielsweise Scaler-CMCs[18] oder TDC-CMCs anbringen. Das GANDALF-Modul verfügt über ähnliche und weitere Möglichkeiten. Eine ausführlichere Beschreibung des GANDALF-Moduls befindet sich in Kapitel 4. Es besteht die Möglichkeit, bis zu 18 GANDALF-Module zentral über einen VXS-Switch auszulesen (siehe Kapitel 4). Diese Aufgabe übernimmt das Freiburger TIGER[19]-Modul [44]. Desweiteren kommen für die Auslese der Silicon- und GEM-Detektoren die in München entwickelten GeSiCA[20]-Module zum Einsatz.

Die Steuerung der Datenauslese übernimmt das Trigger Control System (TCS) [45], welches Trigger über optische Fasern an alle Auslesemodule verteilt. Für eine nachträgliche Zuordnung der Datanpakete zu den Triggern werden noch weitere Informationen zu den Triggern versendet (siehe Abschnitt 4.5). Die Daten der Auslesemodule werden über Glasfasern mit bis zu 160 MByte/s zu den Readout Buffer-PCs (ROBs) weitergeleitet. Von dort aus erfolgt eine Weiterleitung an Eventbuilder-PCs, die vollständige Events generieren, indem die Daten aller Detektoren zu einem bestimmten Event zusammengefasst werden. Anschließend erfolgt die Weiterleitung der Daten an einen zentralen Datenspeicher (CDR[21]) des CERN, zur Speicherung auf Festplatten und später auf Magnetbändern.

[14] DAQ = Data Acquisition System
[15] ADC = Analog-to-Digital-Converter
[16] TDC = Time-to-Digital-Converter
[17] CATCH = COMPASS Accumulate Transfer and Control Hardware
[18] CMC = Common Mezzanine Card, Aufsteckkarte für das CATCH-Modul
[19] TIGER = Trigger Implementation for GANDALF Electronic Readout
[20] GeSiCA = GEM and Silicon Control and Acquisition
[21] CDR = Central Data Recording

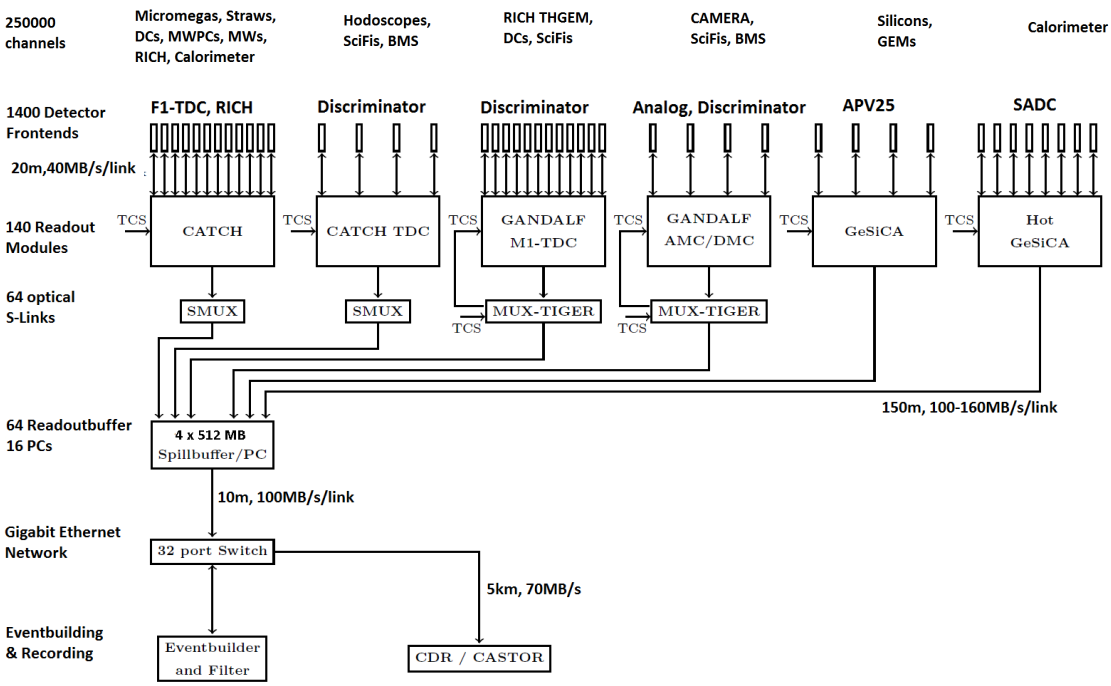

Abbildung 3.11: *Darstellung des Datennahmesystems am COMPASS-II Experiment [46].*

4. Das GANDALF Framework

Wie im vorangehenden Kapitel erläutert, werden für die Auslese und Verarbeitung der Detektorsignale zum Teil sehr hohe Anforderungen an die Elektronik gestellt. Zur Auslese der szintillierenden Fasern wird das in Freiburg entwickelte GANDALF-Modul zum Einsatz kommen. Dies ist ein VXS/VME64x-Modul, welches zur Echtzeitanalyse und Digitalisierung hochfrequenter Signale geeignet ist und die CATCH-Module zukünftig vermehrt ersetzen wird. Bei der Entwicklung des GANDALF-Moduls wurde großen Wert auf vielseitige Anwendungsmöglichkeiten gelegt. So können je nach Einsatzzweck verschiedene Aufsteckkarten (Mezzanine Cards) angebracht werden. Dieses Prinzip hat sich schon bei den CATCH-Modulen bewährt, die bei COMPASS seit Jahren erfolgreich eingesetzt werden. Das Herzstück des GANDALF-Moduls ist ein Xilinx Virtex-5 FPGA, der sich auf der Hauptplatine (Mainboard) befindet. Dieser verfügt über sehr leistungsstarke Logikeinheiten. Außerdem ist das Mainboard mit viel Speicher ausgestattet, der als Puffer für große Datenmengen verwendet werden kann. Das GANDALF-Modul kam ursprünglich als Transientenrekorder für die Auslese eines Rückstoß-Proton-Detektors zum Einsatz [20, 47, 48]. Außerdem konnten bereits weitere Funktionen wie beispielsweise ein 128-Kanal-TDC [49] beziehungsweise ein 64-Kanal-Meantimer [50] implementiert werden. Der im Rahmen dieser Diplomarbeit entwickelte FPGA-basierte Scaler und dessen Vereinigung mit dem TDC auf einem einzigen GANDALF-Modul wird in den Kapiteln 5 und 6 beschrieben.

4.1 Das GANDALF Mainboard

Das GANDALF Mainboard ist ein 6U VME64x/VXS Modul (233x160mm^2 [51]), welches über zwei Steckplätze für digitale oder analoge Aufsteckkarten verfügt. Diese dienen als Verbindung der Detektorsignale mit einem Xilinx Virtex-5 FPGA (Modell: XC5VSX95T [13]), im Folgenden DSP[1]-FPGA genannt. Dieser verfügt über 58880 Flipflops auf insgesamt 58800/4 = 14720 Virtex-5-Slices und mit 8 Mbit BlockRAM über genügend Speicher um auch anspruchsvolle Logiken und Funktionen implementieren zu können. Diese können mit einer maximalen Frequenz von bis zu 500 MHz getaktet werden (*CLK500MHz*). Unter anderem kann für die effiziente Implementierung bestimmter Operationen auf 640 sogenannte DSP-Slices zurückgegriffen werden. Diese werden bei der Entwicklung des Scalers (siehe Kapitel 5) unter anderem für Additionen verwendet.

Die Hauptplatine verfügt über einen weiteren FPGA (Modell: XC5VLX30T [13]), im Folgenden MEM-FPGA genannt. Dieser dient als Schnittstelle zum Hauptspeicher, welcher sich hauptsächlich aus dem 144 Mbit QDRII+[2] Speicher und dem 4 Gbit DDR2[3] Speicher zusammensetzt. Dieser findet beispielsweise in der Zwischenspeicherung von Daten Verwendung, bevor die Übermittlung an das zentrale Datennahmesystem erfolgt. Der QDRII+ Speicher kann aufgrund unabhängiger Ports für Lese- und Schreibzugriffe jeweils gleichzeitig mit bis zu 14, 4 Gbit/s beschrieben beziehungweise ausgelesen werden, wohingegen der DDR2 Speicher mit bis zu 4 Gbit/s beschrieben oder ausgelesen werden kann. Die Kommunikation des DSP- und des MEM-FPGA erfolgt über eine serielle Hochgeschwindigkeits-Schnittstelle, basierend auf dem Aurora Protokoll [52]. Über sogenannte „Rocket IO GTP Transceiver"-Elemente [53] ist über insgesamt acht differentielle Leitungen eine maximale Datenübertragungsrate von bis zu 25 Gbit/s möglich.

Das GANDALF-Modul verfügt außerdem über einen Xilinx Cool-Runner CPLD[4], welcher zur Konfiguration des DSP- beziehungsweise des MEM-FPGAs über die VME-CPU dient. Somit lassen sich die FPGAs auch während des Experiments zum Beispiel über einen Konfigurationsspeicher (Config-Mem), der mit dem CPLD verbunden ist, von außen konfigurieren (siehe Abschnitt 4.6). Der Vorteil eines CPLDs besteht unter anderem darin, dass er nach dem Einschalten des Moduls nicht erneut programmiert werden muss, während man FPGAs jedes mal mit einer sogenannten „Bitstream"-Datei initialisieren muss. Die Programmierung der FPGAs erfolgt in der Regel über den VME-Bus, jedoch ist dies auch mit dem „SystemACE"-Tool über eine CompactFlash-Karte auf der Hauptplatine möglich [54].

Für den Empfang der TCS-Signale kommt außerdem eine weitere Aufsteckkarte zum Einsatz. Diese wird in Abschnitt 4.4.1 beschrieben. Eine Darstellung des GANDALF-Moduls befindet sich in Abbildung 4.1.

[1] DSP = Digital Signal Processing
[2] QDR = Quad Data Rate
[3] DDR = Double Data Rate
[4] CPLD = Complex Programmable Logic Device

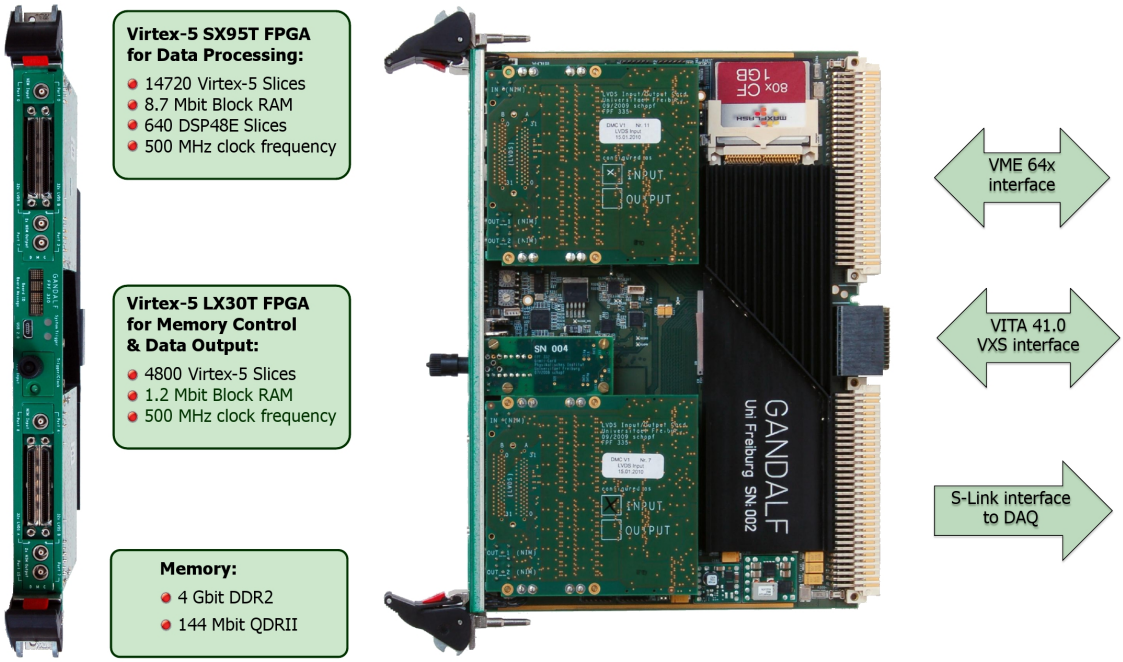

Abbildung 4.1: *Abbildung des GANDALF-Moduls mit zwei digitalen 64-Kanal-Aufsteckkarten (Input). In der Mitte ist die Takt/Trigger-Interface-Karte (Gimli-Karte) zu sehen. Außerdem sind die wichtigsten Schnittstellen und die Spezifikationen der auf dem GANDALF-Modul zum Einsatz kommenden FPGAs sowie des Hauptspeichers angegeben [49].*

4.2 Schnittstellen des GANDALF-Moduls

4.2.1 VME64x

Das GANDALF-Modul (VME-Slave) wird über zwei 160-polige Anschlüsse (P1, P2) mit der Backplane des VME[5]-Crates verbunden (siehe Abbildung 4.1 rechts). Dies ermöglicht eine Konfiguration aller GANDALF-Module im Crate über eine Linux-basierte VME-CPU (VME-Master), welche im gleichen Crate stecken muss. Dabei werden die VME64x-Spezifikationen erfüllt, welche in [51] definiert sind. Über den VME-BUS, der sich aus je einem 32 Bit breiten Adress- und Datenbus zusammensetzt, ist eine Übertragungsrate von bis zu 40 MByte/s möglich. Über die VME64x-Schnittstelle können interne Parameter mit einfachen Befehlen abgefragt respektive verändert werden. Somit kann beispielsweise eine Überwachung aller GANDALF-Module im VME-Crate durch Abfrage der Temperatur auf dem Mainboard erfolgen.

4.2.2 VXS

Es besteht die Möglichkeit, Daten von bis zu 18 GANDALF-Modulen im VME-Crate zentral über ein TIGER-Modul zu übertragen [44]. Die dafür benutzte VXS-Schnittstelle befindet sich zwischen P1 und P2 (siehe Abbildung 4.1). Die Kommunikation erfolgt über acht unidirektionale Hochgeschwindigkeitsleitungen (sie-

[5]VME = VERSA Module Eurocard

he Abbildung 4.2). VXS ist ein ANSI[6]-Standard und wird in [55] definiert. Dieser stellt eine Erweiterung des VME64x-Standards dar. Die Datenübermittlung erfolgt über optische Fasern. Dazu wird an der Rückseite der VME-Backplane eine SLink-Interfacekarte angebracht [56]. Diese erlaubt eine Datenübertragungsrate von bis zu 160 MByte/s. Die Datenübertragung an die DAQ der in diesem Projekt entwickelten GANDALF-Firmware findet am COMPASS-II Experiment über einen „Readout"-TIGER [44] statt.

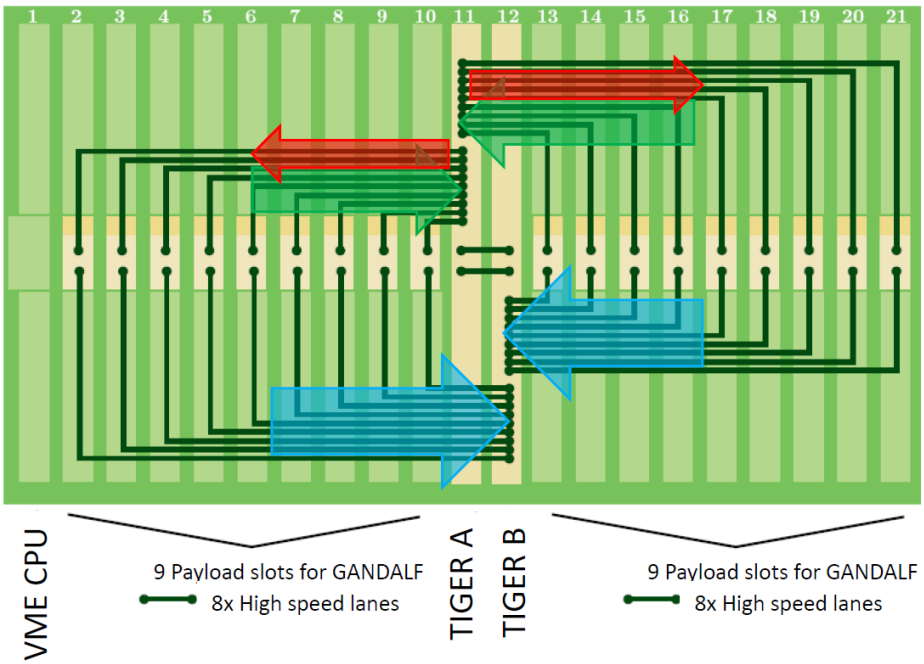

Abbildung 4.2: *Schematische Darstellung der VXS-Backplane: Der erste Slot ist für die VME-CPU bestimmt, welche keine VXS-Verbindung hat. GANDALF-Module können auf den Positionen 2 bis 10 und 13 bis 21 platziert werden. Für TIGER-Module sind die Positionen 11 und 12 vorgesehen. Über zwei Leitungen werden TCS-Signale (Takt- und Datensignal) an die GANDALF-Module übertragen (rot, TIGER A). Die Auslese der GANDALF-TCS-Daten findet über vier Leitungen statt (grün, TIGER A). Für die Übertragung von ungetriggerten Hit-Informationen sind acht Leitungen vorgesehen. Dieser Datenstrom wird von TIGER B empfangen (blau) [57].*

4.2.3 Takt- & Trigger-Schnittstelle

Für die Übertragung der Signale vom TCS auf den DSP-FPGA besitzt das GANDALF-Modul eine weitere Schnittstelle für die sogenannte „Gimli-Karte" (siehe Abschnitt 4.4.1), womit sowohl ein sehr präzises 155, 52 MHz-Taktsignal als auch Trigger und deren zugehörige Informationen übertragen werden (siehe Abschnitt 4.5). Ein einheitliches Taktsignal ist vor allem für das synchrone Schalten bei der Verwendung mehrerer GANDALF-Module essentiell.

[6]ANSI = American National Standards Institute

4.3 Taktsignale

Neben dem 155, 52 MHz-Taktsignal besteht auf dem GANDALF-Modul die Möglichkeit nach Bedarf weitere jitterarme Taktsignale zu erzeugen. Diese können aus dem TCS-Signal mit einem sogenannten Clock Multiplier Chip (Si5326) von Silicon Labs [58] erzeugt und über zwei Signalausgänge auf den beiden FPGAs bereitgestellt werden. Dabei können Frequenzen ν_{Si} zwischen 2 kHz bis maximal 945 MHz

$$\nu_{Si} = \nu_{TCS} \cdot \frac{n_1}{n_2}, \tag{4.1}$$

für natürliche Zahlen n_1, n_2 generiert werden. Auf diesem Weg wird das in Abschnitt 4.1 erwähnte Signal *CLK500MHz* erzeugt.

Desweiteren befindet sich ein programmierbarer Takt-Synthesizer CDCE949 [59] auf der Hauptplatine, der über einen Oszillator einen 27 MHz-Referenztakt erhält und ein 40 MHz-Taktsignal (*CLK40MHz*) generiert, welches an DSP- und MEM-FPGA sowie an den CPLD weitergeleitet wird. Eine schematische Übersicht der wichtigsten Taktsignale auf dem GANDALF-Modul zeigt Abbildung 4.3. Zusätzlich ist auf dem GANDALF-Modul ein Schwingquarz vorhanden, der mit dem CPLD verbunden ist und ein 40 MHz Taktsignal unabhängig vom TCS bereitstellt. Dieses Signal wurde in Abbildung 4.3 der Übersicht halber weggelassen.

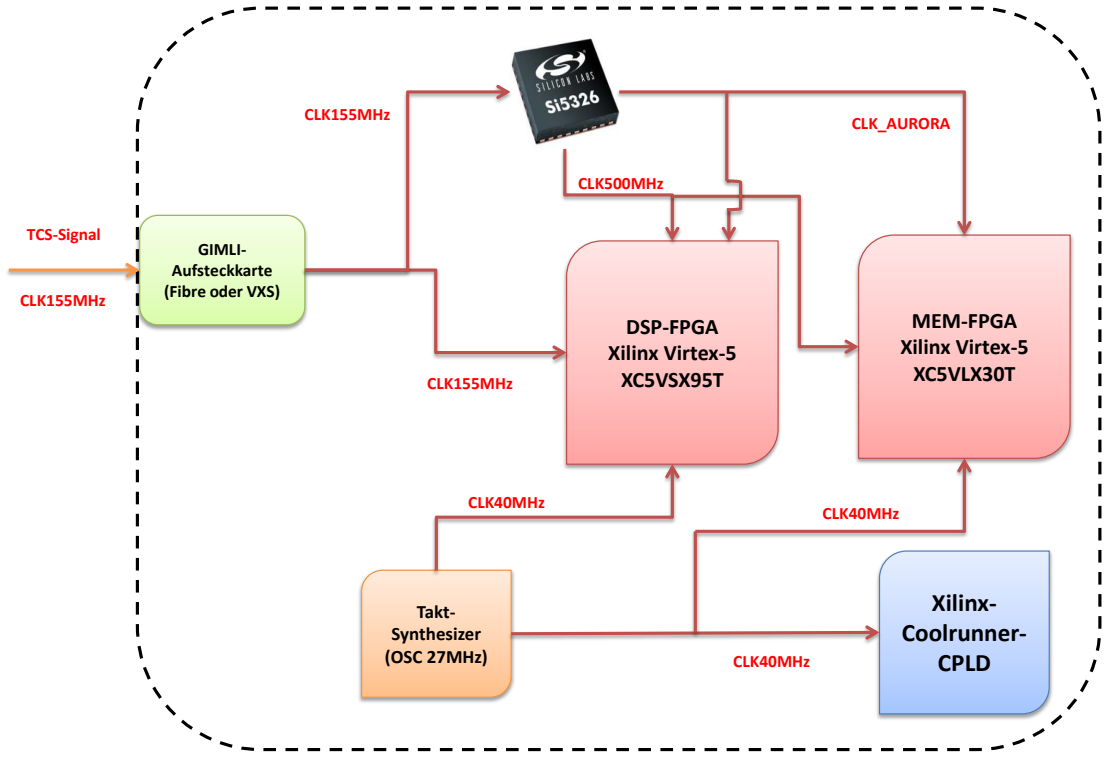

Abbildung 4.3: *Schematische Darstellung der Taktverteilung und Erzeugung der wichtigsten Taktsignale auf dem GANDALF-Modul.*

4.4 Aufsteckkarten

4.4.1 Die Gimli-Karte

Wie in Abschnitt 4.2.3 angedeutet, wird auf den Auslesemodulen ein vom TCS erzeugtes optisches Signal entgegengenommen, welches über Glasfasern verteilt wird. Dieses Signal enthält das 155, 52 MHz-Referenztaktsignal, das Triggersignal, sowie wichtige Informationen zu den Triggern wie beispielsweise Eventnummer, Spillnummer und Art des Events. Die Umwandlung des optischen in ein elektrisches Signal erfolgt auf einer Fibre-Gimli-Karte (siehe Abbildung 4.4). Diese enthält einen CLC016-Chip [60], welcher den Referenztakt aus dem seriellen Datenstrom wiederherstellt. Es besteht außerdem die Möglichkeit, das TCS-Signal über die Backplane entgegenzunehmen. Die Übertragung an die GANDALF-Module über den VXS-Bus übernimmt ein TIGER-Modul [44] (vergleiche Abbildung 4.2, roter Pfeil). Dazu ist eine VXS-Gimli-Karte nötig, die das TCS-Signal über den FPGA entgegennimmt, um es von dort aus wieder zurückzuleiten. Damit kann das Signal über die gleichen Pins am FPGA empfangen werden. Die Entgegennahme der TCS-Signale in diesem Projekt findet über ein TIGER-Modul (Gimli-Typ VXS) statt.

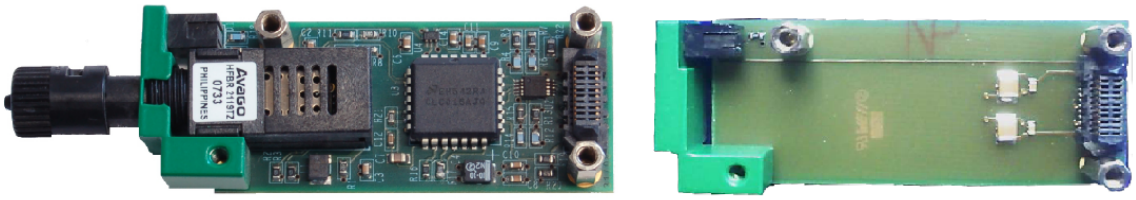

Abbildung 4.4: *Bild einer Fibre-Gimli (links) und einer VXS-Gimli (rechts) [57].*

4.4.2 Die digitale Mezzanine-Card

Neben der Möglichkeit, zwei AMCs[7] anzubringen, welche verwendet werden können um insgesamt 16 analoge Eingangssignale pro GANDALF-Modul zu digitalisieren, können auch digitale Mezzanine-Cards (DMC) aufgesteckt werden. Für die Entwicklung der Firmware zur Auslese der szintillierenden Fasern werden DMCs benötigt, die das Detektorsignal für die Signalverarbeitung zum DSP-FPGA weiterleiten. Das GANDALF-Modul erlaubt das Anbringen zweier DMCs mit jeweils 64 differentiellen Eingangssignalen (LVDS oder LVPECL[8]), die über differentielle Buffer mit dem DSP-FPGA verbunden werden. Somit können mit einem GANDALF-Modul maximal 128 Detektorkanäle verarbeitet werden.

Die Anbindung an das Eingangssignal erfolgt mit jeweils zwei differentiellen 32-Kanal VHDCI-Steckern [61] pro DMC. Die Signale werden von NB4N855S Clock- beziehungsweise Datenbuffern gepuffert, die laut Datenblatt [62] mit maximal 1 ps einen sehr kleinen RMS-Jitter aufweisen. Außerdem garantiert der Hersteller Eingangsfrequenzen bis 1 GHz verarbeiten zu können. Jede DMC besitzt zwei NIM-Ausgänge und einen NIM-Eingang, welche sich neben den VHDCI-Steckverbindungen befinden (siehe Abbildung 4.5). Diese können über LEMO-Verbindungen externe Signale zum DSP-FPGA beziehungsweise interne Signale nach außen weiterleiten.

[7] AMC = Analog Mezzanine-Card
[8] LVPECL = Low Voltage Positive Emitter Coupled Logic

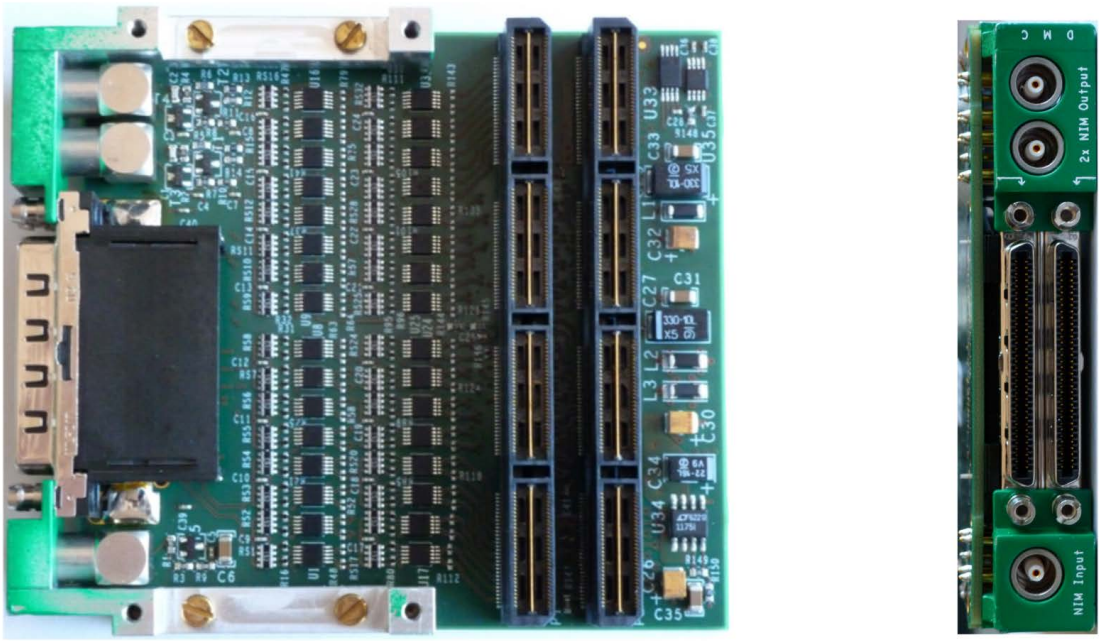

Abbildung 4.5: *Das Bild zeigt eine DMC (links), die über zwei VHDCI-Stecker die Signale von insgesamt 64 Detektorkanälen entgegennehmen beziehungsweise im „Output-Modus“ diese nach außen weiterleiten kann. Rechts sind die VHDCI-Anschlüsse und der NIM-Eingang beziehungsweise die zwei NIM-Ausgänge von vorne zu sehen [49].*

Es besteht außerdem die Möglichkeit durch gedrehte Montage der Signalbuffer die DMC als 64-Kanal Output-Karte zu produzieren. Diese werden zum Beispiel zum Testen verwendet, indem ein „Output-GANDALF“ Testsignale erzeugt und über LVDS-Kabel an einen „Input-GANDALF“ sendet, der die zu testende Logik enthält. Somit kann die Einbindung in das Experiment unter Laborbedingungen simuliert werden (siehe Kapitel 7).

4.5 Das TCS Interface

Das Trigger Control System am COMPASS-II Experiment besteht aus zwei Komponenten. Ob ein „interessantes“ Event stattgefunden hat, wird anhand einer Triggerlogik entschieden, welche mit einer dieser beiden Komponenten (TCS-Controller) verbunden ist [45]. Wird eine Triggerentscheidung getroffen, so generiert der TCS-Controller charakteristische Markierungen zu dem Event, welche an alle Auslesemodule verteilt werden und mit in die Daten geschrieben werden. Bei der Triggergenerierung wird eine minimale Totzeit von $0,4\,\mu$s eingehalten, sodass für diese Zeit nach einem Trigger kein zweiter kommen darf. Eine Zusammenfassung der vom TCS-Controller generierten Triggerinformationen befindet sich in Tabelle 4.1.

Die zweite Komponente ist ein TTCex-Modul [63], welches dazu verwendet wird, die Triggerinformation in einen seriellen Datenstrom zu kodieren und in ein optisches Signal zu konvertieren, welches synchron über Glasfasernleitungen an alle

Auslesemodule des Experiments verteilt wird. Diese können beispielsweise von einer Fibre-Gimli in elektrische Signale umgewandelt und auf den FPGAs bereitgestellt werden. Das TCS Interface stellt eine VHDL-Komponente dar und wird auf dem DSP-FPGA implementiert. Dort wird der Datenstrom mit einem Seriell-zu-Parallel-Wandler dekodiert.

Tabelle 4.1: *Beschreibung der Triggerinformationen, die vom TCS-Controller erzeugt werden. Die Breite der Signalbusse ist in Klammern () angegeben [64].*

Information	**Beschreibung**
eventnumber (20)	Diese Zahl wird nach jedem Trigger um 1 erhöht und gibt somit die Anzahl der seit dem BOS stattgefundenen Trigger an.
spillnumber (11)	Diese Zahl wird nach jedem Spill um 1 erhöht und gibt somit die Anzahl der seit dem Beginn eines Runs (aktive Datennahme) stattgefundenen Spills an.
eventtype (5)	Diese Zahlenkombination gibt die Art des Triggers an: 00000 - Physikalisches Event 00001 - Kalibrationstrigger ⋮ 11011 - Kalibrationstrigger 11100 - Markiert das erste Event eines Run 11101 - Markiert das letzte Event eines Run 11110 - Markiert das erste Event im Spill 11111 - Markiert das letzte Event im Spill

Wie in Abschnitt 3.1 erläutert, erfolgt die Auskopplung der Protonen aus dem SPS mit abwechselnden On- und Offspill-Phasen, wobei die aktive Onspill-Phase, in der Teilchen am Experiment ankommen etwa $9,6\,\mathrm{s}$, die inaktive Offspill-Phase etwa $38,4\,\mathrm{s}$ dauert. Die aktive Phase wird mit dem Beginn-Of-Spill-Signal (BOS) und die inaktive Phase mit dem End-Of-Spill-Signal (EOS) eingeleitet. Diese Signale werden ebenfalls empfangen und können beispielsweise als Resetsignal für Daten-Fifos[9] verwendet werden (siehe Kapitel 5).

4.6 Das CPLD Interface

Wie in Abschnitt 4.1 erwähnt verfügt das GANDALF-Modul über einen Xilinx Cool-Runner CPLD, welcher eine Konfiguration des DSP- beziehungsweise des MEM-FPGAs über die VME64x-Schnittstelle erlaubt. Das CPLD Interface wird auf dem DSP-FPGA implementiert und steuert die Kommunikation zum CPLD. Als Schnittstelle dient unter anderem ein True Dual-Port RAM Konfigurationsspeicher, der beiden Ports simultanen Zugriff auf den dessen Inhalt gewährt [65]. Dies ermöglicht ein Lesezugriff beziehungsweise eine Konfiguration interner Parametern von außen.

[9] FIFO = First In - First Out

Dabei können Änderungen von Parametern über die netzwerkgesteuerte VME-CPU vorgenommen werden. In Kapitel 5 und 6 beziehungsweise Kapitel C werden Parameter erklärt, die über den Konfigurationsspeicher eingestellt werden können. Beispiele solcher Parameter sind „Triggerverzögerung“, „Gateverzögerung“, „Randomtrigger-Frequenz“ und „Randomtrigger-Modus“. Die Speicherbelegung dieser und weiterer Parameter kann [66] entnommen werden.

Eine weitere vor allem für dieses Projekt wichtige Funktion des CPLD Interface stellt der sogenannte „Spy-Fifo“ dar. Dabei handelt es sich um einen 32 Bit breiten Fifo, der direkt über die VME64x-Schnittstelle ausgelesen werden kann. Wie in Kapitel 6 ausführlicher beschrieben, verfügt die im Rahmen dieser Arbeit entwickelte Firmware über die Möglichkeit, einen 96-Kanal Scaler und einen 96-Kanal TDC gleichzeitig auf zwei unterschiedliche Arten auszulesen. Dabei werden die TCS-Daten an ein TIGER-Modul gesendet und über eine SLink-Schnittstelle an die DAQ weitergeleitet (siehe Abschnitt 4.2.2). Zusätzlich werden die zu einem intern generierten Pseudo-Randomtrigger (siehe Abschnitt 6.2) gehörenden Datenpakete mit dem Spy-Fifos direkt über die VME-Backplane ausgelesen. Diese Daten gelangen über die netzwerkgesteuerte VME-CPU unabhängig von der DAQ an ihren gewünschten Bestimmungsort.

Das CPLD Interface bietet noch weitere Möglichkeiten. Beispielsweise können insgesamt 256 sogenannten „Fast-Register“ gesetzt werden. Eine detailliertere Beschreibung des CPLD Interface befindet sich in [20, 48].

4.7 Field Programmable Gate Array

4.7.1 Allgemeines zu FPGA

Ein FPGA ist ein integrierter Schaltkreis (IC[10]) in den eine spezifische digitale Logik einprogrammiert werden kann. Die Grundstruktur eines FPGAs ist eine Matrix aus Logikblöcken, die aus Lookup-Tabellen und Flipflops bestehen. Als Verbindungsstruktur dient eine konfigurierbare Schaltmatrix, durch die Logikblöcke miteinander verknüpft werden können. Dieses Verknüpfungsnetz wird als „Routing“ bezeichnet. FPGAs zeichnen sich vor allem durch ihre einfache Rekonfigurierbarkeit aus. So kann eine digitale Schaltung bei FPGAs sehr einfach korrigiert werden, während dies bei ASICs[11] nicht möglich ist. ASICs werden nur für eine bestimmte Anwendung gefertigt und können im Nachhinein nicht mehr verändert werden. Im Vergleich fallen bei FPGAs für geringe Stückzahlen weniger Kosten an, da keine hohen Fixkosten durch Masken entstehen.

Für die Integration einer Schaltfunktion in den FPGA wird eine Konfigurationsdatei benötigt. Der erste Schritt dazu besteht aus der Modellierung der gewünschten digitalen Funktion durch Hardwarebeschreibungssprachen wie VHDL[12] [67] oder Verilog[13] [68]. Im weiteren Verlauf wird der die Schaltung beschreibende Code synthetisiert, die benötigten Logikblöcke im FPGA ausgewählt beziehungsweise platziert und diese miteinander vernetzt, woraus man schließlich die Konfigurationsdatei erhält. Dazu müssen kombinatorische von getakteten Prozessen getrennt modelliert werden, wobei die Laufzeiten durch kombinatorische Logik kleiner als die Periode der damit verbundenen getakteten Logik sein muss. Für die einzelnen Schritte wird in der Regel folgende Entwurfsmethodik angewandt (vergleiche Abbildung 4.6).

- Die Schaltungseingabe geschieht mittels Hardwarebeschreibungssprachen über einen Texteditor.
- Durch eine Simulation wird geprüft, ob das VHDL-Modell die gewünschte Funktion aufweist und damit als Lösung des Schaltungsentwurfs angesehen werden kann. In diesem Projekt wird hierfür die Simulationssoftware „ModelSim“ [69] benutzt. Dazu wird eine sogenannte VHDL-Testbench verwendet, die die Hardware-Testumgebung simuliert und in die das zu testende Modell (UUT[14]) eingebettet ist. Anschließend werden Testsignale (Stimuli) auf die Eingänge der UUT gegeben und der zeitliche Verlauf der Ausgangssignale überprüft. Entspricht der Verlauf nicht dem gewünschten Ergebnis, so können gezielt interne Signale betrachtet und gegebenenfalls Anpassungen am VHDL-Code vorgenommen werden.
- Der nächste Schritt besteht in der Umwandlung des verifizierten Codes in eine generische Netzliste, wobei noch unberücksichtigt bleibt, auf welche Art die

[10] IC = Integrated Circuit
[11] ASIC = Application-specific integrated circuit
[12] VHDL = Very High Speed Integrated Circuit Hardware Description Language
[13] Verilog = Verifying Logic
[14] UUT = Unit Under Test

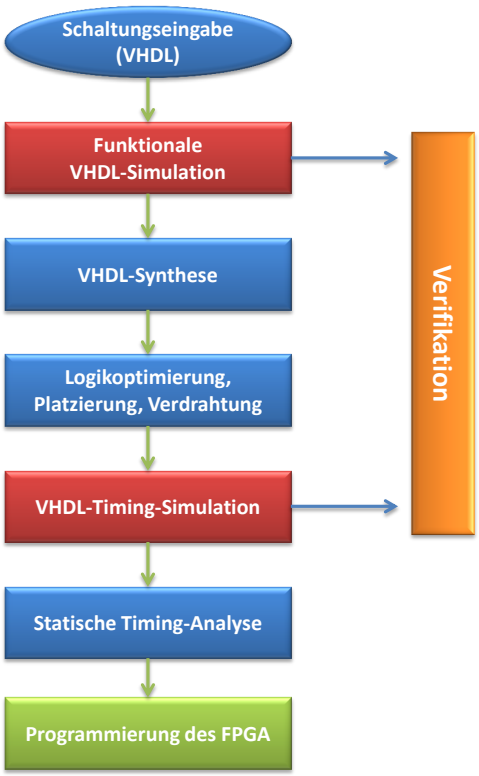

Abbildung 4.6: *Schematische Darstellung der Entwurfsmethodik bei der FPGA-Programmierung [12].*

Umsetzung erfolgt. Das heißt mit welchen logischen Bausteinen der Entwurf implementiert werden soll. Diese Aufgabe übernimmt ein RTL[15]-Synthesetool.

- Nun erfolgt die eigentliche Implementierungsphase PAR[16], in der die generische Netzliste unter Berücksichtigung der tatsächlich vorhandenen Ressourcen optimiert und in eine elektrische Schaltung auf dem FPGA platziert wird, wobei auch die Verdrahtung der Logikbausteine erfolgt.

- Danach erfolgt eine Timing-Simulation. Dazu werden alle Signallaufzeiten der verwendeten Bausteine und deren Verdrahtung berücksichtigt und geprüft, ob das Verhalten der implementierten Schaltung dem des Modells auf Register-Transfer-Ebene entspricht.

- Vor der Generierung der Konfigurationsdatei wird die maximal mögliche Taktfrequenz der Schaltung mit einer Timinganalyse ermittelt. Diese ergibt sich aus jener kombinatorischen Logik, welche die größte Signallaufzeit zwischen zwei Registern besitzt. Die gewünschte Taktfrequenz kann zuvor unter Eingabe sogenannter „timing-constraints“ berücksichtigt werden. Das Implementierungstool versucht dann, die Logikressourcen und deren Verdrahtung so auszuwählen, dass die gewünschten Bedingungen erfüllt werden.

[15]RTL = Register Transfer Level
[16]PAR = Place And Route

- Schließlich erfolgt die Programmierung des FPGAs mit der erhaltenen Konfigurationsdatei, womit die Verifikation der gewünschten Schaltung auf Hardwareebene erfolgen kann. Funktionieren manche Prozesse im FPGA nicht wie gewünscht, so ist dies meist auf fehlgeschlagene Timing-Bedingungen zurückzuführen (letzter Schritt). Dies macht oft eine Optimierung und damit Abänderungen des VHDL-Codes notwendig.

Abgesehen von der funktionalen VHDL-Simulation wird für die Schritte bis zur Erstellung der Konfigurationsdatei die ISE-Design Suite [70] verwendet.

4.7.2 Der Xilinx Virtex-5 FPGA

Der in Kapitel 5 beschriebene 96-Kanal-Scaler und dessen Vereinigung mit einem 96-Kanal-TDC (siehe Kapitel 6) wird auf einem Xilinx Virtex-5 SX95T-FPGA (DSP-FPGA) implementiert. Im Folgenden werden dessen wichtigste Komponenten kurz erläutert.

Der DSP-FPGA besteht aus 7360 konfigurierbaren Logikblöcken (CLB[17]), welche in einer kartesischen Struktur (46 Spalten und 160 Zeilen) auf dem FPGA angeordnet sind. Die CLBs bestehen wiederum aus jeweils zwei Virtex-5 „Slice“-Elementen, die über eine sogenannte „Switch Matrix“ mit dem äußeren Verbindungsnetz verknüpft sind. Über eine sogenannte „Carry Chain“ (CIN, COUT) werden Slice-Elemente benachbarter CLB miteinander direkt verbunden (vergleiche Abbildung 4.7 oben). Alle Slice-Elemente verfügen über vier LUTs[18] mit sechs Eingängen und zwei Ausgängen und außerdem über vier Speichereinheiten, die beispielsweise als D-Flipflop verwendet werden können (siehe 5.1). Die LUTs werden für die Implementierung Boolscher Funktionen mit bis zu sechs Variablen verwendet.

Auf dem Virtex-5 SX95T-FPGA steht mit insgesamt 244 x 36-Kbit BlockRAM-Einheiten zusätzlich 8,784 Mbit Speicher zur Verfügung, der auf vielfältige Art genutzt werden kann. Im vorliegenden Projekt wird dabei hauptsächlich die BlockRAM basierte FIFO-Funktion genutzt [72]. Dabei handelt es sich um einen Speicher welcher eingespeiste Datenworte in der selben Reihenfolge wieder ausgibt. Der Lese- und Schreibport sind voneinander unabhängig und werden über ein *readenable*- und *writeenable*-Port gesteuert, sodass ein FIFO auch asynchron mit unterschiedlichen Frequenzen betrieben werden kann. So kann ein Fifo als Schnittstelle zwischen Logik dienen, die mit unterschiedlicher Frequenz getaktet ist. Ein Fifo kommt außerdem meist als Zwischenspeicher an jenen kritischen Stellen zur Anwendung, wo der Datenfluss zeitweise sehr groß sein kann.

Wie in Unterkapitel 4.1 angedeutet, stehen für die effiziente Implementierung bestimmter Operationen auf dem Virtex-5 SX95T-FPGA 640 sogenannte DSP-Slices zur Verfügung, welche in diesem Projekt hauptsächlich für Additionen verwendet werden (siehe Kapitel 5 und 6).

[17] CLB = Configurable Logic Block
[18] LUT = Look Up Table

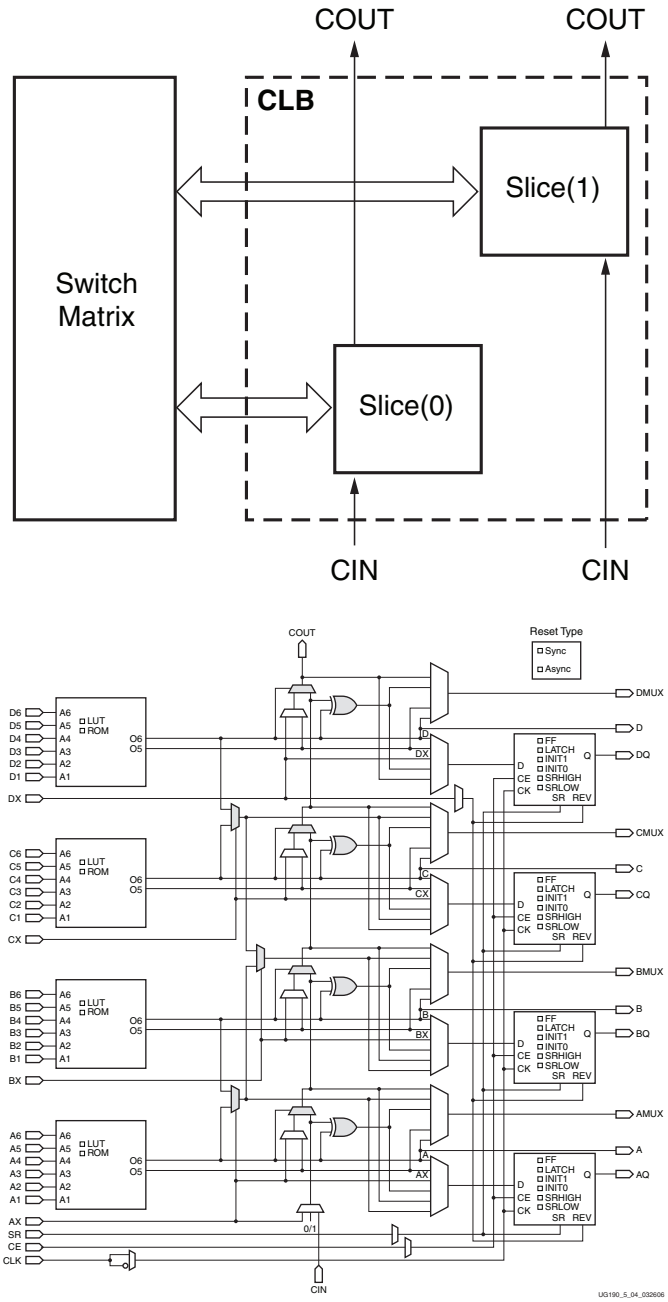

Abbildung 4.7: *Schematische Darstellung eines konfigurierbaren Logikblocks (CLB), welcher aus zwei Slice-Elementen und deren dazugehörige Switch Matrix besteht (oben). Über die Carry Chain (CIN, COUT) werden Slices benachbarter CLB direkt miteinander verbunden. Unten befindet sich eine Darstellung eines Slice-Elements. Der Anschluss an die vier Speichereinheiten erfolgt über die Ports mit den Bezeichnungen AX,BX,CX,DX. Die LUTs werden über die Eingänge A1,...,A6, B1,...,B6, C1,...,C6 und D1,...,D6 angesprochen. In der Mitte der Slice-Elemente befinden sich außerdem programmierbare Multiplexer und die Carry Chain [71].*

5. Entwicklung eines Zählers mit totzeitfreier Auslese

Im Rahmen dieser Diplomarbeit wird ein FPGA-basierter Scaler mit 96 Kanälen entwickelt, der hochfrequente digitale Signale fehlerfrei zählen kann. Dieser wird auf einem Xilinx Virtex-5 FPGA (siehe Kapitel 4) implementiert und kommt auf einem GANDALF-Modul zur Auslese szintillierender Fasern zur Anwendung.

Im vorliegenden Kapitel wird die Funktionsweise eines Zählers und deren elektronische Grundlagen beschrieben. Dafür werden verschiedene Zählercodes und deren Vor- und Nachteile erörtert. Anschließend wird das Scaler-Design erläutert. Herzstück des Scalers ist ein schneller Johnson-Ringzähler, der sich durch seine einfache Logik und fehlerfreie Auslese auszeichnet.

Im Rahmen einer früheren Diplomarbeit wurde eine 32-Kanal-Scaler-Aufsteckkarte für das CATCH-Modul entwickelt [73]. Darin kommen ebenfalls Johnson-Ringzähler zur Anwendung. Dieses Konzept hat sich über die letzten Jahre als erfolgreich erwiesen.

5.1 Prinzip eines Scalers

Ein Scaler basiert auf einer Funktion, die bei jeder steigenden Flanke des Eingangssignals aus einem aktuellen Zählerstand den nächsten bestimmt:

$$j \rightarrow j + 1. \tag{5.1}$$

Die Speicherung des Zählerstands geschieht mit Flipflops, die elektronisch als Bit-Speicher angesehen werden können. Flipflops können zwei stabile Zustände $(0, 1)$ annehmen. Dabei wird der Wert des am Dateneingang (D) des Flipflops anliegenden Signals genau dann auf den Ausgang weitergeleitet, wenn am Clock-Eingang (clk) eine ansteigende Flanke erfolgt. Die einfachste Form eines Flipflops besteht also aus zwei Eingängen und einem Ausgang.

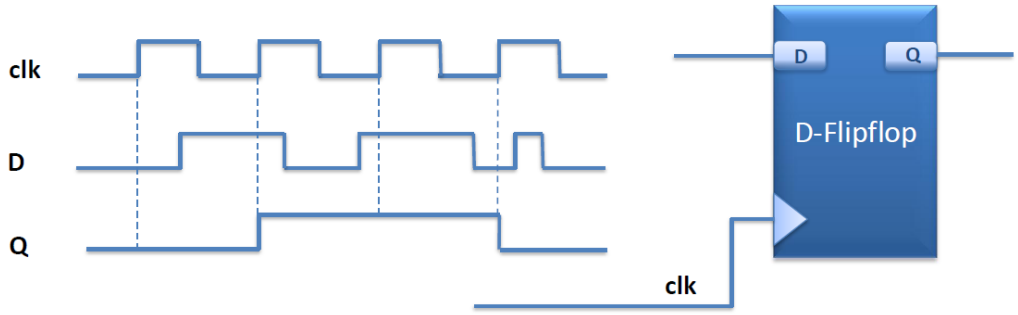

Abbildung 5.1: *Prinzip eines Flipflops*

Die für einen Zähler verwendete Logik berechnet nach jeder steigenden Flanke des Eingangssignals (clk) den neuen Zählerstand, der in einem Register gespeichert wird. Der einfachste Zähler besteht somit aus einem Addierer und einem Register. Bei der Inkrementierung vom Zustand $7 = 0111$ auf $8 = 1000$ im Binärcode ändern sich beispielsweise vier Bits. Je nach Anzahl der Bitstellen vergrößert sich die maximale Anzahl der sich ändernden Bits. Der Zähler benötige die Zeit Δt, um einen neuen Zählerstand zu berechnen. Δt ist von der Komplexizität der Zählerlogik abhängig. Sie bestimmt also maßgeblich die maximale Frequenz $\nu_{max} = 1/\Delta t$, unterhalb derer Signale fehlerfrei gezählt werden können.

Die Bestimmung des LSB[1] für den nächsten Zählerstand im Binärcode ist nur abhängig von sich selbst:

$$Bit_0^* = f(Bit_0) = Bit_0^{-1}. \tag{5.2}$$

Für die höherwertigen Bits werden die Funktionen komplexer. So ist die Logik für das zweite Bit von zwei Variablen

$$Bit_1^* = f(Bit_0, Bit_1) = (Bit_0^{-1} \wedge Bit_1) \vee (Bit_0 \wedge Bit_1^{-1}), \tag{5.3}$$

[1] LSB = Least Significant Bit

und das n te Bit von n Variablen abhängig,

$$Bit^*_{n-1} = f(Bit_0, Bit_1, ...Bit_{n-1}). \tag{5.4}$$

Sofern die Zeit für die Berechnung aller Bitstellen nicht ausreicht, so läge der bei einer zu früh ankommenden Signalflanke auftretende Fehler in obigem Beispiel bei bis zu acht.

Ein weiteres Problem besteht in der Auslese des Zählerstandes, wozu ein zweites Register benutzt wird. Wird der Zählerstand zeitgleich zu einem Auslesevorgang inkrementiert, kann dies zu Fehlern in diesem Register führen. Das Problem kann durch Verwendung mehrerer Zähler in einer Kaskade umgangen werden, was folgendermaßen realisiert werden könnte.

Für den Zähler wird ein Ringzähler mit einfacher Logik (Δt möglichst klein) verwendet, der von 0 bis $N-1$ zählt $(0, 1, 2, ...N-1, 0, 1, 2...)$ und nach jedem vollen Umlauf einen weiteren Zählerstand um $+1$ inkrementiert. Damit ergibt sich die Anforderung für die maximale Berechnungszeit $T = N \cdot \Delta t$ an den zweiten Zähler. Ein 4 Bit-Binär-Ringzähler mit beispielsweise $\Delta t = 2\,\text{ns}$ könnte ein regelmäßiges 500 MHz-Signal fehlerfrei zählen. Damit ergibt sich die Anforderung $\Delta T < 2^4 \cdot 2\,\text{ns}$ $= 32\,\text{ns}$ an den zweiten Zähler.

Eine andere Umsetzung mit Ringzählern besteht darin, diesen getaktet (CLK) auszulesen und die Werte aufzuaddieren. Eine Auslese findet also alle $\Delta T = 1/\nu_{CLK}$ statt. Diese Möglichkeit kommt für das Scaler-Design zur Anwendung.

5.2 Verschiedene Codes und deren Vor- und Nachteile

Es bleibt die Logik des Vorzählers zu erörtern, um eine möglichst hohe Frequenz fehlerfrei zählen zu können. Außerdem besteht für die zweite Möglichkeit das Problem, einen falschen Zählerstand im ausgelesenen Register zu erhalten, wenn die Auslese zeitgleich zu einer Erhöhung des Zählerstands erfolgt.

Diese Problematik tritt bei Codes mit Hammingdistanzen $=\text{const.} = 1$ nicht auf. Die sogenannte Hammingdistanz bezeichnet die Anzahl der sich ändernden Bits zweier aufeinander folgenden Werte. Dieser Abstand kann für einen binären Zählvorgang groß sein ($0111 \rightarrow 1000$). Im Vergleich dazu wäre bei einem Code mit Hammingdistanz $= 1$ die Auslese immer ± 1 richtig. Dafür eignet sich beispielsweise der Gray- und der Johnson-Code. In Tabelle 5.1 werden vier verschiedene Codes verglichen. In Tabelle 5.2 sind deren Eigenschaften dargestellt. Im nachfolgenden Abschnitt werden Vor- und Nachteile der Codes erläutert.

5.2.1 Binär- und Gray-Code

Ein Binärzähler hat keine undefinierten Zustände und ist somit sehr platzsparend, da nur wenige Flipflops benötigt werden. Den gleichen Vorteil bringt auch der Gray-Code mit sich, der zudem eine konstante Hammingdistanz von 1 hat. Allerdings wird

Tabelle 5.1: *Darstellung verschiedener Ringzähler und deren definierte Zustände von 0 bis 15.*

Zustand	**Binär**	**Gray**	**Johnson**	**One-Hot**
0	0000	0000	00000000	1000000000000000
1	0001	0001	10000000	0100000000000000
2	0010	0011	11000000	0010000000000000
3	0011	0010	11100000	0001000000000000
4	0100	0110	11110000	0000100000000000
5	0101	0111	11111000	0000010000000000
6	0110	0101	11111100	0000001000000000
7	0111	0100	11111110	0000000100000000
8	1000	1100	11111111	0000000010000000
9	1001	1101	01111111	0000000001000000
10	1010	1111	00111111	0000000000100000
11	1011	1110	00011111	0000000000010000
12	1100	1010	00001111	0000000000001000
13	1101	1011	00000111	0000000000000100
14	1110	1001	00000011	0000000000000010
15	1111	1000	00000001	0000000000000001
0	0000	0000	00000000	1000000000000000

Tabelle 5.2: *Eigenschaften der Codes (n = Anzahl der Bitstellen).*

	Binär	**Gray**	**Johnson**	**One-Hot**
Maximale Hammingdistanz	n	1	1	2
Definierten Zustände	2^n	2^n	$2 \cdot n$	n
Undefinierten Zustände	0	0	$2^n - 2 \cdot n$	$2^n - n$

ein Dekoder benötigt, sofern die Zählerstände ins Binärsystem dekodiert werden sollen. Der nach Frank Gray benannte Code stellt eine Umordnung des Binärcodes dar und lässt sich mit einer *xor*-Verknüpfung durch folgenden Pseudocode realisieren:

$$x_b \rightarrow x_g = x_b \veebar x_{sh}. \tag{5.5}$$

Hierbei stellt x_b eine Dualzahl im Binärcode und x_g die gewünschte Zahl im Gray-Code dar. x_{sh} erhält man durch einen Rechts-Shift der Zahl x_b um ein Bit. Sowohl der Binär- als auch der Gray-Code haben trotz der genannten Vorteile das Problem, dass ihre zugrunde liegende Logik komplex ist. Je nach Übergang $j \rightarrow j + 1$ kann die benötigte Zeit Δt groß werden.

5.2.2 One-Hot- und Johnson-Code

Die in Tabelle 5.1 und 5.2 dargestellten Johnson- und One-Hot-Codierungen erscheinen zunächst aufwendig und ineffizient, da im Vergleich zur Binär- oder Gray-

Codierung undefinierte Zustände auftreten. Dennoch haben diese bezüglich ihrer Berechnungszeiten Δt einen großen Vorteil.

Die One-Hot-Codierung lässt sich durch ein 16 Bit-Schieberegister mit Anfangszustand 1000000000000000 realisieren. Hierbei handelt es sich um einen geschlossenen Ring aus 16 Flipflops, deren CLK-Eingänge mit dem Eingangssignal verbunden sind. Der Dateneingang ist jeweils mit dem Datenausgang eines benachbarten Flipflops verbunden, wodurch eine geschlossene Kette entsteht. Somit wird das aktive Bit bei jeder ansteigenden Flanke des Eingangssignals um eine Stelle weitergeschoben. Das Register für die Auslese des Ringzählers besteht ebenfalls aus 16 Flipflops, die jeweils mit den Datenausgängen (Q) der Flipflops aus dem Schieberegister verbunden sind.

Diese Methode besitzt aufgrund ihrer simplen Logik eine sehr geringe Berechnungszeit Δt, würde sich also für die Zählung von hochfrequenten Signalen eignen. Dennoch besitzt sie den Nachteil, dass sie bezüglich der Anzahl der benötigten Flipflops sehr ineffizient ist. So werden für n Zustände insgesamt $2n$ Flipflops für Ringzähler und Register benötigt. Zudem ist die Hammingdistanz immer = 2, was in einen undefinierten Zustand im zweiten Register, also etwa 0000000000000000 oder 0...0110...0, führen könnte, sofern die Erhöhung des Zählerstandes zeitgleich zu einer Auslese erfolgt.

Der beste Kompromiss zwischen Effizienz und Komplexität der Logik stellt wohl die Johnson-Codierung dar. Zwar existieren ebenfalls undefinierte Zustände, jedoch benötigt dieser im Vergleich zur One-Hot-Codierung für die gleiche Anzahl definierter Zustände nur halb so viele Flipflops. Der Johnson-Zähler besteht aus einem geschlossenen Schieberegister, wobei das erste und das letzte Flipflop durch einen Inverter miteinander verknüpft sind. Die Flipflops des daran anschließenden Registers sind, wie beim One-Hot-Counter, jeweils mit den Ausgängen der Flipflops aus dem Vorzähler verbunden. Der Johnson-Ringzähler besitzt eine sehr kleine Berechnungszeit Δt. Außerdem ist die Wahrscheinlichkeit für undefinierte Zustände nahezu unmöglich, da immer nur ein Flipflop schaltet (siehe Tabelle 5.1). Für das im Folgenden beschriebene Scaler-Design werden 8 Bit-Johnson-Ringzähler als schnelle Vorzähler verwendet, die zu regelmäßigen Zeitpunkten ausgelesen werden.

5.3 Das Scaler-Design

5.3.1 Vorgaben

- **Maximal zählbare Frequenz** Der Scaler soll im Experiment die hochfrequenten Myon-Signale der szintillierenden Fasern zählen. Diese besitzen eine sehr geringe Totzeit von $t_d = 5\,\text{ns}$. Wird zum Zeitpunkt t_0 ein Signal erzeugt, so kann erst zum Zeitpunkt $t_0 + t_d$ ein weiteres Teilchen nachgewiesen werden. Damit ergibt sich als direkte Vorgabe an den Scaler, ein regelmäßiges Signal mit Frequenz

$$\nu = \frac{1}{t_d} = 200\,\text{MHz}, \tag{5.6}$$

fehlerfrei zählen zu können, also $\Delta t \overset{!}{<} t_d$.

- **Breite des Gesamtzählerstandes**

  Betrachtet man die Totzeit szintillierender Fasern von $t_d = 5\,\text{ns}$, so kann ein Zählerstand bei einer Spilldauer von $\approx 10\,\text{s}$ maximal $10 \cdot 200 \cdot 10^6 = 2 \cdot 10^9$ betragen, dieser Fall wird am Experiment jedoch nicht auftreten, die Myonen aufgrund der Aufweitung des Teilchenstrahls auf mehrere Kanäle verteilt gemessen werden. Mit einer Breite von 31 Bit sind die Zählerstände mit bis zu $2^{31} \approx 2,147 \cdot 10^9$ also sehr großzügig bemessen.

- **Totzeitfreie Auslese**

  Die Triggerfrequenz am COMPASS-II Experiment kann bis zu 100 kHz betragen und der minimale Zeitabstand zweier Trigger beträgt $0,4\,\mu\text{s}$. Die Auslese sollte nach dieser Zeit also wieder möglich sein. Dabei ist es wichtig, dass der Zählvorgang währenddessen nicht unterbrochen wird, also eine totzeitfreie Auslese stattfinden kann. Andernfalls würde man einen Fehler im Zählerstand erhalten, der dann zu niedrig wäre, da innerhalb der Totzeit keine Signale gezählt werden könnten.

- **Verzögerung des Triggers**

  Wird am Experiment ein TCS-Trigger erzeugt, so benötigt sowohl seine Erzeugung, als auch seine Laufzeit bis zum Auslese-Modul eine Zeit t_{TR} (siehe Abbildung 5.2). Das Trigger-Signal kommt bezüglich des Events auf das getriggert wurde verzögert ($\approx \mu s$) an. Die Verzögerung bewirkt, dass ein zu hoher Zählerstand ausgelesen wird, weil zwischen dem Zeitpunkt des Events und dem Zeitpunkt des Trigger-Signals weitergezählt wird. Daher gilt es, diese Verzögerung auszugleichen.

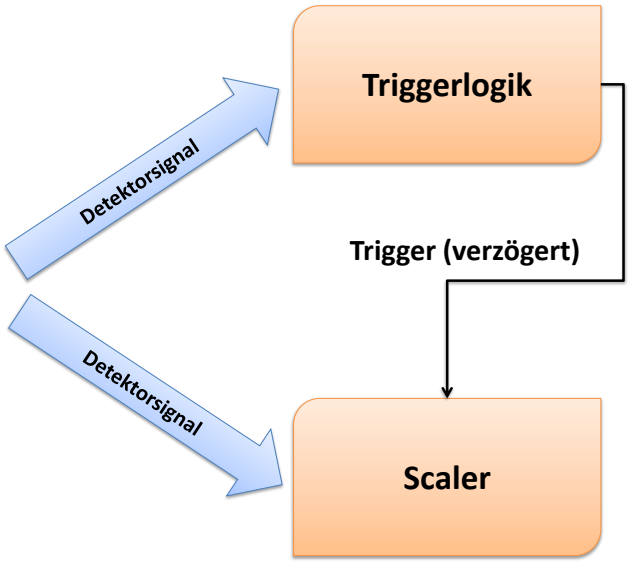

Abbildung 5.2: *Schematische Darstellung der Triggerverzögerung.*

- **Das Gate und dessen Verzögerung**

  Trigger beziehungsweise Veto-Trigger sorgen für Totzeiten, in denen keine weiteren Ereignisse akzeptiert werden können. Um dies zu berücksichtigen, soll eine Gate-Funktion implementiert werden. Sie sorgt dafür, dass in diesen Zeitabschnitten nicht weitergezählt wird. Da das Gate-Signal, wie das Trigger-Signal, verzögert am Auslese-Modul ankommt, gilt es, diese Verzögerung auszugleichen.

- **Randomtrigger**

  Neben den TCS-Triggern wird ein von der DAQ unabhängiger Randomtrigger benötigt. Dabei wird auf eine Gate-Funktion verzichtet, das heißt, es ist notwendig zwei verschiedene Auslese-Register zu implementieren. Der Randomtrigger soll als sogenannter Pseudo-Randomtrigger auf dem FPGA generiert werden (siehe Kapitel 6.2). Die Random-Daten sollen über die VME-Backplane ausgelesen werden, um damit verschiedene Strahlattribute auf einem Bildschirm im COMPASS-Controlroom zu visualisieren.

- **Konfigurierbare Funktionen**

  Einige Funktionen der auf den FPGA programmierten Logik sollen auch während dem laufenden Experiment programmierbar sein, ohne den FPGA neu laden zu müssen. Dies ist beispielsweise für die Konfiguration der Trigger- und Gateverzögerung der Fall.

- **Ressourcenverbrauch**

  Eines der wichtigsten Ziele ist es, möglichst wenig FPGA-Ressourcen zu verbrauchen, da noch andere erweiternde Funktionen geplant sind. So sind für dieses Projekt 96 Scaler- und TDC-Kanäle vorgesehen. Im Rahmen einer früheren Diplomarbeit [49] wurde bereits ein TDC für das GANDALF-Modul entwickelt und getestet (siehe Abschnitt 6.1). Der VHDL-Code dieser Entwicklung wird bei der Vereinigung (siehe Kapitel 6) beider Komponenten verwendet.

5.3.2 Johnson-Ringzähler und Johnson-Register

Wie in Abschnitt 5.2 erörtert, fällt die Wahl des Zählmechanismus auf den Johnson-Ringzähler. In Abbildung 5.3 wird der Zählvorgang schematisch erklärt.

Der Zustand des Johnson-Ringzählers wird mit einem regelmäßigen Taktsignal (*CLK38MHz*) alle $25,72\,\text{ns}$ ausgelesen und in ein 8 Bit-Register (Johnson-Register) übernommen. Das *CLK38MHz*-Signal wird aus dem TCS-Taktsignal erzeugt (siehe Abschnitt 4.3). Es hat die Frequenz $\nu = 38,88\,\text{MHz}$ ($1/4 \times 155,52\,\text{MHz}$). Da die Hammingdistanz des Johnson-Ringzählers immer $= 1$ ist, kann es zu keinen undefinierten Zuständen im Johnson-Register kommen, womit der momentane Fehler darin maximal ± 1 beträgt.

5.3.3 Dekoder und Triggerverzögerung

Die Daten im Johnson-Register werden zunächst ins Binärsystem dekodiert (siehe Abbildung 5.4). Damit werden im weiteren Verlauf Flipflops eingespart. Die Logik

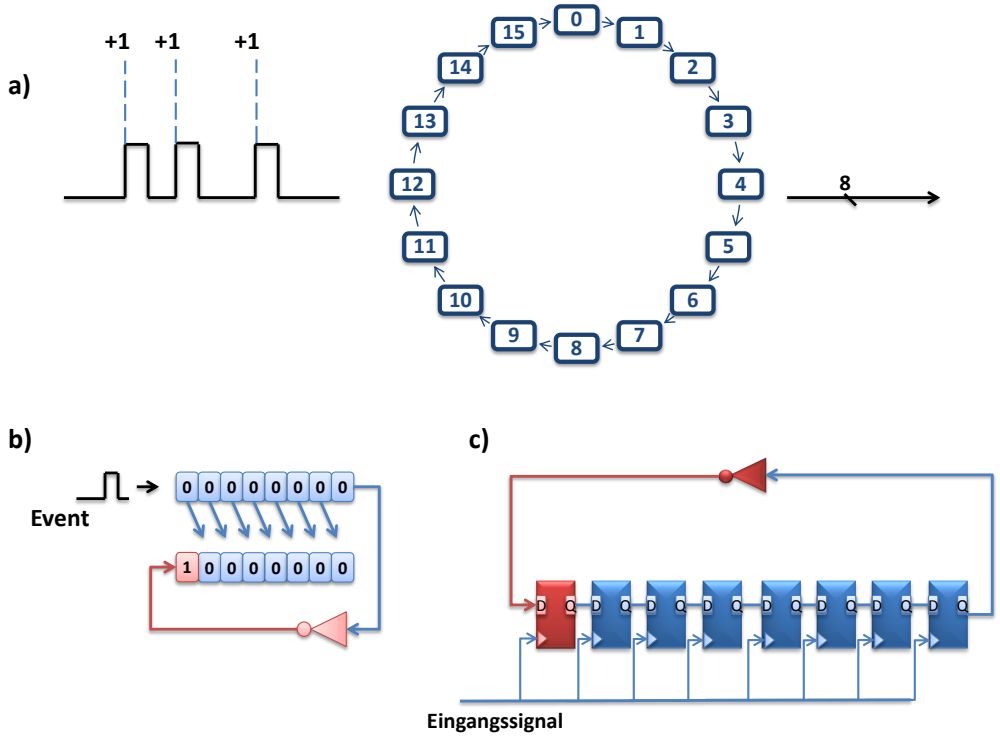

Abbildung 5.3: *(a) Darstellung eines Ringzählers mit 16 Zuständen. (b) Zählmechanismus im Johnson-Code: Der Inhalt aller Flipflops wird bei einer steigenden Flanke des Eingangssignals um eine Position versetzt, während eines über einen Inverter mit der erste Position verbunden ist. (c) Logik eines 8 Bit-Johnson-Ringzählers. Die Resetfunktion durch das BOS- und EOS-Signal bleibt in der Abbildung unberücksichtigt.*

des Dekoders lässt sich mit einer $8 \rightarrow 4-$Bit-Wahrheitstabelle realisieren, welche man Tabelle 5.1 entnehmen kann. Der Dekodiervorgang kann ungetaktet erfolgen und wird mit der LUT-Funktion der Slice-Elemente realisiert. Ändert sich der Wert am Eingang des Dekoders, so ändert sich dementsprechend der Wert am Ausgang, wofür es keines Taktsignals bedarf.

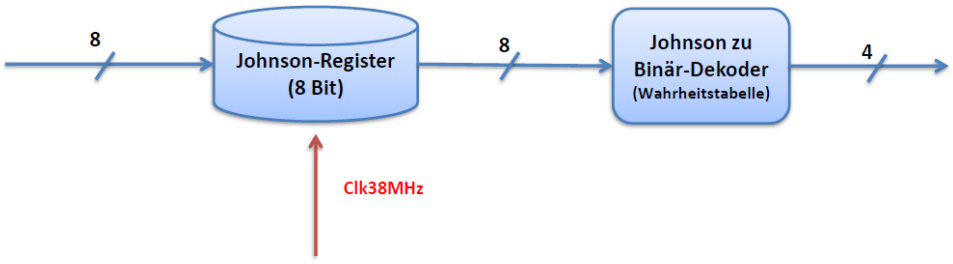

Abbildung 5.4: *Dekodiervorgang mittels einer* $8 \rightarrow 4$*-Wahrheitstabelle.*

Hinter dem Dekoder wird die Triggerverzögerung mit einem BlockRAM-basierten Schieberegister realisiert. Die Funktionsweise lässt sich anhand Abbildung 5.5 ver-

anschaulichen. Es handelt sich hierbei um eine 512 Worte tiefe Dual-Port-RAM-Einheit, wobei der *read*- beziehungsweise *write*-Pointer der gleichen Taktung unterliegen (*CLK38MHz*). Zu Beginn eines Spills wird die Verzögerungseinheit mit Hilfe des BOS-Signals resettiert. Außerdem werden die Startadressen festgelegt, auf die die Pointer zu Beginn des Spills zeigen. Der *write*-Pointer wird auf die Adresse „0“ gesetzt. Um die Startposition des *read*-Pointers festzulegen, wird die dafür festgelegte Adresse im Konfigurationsspeicher, der mit dem Cool-Runner CPLD (siehe Abschnitt 4.6) verbunden ist, ausgelesen. Damit erhält man ein 9 Bit breites Signal (ADDR0), welches die gewünschte Triggerverzögerung in Einheiten des 38, 88 MHz-Taktsignals enthält.

Nach dem BOS-Signal werden die Adressen auf die die beiden Pointer zeigen pro Taktzyklus um eine Stelle inkrementiert und bei der Adresse 511 auf 0 zurückgesetzt, womit der *read*-Pointer dem *write*-Pointer im gleichen Abstand nachläuft. Damit werden zu Beginn $511 - z$ mal Nullen ausgelesen, während schon dekodierte Werte $\neq 0$ aus dem Johnson-Register in die BlockRAM-Einheit hineingeschrieben werden (siehe Abbildung 5.5). Bei z handelt es sich um den einstellbaren Wert für die Triggerverzögerung, welche $(511 - z) \cdot 25,72\,\text{ns}$ beträgt. Mit dem Schieberegister lassen sich Triggerverzögerungen bis zu $13,2\,\mu\text{s}$ einstellen, was für die Anforderungen am Experiment ausreichend ist. Dort beträgt die Triggerverzögerung wenige μs. Da auf dem Xilinx Virtex-5 FPGA BlockRAM in 18 kBit-Blöcken ($512 \cdot 34$ Bit, davon $512 \cdot 2$ optionale Paritätsbits) vorliegt, werden jeweils acht Scaler-Kanäle ($8 \cdot 4$ Bit) zusammengefasst. Die Triggerverzögerung hätte man nicht mit auf Flipflop basierenden Schieberegistern implementieren können, da die dafür benötigte Anzahl an Flipflops zu groß wäre.

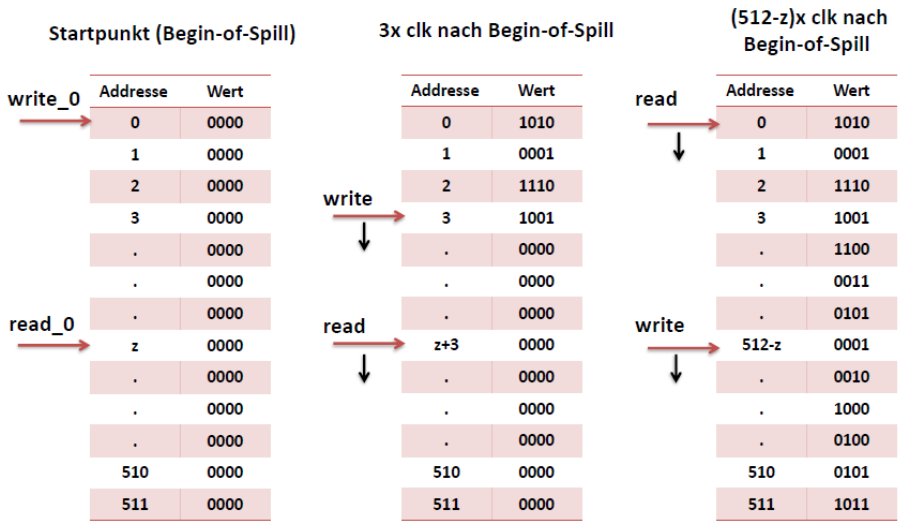

Abbildung 5.5: *Ausgleich der Triggerverzögerung mit einem* 512 *Worte tiefen BlockRAM-basierten Schieberegister. write_0 und read_0 bezeichnen den Zustand der read- und write-Pointer zu Beginn eines Spills.*

5.3.4 Subtraktion, Addition und Gate-Funktion

Mit einem Subtrahierer und einem Addierer wird der aktuelle 31 Bit-Gesamtzählerstand bestimmt. Die Funktion des Subtrahierers wird in Abbildung 5.6 ver-

anschaulicht. Das 4 Bit-Signal aus dem BlockRAM-Schieberegister wird mit dem *CLK38MHz*-Signal getaktet in ein 4 Bit-Register geschrieben. Der Ausgang des Registers ist mit dem Eingang eines weiteren 4 Bit-Registers verbunden, welches das um 25, 72 ns verzögere Datenwort enthält. Dies ist mit den Bezeichnungen „new“ und „old“ angedeutet.

In den Registern stehen also die aufeinander folgenden aus dem Johnson-Ringzähler ausgelesenen Werte, verzögert durch das BlockRAM-basierte Schieberegister. Durch Subtraktion des alten vom neuen Werts erhält man die Anzahl der Signalflanken zwischen den zwei Auslesen. Dabei gilt es, ein Überrollen ($15 \rightarrow 0$) zu berücksichtigen. Diese Anforderung erfüllt ein Subtrahierer im Zweierkomplement [12]. In dieser Darstellung lassen sich auch negative Binärzahlen darstellen. Um eine negative Zahl, beispielsweise -4, im Zweierkomplement zu erhalten, werden alle Bitstellen des Betrags dieser Zahl invertiert ($+4 = 0100 \rightarrow 1011$) und die Zahl 1 addiert ($\rightarrow 1100$). Für die Berechnung von beispielsweise $5 - 4$ erhält man folgendes Ergebnis:

$$\begin{aligned} & 0101 \\ + & \,1100 \\ = & \not{1}0001. \end{aligned}$$

Unter Vernachlässigung des Übertragsbits (MSB), erhält man die richtige Lösung $5-4 = 1$. Betrachtet man einen Fall in dem ein Überrollen stattfindet, beispielsweise $2-13$, so erhält man mit der Zweierkomplement-Darstellung der Zahl $-13_{2er} = 0011$ das gewünschte Ergebnis 5:

$$\begin{aligned} & 0010 \\ + & \,0011 \\ = & \,0101. \end{aligned}$$

Die Funktion eines Subtrahierers im Zweierkomplement lässt sich durch Inverter und Addierer realisieren. Die Implementierung des Subtrahierer kann ungetaktet erfolgen, da dieser nur kombinatorische Logik enthält.

Das Ausgangssignal des Subtrahierers wird mit Eingang A eines 31-Bit-Addierers verbunden, dessen Ausgang auf Eingang B liegt (siehe Abbildung 5.7). Der Zählerstand wird zu Beginn eines Spills durch das BOS-Signal resettiert. Es entsteht alle 25, 72 ns ein neuer Zählerstand, der pro Taktzyklus um maximal 15 erhöht werden kann. Wie in Abbildung 5.7 ersichtlich, wird der 31 Bit-Zählerstand mit jedem Trigger, sowohl TCS- als auch Randomtrigger, in 31 Bit-Auslese-Register geschrieben, wofür zwei verschiedene benutzt werden müssen. Es werden ebenso zwei 31 Bit-Addierer (A1, A2) benötigt, da man einerseits die mit dem Gate-Signal korrigierten Zählerstände (TCS-Trigger, A1) für die DAQ-Auslese und andererseits die unkorrigierten Daten für das Beam-Monitoring (Pseudo-Randomtrigger, A2) benötigt. Da die beiden Zählerstände im Laufe eines Spills mehr und mehr voneinander abweichen, muss auf zwei Addierer zurückgegriffen werden. Im Gegensatz zu A2, besitzt A1 für die Implementierung der Gate-Funktion einen CE-Eingang. A2 addiert die Werte unabhängig davon zu jeder steigenden Takt-Flanke auf.

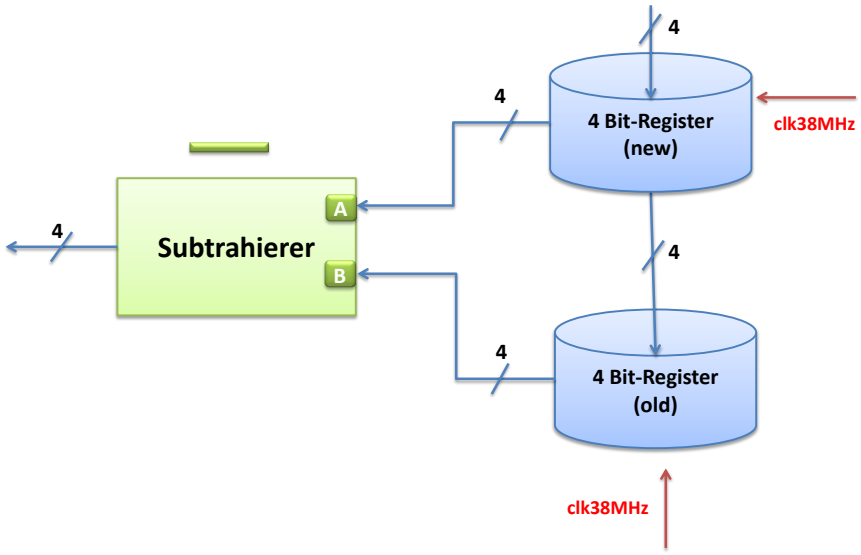

Abbildung 5.6: *Ein Subtrahierer bestimmt die Anzahl der steigenden Signalflanken zwischen zwei Auslesen des Johnson-Ringzählers. Die Werte des Ringzählers sind durch das BlockRAM-basierte Schieberegister verzögert.*

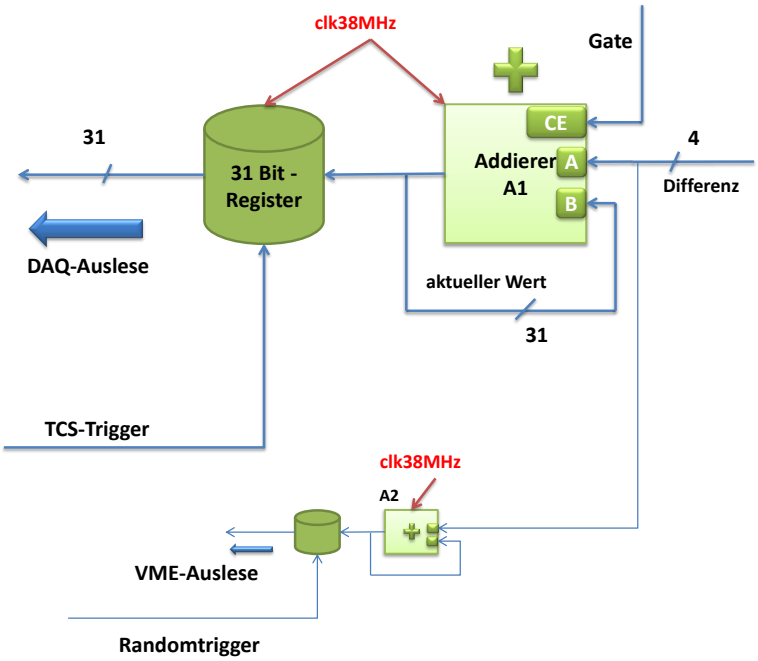

Abbildung 5.7: *In regelmäßigen Zeitabschnitten wird einem aktuellen Wert (B) die Anzahl der zwischen zwei Auslesevorgängen des Johnson-Ringzählers stattgefundenen Events (A) aufaddiert. Dabei besitzt nur einer der beiden Addierer einen Clock Enable (CE)-Eingang, der für die Gate-Funktion verwendet wird. Der aktuelle Wert wird zum Zeitpunkt eines Triggers in das 31 Bit breites Auslese-Register geschrieben.*

Folgende Punkte gilt es zu beachten:

- Es wird mit jedem Trigger pro Scalerkanal ein weiteres 1 Bit-Register benötigt, worin die Information gespeichert wird, ob zum Zeitpunkt des Triggers Events gezählt worden sind. Liegen am Ausgang des Subtrahierers Nullen „0000“, so wird eine „0“ in dieses Register geschrieben. In allen anderen Fällen wird eine „1“ gespeichert. Dieses Bit wird als Teil des Datenformats für die TCS-Auslese benötigt.

- Die seit dem BOS-Signal vergangene Zeit wird mit einem 30 Bit-Binär-Counter unter Benutzung eines DSP-Slices in Einheiten des Taktsignals *CLK38MHz* gezählt. Die Zeitinformation wird mit jedem TCS-Trigger in ein 30 Bit-Register geschrieben. Ebenso wird die Zeit mit einem weiteren Counter gezählt und bei jedem Randomtrigger gespeichert. Der Zeitpunkt des Resets des zweiten Counters entspricht nicht dem des ersten, da ein vom TCS unabhängiges Begin-Of-Spill-Signal erzeugt wird (siehe Abschnitt 6.2.2), womit sich auch die momentanen Zustände der beiden Counter unterscheiden. Mit dem Zählvorgang der seit dem BOS-Signal erfolgten Taktzyklen, lässt sich den Triggern und den dazugehörigen Daten eine Zeitinformation hinzufügen.

- Die Zeit, die sich aus der Erzeugung des Gate-Signals und seines Weges bis zum GANDALF-Modul zusammensetzt, muss, ähnlich wie die Triggerverzögerung, ebenfalls berücksichtigt werden. Die Einstellung der Verzögerung kann über den Konfigurationsspeicher vorgenommen werden. Verwendet wird ein 8 Bit-Signal, womit Verzögerungen bis zu $6,6\,\mu$s einstellbar sind. Die Verzögerung wird in $25,72$ ns-Schritten eingestellt. Das Gate-Signal kann auf alle Scalerkanäle verteilt werden, das heißt es genügt ein einziges Schieberegister, das mit Flipflops umgesetz wird.

5.3.5 Zusammenfassung des 1-Kanal-Scaler-Designs

Das 1-Kanal-Scaler-Design wird in wenigen Worten zusammengefasst, siehe Abbildung 5.8. Verwendet wird ein 8 Bit-Johnson-Ringzähler, dessen Logik sehr einfach ist, sodass die Zeit, die bei einer steigenden Flanke des Eingangssignals für die Bestimmung des neuen Zustands benötigt wird sehr gering ist. Der Ringzähler wird mit einem maximalen Fehler von ± 1 alle $25,72$ ns getaktet ausgelesen. Die Triggerverzögerung wird mit Hilfe eines BlockRAM-basierten Schieberegisters bewerkstelligt. Die verwendeten BlockRAM-Einheiten haben eine Tiefe von 512 Worte, sodass Verzögerungen bis $13,2\,\mu$s über den Konfigurationsspeicher einstellbar sind. Anschließend werden zwei aufeinander folgende Werte subtrahiert und das Ergebnis einem aktuellen 31 Bit-Zählerstand aufaddiert. Dazu werden zwei verschiedene Addierer A1 und A2 benötigt, wovon nur A1 einen CE-Eingang für die Gate-Funktion besitzt. Das Gate-Signal wird mit einem konfigurierbaren Schieberegister verzögert, wobei Zeiten bis $6,6\,\mu$s einstellbar sind. Die Werte, die am Ausgang der Addierer anliegen, werden mit jedem Trigger (TCS- oder Randomtrigger) in 31 Bit-Auslese-Registern gespeichert.

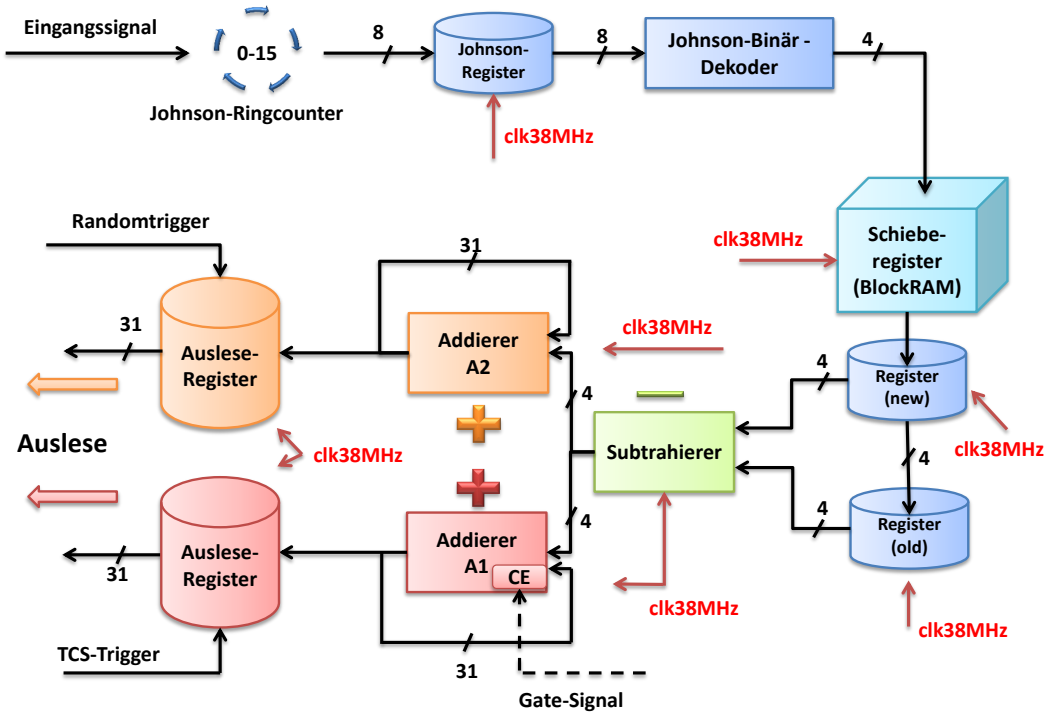

Abbildung 5.8: *Schematische Zusammenfassung des 1-Kanal-Scaler-Designs.*

5.3.6 Implementierung des 1-Kanal-Scaler-Designs

Betrachtet man das 1-Kanal-Scaler-Design, so stellt man fest, dass sehr viele Register und damit Flipflops benutzt werden, was laut Vorgaben vermieden werden sollte. Mit der Idee der Vereinigung des 96-Kanal-Scalers mit einem 96-Kanal-TDC auf einem GANDALF-Modul stellt sich die Frage, wie man die Anzahl der für den Scaler verwendeten Flipflops verringern könnte. Da der Xilinx Virtex-5 FPGA (siehe Kapitel 4) über 640 DSP-Slices verfügt, wovon im vorhandenen TDC-Design nur sehr wenige benutzt werden, bot es sich unter diesem Gesichtspunkt an, Flipflops durch Benutzung von DSP-Slices einzusparen. In Abbildung 5.9 wird die Ersetzung von Registern mit 48 Bit-DSP-Addierern veranschaulicht. Der 31 Bit-Addierer A2, das Johnson-Register und die beiden 4 Bit-Register „new“ und „old“ können auf einem DSP-Slice-Element zusammengefasst werden. Dafür darf nur der Addierer A2 ohne CE-Eingang herangezogen werden, da sonst die Auslese des Johnson-Ringzählers durch das Gate-Signal verhindert werden könnte. Das Johnson-Register und die beiden 4 Bit-Register werden nicht durch Überträge verfälscht, da am Eingang B des DSP-Addierers (siehe Abbildung 5.9 links) nur kleine Werte bis zu 15 aufaddiert werden und der maximale Zählerstand während eines Spills maximal $2 \cdot 10^9$ betragen kann. Der Addierer A1 wird ebenfalls mittels eines 48 Bit-DSP-Addierers umgesetzt, allerdings nur unter Benutzung von 31 Bitstellen und ohne zusätzliche Register.

Des Weiteren sind die beiden 31 Bit-Auslese-Register pro Scalerkanal zu erwähnen, in denen die getriggerten Zählerstände (TCS- und Randomtrigger) gespeichert sind.

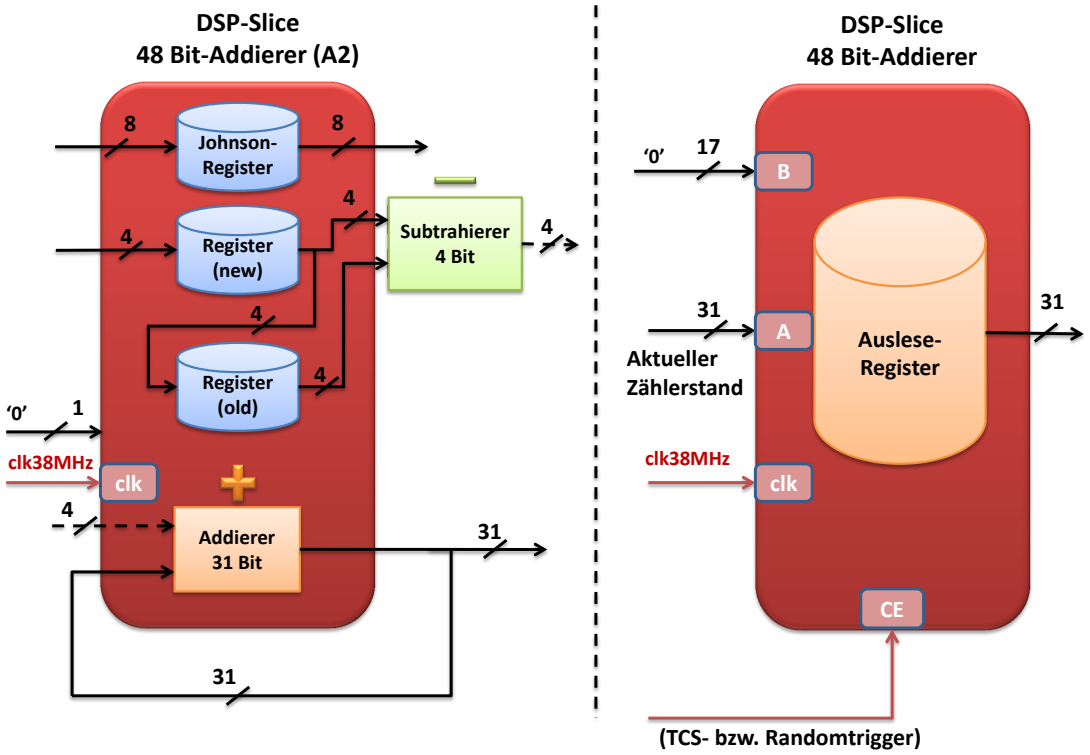

Abbildung 5.9: *Schematische Veranschaulichung der Ersetzungen einzelner Register durch DSP-Slices zur Einsparung von Flipflops.*

Diese können ebenfalls mit 48 Bit-DSP-Addierern umgesetzt werden, indem der aktuelle Zählerstand mit dem Eingang „A“ des Addierers verbunden wird (siehe Abbildung 5.9 rechts). Eingang „B“ wird der Wert 0 zugewiesen. Das Trigger-Signal wird mit dem CE-Eingang des Addierers verbunden, sodass die Zählerstände nur zu diesen Zeitpunkten gespeichert werden.

Summiert man die Anzahl der Flipflops n, die mit den Ersetzungen pro Kanal eingespart werden können, so erhält man

$$n = \underbrace{8+4+4+31}_{\text{enthält Addierer A2}} + \underbrace{31}_{\text{enthält Addierer A1}} + \underbrace{2\cdot 31}_{\text{2x 31 Bit-Auslese-Register}} = 140. \qquad (5.7)$$

Bei 96 Kanälen liegt die Gesamtzahl der eingesparten Flipflops bei 13440, was etwa 1/4 der gesamten Flipflops des Virtex-5 XC5VSX95T-FPGA (siehe Abschnitt 4.1) entspricht. Dafür werden $4 \cdot 96 = 384$ DSP-Slices benötigt.

5.3.7 Multiplexer und Zwischenspeicherung

Die Daten in den Auslese-Registern werden für beide Trigger seperat in Fifos gespeichert, wofür jeweils 32 Kanäle zusammengefasst werden. Wie in Abbildung 5.10 veranschaulicht, werden für 96 Kanäle also jeweils drei Multiplexer benötigt. Die Multiplexer, die die TCS-Daten verarbeiten, werden mit dem *CLK38MHz*-Signal getaktet und leiten die Daten in 128 Bit breite Fifos, welche jeweils eine Tiefe von

512 Worten besitzen. Aufgrund der Breite des Fifos kann ein Multiplexer pro Taktzyklus vier Datenworte verarbeiten, womit nach acht Taktzyklen alle Zählerstände abgespeichert sind. In einem neunten Wort wird die zwischen BOS-Signal und Trigger-Signal vergangene Zeit zusammen mit der Informationen ob zum Zeitpunkt des Triggers Events stattgefunden haben, zusammengefasst. Ein TCS-Event ist nach etwa $0,25\,\mu$s abgearbeitet, was unterhalb der mininalen Totzeit des TCS-Triggers von $0,4\,\mu$s liegt. Die Verwendung von 128 Bit breiten Fifos statt etwa 32 Bit-Fifos wird damit auch ersichtlich. Für die Speicherung in den Zwischenbuffern werden insgesamt zwölf 18 kBit Fifos benötigt. Diese können etwa 60 komplette Scaler-Events zwischenspeichern.

Die Zwischenspeicherung der Pseudo-Random-Daten erfolgt über insgesamt drei 1024×32 Bit-Fifos. Dafür werden 32 Scalerkanäle zusammengefasst, deren Zählerstände mit dem *CLK38MHz*-Signal getaktet, nacheinander ausgelesen und in die Fifos geschrieben werden. Der Zeitpunkt des Randomtriggers und die Information, ob gerade Events stattgefunden haben, wird in zwei weiteren 32 Bit Datenworten zusammengefasst. Der Multiplexer hat nach etwa $0,9\,\mu$s alle zu einem Randomtrigger gehörenden Datenworte zwischengespeichert. Nach der Erzeugung eines Pseudo-Randomtriggers (siehe Abschnitt 6.2) wird ein erneutes Auslösen eines Triggers in der Zeit bis $0,9\,\mu$s danach verhindert, um die Verarbeitung der Datenworte des Multiplexers nicht durcheinander zu bringen. In den Zwischenbuffern können etwa 30 ($\approx 1024/34$) komplette Events gespeichert werden.

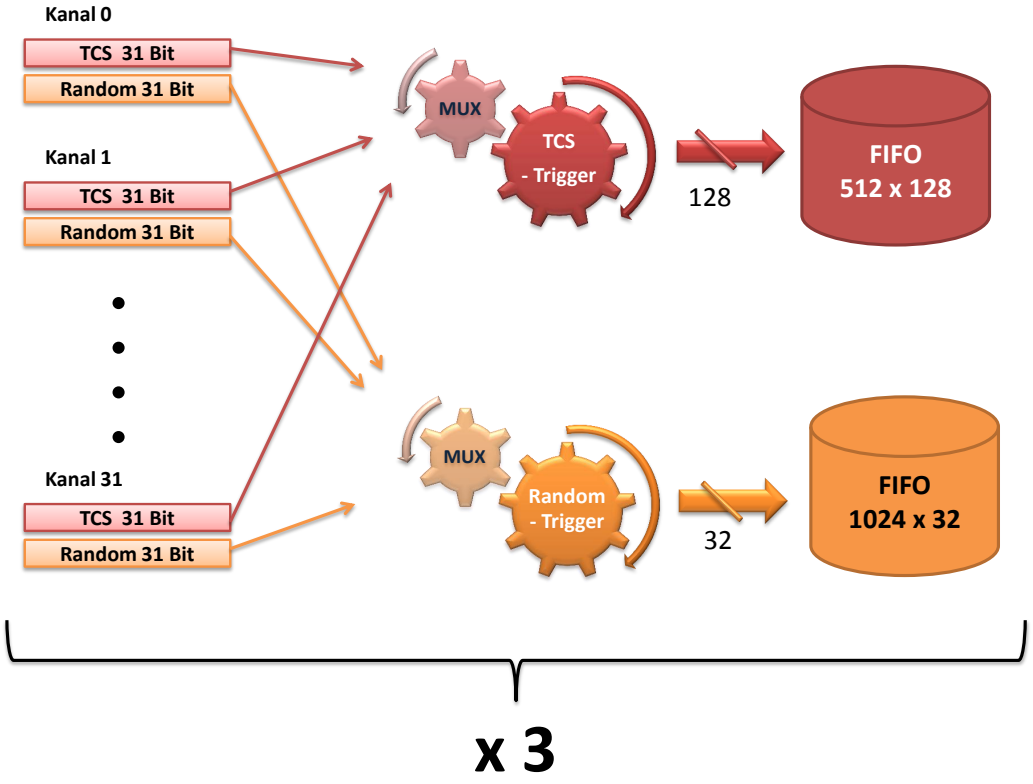

Abbildung 5.10: *Schematische Darstellung der Multiplexer und Speicherung der Daten in Fifos.*

5.3.8 Ressourcenverbrauch

In folgender Tabelle sind die benötigten FPGA-Ressourcen zusammengefasst. Vor allem aufgrund der Ersetzung vieler Register durch DSP-Slices werden nur sehr wenige Flipflops benötigt, was eine Kombination des Scalers mit anderen Komponenten erleichtert. Ebenso wurde Wert darauf gelegt, die BlockRAM-Einheiten in voller Breite auszunutzen. Der Anteil der benutzten BlockRAM-Einheiten entspricht dabei dem Anteil des benutzten Speichers.

Tabelle 5.3: *Ressourcenverbrauch des 96-Kanal-Scalers auf dem Xilinx Virtex-5 FPGA.*

	Verbrauch	**Verfügbar**	**Anteil** [%]
Slice-Register	3847	58880	6,53
Slice-LUTs	6368	58880	10,82
Besetzte Slices	3414	14720	23,19
BlockRAM-Einheiten	15	244	6,15
BlockRAM-Speicher	540 kBit	8784 kBit	6,15
DSP-Slices	386	640	60,31

6. Vereinigung von Scaler und TDC auf einem GANDALF-Modul

Ein großes Ziel bei der Entwicklung eines Echtzeit-Strahlprofil-Monitoring-Systems war es, eine FPGA-Firmware zu erstellen, die es erlaubt ein GANDALF-Modul gleichermaßen als Scaler und als TDC[1] zu verwenden. Dafür könnte man im einfachsten Fall für jede Komponente jeweils ein Modul benutzen, jedoch können durch die Vereinigung von Scaler- und TDC in einem Auslese-Modul Ressourcen und damit Geld eingespart werden. Eine Vereinigung macht vor allem auch LVDS-Signalsplitter [74] überflüssig. Bisher wurden die Detektorsignale der szintillierenden Fasern über LVDS-Splitter auf Scaler- und TDC-Einheiten verteilt, die auf den CATCH-Modulen angebracht sind.

Der TDC des Echtzeit-Strahlprofil-Monitoring-Systems basiert auf der Firmware des GANDALF-128-Kanal TDC, der im Rahmen der Diplomarbeit von M. Büchele [49] erfolgreich entwickelt und getestet wurde. Für dieses Projekt wurde der dazugehörige VHDL-Code zur Verfügung gestellt.

Die Vereinigung beider Komponenten auf einem GANDALF-Modul stellte vor allem aufgrund der begrenzten FPGA-Ressourcen eine große Herausforderung dar. Sowohl Scaler als auch TDC müssen in das COMPASS-Datennahme-System integriert werden. Unabhängig davon werden die Daten für das Monitoring-System über die VME-Backplane ausgelesen.

[1] TDC = Time-to-Digital Converter

6.1 Die Funktionsweise des TDC

Mit einem TDC misst man den Zeitpunkt einer Zustandsänderung auf einem digitalen Signal, wobei die Zeitmarke anschließend in digitaler Form ausgegeben wird. Somit lassen sich beispielsweise sehr präzise Zeitintervalle zwischen zwei Detektorsignalen bestimmen. Die spezifische Quantisierungsbreite LSB[2] eines TDC ist dabei eine wichtige Kenngröße. Für einen zählerbasierten TDC ist diese umgekehrt proportional zur Rate, mit der das Eingangssignal abgetastet wird.

$$\mathrm{LSB} = T_{clk} = \frac{1}{f_{clk}} \tag{6.1}$$

Die maximale Frequenz des Xilinx Virtex-5 FPGAs liegt bei etwa 500 MHz [75], was einer Quantisierungsbreite von 2 ns entspräche. Diese wird durch das im Folgenden beschriebene Konzept verbessert.

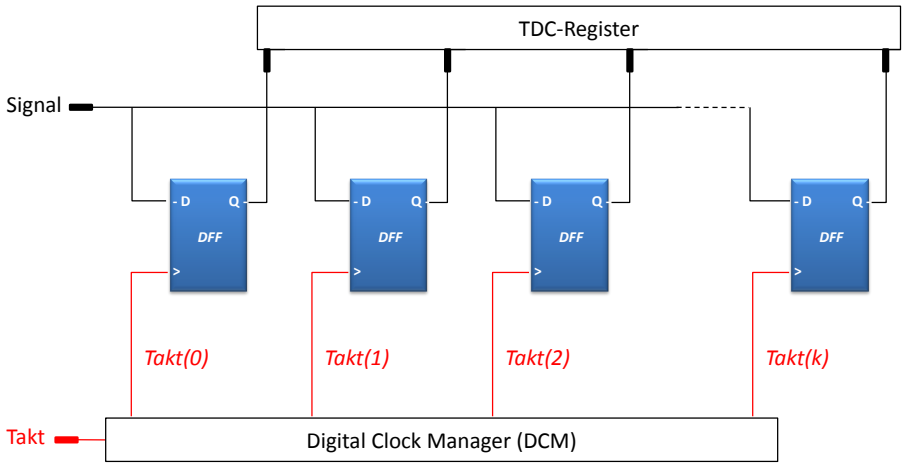

Abbildung 6.1: *Shifted Clock Sampling: Ein DCM generiert die phasenverschobenen Taktsignale Takt(i) [49].*

Beim sogenannten „Shifted Clock Sampling“ werden k phasenverschobene Taktsignale verwendet, die mit den Taktsignaleingängen von k Flipflops (TDC-Register) verbunden werden. Dabei ist eine äquidistante Phasenverschiebung $\Delta\alpha$ notwendig. Die Summe aller Verschiebungen muss 2π, also einer kompletten Taktperiode entsprechen,

$$k\Delta\alpha \stackrel{!}{=} 2\pi. \tag{6.2}$$

Das Detektorsignal wird jeweils auf die Eingänge der k Flipflops geleitet (siehe Abbildung 6.1). Aufgrund der phasenverschobenen Taktsignale wird die Taktperiode in äquidistante Schritte unterteilt, sodass man durch das Auslesen des TDC-Registers eine um den Faktor $1/k$ verkleinerte Quantisierungsbreite

[2] LSB = Least Significant Bit

$$\mathrm{LSB}_{SCS} = \frac{T_{clk}}{k} = \frac{1}{k \cdot f_{clk}}, \tag{6.3}$$

erhält.

Um den charakteristischen Fehler auf die Zeitmessung klein zu halten, ist es wichtig, dass das Eingangssignal möglichst gleichzeitig an allen k Flipflops ankommt, beziehungsweise die Laufzeitunterschiede (Routing Skew) gering sind. Außerdem müssen Unterschiede in der Phasenverschiebung der Taktsignale minimiert werden. Der TDC in diesem Projekt basiert auf dem Shifted Clock Sampling Konzept unter Verwendung von 16 phasenverschobenen Taktsignalen.

Der Virtex-5 FPGA ist in 16 sogenannte „Clockregionen“ aufgeteilt, worin jeweils maximal zehn verschiedene Taktsignale verwendet werden können. Aus diesem Grund wird die Hälfte der 16 TDC-Flipflops von acht phasenverschobenen Taktsignalen getaktet. Die restlichen Flipflops reagieren auf die fallende Flanke der Taktsignale, sodass eine äquidistante Phasenverschiebung von $\pi/8$ mit nur acht statt 16 Taktsignalen erreicht werden kann.

Unter Benutzung einer Taktfrequenz von $388,80\,\mathrm{MHz}$ (*CLK388MHz*) erhält man eine Digitalisierungsbreite von circa 161 ps. Das *CLK388MHz*-Signal wird mit dem Si5326-Clock Multiplier Chip auf dem GANDALF-Modul (siehe Kapitel 4) aus dem TCS Taktsignal (*CLK155MHz*) erzeugt. Die gewünschten Phasenverschiebungen erreicht man mit sogenannten PLLs[3], welche zur Frequenzsynthese von Taktsignalen benutzt werden. Eine detaillierte Beschreibung des Auslesevorgangs befindet sich in [49] beziehungsweise [76, 77].

Die Zeitmarken der auf dem Eingangssignal gemessenen Hits werden in sogenannten „Hitbuffern“ gespeichert, welche mit jedem Triggersignal ausgelesen werden. Dabei wird ein über den Konfigurationsspeicher einstellbares Zeitfenster und eine Triggerverzögerung berücksichtigt. Damit können die zum Zeitpunkt des interessanten Events stattgefundenen Hits selektiert und in Fifos zwischengespeichert werden (siehe Abbildung 6.2).

Zur Speicherung der selektierten Zeitmarken wird ein Output-Fifo von je acht TDC-Kanälen geteilt. Für die Vereinigung des entwickelten Scalers mit dem 96-Kanal-TDC müsste die Zustandsmaschine für die Auslese der zwölf Output-Fifos modifiziert werden, sodass eine simultane Datennahme für Scaler- und TDC-Daten auf einem einzigen GANDALF-Modul ermöglicht wird.

Dabei soll zwischen TCS- und Randomtriggern unterschieden werden. Während die Auslese der TCS-getriggerten Daten zum zentralen Datennahmesystem über einen Readout-TIGER beziehungsweise das SLink-Protokoll [56] vonstatten geht, werden die Daten der Randomtrigger über die VME-Backplane ausgelesen, analysiert und in Echtzeit visualisiert (siehe Abschnitt 7.2.2). Eine detaillierte Beschreibung des Auslesevorgangs befindet sich in Abschnitt 6.3.

[3] PLL = Phase Locked Loop

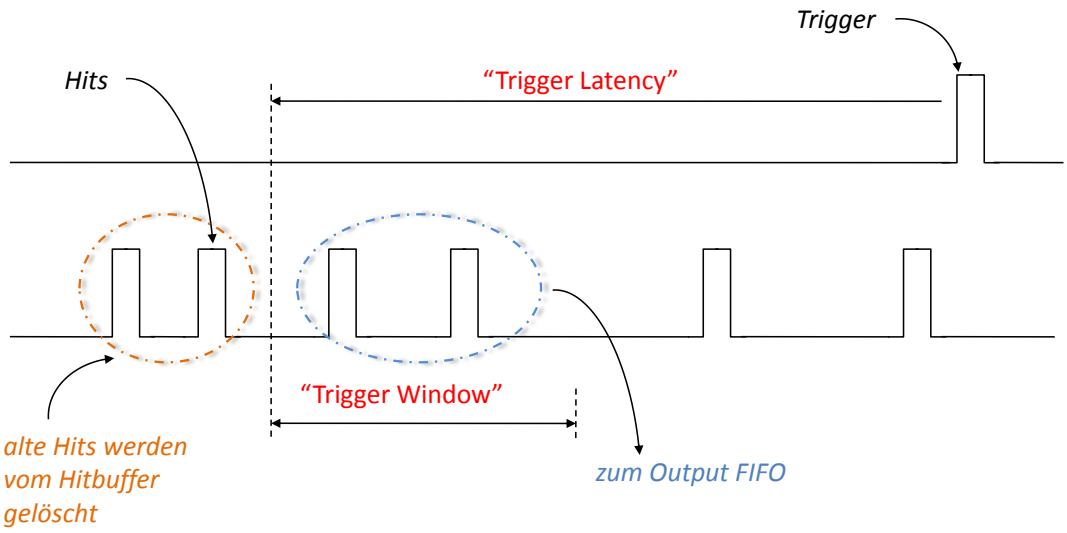

Abbildung 6.2: *Hits die innerhalb des Triggerfensters liegen werden in Output-Fifos zwischengespeichert, Hits älter als die Triggerverzögerung können vom Hitbuffer gelöscht werden [49].*

6.2 Der Pseudo-Randomtrigger

Im Rahmen dieser Diplomarbeit wurde ein auf der Hardwarebeschreibungssprache VHDL basierendes Pseudo-Randomsignal entwickelt. Dieses wird für das Beam-Monitoring als Trigger benötigt und wird unabhängig vom TCS erzeugt.

6.2.1 Linear rückgekoppelte Schieberegister

Der Randomtrigger basiert auf mehreren Schieberegistern, die über *xor*-Verknüpfungen rückgekoppelt werden. Diese werden im Folgenden mit LFSR[4] abgekürzt. In Abbildung 6.3 wird die Funktionsweise anhand eines 4 Bit LFSR veranschaulicht. Mit jeder steigenden Taktflanke verschiebt sich der gesamte Inhalt des LFSR um eine Stelle nach links, wobei das *xor*-verknüpfte Signal ins LSB rückgekoppelt wird. Der sich damit ergebende neue Zustand ist nicht als Zufallszahl zu betrachten, lediglich das rückgekoppelte Bit ist als Zufallsbit anzusehen [78]. Mit diesem Prinzip entsteht ein zyklischer Prozess, der sich nach

$$2^n - 1 \overset{n=4}{=} 15 \tag{6.4}$$

Takten wiederholt, was auch den Begriff „Pseudo-Random“ erklärt.

Mathematisch lässt sich der Wert des Rückkopplungsbits eines LFSR der Breite n durch ein Rückkopplungspolynom f über dem Körper $\mathbb{F}_2$ beschreiben[5]:

$$f: \ \mathbb{F}_2^n \to \mathbb{F}_2, \quad f(X) = \sum_{i=1}^{n} \nu_i X^i + 1, \tag{6.5}$$

[4] LFSR = Linear Feedback Shift Register

[5] $\mathbb{F}_2$ ist ein kommutativer Ring mit Eins ($\mathbb{Z}/r\mathbb{Z}$ für $r \geq 2$). Handelt es sich bei r um eine Primzahl, so ist $\mathbb{Z}/r\mathbb{Z}$ ein endlicher Körper mit r Elementen und wird mit $\mathbb{F}_r$ bezeichnet. Für den Körper $\mathbb{F}_2 = \{0, 1\}$ gelten folgende Eigenschaften:
$1 + 1 = 0 + 0 = 0 \cdot 0 = 0 \cdot 1 = 1 \cdot 0 = 0, \qquad 0 + 1 = 1 + 0 = 1 \cdot 1 = 1.$
Die Elemente „0“ und „1“ repräsentieren die Restklassen $2\mathbb{Z}$ beziehungsweise $1+2\mathbb{Z}$.

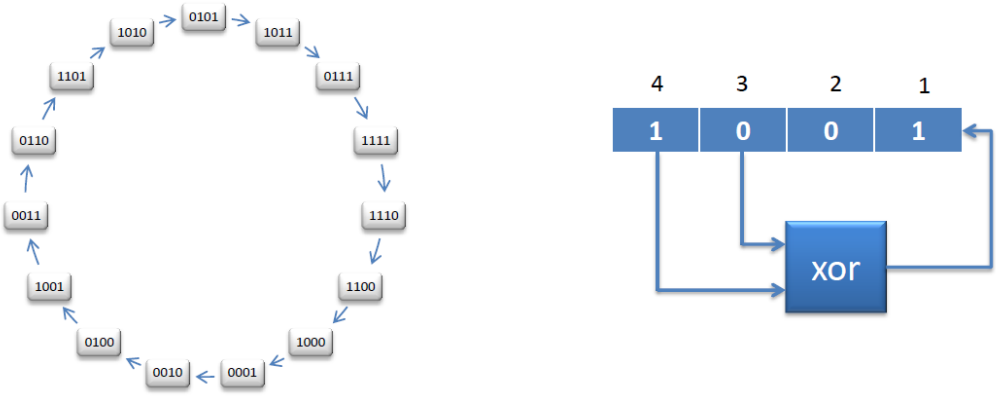

Abbildung 6.3: *Darstellung einer 4 Bit LFSR-Sequenz maximaler Periodizität [78]. Das Rückkopplungspolynom $f(X) = X^4 + X^3 + 1$ definiert die Verbindungsstellen der xor-Verknüpfungen.*

mit den Koeffizienten $\nu_1, ..., \nu_n \in \mathbb{F}_2$. Aufgrund der Linearität der Funktion f lässt sich die Zustandsüberführung als Matrixmultiplikation schreiben:

$$\vec{a}_{t+1} = A \cdot \vec{a}_t. \tag{6.6}$$

Hierbei ist $\vec{a}_t \in \mathbb{F}_2^n$ der Zustand des LFSR zum Zeitpunkt t und

$$A = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & 1 \\ \nu_n & \nu_{n-1} & \dots & \nu_2 & \nu_1 \end{bmatrix} \in \mathbb{F}_2^{n \times n} \tag{6.7}$$

bezeichnet die Rückkopplungsmatrix [79]. Der Zustand $\vec{a}_t$ lässt sich anhand des Initialisierungszustands $\vec{a}_0 = (a_0^1, ..., a_0^n)$ per Rekursion bestimmen.

Eine *xor*-Verknüpfung kann als mod 2-Addition interpretiert werden[6]. Somit kann das Rückkopplungspolynom über dem Körper $\mathbb{F}_2$ mit *xor*-Verknüpfungen realisiert werden. Diese werden an jenen Stellen i benutzt, für die $\nu_i = 1$ gilt. Wird der Nullzustand angenommen, so ergibt sich ein LFSR mit Periodizität 1, was unbrauchbar ist. Existiert ein Zustand für den es zwei oder mehrere verschiedene Vorgänger gibt, so besitzt die Folge eine Vorperiode. Dies ist äquivalent dazu, dass die Determinante der Zustandsüberführungsmatrix $\det(A)$ verschwindet, da die Abbildung nicht injektiv und somit nicht invertierbar ist. Aus Gleichung 6.7 erhält man

$$\det(A) = \nu_n, \tag{6.8}$$

womit eine periodische Folge genau dann entsteht, sofern $\nu_n = 1$ gilt, der Grad des Rückkopplungspolynoms also der Breite des LFSR entspricht.

[6] $1 \veebar 0 = 0 \veebar 1 = 1$,
$0 \veebar 0 = 1 \veebar 1 = 0$.

Ein LFSR maximaler Periodizität $2^n - 1$ ergibt sich durch geeignete Wahl des Rückkopplungspolynoms vom Grad n. Dies ist gegeben für ein irreduzibles Polynom[7] $f(X) \in \mathbb{F}_2[X]$, falls X ein Erzeuger der multiplikativen Gruppe $(\mathbb{Z}/f(X)\mathbb{Z})^*$ ist [80]. Eine Potenzierung von X mod $(\mathbb{Z}/f(x)\mathbb{Z})$ liefert alle $2^n - 1$ Restklassen, die von null verschieden sind. Ein Polynom mit $2^n - 1$ Zuständen heißt *primitiv*. Dabei gilt, dass jeder vom Nullzustand verschiedene Initialisierungszustand $\vec{a}_0$ einen Ausgabestrom maximaler Periode erzeugt. In Abbildung 6.4 sind Kombinationen für Sequenzen maximaler Periodizität in Abhängigkeit der Anzahl der Bitstellen dargestellt.

Das LFSR soll so konzipiert sein, dass sich die in ihm enthaltenen Zustände über den Zeitraum eines Spillzyklus nicht wiederholen. Werden die LFSR mit dem 38, 88 MHz-Taktsignal getaktet, so ergibt sich bei einem 31 Bit-LFSR mit maximaler Periodizität eine Umlaufzeit von

$$T_{LFSR} = (2^n - 1) \cdot \frac{1}{\nu_{CLK}} \overset{n=31}{\approx} 55,23\,\mathrm{s}. \tag{6.9}$$

Verglichen mit der Dauer eines Spillzyklus (siehe Kapitel 3) sind also 31 Bit-LFSR ausreichend.

a)

N	XOR
3	3,2
4	4,3
5	5,3
6	6,5
7	7,6
8	8,6,5,4
9	9,5
10	10,7
11	11,9
12	12,6,4,1

N	XOR
13	13,4,3,1
14	14,5,3,1
15	15,14
16	16,15,13,4
17	17,14
18	18,11
19	19,6,2,1
20	20,17
21	21,19
22	22,21

N	XOR
23	23,18
24	24,23,22,17
25	25,22
26	26,6,2,1
27	27,5,2,1
28	28,25
29	29,27
30	30,6,4,1
31	31,28
32	32,22,2,1

b)

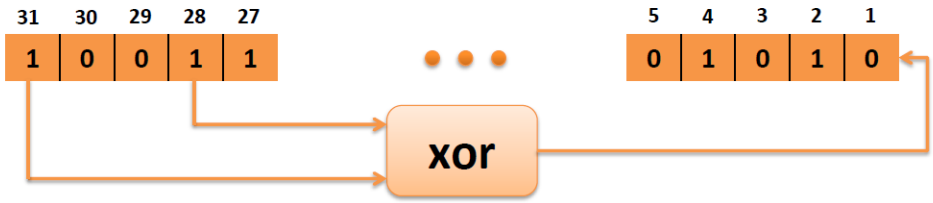

Abbildung 6.4: *(a) Tabelle xor-verknüpfter LFSR der Breite N mit maximaler Periodizität [81]. (b) Beispielhafte Darstellung eines 31 Bit-LFSR. Für die Realisierung des Pseudo-Randomtriggers werden ebenfalls 31 Bit breite LFSR benutzt. Das zugehörige Rückkopplungspolynom lautet:* $f(X) = X^{31} + X^{28} + 1$.

Zur Generierung des Zufallstriggers werden insgesamt 18 Schieberegister mit unterschiedlichen Anfangszustände benutzt. Dabei muss darauf geachtet werden, dass der Unterschied der Zustände nicht zu klein gewählt wird. Ist die Zeit, beziehungsweise die Anzahl der Schritte, nach der ein LFSR den Anfangszustand eines anderen LFSR annimmt zu klein, so besteht die Gefahr einer Korrelation, was bei der Generierung eines Zufallsignals unerwünscht wäre. Mit Hilfe der Simulation eines vollen

[7] Irreduzibilität bedeutet, dass keine Polynome $h(X), g(X) \in \mathbb{F}_2[X]$ mit $f(X) = h(X) \cdot g(X)$ existieren.

Umlaufs eines 31 Bit-LFSR maximaler Periodizität konnten die Anfangszustände der 18 Schieberegister geeignet gewählt werden.

Betrachtet man die Zufallsbits der LFSR, so kann mit einer beliebigen Kombination ein Zufallstrigger ausgelöst werden. In diesem Projekt wird die Triggerentscheidung anhand folgender Bedingung gefällt:

$$\mathbf{LSB}_1 = \mathbf{LSB}_2 = \cdots \mathbf{LSB}_{18} = 1, \tag{6.10}$$

wobei jede andere Bedingung $\{\mathbf{LSB}_1, \mathbf{LSB}_2, \cdots, \mathbf{LSB}_{18}\}$ denselben Zweck erfüllen würde. Die Triggerbedingung lässt sich am besten mit einer geforderten Kombination aus insgesamt 18 Münzwürfen vergleichen. Enthalten alle LSB den Zustand '1' so wird ein Trigger synchron zum *CLK38MHz*-Signal erzeugt. Dies geschieht im Durchschnitt einmal pro

$$t \;=\; 2^{18} \;\cdot\; \frac{1}{\nu_{CLK}} \tag{6.11}$$

mit einer Durchschnittsfrequenz von

$$\nu_{min} = \frac{1}{t} \;\approx\; 148\,\mathrm{Hz}. \tag{6.12}$$

Die Triggerfrequenz kann vergrößert werden, indem man die Anzahl der in die Triggerbedingung einbezogenen Zufallsbits verkleinert. Diese Einstellung lässt sich jederzeit über den Konfigurationsspeicher vornehmen. Damit lassen sich Triggerfrequenzen

$$\nu = \nu_{min} \cdot 2^n, \qquad \mathrm{n} = 0, 1, ..., 6 \tag{6.13}$$

einstellen, wobei n die Anzahl der nicht für die Triggerbedingung verwendeten LFSR bezeichnet. Die maximale Triggerfrequenz wird durch die Auslesegeschwindigkeit über die VME-Backplane vorgegeben. Mit einer maximalen Bandbreite von $\approx 10\,\mathrm{MB/s}$ liegt die größtmögliche Triggerrate bei ungefähr 10 kHz. Dies stellt eine untere Schranke für die Anzahl der LFSR dar, die für die Triggerbedingung herangezogen werden müssen, sodass mindestens zwölf LFSR ($\hat{=} 9,49\,\mathrm{kHz}$) benutzt werden. Findet ein Trigger statt, so wird ein Zustand aktiv, der über den Zeitraum von $1\,\mu$s keinen neuen zulässt. Diese Totzeit ist für die in Abschnitt 5.3.7 beschriebenen Scaler-Multiplexer vonnöten, da diese ungefähr $0,9\,\mu s$ benötigen, um die Werte der Auslese-Register in Fifos zu schreiben.

6.2.2 Erzeugung einer künstlichen Spillstruktur

Über das Trigger-Control-System wird neben den Triggersignalen auch das Begin- und End-Of-Spill-Signal an alle Auslese-Module verteilt. Ist das Datennahmesystem gerade inaktiv, so werden diese Signale nicht übertragen. Da eine Darstellung des Strahlprofils während dieser Zeit trotzdem wünschenswert ist, werden diese Startsignale für die Auslese der Pseudo-Random-Daten intern generiert. Damit ist man auch weitestgehend unabhängig vom TCS.

Für die Erzeugung einer künstlichen Spillstruktur wird die momentane Rate eines Scalerkanals, der mit die höchste Rate der szintillierenden Fasern aufweist, bestimmt. Dazu wird das von diesem Kanal empfangene Detektorsignal geteilt und auf

eine zusätzliche Scalereinheit geleitet. Der Zählvorgang ist abgesehen von der nicht benötigten Verzögerungseinheit mit der in Kapitel 5 beschriebenen Logik identisch.

Der Zählerstand dieser Scalereinheit wird alle 100 ms ausgelesen und resettiert. Wird ein über den Konfigurationsspeicher einstellbarer Wert im Bereich von 1 kHz bis 1 MHz überschritten, so wird das künstliche Begin-Of-Spill-Signal erzeugt und das Onspill-Signal für zwei Sekunden auf den Wert '1' gesetzt. Sobald der eingestellte Wert wieder unterschritten wird, wird das künstliche End-Of-Spill-Signal generiert und das Onspill-Signal für mindestens zwei Sekunden auf den Wert '0' gesetzt. Dadurch wird vermieden, dass sowohl ein Unterschreiten der Schranke kurz nach dem Begin-Of-Spill-Signal durch mögliche Fluktuationen ein End-Of-Spill-Signal hervorruft, beziehungsweise dass durch ein sofortiges Überschreiten der Schranke nach einem End-Of-Spill-Signal ein erneutes Begin-Of-Spill-Signal erzeugt wird. Auf diese Weise erhält man eine vom TCS unabhängige Spillstruktur.

Die Generierung eines Randomtriggers ist nur während der künstlichen Onspill-Zeit möglich. Das Reset-Signal für die Fifos, in denen die Pseudo-Random-Daten zwischengespeichert werden, orientiert sich ebenfalls an den künstlich erzeugten Spill-Signalen.

6.2.3 Test auf zeitliche Zufälligkeit des Signals

Mit einem C++ Programm wurde die zeitliche Abfolge des Randomtriggers unter Verwendung von zwölf, acht und sechs LFSR für einen vollen Umlauf simuliert. In erster Ordnung werden die Zeitintervalle zwischen zwei Triggern in einem Histogram aufgetragen. In zweiter Ordnung werden die zwischen drei aufeinander folgenden Triggern, beziehungsweise in $n$ter Ordnung die Zeit nach $n+1$ Triggern in Histogrammen aufgetragen. Danach erfolgte eine Anpassung an Wahrscheinlichkeitsdichte-Funktionen [82]

$$w_n(t-t_0) = r\,\frac{(r(t-t_0))^{n-1}e^{-r(t-t_0)}}{(n-1)!}. \tag{6.14}$$

Gleichung 6.14 lässt sich folgendermaßen verstehen. Wenn zum Zeitpunkt t_0 ein Trigger erzeugt wird, so ist für einen Randomtrigger die Wahrscheinlichkeit, n weitere Trigger im Zeitintervall $[t_0, t]$ zu finden, durch die Funktion $w_n(t-t_0)$ gegeben. $w_n(t-t_0)$ kann als Poissonverteilung mit Erwartungswert rt, gewichtet mit der Triggerrate r, aufgefasst werden. Für $n = 1$ erhält man beispielsweise eine Exponentialfunktion. In Abbildung 6.5, 6.6 und 6.7 sind die Ergebnisse der Simulation bis zur sechsten Ordnung dargestellt. Dabei wurden jeweils Funktionen $c \cdot w_n(t-t_0)|_{t_0=0}$ an die Daten gefittet, wobei c ein weiterer Fitparameter ist, durch den die Normierung berücksichtigt wird. Die Zeit t ist in Einheiten der Periodendauer (T) des verwendeten Taktsignals angegeben.

Deutlich zu erkennen ist, dass die simulierten Daten mit den theoretischen Anpassungskurven sehr gut übereinstimmen. Bei der Simulation in Abbildung 6.5 wurden zwölf LFSR verwendet, was einer Rate von $1/4096 = 2^{-12} \approx 2,44 \cdot 10^{-4}$ pro Taktzyklus entspricht (siehe Gleichung 6.13). Für Abbildung 6.6 wurden nur acht LFSR verwendet, sodass die Rate etwa $1/256 = 2^{-8} \approx 3,91 \cdot 10^{-3}$ bezogen auf einen

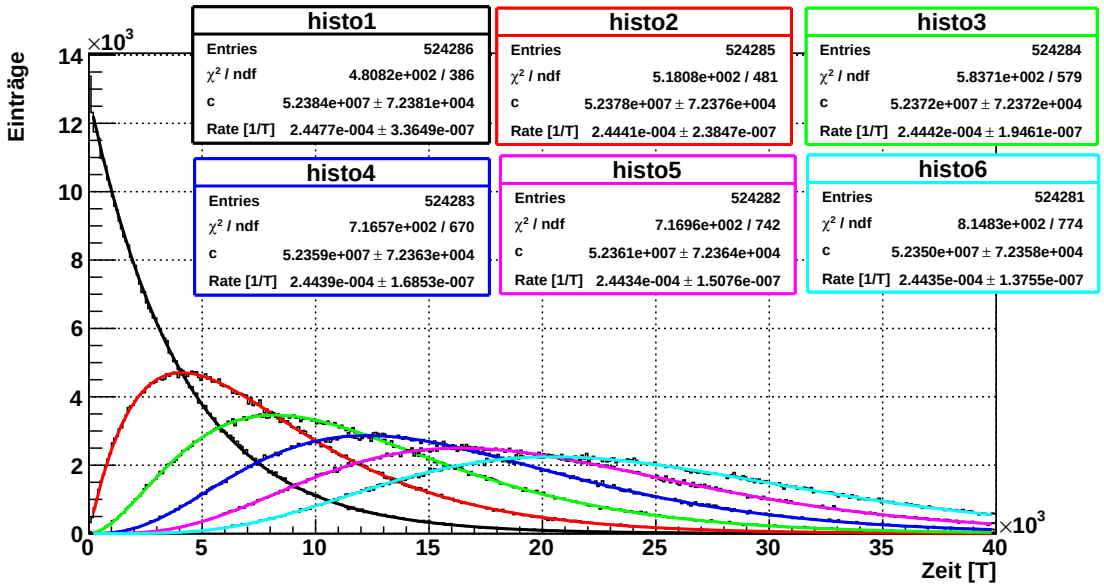

Abbildung 6.5: *Häufigkeit für $n = 1, 2, 3, 4, 5, 6$ weitere Trigger im Zeitintervall $[t_0, t]$ wenn zum Zeitpunkt t_0 ein Trigger stattgefunden hat, unter Verwendung von zwölf LFSR.*

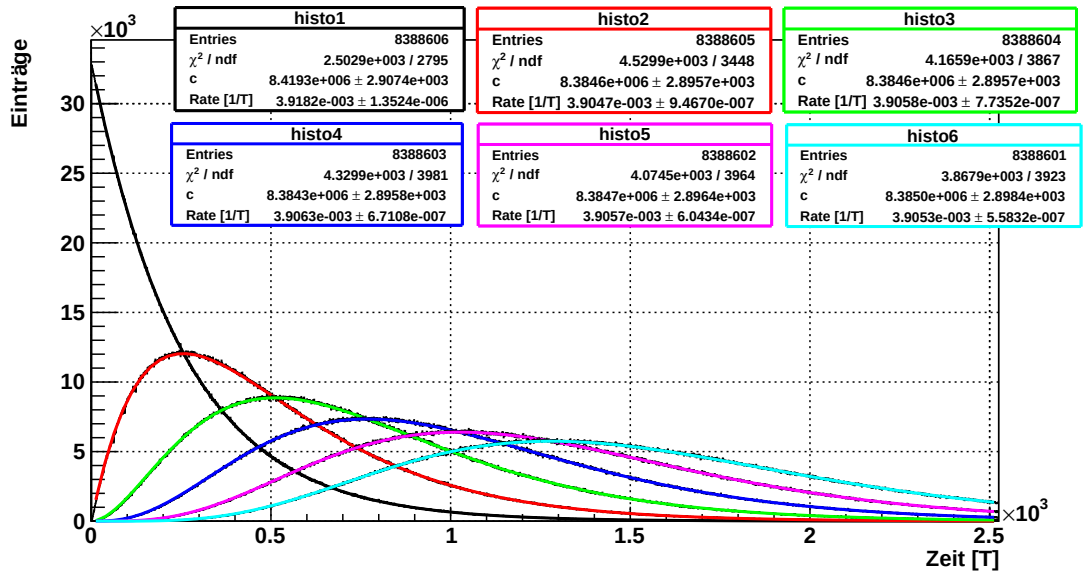

Abbildung 6.6: *Häufigkeit für $n = 1, 2, 3, 4, 5, 6$ weitere Trigger im Zeitintervall $[t_0, t]$ wenn zum Zeitpunkt t_0 ein Trigger stattgefunden hat. Es wurden acht LFSR benutzt.*

Taktzyklus T beträgt. Die Anpassung an die Poissonkurven wird ab einer Anzahl von sechs LFSR zunehmend schlechter (vergleiche Abbildung 6.7). Auch stimmen die Fitparameter „Rate“ nicht mehr im Rahmen ihrer Fehlergrenzen überein.

In einer Testmessung konnte die Funktion des Zufallstriggers auch mit echten Signalen verifiziert werden. Hierzu wurde ein GANDALF-Modul als Patterngenerator im Output-Modus betrieben (siehe Abschnitt 7.1), welcher insgesamt zehn

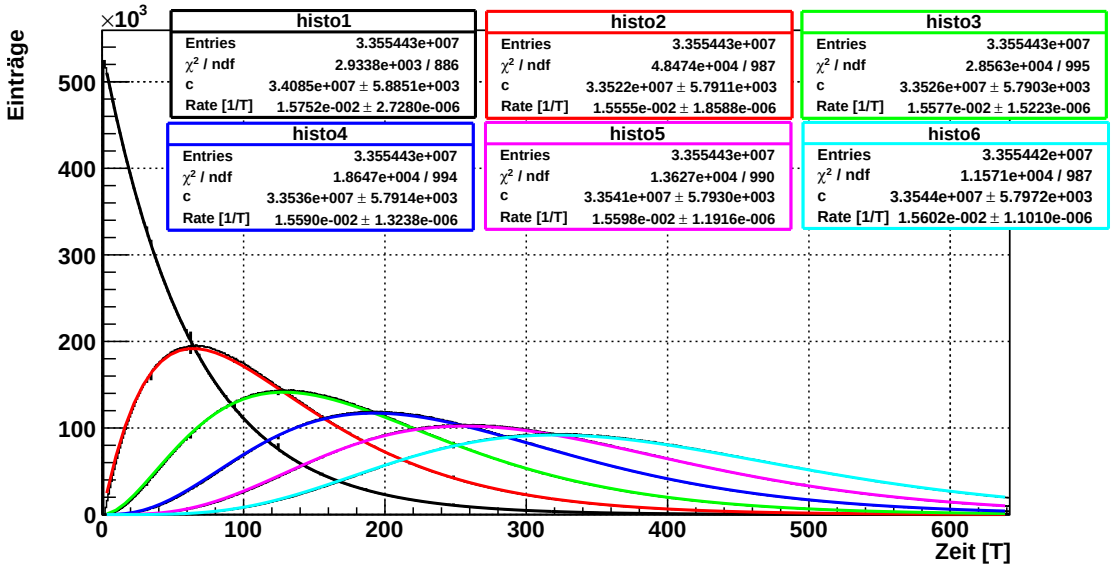

Abbildung 6.7: *Häufigkeit für $n = 1, 2, 3, 4, 5, 6$ weitere Trigger im Zeitintervall $[t_0, t]$ wenn zum Zeitpunkt t_0 ein Trigger stattgefunden hat (sechs LFSR).*

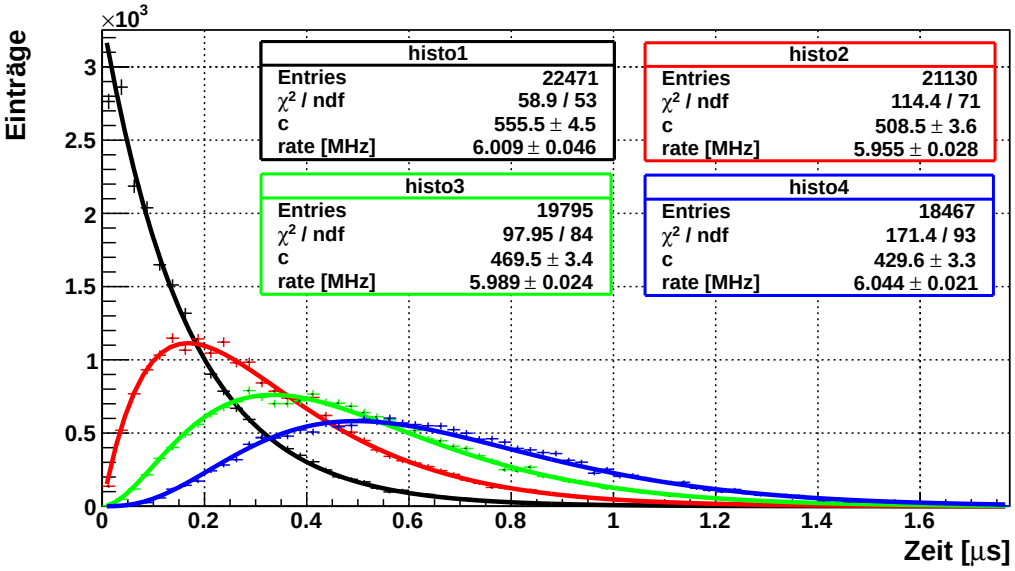

Abbildung 6.8: *Ergebnisse einer Testmessung bis zur vierten Ordnung.*

Eingangssignale für einen Input-GANDALF mit TDC-Funktion lieferte. Jedes dieser zehn Signale wurde mit neun LFSR unterschiedlicher Anfangszustände generiert, welche mit 311 MHz getaktet waren. Die zugehörigen Histogramme des Tests bis zur vierten Ordnung sind in Abbildung 6.8 dargestellt.

Die Ergebnisse der Simulation und der Testmessung zeigen, dass der Zufallstrigger mindestens bis zur sechsten Ordnung zeitlich zufällige Signale liefert, wodurch das Pseudo-Randomtrigger-Signal zur Strahlanalyse geeignet ist.

Eine schematische Zusammenfassung der Erzeugung des Randomtriggers befindet sich in Abbildung 6.9.

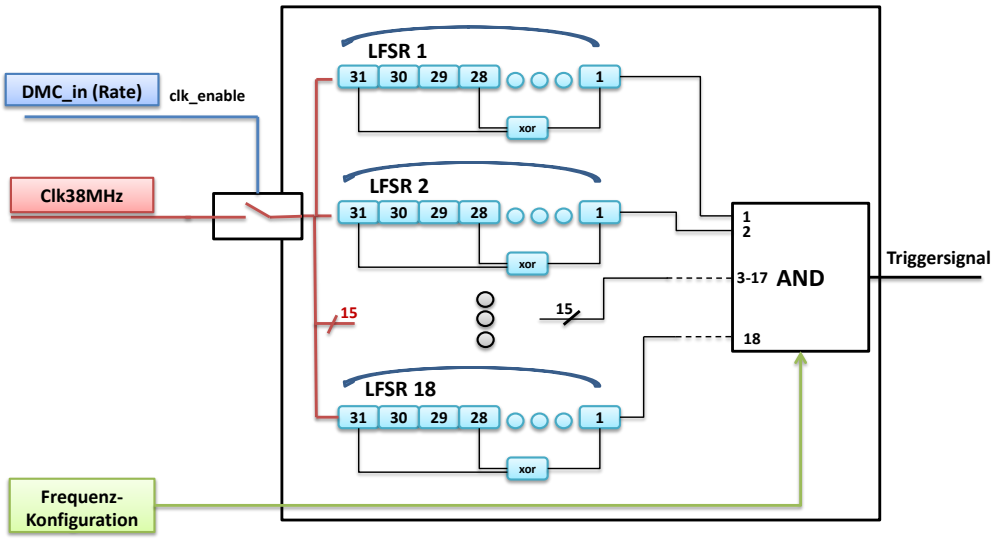

Abbildung 6.9: *Schematische Darstellung des Randomtriggers - Ein Trigger wird generiert, wenn eine konfigurierbare Anzahl von 31 Bit-LFSR im LSB den Zustand '1' aufweisen. Durch eine Messung des Eingangssignals eines ausgewählten Kanals wird eine künstliche Spillstruktur erzeugt, die die Generierung der Trigger startet, sobald Strahlteilchen am Experiment ankommen.*

6.3 Die Datenauslese

Eine große Herausforderung stellte die Datenauslese dar, zumal die Firmware aus zwei Komponenten besteht, die jeweils auf zwei verschiedene Arten ausgelesen werden müssen.

Für die Datenauslese wird eine korrekte Zuordnung der Daten zu den verschiedenen Triggertypen benötigt. Außerdem muss auf eine möglichst schnelle Abarbeitung der Daten geachtet werden, da die Datenrate unter Benutzung beider Komponenten je nach Triggerrate enorm hoch sein kann. Die 96 Eingangssignale werden im FPGA geteilt und mit den jeweiligen Scaler- und TDC-Einheiten verbunden (siehe Abbildung 6.10).

Die Triggerzuordnung wird mit Hilfe eines Fifos bewerkstelligt, der die Reihenfolge der eingehenden Trigger speichert (*Trigger-Fifo*). Dazu werden TCS-Trigger und Pseudo-Randomtrigger zum einen über eine logische „oder"-Verknüpfung mit dem *writeenable* Port des Fifos und zum anderen mit unterschiedlichen Bitstellen des Dateneingangs verbunden. Kommen beide Trigger zur selben Zeit, so wird der Randomtrigger um einen Taktzyklus (25, 72 ns) verzögert. Damit wird entweder das Wort „01" (Pseudo-Random) oder „10" (TCS) im Fifo gespeichert.

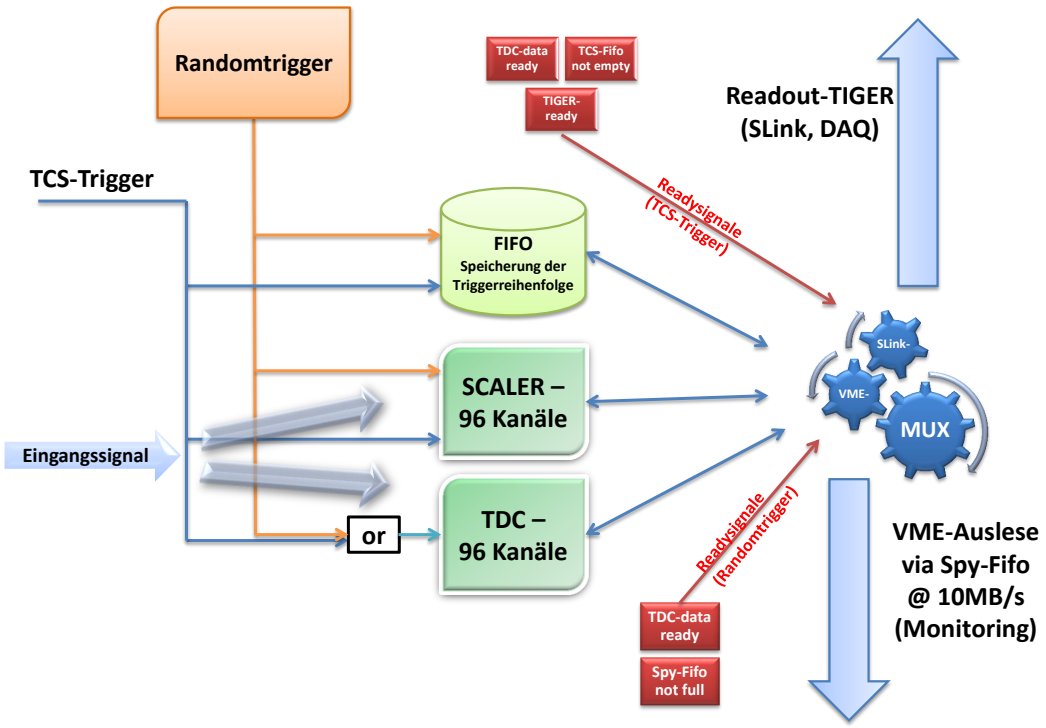

Abbildung 6.10: *Schematische Darstellung des FPGA Designs und dessen Auslese.*

Für die Auslese wird eine endlichen Zustandsmaschine *FSM1*[8] [12] verwendet. Diese enthält die übergeordneten Zustände *st_neutral*, *st_slink* und *st_vme* (vergleiche Abbildung 6.11).

Zu Beginn eines Auslesevorgangs wird auf ein Steuersignal gewartet, (*TDC-data ready (1)*) welches anzeigt, dass Daten in den TDC-Output-Fifos zur Auslese bereit liegen. Im nächsten Schritt erfolgt die Auslese des *Trigger-Fifos (2)*, womit einer der beiden Zustandsübergänge {*st_neutral* → *st_slink (3a)*, *st_neutral* → *st_vme (3b)*} einhergeht.

Befindet sich *FSM1* im übergeordneten Zustand *st_slink*, so wird auf ein TCS-Wort gewartet *(4a)*. Dieses enthält die Eventnummer, Spillnummer und den Eventtyp (siehe Abschnitt 4.5). Anschließend muss die Größe des gesamten Events berechnet werden, da die Anzahl der TDC-Hits pro Event variieren kann. Zur TDC-Eventsize wird ein konstanter Offset (113) addiert, welcher die Anzahl der Scalerworte (108) und die für die Datennahme benötigten SMux[9]- beziehungsweise SLink-Header (2+3) enthält. Die Eventgröße wird für den Eventbuilder benötigt und wird mit in die SLink-Header geschrieben. Anschließend werden nacheinander alle zwölf TDC-Output-Fifos ausgelesen und in einem 32 Bit breiten Fifo (*SLinkdata-Fifo*) zusammengefasst *(5a)*, welcher direkt mit dem TIGER-Ausgangsstrom verbunden ist. Danach geschieht das gleiche mit den drei Fifos die die TCS-Scalerdaten enthalten (vergleiche Abschnitt 5.3.7, *(6a)*). Die Abarbeitung eines kompletten Events

[8]FSM = Finite State Machine
[9]SMux = SLink Multiplexer

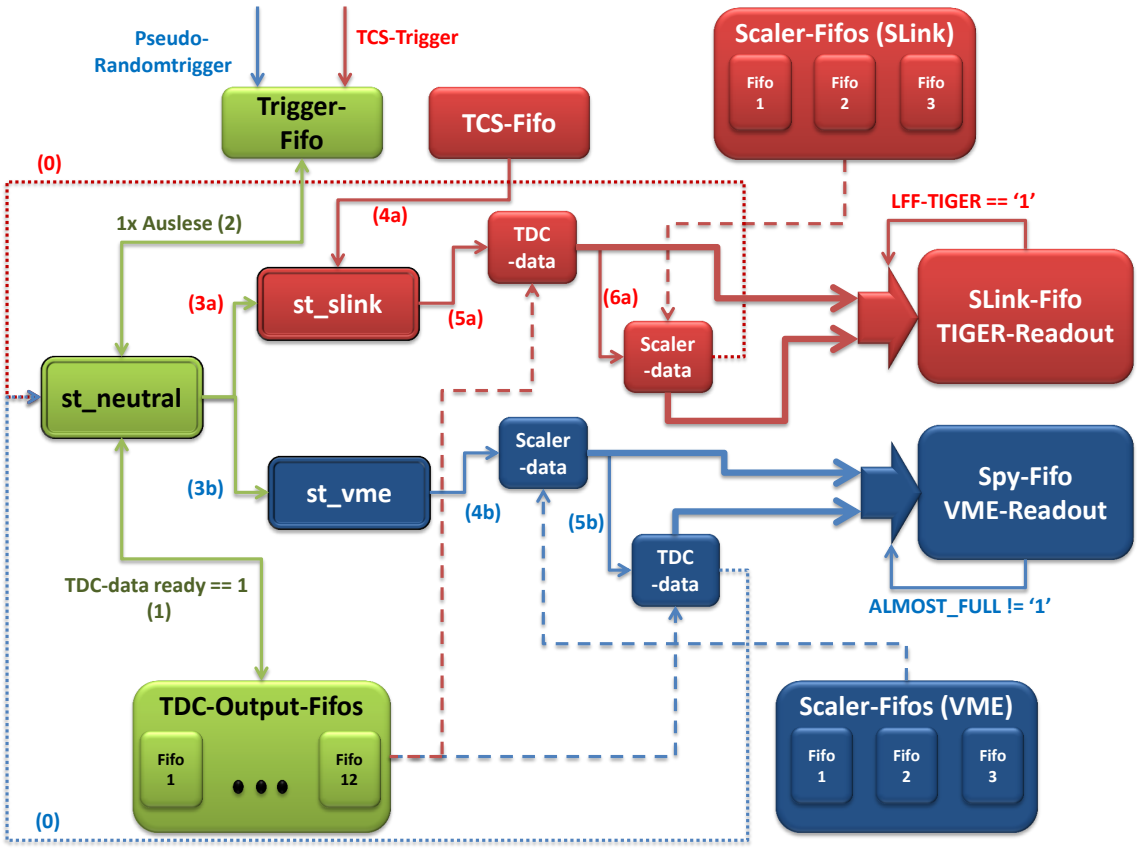

Abbildung 6.11: *Funktionsweise der endlichen Zustandsmaschine „FSM1“. Die Zustandsübergänge sind in Klammern () angegeben und die übergeordneten Zustände sind farblich markiert – st_neutral (grün), st_slink (rot), st_vme (blau).*

bis zum Zustandsübergang *(0)* dauert ungefähr $4,5\,\mu$s, wobei die Zeit je nach Größe des TDC-Events variieren kann. Das Datenformat der TCS-Daten wird in Kapitel B erklärt. Die Übertragung der Daten vom TIGER-Modul an den ROB ist vom Steuersignal *LFF*[10] abhängig, welches anzeigt, ob die SLink-Karte für die Aufnahme weiterer Datenwörter bereit ist. Ist dieses Signal im Zustand '0', so muss die Datenübertragung unterbrochen werden (siehe Abbildung 6.12). Auf ähnliche Weise funktioniert die Übertragung der Daten an das TIGER-Modul, welches seinerseits Steuersignale (LFF-TIGER in Abbildung 6.11) zur Auslese des *SLinkdata-Fifos* an das GANDALF-Modul verschickt.

Befindet sich *FSM1* im übergeordneten Zustand *st_vme*, so kann sofort mit der Abarbeitung der Pseudo-Random-Daten begonnen werden, da nicht auf ein TCS-Wort gewartet werden muss. Die Datenworte des Scalers und des TDC zu den Pseudo-Random-Daten werden anschließend nacheinander in den *Spy-Fifo* geschrieben (siehe Abbildung 6.11).

Die Auslese der *Spy-Fifo*-Daten geschieht über die VME-Backplane. Da die Datenübertragungsrate ($\approx$ 10 MByte/s) verglichen mit der Auslese über das TIGER-Modul deutlich langsamer ist, kann der *Spy-Fifo* bei einer zu großen Triggerrate volllaufen, was zum Datenverlust einzelner Datenwörter führen würde. Dies wieder-

[10] LFF = Link Full Flag

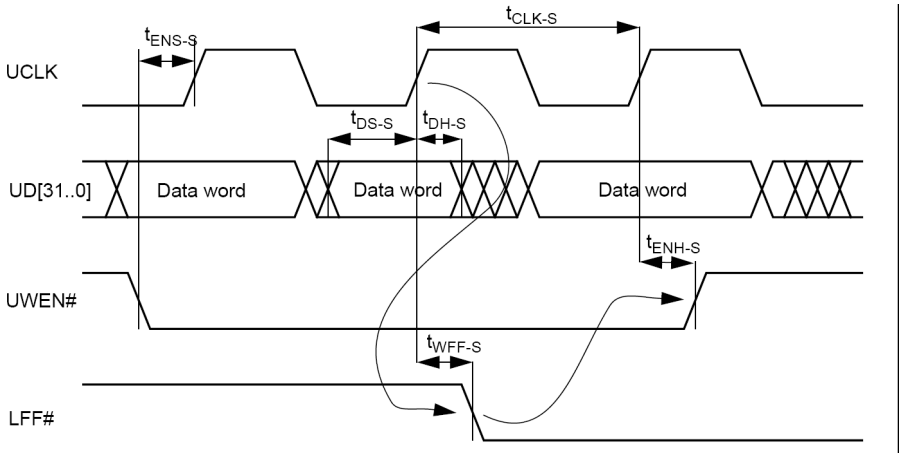

Abbildung 6.12: *Das SLink-Protokoll ermöglicht über einen 32 Bit breiten Datenbus UD und einer Taktfrequenz von* 40 *MHz eine Datenübertragungsrate von bis zu* 160 *MByte/s. Der Transfer findet nur für UWEN = '0' (aktive low) bei jeder steigenden Taktflanke UCLK statt. Wenn das Steuersignal LFF auf '0' steht, können noch zwei weitere Datenworte folgen, bevor die Übertragung unterbrochen werden muss (UWEN⇒ '1') [56].*

um bringt Schwierigkeiten für eine spätere Analyse der Daten mit sich. Das Problem kann umgangen werden, wenn am Anfang des Auslesevorgangs anhand des konfigurierbaren *Almost-Full-Flags* des *Spy-Fifos* geprüft wird, ob genügend Speicherplatz vorhanden ist. Andernfalls wird das komplette Event verworfen. Zusätzlich wird nach jedem künstlichen End-of-Spill-Signal ein weiteres Datenwort in den *Spy-Fifo* geschrieben, welches das Ende eines Spills markiert. Dieses Kontrollwort wird benötigt um die fertige Rohdaten-Datei abzuschließen. Das Datenformat für die Pseudo-Random-Daten wird in Kapitel A erläutert.

In Tabelle 6.1 ist der Ressourcenverbrauch der Firmware auf dem Xilinx Virtex-5 FPGA dargestellt.

Tabelle 6.1: *Ressourcenverbrauch der Firmware auf dem Xilinx Virtex-5 FPGA.*

	Verbrauch	**Verfügbar**	**Anteil** [%]
Slice-Register	26176	58880	44,52
Slice-LUTs	22708	58880	38,57
Besetzte Slices	12904	14720	87,66
BlockRAM-Einheiten	173	244	70,90
BlockRAM-Speicher	4212 kBit	8784 kBit	47,95
DSP-Slices	495	640	77,34

7. Verifikation und Einbindung in das COMPASS-II Experiment

In diesem Kapitel werden Labormessungen des Scalers und Analysen des Teilchen-Strahls am COMPASS-II Experiment vorgestellt.

Desweiteren wird das Monitoring-System zur Überwachung verschiedener Strahlattribute erklärt. Die FPGA-Firmware wurde für die Auslese eines SciFi-Detektors am COMPASS-II Experiment, welcher sich kurz vor dem Target befindet, eingesetzt. Erste Messungen konnten eine starke Systematik in der Strahlrate aufdecken.

7.1 Labormessungen

Die Funktion des TDC wurde ausführlich in [49] getestet und verifiziert. Für die Scalerfunktion erfolgte ein Vergleich mit der CATCH-Scaler-CMC [73]. Dazu wurde ein GANDALF-Modul mit DMC-Aufsteckkarten als Patterngenerator betrieben, welcher 128 Ausgangssignale für die 32-Kanal Scaler-CMC und die 96 Kanäle des GANDALF-Moduls lieferte. Die Frequenz der Ausgangssignale wurde sukzessive erhöht und die Zählerstände miteinander verglichen. Die Triggerverzögerungen beider Testmodule wurden angepasst. Die Triggerrate wurde auf etwa 10 kHz und die Totzeit auf 1 μs eingestellt. Das Triggersignal wurde mit einem elektronischen Rauschgenerator erzeugt. Die Messdauer betrug $1,9478$ s, was ein Vielfaches der Periodendauer des *CLK38MHz*-Taktsignals entspricht. Das Testsignal war ein Taktsignal mit 50% Duty Cycle[1] und wurde über den Clock Multiplier Chip Si5326 (siehe Kapitel 4.3) aus dem TCS-Taktsignal *CLK155MHz* erzeugt. Die Scaler-CMC wurde im Rahmen einer früheren Diplomarbeit entwickelt [73]. Der Zählmechanismus basiert auf 4 Bit Johnson-Ringzählern und die Funktionalität wurde bis zu einer Eingangsfrequenz von 250 MHz verifiziert.

Das Ergebnis des in Abbildung 7.1 dargestellten Testaufbaus waren Zählerstände, die sich für alle 128 Kanäle um maximal ± 1 unterschieden, sowohl für eine Ein-

[1] Duty Cycle bezeichnet die Dauer des aktiven Zustands im Verhältnis zu einer vollen Taktperiode.

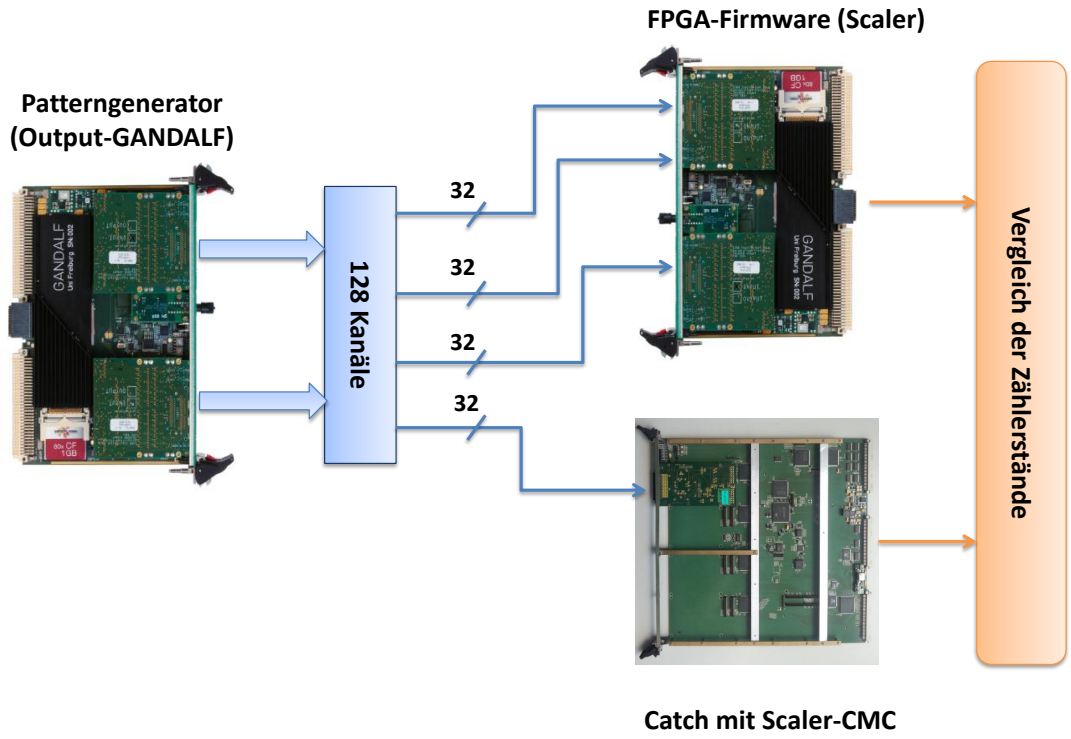

Abbildung 7.1: *Aufbau des Tests zur Überprüfung der Scalerfunktion durch Vergleich mit einer Scaler-CMC.*

gangsfrequenz von $194,4\,\mathrm{MHz}$[2] als auch für $233,28\,\mathrm{MHz}$[3]. Die gleichen Ergebnisse lieferte ein weiterer Test für ein ungetaktetes Eingangssignal. Dieses Zufallsrauschen wurde von einem elektronischen Rauschgenerator erzeugt.

Bei höheren Frequenzen der GANDALF-Output-Signale ($272,16\,\mathrm{MHz}$[4]) funktionierte der Zählmechanismus der Scaler-CMC nur noch für wenige Kanäle. Einige Zählerstände der restlichen Kanäle stimmen trotzdem mit denen des GANDALF-Designs ± 2 überein. Für höhere Frequenzen wurden nur noch die Zählerstände des GANDALF-Designs untereinander verglichen. In Abbildung 7.2 sind die 96 Zählerstände des GANDALF-Designs bei einer Eingangsrate von $311,04\,\mathrm{MHz}$[5] am Ende des Tests aufgetragen. Der Endzustand der Zählerstände beträgt jeweils 605 843 712, in Übereinstimmung[6] mit der Eingangsfrequenz und der Gesamtdauer von $1,9478\,\mathrm{s}$.

Das Testsignal des Output-GANDALFs wurde bei höheren Frequenzen (ab $388,80\,\mathrm{MHz}$[7]) für einige Kanäle zunehmend schlechter, sodass die Zählerstände einiger Kanäle wesentlich unter dem erwarteten Wert lagen. Wurden die Anschlüsse

[2] $5/4 \times 155,52\,\mathrm{MHz}$
[3] $3/2 \times 155,52\,\mathrm{MHz}$
[4] $7/4 \times 155,52\,\mathrm{MHz}$
[5] $2/1 \times 155,52\,\mathrm{MHz}$
[6] $\text{Zählerstand} = \nu \cdot t = 311,04\,\mathrm{MHz} \cdot 1,9478\,\mathrm{s} = 605\,843\,712$
[7] $5/2 \times 155,52\,\mathrm{MHz}$

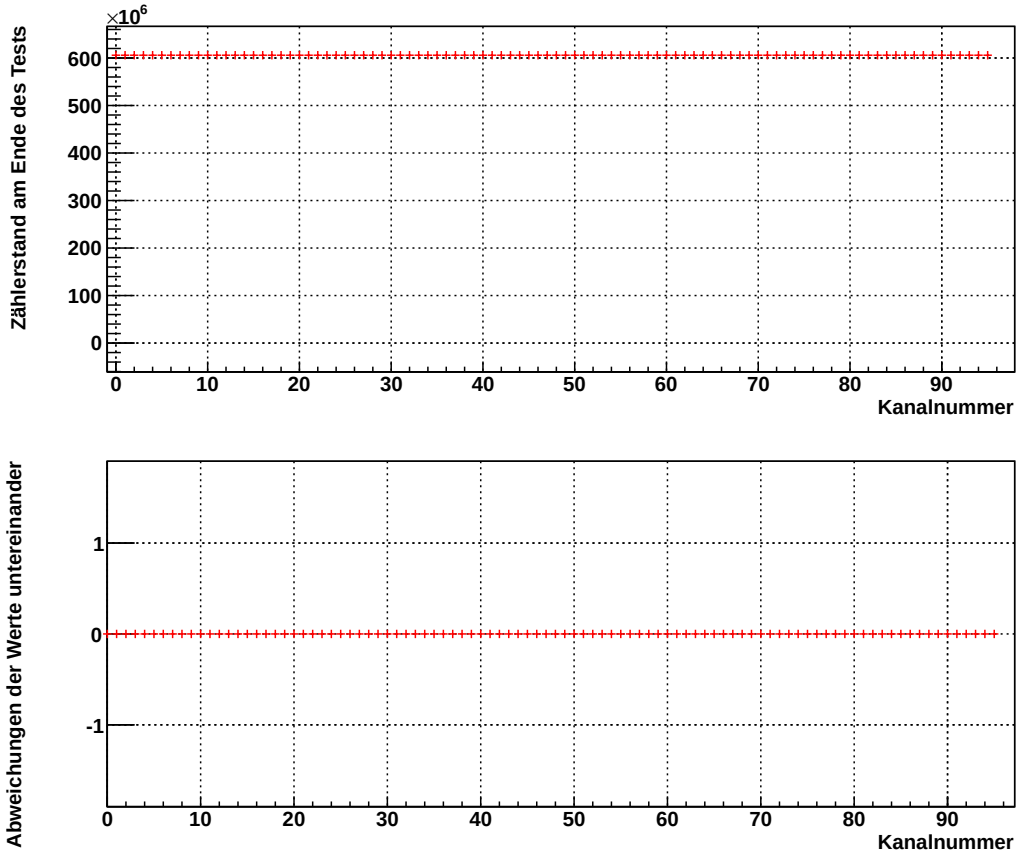

Abbildung 7.2: *Ergebnisse des Labortest mit* $311,04\,MHz$*-Signalen am Ende des Tests. Auftragung der Zählerwerte gegen die Kanalnummer (oben). Unten sind die Abweichungen der Zählerstände vom durchschnittlichen Zählerstand am Ende des Tests aufgetragen.*

an den VHDCI-Steckern des Input-GANDALFs vertauscht, so traten die zu niedrigen Zählerstände an den entsprechend vertauschten Kanälen auf. Dies deutet darauf hin, dass in diesem Frequenzbereich ein Limit für das gemeinsame Schalten von allen Ausgängen der DMC erreicht wird. Auch kann eine nicht optimal geroutete Signalverbindung vom FPGA zu den Buffern auf der DMC zu verstärktem Übersprechen führen und somit für dieses Verhalten verantwortlich sein [83]. Das Problem ließ sich umgehen, indem das Ausgangssignal nur jeweils auf jeden vierten Kanal gegeben wurde (Output-Design G2). Ein Vergleich mit dem 128-Kanal-Output-Design zeigt eine wesentlich bessere Signalform und größere Signalamplitude (vergleiche Abbildung 7.3). Dies bekräftigt oben genannte Vermutung, da die Buffer der Output-DMCs damit entlastet sind und ein Übersprechen von den Nachbarkanälen verringert wurde. Der Test konnte damit für höhere Frequenzen fortgeführt werden.

Erste Unterschiede in den Zählerständen traten ab einer Frequenz von $388,80\,MHz$ auf. Im Endzustand weist etwa jeder achte Kanal eine Abweichung von ± 1 im Vergleich zu den restlichen Zählerständen auf. Bei $466,56\,MHz$[8] trifft dies auf etwa jeden sechsten Kanal zu.

[8] $3/1 \times 155,52\,MHz$

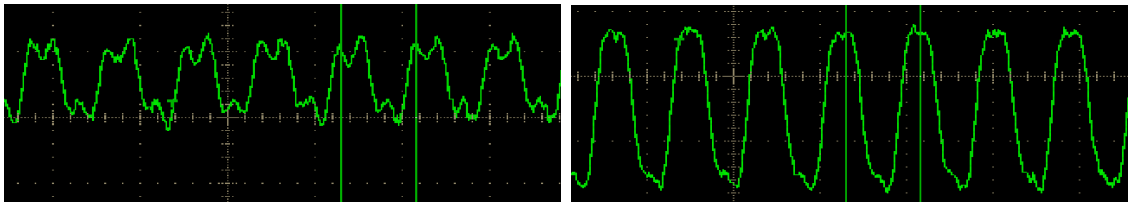

Abbildung 7.3: *Ausgangssignal eines ausgewählten Kanals des 128-Kanal-Output-Designs (links) im Vergleich zu einem typischen Signal des Output-Designs G2 (rechts) bei* $466,56\,MHz$*. Die Skalierung der x-Achsen liegt jeweils bei* $2,5\,ns$ *und die der y-Achsen bei* $200\,mV$*. Neben der deutlich besseren Signalform des G2-Designs ist auch die Amplitude der Signale viel größer.*

In Abbildung 7.4 ist das Ergebnis des Tests bei einer Signalfrequenz von $505,44\,\mathrm{MHz}$[9] dargestellt. Das Signal wurde auf jeden vierten der 96 Eingangsports gegeben[10]. Die gleichen Messungen wurden für die restlichen drei Fälle ebenfalls durchgeführt, jeweils mit den gleichen Ergebnissen. In Abbildung 7.4 sind die jeweiligen Zählerstände am Ende des Tests gegen die Kanalnummer aufgetragen (oben). Die Zählerstände unterscheiden sich zu jedem Trigger-Zeitpunkt um maximal ± 1, so auch im Endzustand (unten). Der Endzustand der Zählerstände beträgt $984\,496\,032(1)$, in Übereinstimmung[11] mit der Eingangsfrequenz und der Gesamtdauer von $1,9478\,\mathrm{s}$.

Zum Vergleich wurden die $505,44\,\mathrm{MHz}$-Signale in einer weiteren Messung über den Clock Multiplier Chip Si5326 intern generiert und auf die Johnson-Ringzähler geleitet. Die Abweichungen vom erwarteten Wert betrug in diesem Fall ebenfalls maximal 1, jedoch traten um 1 zu hohe Zählerstände nicht auf (siehe Abbildung 7.5). Auch konnte dieser Test unter Verwendung aller 96 Kanäle durchgeführt werden.

In Abbildung 7.6 ist die relative Abweichung der gemessenen Gesamtrate (G2-Design) von der erwarteten Durchschnittsrate $24 \cdot 505,44\,\mathrm{MHz} = 12,13056\,\mathrm{GHz}$ bei einer konstanten Triggerfrequenz ($1\,\mathrm{kHz}$) gegen die Zeit aufgetragen. Die Gesamtrate wird folgendermaßen berechnet:

$$\text{Gesamtrate}(t) = \frac{\sum_{ch} \Delta\,\text{hit}(ch,t)}{\Delta\,t}. \tag{7.1}$$

Hierbei gibt $\Delta\,\text{hit}(ch,t)$ die Anzahl der seit dem vorherigen Trigger stattgefundenen Hits auf dem Kanal ch in Abhängigkeit des Trigger-Zeitpunkts t an. $\Delta\,t$ bezeichnet die Zeitdifferenz nacheinander folgender Trigger.

Die kleinen Abweichungen bis etwa $0,8 \cdot 10^{-6}$ lassen sich höchstwahrscheinlich durch leicht unterschiedliches Routing beziehungsweise kleine Laufzeitunterschiede der 24 Eingangssignale erklären. Dies führt dazu, dass je nach Triggerzeitpunkt hin und wieder unterschiedliche Werte (± 1) aus den Johnson-Ringzählern ausgelesen werden, da diese dann leicht asynchron zueinander schalten. Je nach Auslesezeitpunkt der Johnson-Ringzähler, beträgt die Abweichung von $\Delta\,\text{hit}(ch,t)$ vom

[9] $13/4 \times 155,52\,\mathrm{MHz}$

[10] In diesem Beispiel exemplarisch für Kanal $2, 6, 10, ..., 94$

[11] $\text{Zählerstand} = \nu \cdot t = 505,44\,\mathrm{MHz} \cdot 1,9478\,\mathrm{s} = 984\,496\,032$

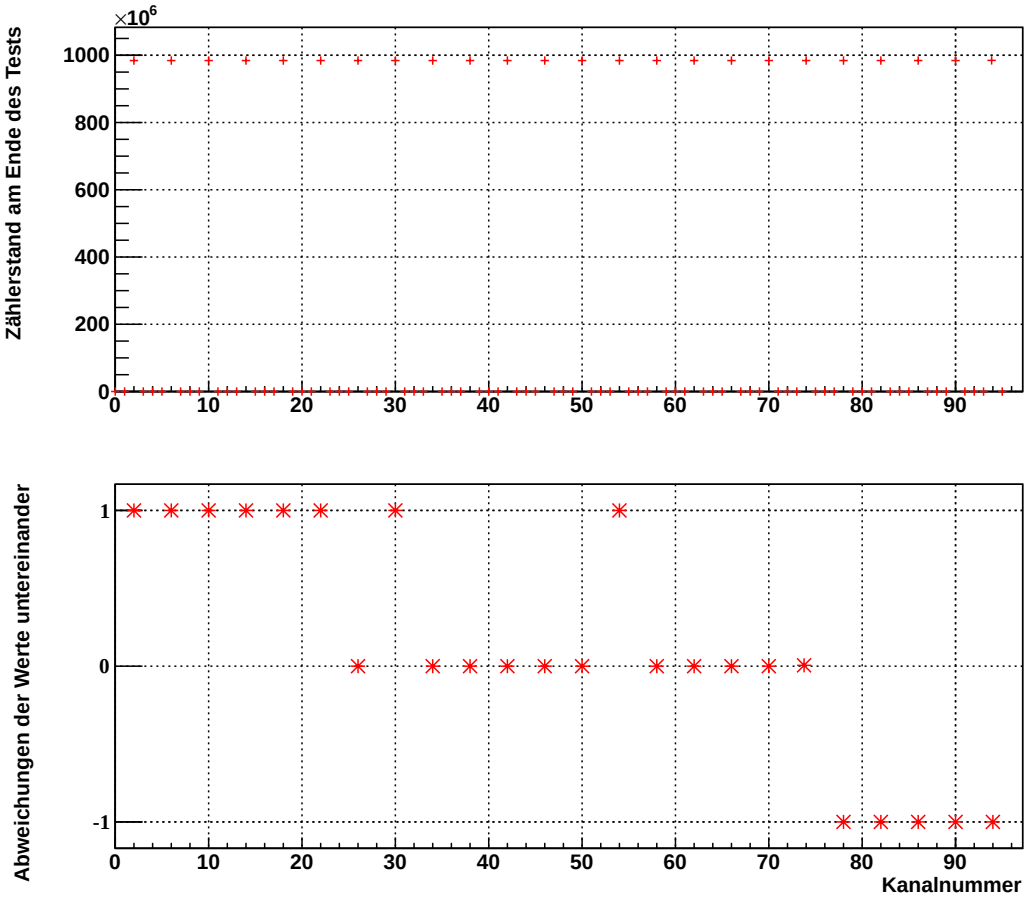

Abbildung 7.4: *Ergebnisse des Labortest mit* $505,44\,MHz$*-Signalen am Ende des Tests (exemplarisch für Kanal 2,6,10,...). Auftragung der Zählerwerte gegen die Kanalnummer (oben). Unten sind die Abweichungen der Zählerstände untereinander aufgetragen.*

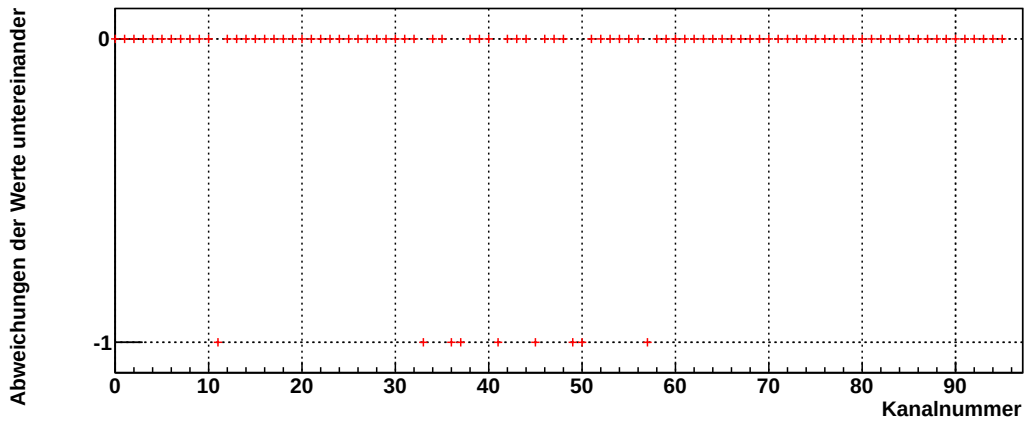

Abbildung 7.5: *Ergebnisse mit* $505,44\,MHz$*-Signalen, die über den SI5326-Chip des zu testenden GANDALF-Moduls intern generiert wurden. Es sind die Abweichungen der Zählerstände vom erwarteten Wert am Ende des Tests aufgetragen.*

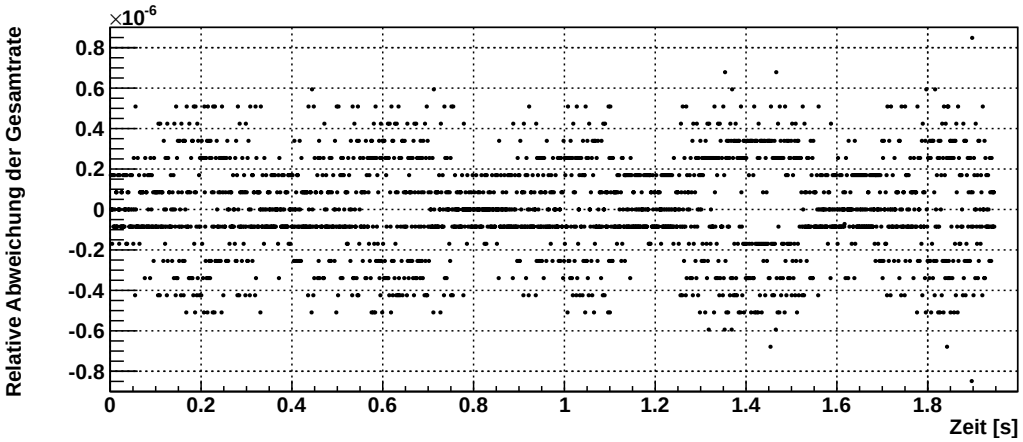

Abbildung 7.6: *Aufgetragen ist die relative Abweichung der gemessenen Gesamtrate von der erwarteten Rate* $24 \cdot 505,44\,MHz$*, wobei sich keine Punkte außerhalb des dargestellten Fensters befinden (Overflow = Underflow = 0). Verwendet wurden Signale des Output-Designs G2. Getriggert wurde zu festen Zeitabständen (1 ms).*

Durchschnittswert $\nu_{505}/\nu_{Trg} = 505440$ maximal ± 1. Die Abweichung der Größe $\xi = \sum_{ch} \Delta\,\mathrm{hit}(ch,t)$ vom Durchschnittswert $\eta = n_{ch} \cdot \nu_{505}/\nu_{Trg}$ beträgt also maximal $\pm n_{ch}$. Die Grenze für die relative Abweichung unter Verwendung von insgesamt $n_{ch} = 24$ Kanälen ergibt sich zu $n_{ch}/\xi = 1,98 \cdot 10^{-6}$. Die relative Abweichung der Gesamtrate nimmt dabei diskrete Werte z/ξ mit $z \in \{-24,-23,...,23,24\}$ an. In diesem Test betrug die maximale relative Abweichung $z/\xi|_{z=\pm 10} = \pm 8,24 \cdot 10^{-7}$ (vergleiche Abbildung 7.6).

Der Effekt lässt sich auch anhand Abbildung 7.7 verdeutlichen. Hierbei ist die zeitabhängige Abweichung des Schwerpunkts von Kanal 48 unter Verwendung einer konstante Triggerrate (1 kHz) dargestellt. Der Schwerpunkt wird folgendermaßen definiert:

$$\mathrm{Schwerpunkt}(t) = \frac{\sum_{ch} ch \cdot \Delta\,\mathrm{hit}(ch,t)}{\sum_{ch} \Delta\,\mathrm{hit}(ch,t)}. \tag{7.2}$$

Die Kanalnummer 48 ergibt sich aus

$$\frac{1}{24}\sum_{k=0}^{23} 4k+2 = 48, \tag{7.3}$$

sofern die gleiche Gewichtung der 24 Kanäle vorliegt, also

$$\Delta\,\mathrm{hit}(ch=2,t) = \Delta\,\mathrm{hit}(ch=6,t) = \cdots = \Delta\,\mathrm{hit}(ch=94,t). \tag{7.4}$$

Es scheint als nähmen die Werte in Abbildung 7.7 ebenfalls diskrete Werte an. Wie in Abbildung 7.6 ist auch eine deutliche Achsensymmetrie bezüglich der Nulllinie zu erkennen. Diese kommt ebenfalls durch oben beschriebene Schwankungen zustande, die mit einem der nächsten Trigger wieder ausgeglichen werden. Davon sind vor

allem Kanäle betroffen, deren Flipflops der Johnson-Register häufig zeitgleich zu einem Trigger schalten.

Für einen Kanal ch' (Triggerzeitpunkt t') mit

$$\Delta\,\mathrm{hit}(ch', t') \;=\; 1 + \underbrace{\varnothing\,(\Delta\,\mathrm{hit}(ch, t'))}_{\text{Durchschnitt aller Kanäle}} \tag{7.5}$$

gilt mit großer Wahrscheinlichkeit für den nächsten Trigger zum Zeitpunkt $t' + 1/\nu_{Trg}$:

$$\Delta\,\mathrm{hit}(ch', t' + 1/\nu_{Trg}) \;=\; \varnothing\,(\Delta\,\mathrm{hit}(t' + 1/\nu_{Trg})) - 1. \tag{7.6}$$

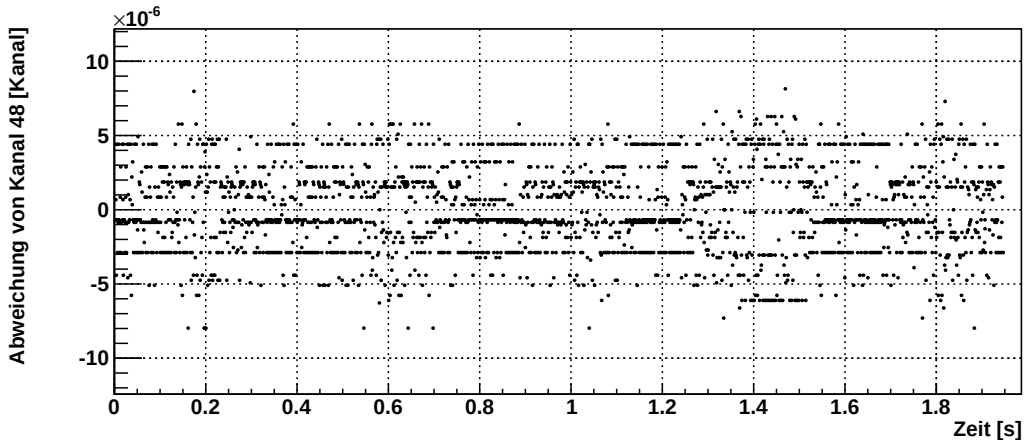

Abbildung 7.7: *Aufgetragen ist die Abweichung des zeitabhängigen Schwerpunkts von Kanal 48 bei konstanter Triggerrate (1 kHz), wobei sich keine Punkte außerhalb des dargestellten Fensters befinden (Overflow = Underflow = 0).*

In Abbildung 7.8 sind obige Tests zusammengefasst.

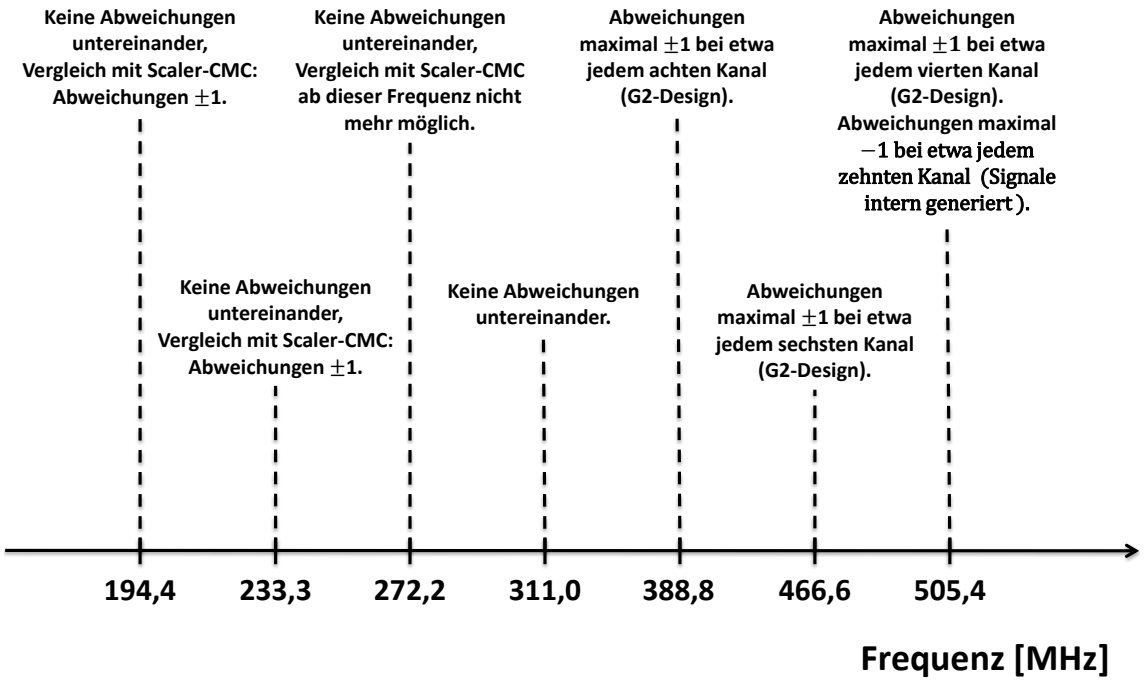

Abbildung 7.8: *Grafische Zusammenfassung der Labormessungen zur Überprüfung der Funktion des Scalers. Diese konnte bis zu einer Eingangsfrequenz von* 505 *MHz mit maximalen Abweichungen von* ±1 *verifiziert werden.*

7.2 Einbindung in das COMPASS-II Experiment

7.2.1 FI02

Der erste Test der Firmware mit „echten“ Detektorsignalen fand im Herbst 2012 statt. Dafür wurden die Eingangsports zweier GANDALF-Module mit den Ausgangssignalen des FI02-Detektors verbunden. Hierbei handelt es sich um ein szintillierendes Faserhodoskop, welches als Tracking-Detektoren für Teilchen dient, die sich sehr nahe am Strahlschwerpunkt befinden. Die Vermessung des Teilchenstrahls geschieht sowohl vor als auch hinter dem Target mittels mehrerer SciFi-Stationen (siehe Kapitel 3). Außerdem können mittels Zeitkorrelationen Myonspuren rekonstruiert werden [34].

FI02 befindet sich kurz vor dem Target und vermisst somit den Myonstrahl vor der Wechselwirkungszone. Die SciFi-Station ist in zwei Projektionsebenen (X,Y) aufgeteilt, die senkrecht zueinander stehen. Der Teilchenstrahl trifft zuerst auf die vertikale Y-Ebene und dann auf die horizontale X-Ebene.

Wie in Abschnitt 3.3.1 beschrieben, hinterlässt ein ionisierendes Teilchen Szintillatorlicht in den Fasern, welches von PMTs (Hamamatsu H6568 [84]) verstärkt wird. Das Signal wird dann über Diskriminatoren [85] zur Auslese-Elektronik geleitet. Jede Ebene der FI02 enthält 96 Kanäle, sodass insgesamt zwei GANDALF-Module

für die Auslese herangezogen werden. FI02 hat eine räumliche Auflösung von ungefähr 130 μm und die intrinsische Effizienz für den Nachweis eines Strahlteilchens ist $\geq 99\%$ [34]. Die Zeitauflösung ist über alle Kanäle annähernd konstant ($350-450$ ps, siehe Abbildung 3.5), wobei diese ratenabhängig ist, sodass die Auflösung in äußeren Kanälen mit geringeren Hitraten etwas besser ist. Die aktive Fläche der X- und Y-Ebene der FI02 misst $3,9\,\mathrm{x}\,3,9\,\mathrm{cm}^2$. Die Fasern haben einen Durchmesser von $0,5$ mm, was ungefähr $1,6\,\%$ der Strahlungslänge entspricht [34].

7.2.2 Das Monitoring-System

Das Monitoring-System besteht aus drei Teilen und ist in Abbildung 7.9 zusammengefasst. Nach jedem Spill wird für die Pseudo-Random-Daten beider GANDALF-Module (FI02-X, FI02-Y) jeweils eine Rohdaten-Datei im hex-Format erzeugt. Dies geschieht, wie in Abschnitt 6.2.2 erklärt, unabhängig von der COMPASS-II-DAQ. Für eine Unterscheidung der Spills wird der Dateiname mit fortlaufenden Indizes versehen. Für die Auslese des 32 Bit breiten Spy-Fifo im DSP-FPGA (vergleiche Abschnitt 4.6 beziehungsweise 6.3) werden die Daten vom CPLD an den VME-Bus weitergeleitet. Die Übertragung vom GANDALF-Modul (Slave) zur VME-CPU (Master) geschieht im Block-Transfer-Modus. Wenn viele Datenworte übertragen werden sollen, ist dieser schneller als ein normales „VME-Read“-Kommando, da die Adresse nur zu Beginn einmal gesetzt werden muss und dann kontinuierlich Daten übertragen werden können. Die Adressen auf Master und Slave werden automatisch mit jedem Datenwort inkrementiert. Die Daten werden per DMA[12] in den Arbeitsspeicher der VME-CPU übertragen und danach in eine Datei im NFS[13] geschrieben.

Für den zweiten Teil des Systems wurde ein Analyse-Tool programmiert, welches die Pseudo-Random-Daten einliest und verschiedene Histogramme sowie zeitabhängige Graphen erzeugt. Dafür wird ROOT [86] verwendet. ROOT ist ein auf C++ Klassen basierendes Softwarepaket zur Datenverarbeitung und Visualisierung, welches am CERN entwickelt wurde und vor allem in der Teilchenphysik angewendet wird.

In der GUI[14] werden dargestellt:

- Integriertes Strahlprofil am Ende des Spills (X- und Y-Ebene)
- Schwerpunkt des Strahls (X-Ebene) in Abhängigkeit der Zeit
- Strahlrate (X-Ebene) in Abhängigkeit der Zeit
- FFT[15] der Strahlrate für Realteil und Absolutwert der Fourier-Koeffizienten
- Kreuzkorrelationen der Detektorsignale in erster bis vierter Ordnung (X-Ebene)

[12] DMA = Direct Memory Access
[13] NFS = Network File System
[14] GUI = Grafische User-Interface
[15] FFT = Fast Fourier Transformation

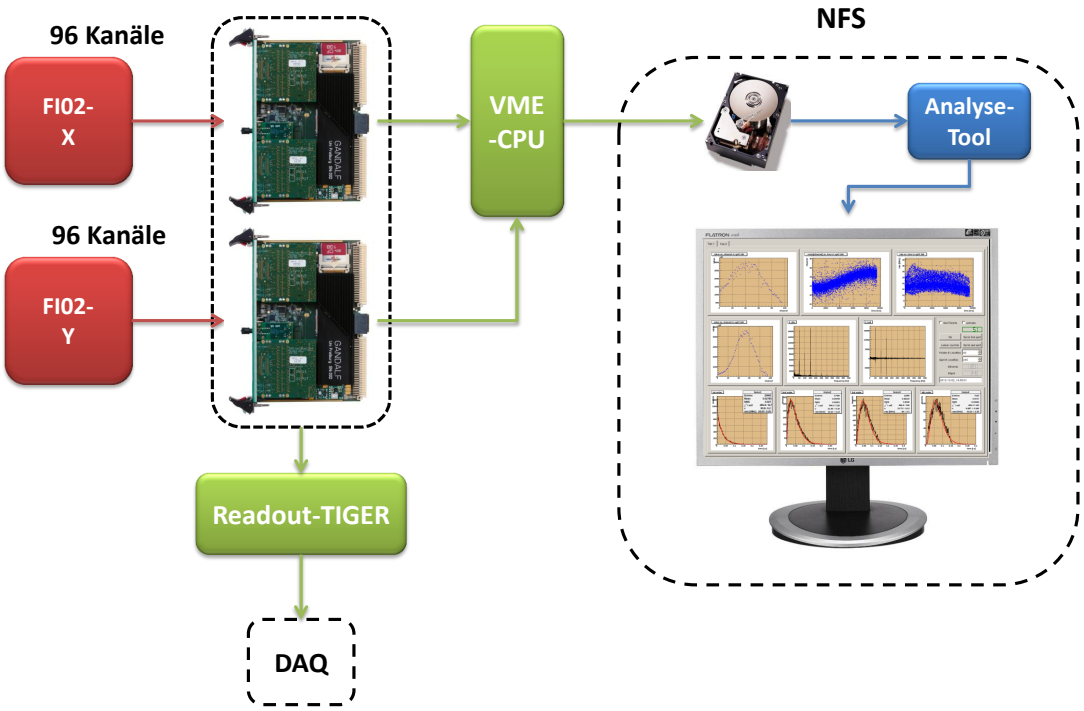

Abbildung 7.9: *Schematische Darstellung des Monitoring-Systems.*

Bis auf das integrierte Strahlprofil in der Y-Ebene werden alle Diagramme mit den Daten der FI02-X-Ebene gewonnen. Für die Daten der Y-Ebene genügt ein Trigger am Ende des Spills. Über den Konfiguratonsspeicher lässt sich dieser Modus einstellen, sodass das Pseudo-Randomtrigger-Signal im FPGA direkt mit dem künstlichen End-Of-Spill-Signal verbunden wird, womit viel weniger Daten anfallen. Bis auf die Kreuzkorrelationen der Myonsignale werden alle Bilder mit der Scalerfunktion des GANDALF-Designs gewonnen. Für die vier Histogramme der Kreuzkorrelationen werden in $n$ter Ordnung (n=1,2,3,4) die Zeitintervalle zwischen $n+1$ aufeinander folgenden Myonsignalen in Histogrammen aufgetragen, wozu die TDC-Zeitmarken verwendet werden. Wird zum Zeitpunkt t_0 ein Myon-Hit gemessen, so ist für ein zeitlich unkorreliertes Signal die Wahrscheinlichkeit, n weitere Myon-Signale im Zeitintervall $[t_0, t]$ zu finden durch die Funktion $w_n(t-t_0)$ gegeben. Die Wahrscheinlichkeitsdichte-Funktionen w_n wurden in Abschnitt 6.2.3 erklärt. An die gemessenen Daten der Kreuzkorrelationen werden Funktionen $c \cdot w_n(t-t_0)|_{t_0=0}$ gefittet, wobei c die Normierung berücksichtigt. Die Histogramme und Graphen werden mit den gleichen Indizes im Dateinamen in einer ROOT-Datei abgespeichert. Das Analyse-Tool schreibt den aktuellen Index in eine Textdatei und wartet danach auf die beiden Rohdaten-Dateien mit dem nächstgrößeren Index.

Der dritte Teil des Systems stellt das grafische User-Interface dar. Dafür wird die C++ Klassenbibliothek „Qt“ [87] beziehungsweise „Qt-ROOT“ [88] verwendet, die eine plattformübergreifende Programmierung grafischer Benutzeroberflächen ermöglicht. Wie die anderen beiden Teile des Systems ist dieser ebenfalls vollkommen

automatisiert. Die oben erwähnte Textdatei wird alle zwei Sekunden ausgelesen, um zu überprüfen, ob sich der Index geändert hat. Ist dies der Fall, so wird die neue ROOT-Datei eingelesen und die darin enthaltenen Graphen und Histogramme geöffnet und auf einem Monitor angezeigt. Außerdem wird die Uhrzeit und das Datum des Spills, der Spill-Index und die Anzahl der Pseudo-Randomtrigger angezeigt (siehe Abbildung 7.10). Ein fortlaufender Zähler (grün) zeigt die Zeit in Sekunden seit der letzten Aktualisierung der Diagramme an. Dieser wird nach längerer Strahlpause rot. Zudem können alle bisher aufgenommenen Graphen und Histogramme durch Auswahl des Spill-Index noch einmal angezeigt werden. Dazu kann die automatische Aktualisierung der GUI deaktiviert werden.

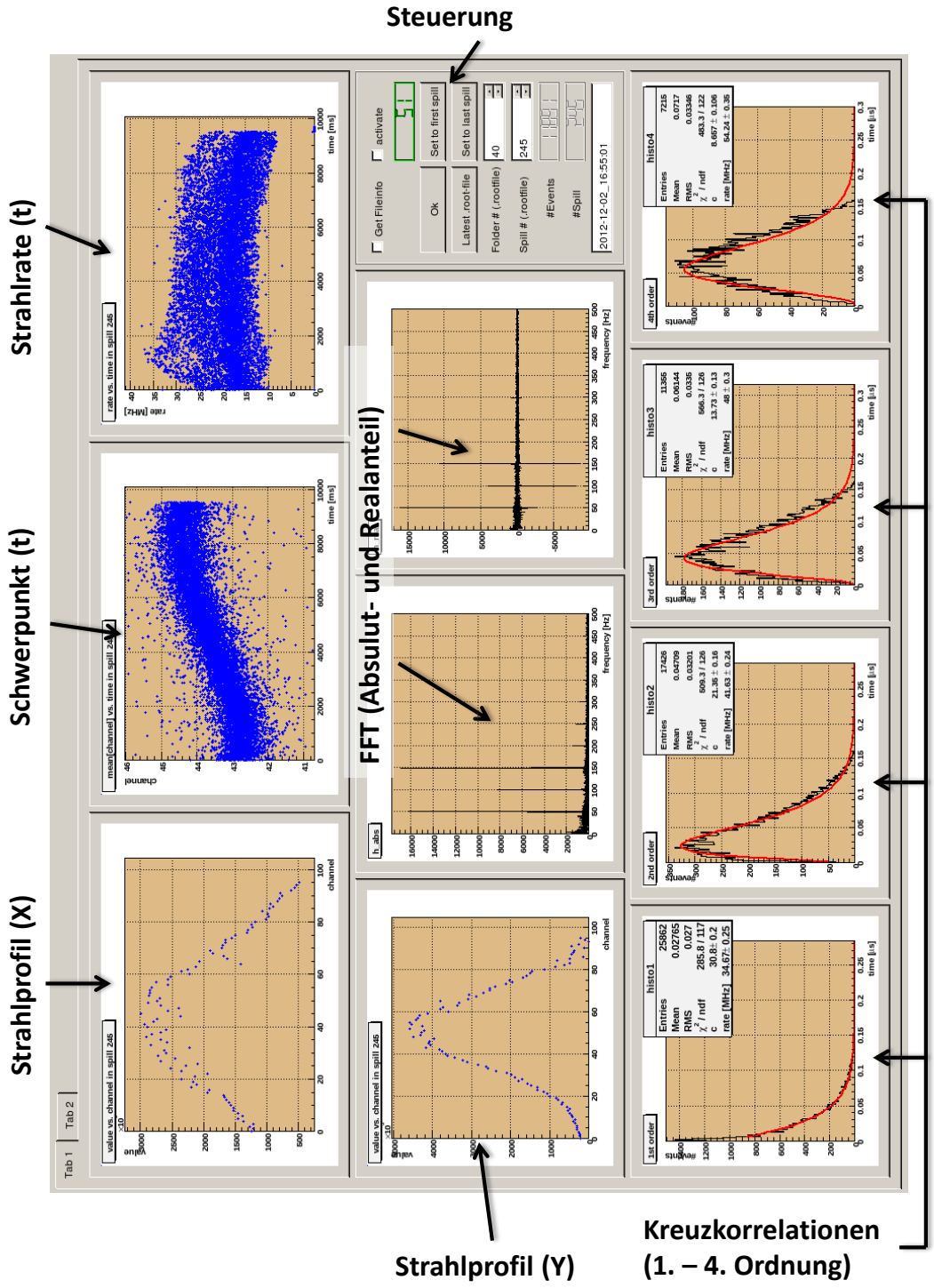

Abbildung 7.10: *Das grafische User-Interface.*

7.3 Ergebnisse der Messungen am COMPASS-II Experiment

7.3.1 Scaler-Daten

Mit dem GANDALF-Design konnten über mehrere Wochen hinweg bis zum Ende der Strahlzeit im Herbst 2012 problemlos Daten genommen werden, sowohl für die COMPASS-II-DAQ als auch für das Beam-Monitoring. Mit dem Beam-Monitoring konnte sehr schnell ein Problem in der Strahlrate festgestellt werden. Diese wird von einer starken 50 Hz-Schwankung geprägt. Wie in Abbildung 7.11 dargestellt, variierte die Strahlrate zwischen Minimum und Maximun teilweise um den Faktor fünf. Neben dieser Komponente zeigt die Frequenzanalyse (FFT) ebenso die auftretenden Oberfrequenzen bei 100 Hz, 150 Hz und 200 Hz. Für die DVCS-Messungen gilt es, systematische Effekte, wie beispielsweise ratenabhängige Effekte in manchen Detektoren, zu vermeiden.

Das Problem wurde mit SPS-Experten besprochen. Es konnte festgestellt werden, dass die 50 Hz-Oszillation mit dem 220 V-Stromnetz synchronisiert ist. Höchstwahrscheinlich wird diese Komponente beim Auskopplungsvorgang beziehungsweise beim Regulieren der Strahlrate auf das Beryllium-Target (T6) induziert.

Für die Strahloptimierung wurde eine Webseite angelegt [89], auf der das Spillprofil und die Fourier-Transformation der Strahlrate nach jedem Spill aktualisiert wird. Dies sollte den SPS-Experten ermöglichen, das Problem durch Feinabstimmung der betroffenen Elemente besser in den Griff zu bekommen. Gegen Ende der Strahlzeit im Herbst 2012 konnte die 50 Hz-Oszillation in der Extraktions-Rückkopplungsschleife etwas abgeschwächt werden, jedoch traten verstärkt andere Frequenzen bis 450 Hz auf. Die Schwankung der Amplitude in der Strahlrate konnte jedoch deutlich gesenkt werden (siehe Abbildung 7.12).

Ebenso konnte festgestellt werden, dass sich der Strahlschwerpunkt in der X-Ebene innerhalb eines Spills um etwa einen Kanal weiterbewegt ($42 \rightarrow 43$). In der Y-Ebene bleibt der Strahlschwerpunkt konstant. Hierzu sind in Abbildung 7.13 Graphen für beide Ebenen mit drei unterschiedlichen Triggerraten dargestellt.

Die Strahlprofile beider SciFi-Ebenen befinden sich in Abbildung 7.14. Diese wiesen durchgehend keine großen qualitativen Veränderungen auf. Zu erkennen ist, dass es vor allem für einige Kanäle der X-Ebene einer Feinjustierung der Diskriminator-Schwellen bedarf.

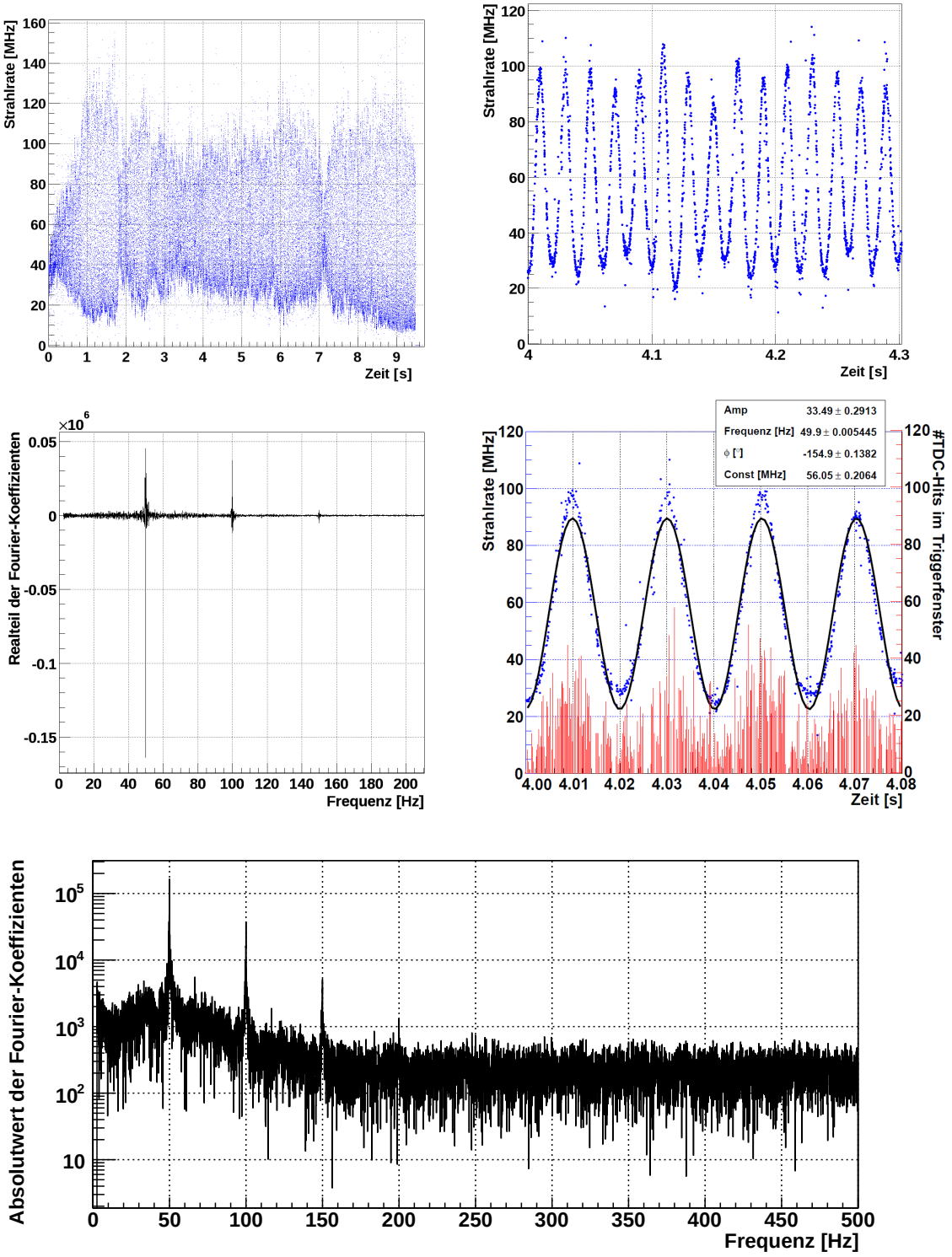

Abbildung 7.11: *Darstellung des Spillprofils (oben links) und ausgewählte Bereiche davon (oben rechts und Mitte rechts) bei einer Triggerfrequenz von* $9,49\,kHz$. *In der Mitte rechts ist zudem die Anzahl der TDC-Hits bei gegebenem Triggerfenster dargestellt (rot), die der gleichen zeitlichen Schwankung unterliegt. Außerdem wurde eine Sinusfunktion an die Strahlrate gefittet. Die Ergebnisse der Fourier-Analyse sind in der Mitte links (Realteil der Koeffizienten) und unten (Absolutwert der Koeffizienten in logarithmischer Darstellung) abgebildet (Spilldatum: 31.10.2012 - 19:36 Uhr).*

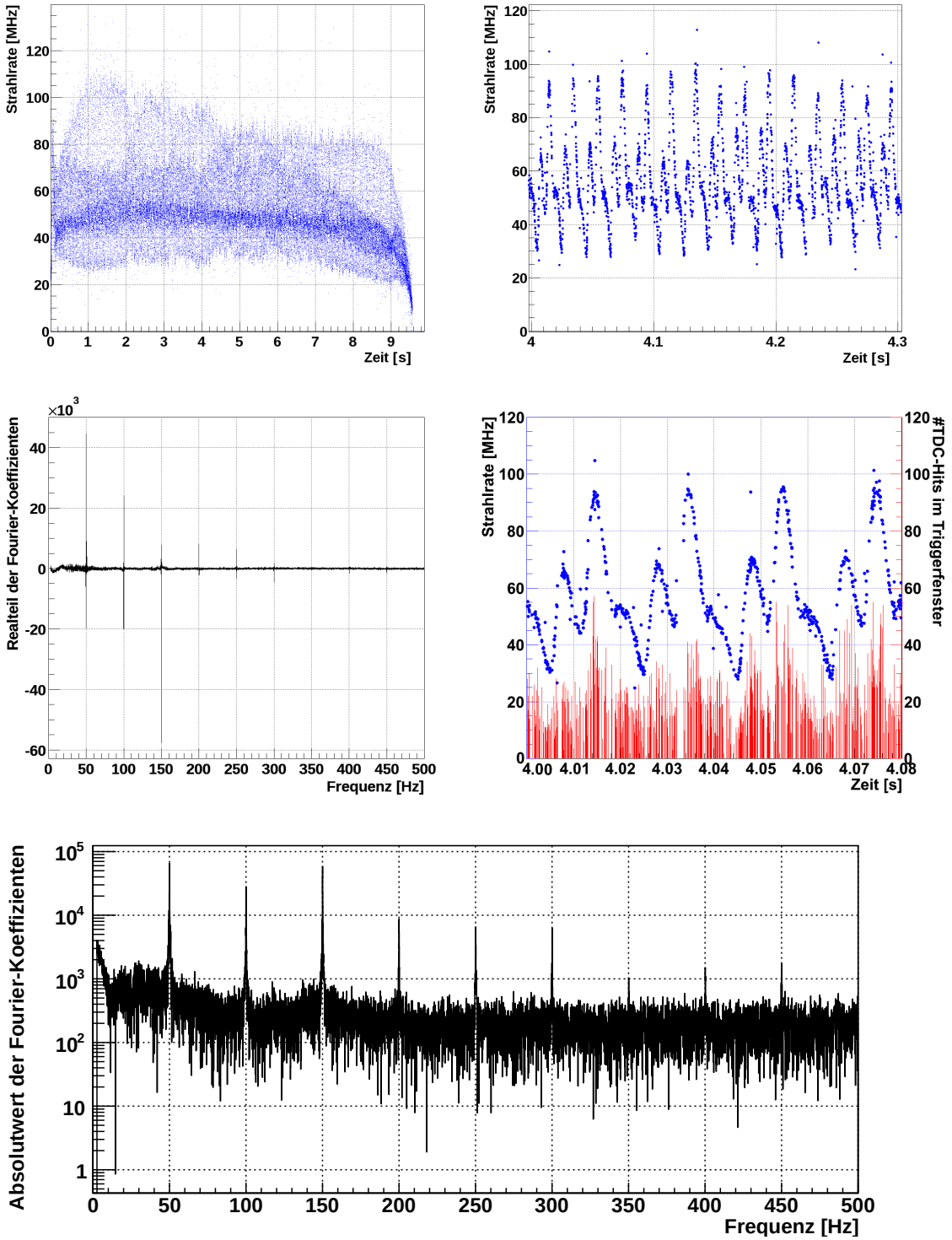

Abbildung 7.12: *Darstellung des Spillprofils (oben links) und ausgewählte Bereiche davon (oben rechts und Mitte rechts) bei einer Triggerfrequenz von* $9,49\,kHz$. *In der Mitte rechts ist zudem die Anzahl der TDC-Hits bei gegebenem Triggerfenster dargestellt (rot), die der gleichen zeitlichen Schwankung unterliegt. Die Ergebnisse der Fourier-Analyse sind in der Mitte links (Realteil der Koeffizienten) und unten (Absolutwert der Koeffizienten in logarithmischer Darstellung) abgebildet (Spilldatum: 29.11.2012 - 23:31 Uhr).*

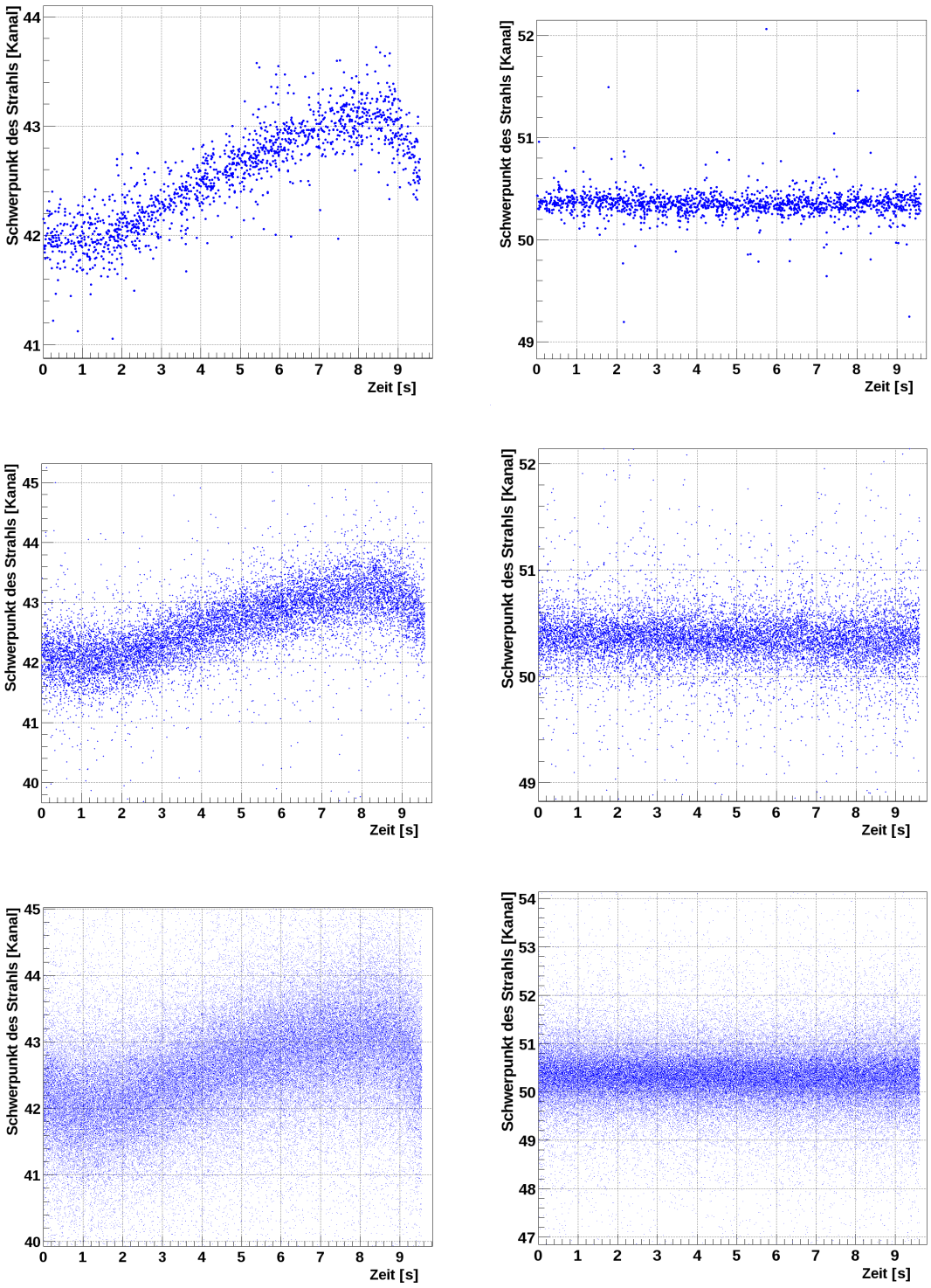

Abbildung 7.13: *Darstellung des Strahlschwerpunks in Abhängigkeit der Zeit für die X-Ebene (links) und die Y-Ebene (rechts). Die Rate des Pseudo-Randomtriggers lag bei* $0,15\,kHz$ *(oben),* $1,19\,kHz$ *(Mitte) und* $9,49\,kHz$ *(unten). Spilldatum: 03.12.2012 - 00:32 Uhr bis 01:03 Uhr).*

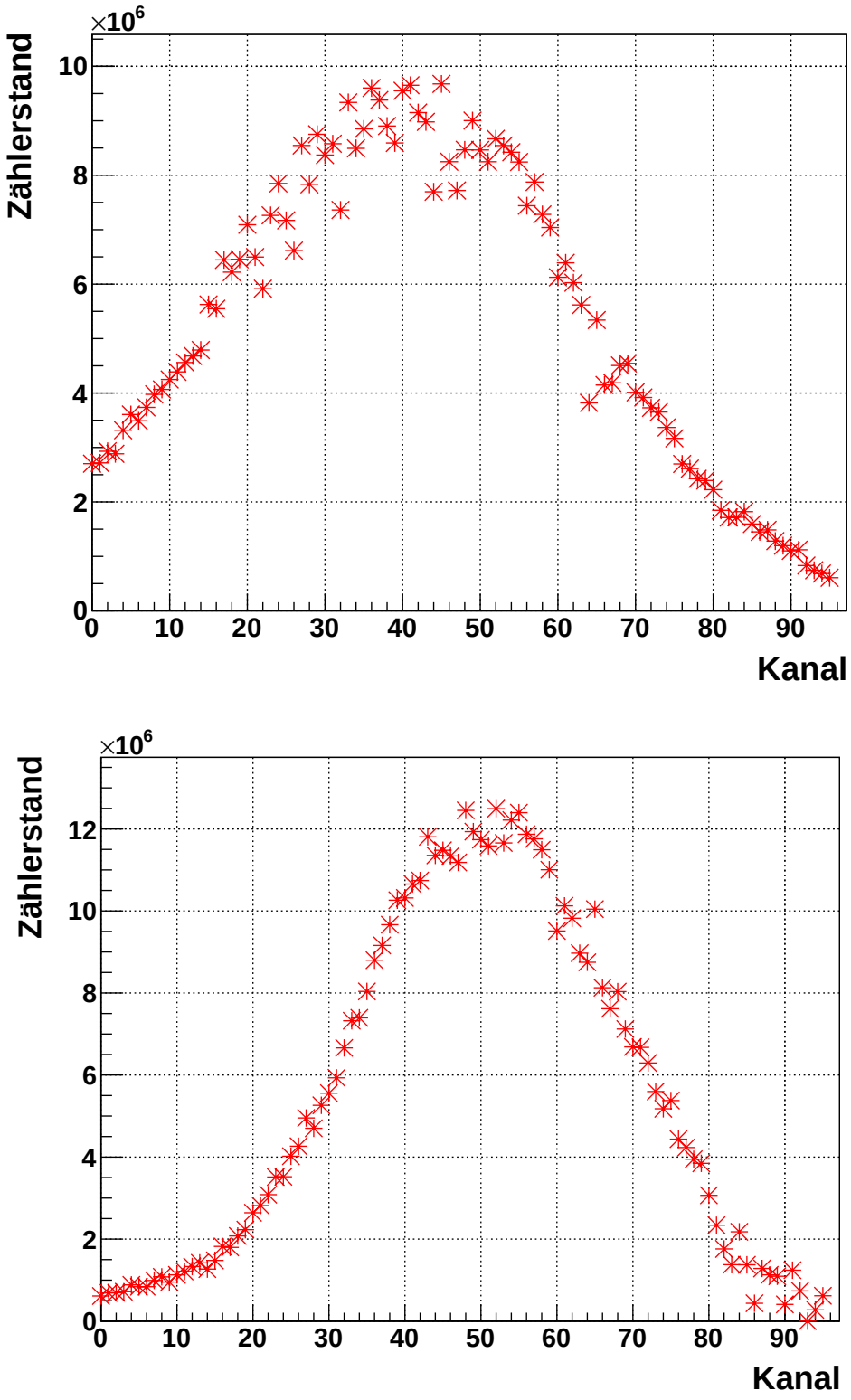

Abbildung 7.14: *Das integrierte Strahlprofil für die X-Ebene (oben) und die Y-Ebene (unten) für einen Spill vom 01.12.2012 - 01:48 Uhr.*

7.3.2 TDC-Daten

Die Kreuzkorrelationen der Myonsignale zeigen im Allgemeinen keine gute Anpassung an die Poissonkurven w_n, wie an den jeweiligen Angaben für χ^2/ndf[16] in Abbildung 7.15 zu erkennen ist.

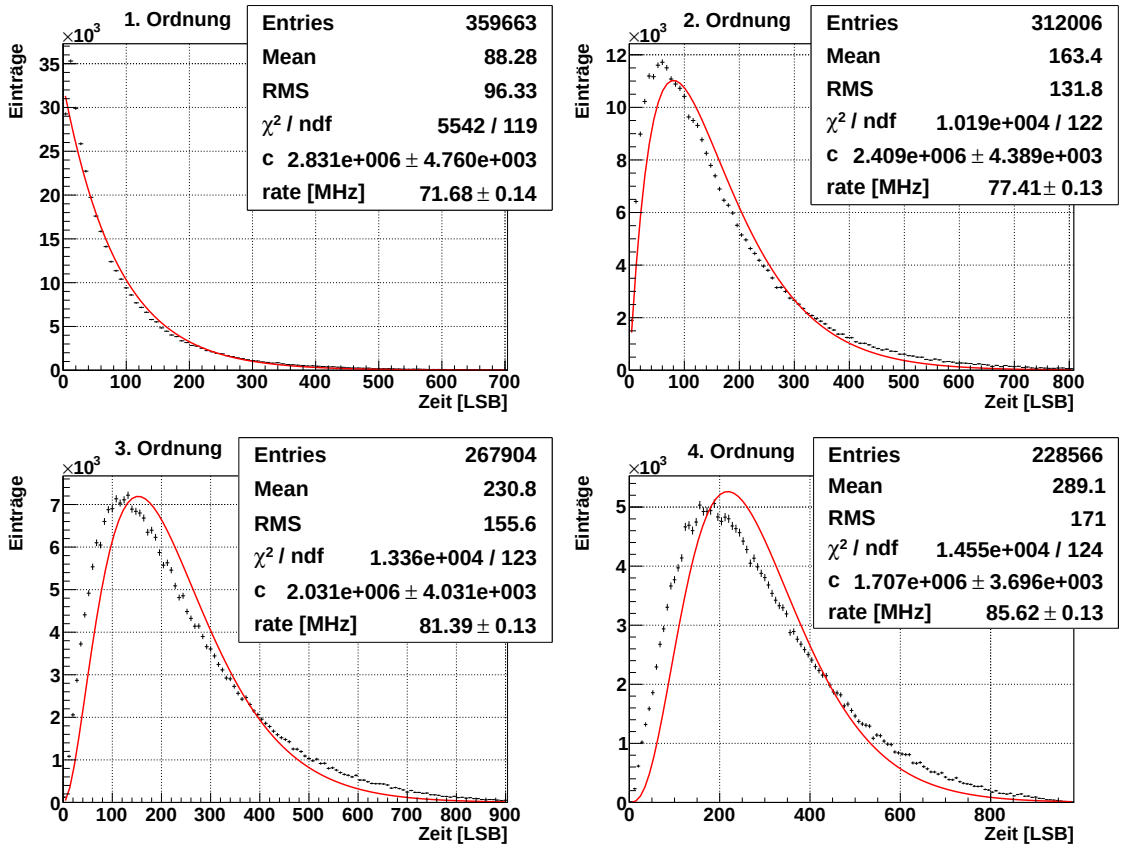

Abbildung 7.15: *Kreuzkorrelationen bis zur vierten Ordnung. Deutlich zu erkennen ist, dass die Poissonkurven w_n keine gute Anpassung an die Datenpunkte darstellen, da sowohl vermehrt kleine als auch große Zeitintervalle zwischen den Myonsignalen auftreten (Spilldatum: 01.12.2012 - 01:48 Uhr).*

Auch stimmt der aus den Fits bestimmte Parameter „rate“ für die Strahlrate nicht innerhalb der Fehlergrenzen überein. Im Vergleich zur theoretischen Anpassungskurve scheinen die Datenpunkte sowohl zu kleinen als auch zu großen Zeitintervallen hin versetzt zu sein. Das heißt es treten vermehrt kleinere und größere Zeitintervalle zwischen den Myonsignalen auf, als es für ein zeitlich unkorreliertes Signal der Fall wäre. Neben der oszillierenden Strahlrate könnte auch ein Übersprechen (Crosstalk) benachbarter SciFi-Kanäle zu systematischen Effekten beitragen, wie es durch die Anordnung der szintillierenden Fasern gegeben sein könnte (siehe Abbildung 3.5). Ein Myon kann abhängig von seiner Flugrichtung in mehreren Kanälen Energie deponieren, da ein Kanal aus mehreren hintereinander liegenden Fasern besteht. Zusätzlich ist eine Energiedeposition in verschiedenen Kanälen durch Übersprechen direkt am Photomultiplier möglich. Der Crosstalk wird bei FI02 durch das sogenannte „PSC[17]“ unterdrückt [90], indem die Amplituden auf benachbarten

[16] ndf = number of degrees of freedom
[17] PSC = Peak-Sensing Circuit

Kanälen miteinander verglichen werden und der schwächere verworfen wird, sofern beide Kanäle zur gleichen Zeit einen Hit aufweisen. Dennoch wirkt das PSC nur auf benachbarte Kanäle, die vom gleichen Photomultiplier verstärkt werden[18].

Um die mögliche Systematik des Übersprechens genauer zu untersuchen, wurden kleine Zeitintervalle zwischen zwei Hits im Bereich von null bis 60 TDC-LSB[19] betrachtet. Bis zu diesem Bereich ließen sich deutliche Systematiken erkennen. Die Häufigkeit der damit verbundenen SciFi-Kanäle ist für einen Spill in Abbildung 7.16 und 7.17 aufgetragen (X-Ebene). Auf der x-Achse ist der Kanal des ersten Hits und auf der y-Achse der des zweiten Hits aufgetragen. Hier ist eine deutliche Systematik zu erkennen. Benachbarte Kanäle weisen im Vergleich zu anderen Kombinationen eine viel größere Häufigkeit auf. Dies macht sich vor allem in den Randbereichen bemerkbar, in denen das PSC nicht zu tragen kommt, da Kanäle unterschiedlicher Photomultiplier nicht miteinander verglichen werden. Dabei beschränkt sich das unkorrigierte Übersprechen nicht nur auf den nächsten Nachbarn [74]. Die Systematik wird in Abbildung 7.20 exemplarisch für das Übersprechen zwischen Kanal 47 und Kanal 48 verdeutlicht.

Abbildung 7.16 ist zu entnehmen, dass innerhalb des gewählten Zeitintervalls keine zwei Hits vom gleichen Kanal nachgewiesen werden können. Dies lässt sich mit der Totzeit der szintillierenden Fasern von einigen Nanosekunden erklären.

Zum Vergleich ist in Abbildung 7.21 die Häufigkeit für Hits zwischen 100 und 160 LSB für den gleichen Spill aufgetragen. Wie erwartet liegt eine nahezu radialsymmetrische Verteilung vor, das heißt für größere Zeitintervalle spielen oben erwähnte Effekte keine Rolle mehr. Die Parameter „MeanX“ und „MeanY“ aus den 2D-Gauss-Fits sind mit den Graphen aus den Scaler-Daten in Abbildung 7.14 respektive 7.13 vereinbar. Für die Y-Ebene können aus den Abbildungen 7.18, 7.19 und 7.22 die gleichen Ergebnisse entnommen werden. Abbildung 7.18 lässt sich entnehmen, dass gar keine Hits auf Kanal 93 zu finden sind, in Übereinstimmung mit dem integrierten Spillprofil für die Y-Ebene (siehe Abbildung 7.14). Ein Vergleich zur X-Ebene lässt darauf schließen, dass ein defekter Detektorkanal vorliegt und dies kein Problem der FPGA-Firmware ist.

[18] Dabei werden jeweils 16 Kanäle von einem Photomultiplier verstärkt.
[19] 60 TDC-LSB entspricht etwa $9,5\,\mathrm{ns}$.

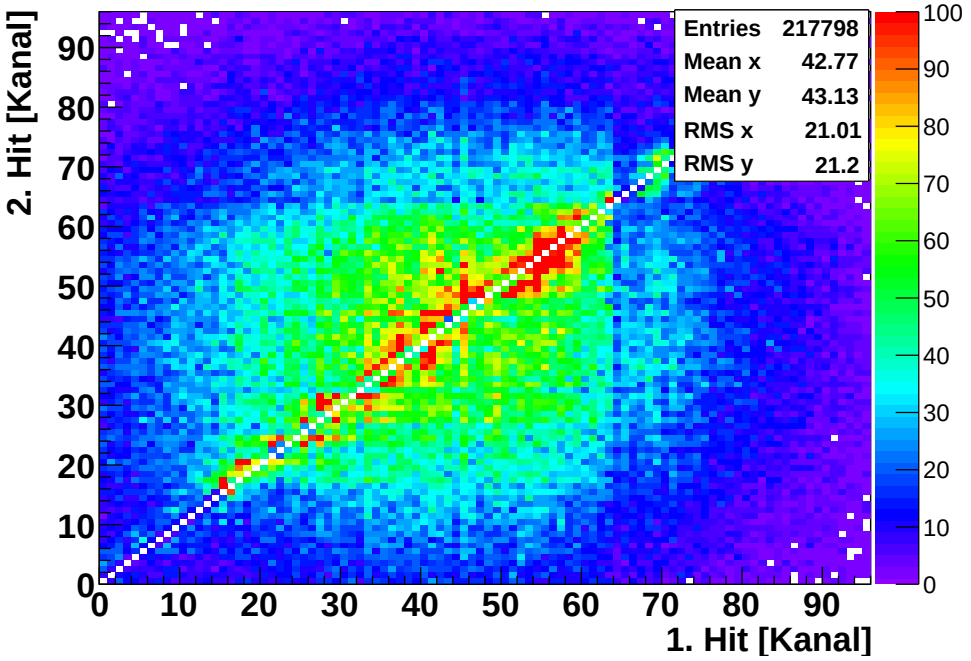

Abbildung 7.16: *2D-Histogramm: Häufigkeitsverteilung der Kanal-Kombinationen für den Nachweis von zeitlich nahe beieinander liegenden Hits bis 60 TDC-LSB (X-Ebene, Spilldatum: 03.12.2012 - 00:38 Uhr).*

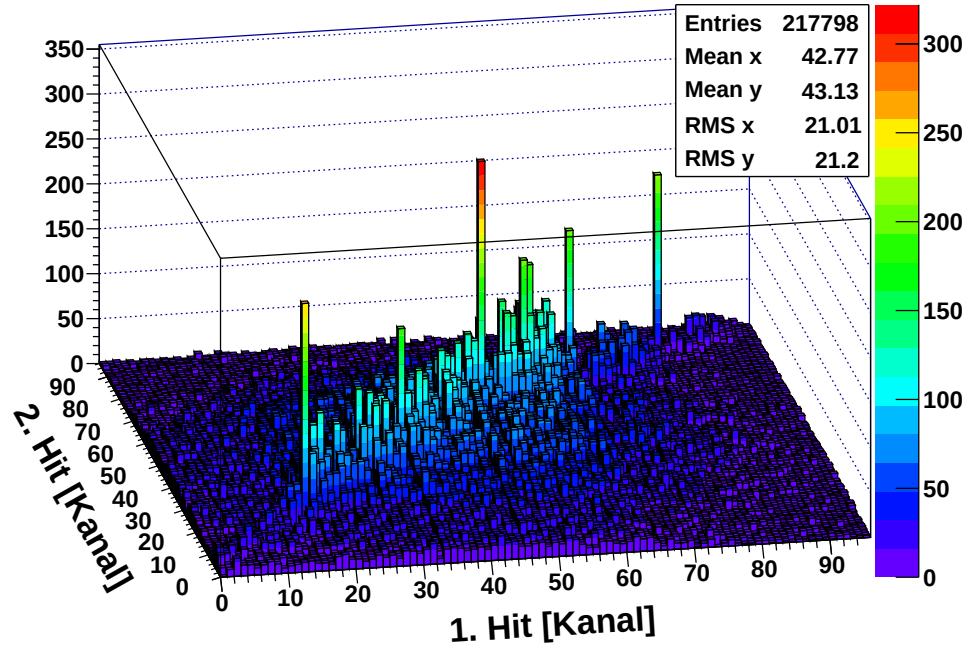

Abbildung 7.17: *3D-Histogramm: Häufigkeitsverteilung der Kanal-Kombinationen für den Nachweis von zeitlich nahe beieinander liegenden Hits bis 60 TDC-LSB (X-Ebene, Spilldatum: 03.12.2012 - 00:38 Uhr).*

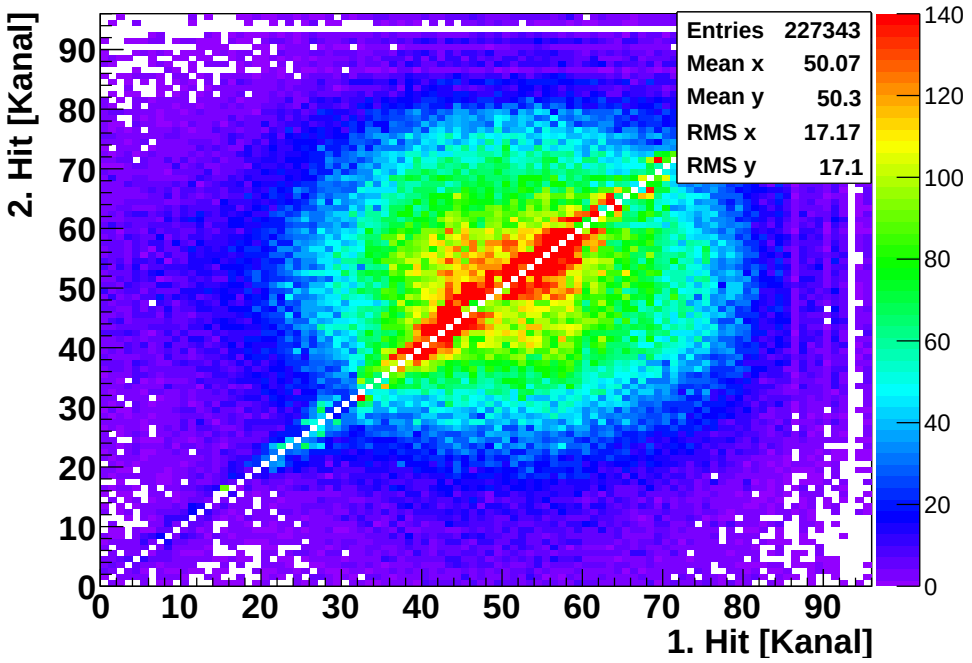

Abbildung 7.18: *2D-Histogramm: Häufigkeitsverteilung der Kanal-Kombinationen für den Nachweis von zeitlich nahe beieinander liegenden Hits bis 60 TDC-LSB (Y-Ebene, Spilldatum: 03.12.2012 - 01:03 Uhr).*

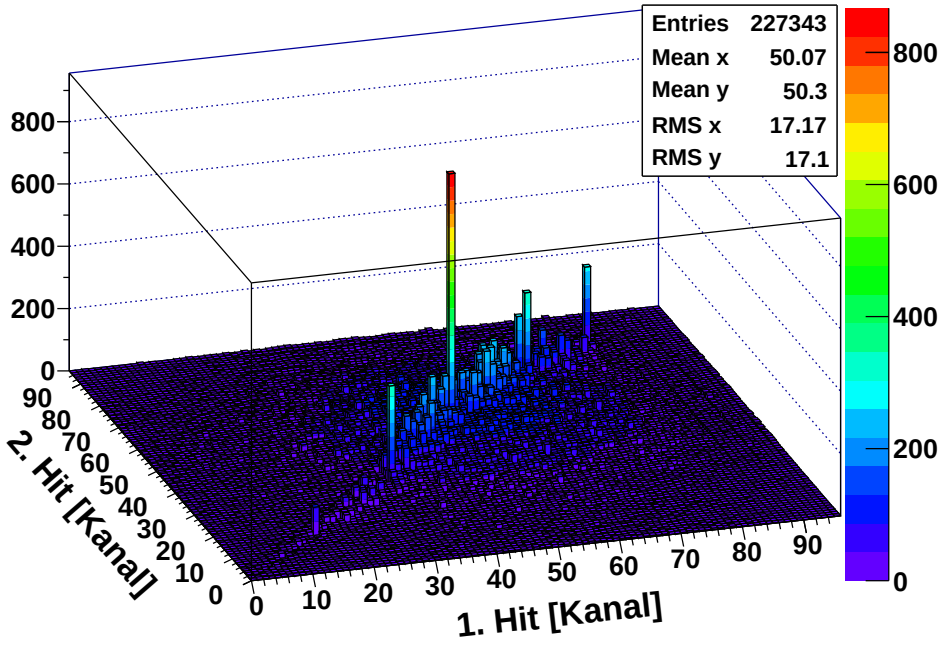

Abbildung 7.19: *3D-Histogramm: Häufigkeitsverteilung der Kanal-Kombinationen für den Nachweis von zeitlich nahe beieinander liegenden Hits bis 60 TDC-LSB (Y-Ebene, Spilldatum: 03.12.2012 - 01:03 Uhr).*

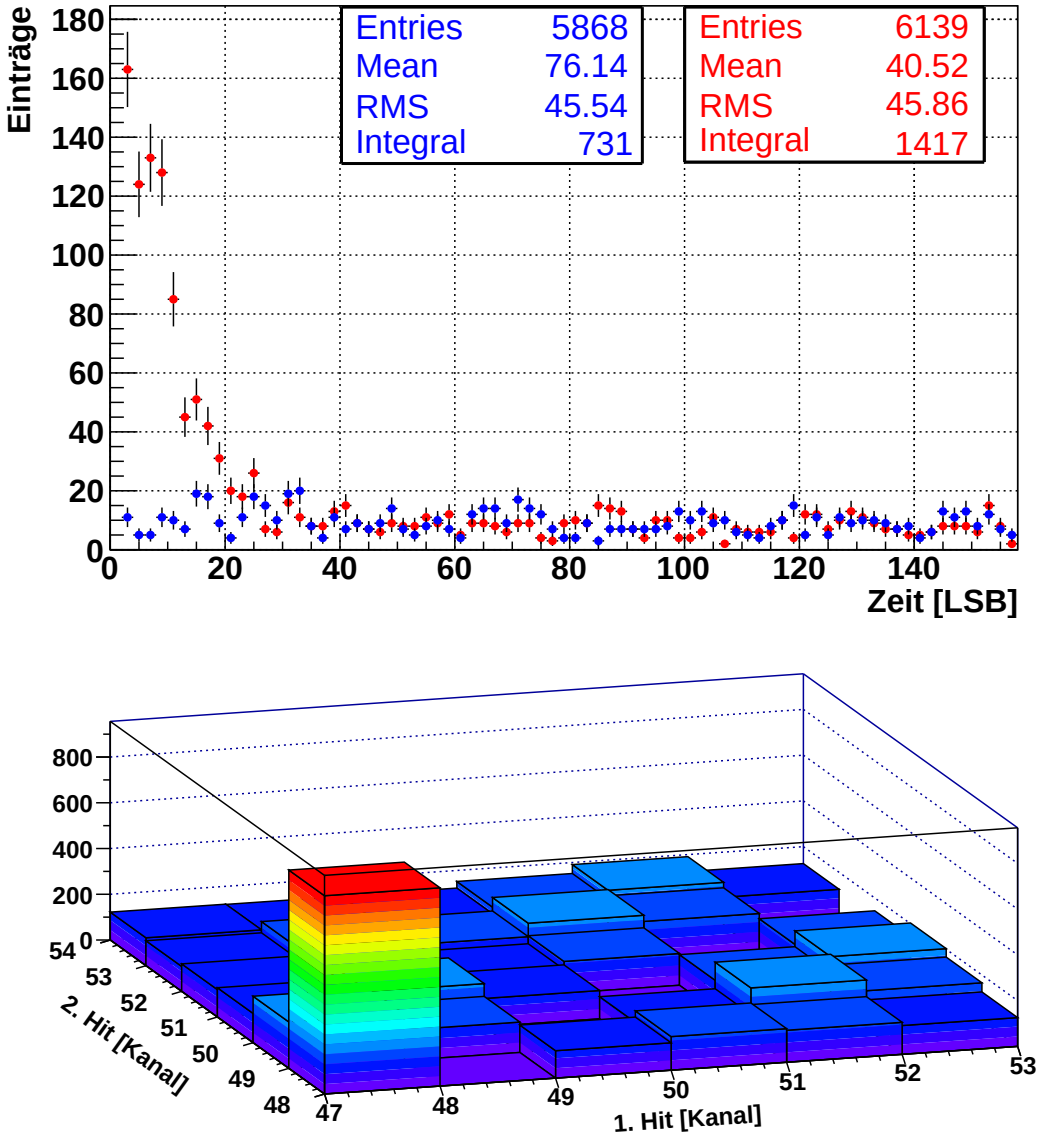

Abbildung 7.20: *Zeitkorrelationen (oben): Aufgetragen ist die Häufigkeit der gemessenen Zeitintervalle zwischen Kanal 47 und Kanal 48 (rot) und zum Vergleich zwischen Kanal 48 und Kanal 49 (blau). Zwischen Kanal 47 und Kanal 48 macht sich das fehlende Peak-Sensing durch eine starke Korrelation der Detektorsignale bemerkbar. Unten ist ein vergrößerter Bereich aus Abbildung 7.19 dargestellt (Y-Ebene, Spilldatum: 03.12.2012 - 01:03 Uhr).*

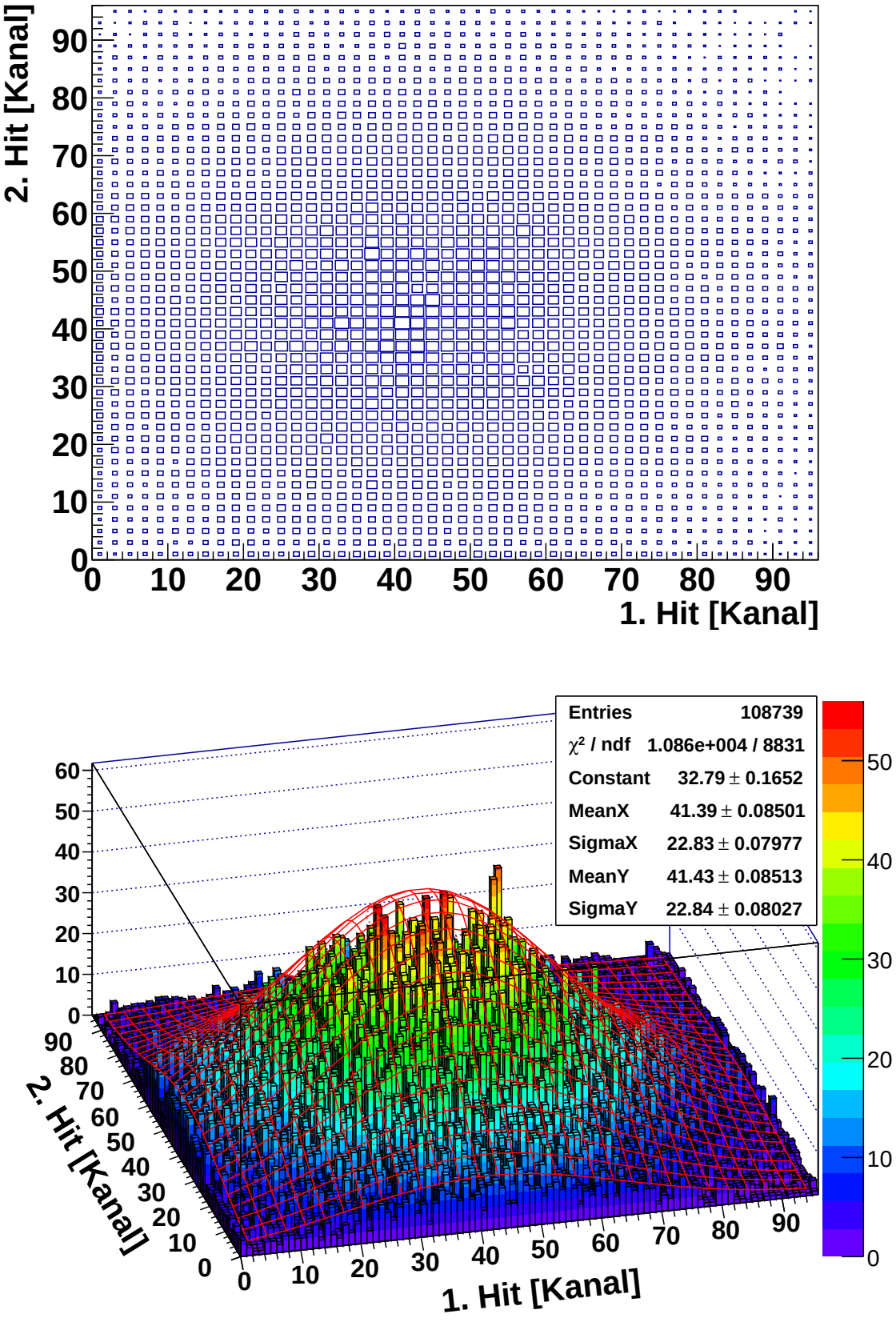

Abbildung 7.21: *Häufigkeitsverteilung der Kanal-Kombinationen für den Nachweis zweier Hits zwischen 100 und 160 TDC-LSB in 2D- und 3D-Darstellung (X-Ebene, Spilldatum: 03.12.2012 - 00:38 Uhr).*

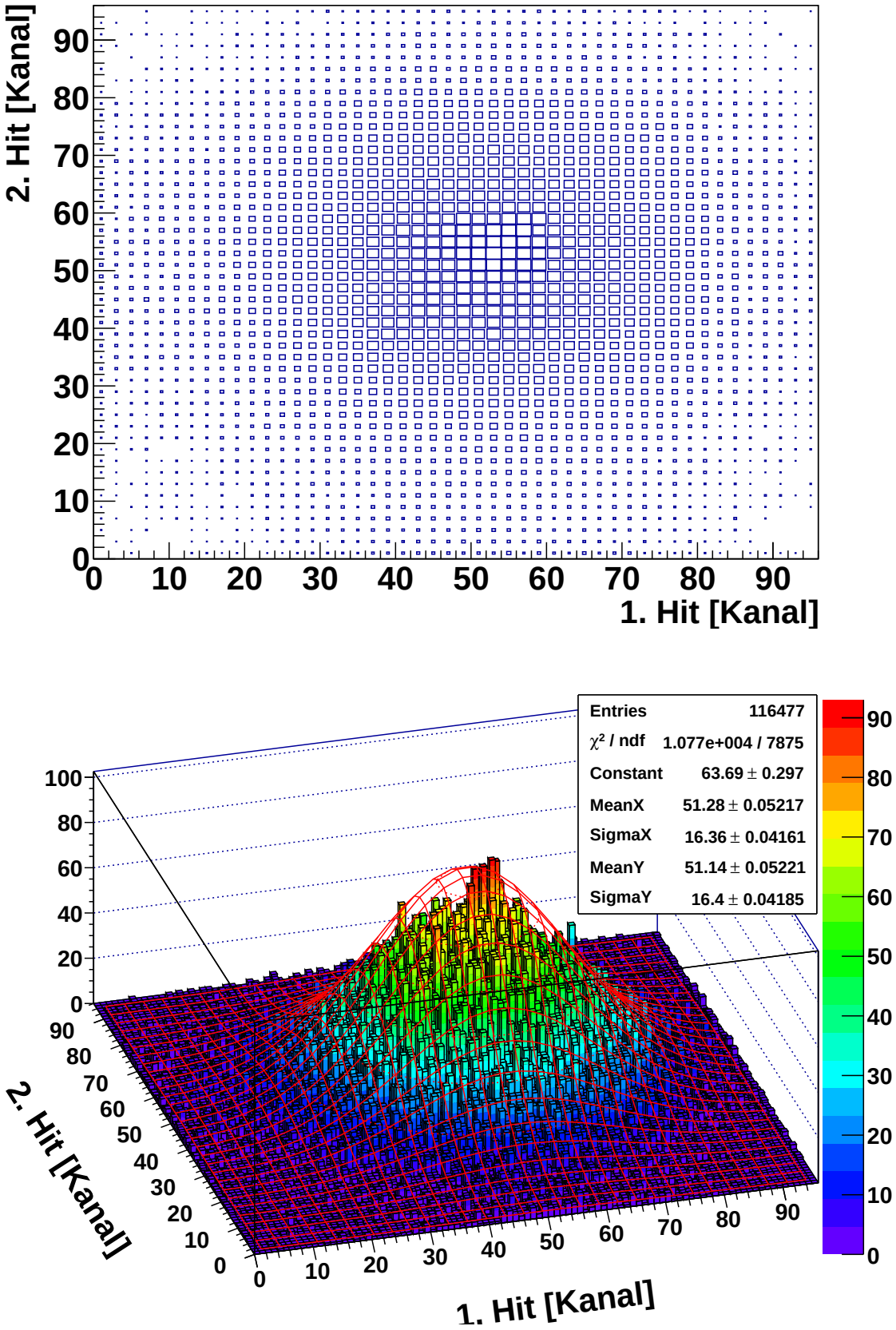

Abbildung 7.22: *Häufigkeitsverteilung der Kanal-Kombinationen für den Nachweis zweier Hits zwischen 100 und 160 TDC-LSB in 2D- und 3D-Darstellung (Y-Ebene, Spilldatum: 03.12.2012 - 01:03 Uhr).*

Eine weitere Ursache für die schlechte Anpassung an die Poissonkurven könnte die oszillierende Strahlrate sein. Dies würde eine Ersetzung der Anpassungsfunktionen w_n durch w_n^{korr}

$$w_n(t-t_0) \quad \longrightarrow \quad w_n^{\text{korr}}(t-t_0) \tag{7.7}$$

$$r\frac{(r(t-t_0))^{n-1}e^{-r(t-t_0)}}{(n-1)!} \quad \longrightarrow \quad \int_{r_{\min}}^{r_{\max}} dr'\phi(r')r'\,\frac{(r'(t-t_0))^{n-1}e^{-r'(t-t_0)}}{(n-1)!} \tag{7.8}$$

mit einer Gewichtungsfunktion ϕ erfordern. Exemplarisch erfolgte ein Schnitt auf die Strahlrate, das heißt, TDC-Hits wurden nur dann für die Auftragung herangezogen, sofern die zugehörige momentane Strahlrate aus den Scaler-Daten in einem bestimmten Intervall liegt. Außerdem wurden alle Hits die zeitlich nacheinander folgten verworfen, sofern sich die zugehörigen Kanalnummern um weniger als drei unterschieden. Hierbei erfolgten exemplarisch Schnitte im Bereich [79 MHz, 81 MHz] und [100 MHz, 105 MHz].

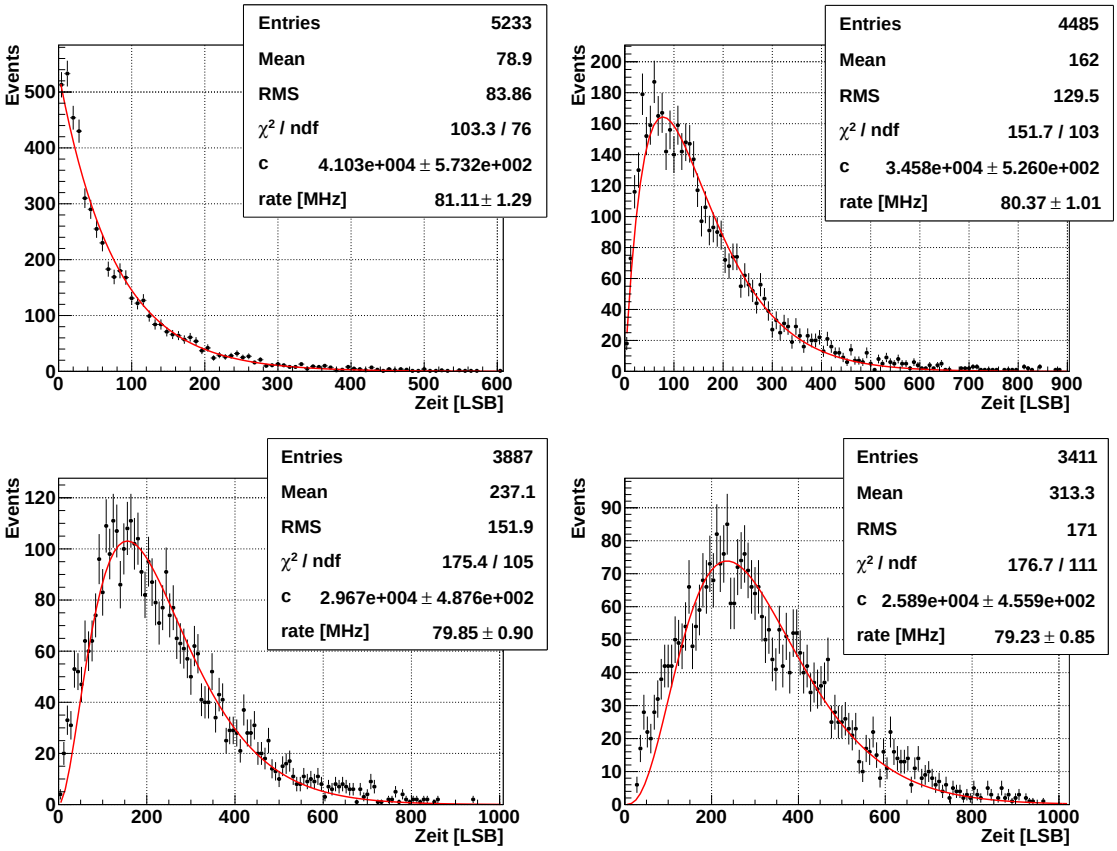

Abbildung 7.23: *Kreuzkorrelationen bis zur vierten Ordnung mit Schnitt auf die Strahlrate [79 MHz, 81 MHz]. Hits die zeitlich nacheinander folgten wurden verworfen, sofern sich die zugehörigen Kanalnummern um weniger als drei unterschieden (Spilldatum: 01.12.2012 - 01:48 Uhr).*

Die Anpassung an die Poissonverteilungen ist nun deutlich besser, wie in Abbildung 7.23 und 7.24 zu erkennen ist. Auch die jeweiligen Parameter „rate“ sind innerhalb ihrer Fehlergrenzen miteinander verträglich. Allerdings müsste für eine

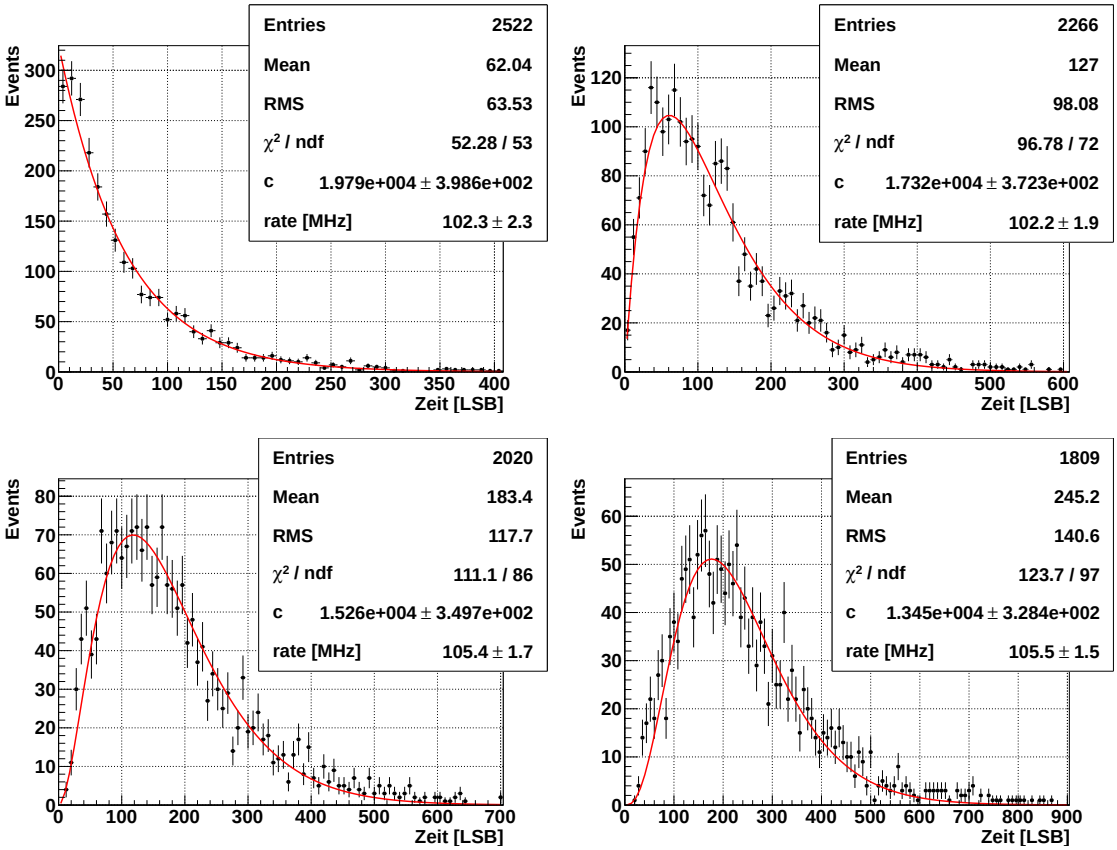

Abbildung 7.24: *Kreuzkorrelationen bis zur vierten Ordnung mit Schnitt auf die (höchste) Strahlrate [100 MHz, 105 MHz]. Hits die zeitlich nacheinander folgten wurden verworfen, sofern sich die zugehörigen Kanalnummern um weniger als drei unterschieden. (Spilldatum: 01.12.2012 - 01:48 Uhr).*

aussagekräftige Beurteilung über die zeitliche Korrelation des Strahls jeder beliebige Schnitt auf die Strahlrate eine Poissonverteilung ergeben. Einer Beurteilung für kleinere Strahlraten kommt erschwerend hinzu, dass das Hit-Fenster des TDC begrenzt ist. Außerdem geht durch die Schnitte sehr viel Statistik verloren. Der Kanalschnitt verfälscht zudem das Ergebnis da auch Hits verworfen werden, die nicht durch Crosstalk hervorgerufen wurden.

8. Zusammenfassung

Ziel dieser Arbeit war die Entwicklung eines voll automatisierten Monitoring-Systems zur Visualisierung verschiedener Strahleigenschaften am COMPASS-II Experiment in Echtzeit. Dafür wurde eine FPGA-Firmware erstellt, welche auf dem GANDALF-Modul in einem Xilinx Virtex-5 FPGA zum Einsatz kommt. Das GANDALF-Modul ist ein VXS/VME64x-Modul, welches zur Echtzeitanalyse und Digitalisierung hochfrequenter Signale geeignet ist.

Die Daten werden über die Auslese des FI02-Detektors gewonnen. Dabei handelt es sich um szintillierende Fasern vor dem Target, die als Tracking-Detektoren für Strahlteilchen dienen.

Für Ratenmessungen des Strahls wird ein Scaler mit totzeitfreier Auslese für 96 Kanäle benötigt. Dies kann mit dem Konzept eines Zählers und eines Addierers erreicht werden. Der Zähler ist ein Johnson-Ringzähler, welcher aufgrund seiner einfachen Logik eine sehr geringe Rechenzeit benötigt, um bei einer Signalflanke den nächsten Zustand zu bestimmen. Außerdem wird bei einem Zustandsübergang aufgrund der konstanten Hammingdistanz von 1 eine fehlerfreie Auslese ± 1 garantiert. Der Addierer ist mit $38,88\,\mathrm{MHz}$ getaktet und addiert einem aktuellen Zählerstand die Differenz zweier ausgelesenen Zustände des Johnson-Ringzähler auf. Ein TCS-Event ist nach etwa $0,25\,\mu\mathrm{s}$ abgearbeitet, was unterhalb der geforderten miminalen Totzeit des TCS-Triggers von $0,4\,\mu\mathrm{s}$ liegt. Über den Konfigurationsspeicher lassen sich außerdem Triggerverzögerungen bis $13,2\,\mu\mathrm{s}$ und Gateverzögerungen bis $6,6\,\mu\mathrm{s}$ einstellen. Die Funktionsweise des Scalers konnte in einer Labormessung bis zu einer Eingangsfrequenz von $505\,\mathrm{MHz}$ mit einem Auslesefehler von ± 1 verifiziert werden.

Für die Strahlanalyse wird ein Pseudo-Randomtrigger auf VHDL-Basis erstellt. Dieser basiert auf mehreren rückgekoppelten Schieberegistern über deren Rückkopplungsbit ein zufälliges Signal generiert wird. Die Triggerrate lässt sich steuern, indem die Anzahl der in die Triggerbedingung eingehenden Rückkopplungsbits verändert wird. Die zeitliche Zufälligkeit des Triggersignals konnte mit einer Simulation bis zur sechsten Ordnung bestätigt werden. Die Generierung von Randomtriggern wird während einer künstlichen Onspill-Zeit aktiviert, wofür die momentane Rate eines

Scalerkanals, der mit die höchste Rate der szintillierenden Fasern aufweist, bestimmt wird. Dadurch können Randomtrigger unabhängig vom TCS generiert werden.

Die FPGA-Firmware wurde um einen 96-Kanal TDC erweitert, der auf dem Shifted Clock Sampling Konzept unter Verwendung von 16 phasenverschobenen Taktsignalen basiert [49]. Dieser gibt Zeitmarken mit einer Digitalisierungsbreite von 161 ps aus. Sowohl Scaler als auch TDC konnten auf einem GANDALF-Modul in das COMPASS-Datennahme-System integriert werden. Unabhängig davon werden die Daten für das Monitoring-System über eine VME-Backplane ausgelesen. Für eine korrekte Zuordnung der Daten zu den verschiedenen Triggertypen wird ein Fifo verwendet, der die Reihenfolge der Trigger speichert.

Ein Analysetool verarbeitet die Pseudo-Random-Daten des Scalers und des TDC und generiert verschiedene Graphen und Histogramme. Für die Visualisierung wurde eine grafische Benutzeroberfläche programmiert worauf die Diagramme nach jedem Spill aktualisiert werden. Die Informationen werden auch im CERN Intranet veröffentlicht, sodass Mitarbeiter des SPS Beschleunigers diese zur Strahloptimierung nutzen können. Erste Messungen am COMPASS-II Experiment konnten eine starke 50 Hz-Oszillation in der Strahlrate aufdecken. Außerdem konnte eine Veränderung des Strahlschwerpunkts in der horizontalen Ebene des FI02-Detektors im Verlauf eines Spills um etwa einen Kanal festgestellt werden. Die Kreuzkorrelationen der Myonsignale zeigen im Allgemeinen keine gute Anpassung an die Poissonkurven w_n. Ein Übersprechen benachbarter SciFi-Kanäle und vor allem die oszillierende Strahlrate könnten diese Systematik erklären.

A. Datenformat der Pseudo-Random-Daten

Im Folgenden wird das Datenformat der Pseudo-Randomtrigger-Daten erklärt. Pro Trigger wird ein Scaler-Datenblock (siehe Tabelle A.2) und ein TDC-Datenblock (siehe Tabelle A.3) erzeugt und über die VME-Backplane ausgelesen. Nach jedem Spill wird ein End-of-Spill-Marker erzeugt und ausgelesen (siehe Tabelle A.4). In Tabelle A.1 ist die Grundstruktur der Daten dargestellt.

Tabelle A.1: *Die Grundstruktur des Datenformats.*

Scaler-Daten (1. Event)
TDC-Daten (1. Event)
Scaler-Daten (2. Event)
TDC-Daten (2. Event)
⋮
Scaler-Daten (letztes Event im Spill)
TDC-Daten (letztes Event im Spill)
End-of-Spill-Marker

Tabelle A.2: *Scaler-Datenformat für Pseudo-Randomtrigger-Daten (32 Bit). Mit (∗) markierte Signale kommen nicht vom TCS, sondern werden intern generiert.*

Datenwort	**Beschreibung**
0x00000000	Kontrollwort, leitet die Auslese ein
0x7aaaaaaa	Zustand „st_scdata“
’0’ & spillNr* (11) & eventNr* (20)	Eventheader
’00’ & geoID1 (10) & eventNr* (20)	Header (Block 1)
’1’ & scalervalue (31)	Zählerstand des 0. Kanals
⋮	⋮
’1’ & scalervalue (31)	Zählerstand des 31. Kanals
’1’ & time(14 → 0) & pattern(15 → 0)	Zeit- und Hitinformation (1)
’1’ & time(30 → 15) & pattern(31 → 16)	Zeit- und Hitinformation (2)
’01’ & geoID1 (10) & eventNr* (20)	Trailer (Block 1)
’00’ & geoID2 (10) & eventNr* (20)	Header (Block 2)
’1’& scalervalue (31)	Zählerstand des 32. Kanals
⋮	⋮
’1’ & scalervalue (31)	Zählerstand des 63. Kanals
’1’ & time(14 → 0) & pattern(47 → 32)	Zeit- und Hitinformation (1)
’1’ & time(30 → 15) & pattern(63 → 48)	Zeit- und Hitinformation (2)
’01’ & geoID2 (10) & eventNr* (20)	Trailer (Block 2)
’00’ & geoID3 (10) & eventNr* (20)	Header (Block 3)
’1’ & scalervalue (31)	Zählerstand des 64. Kanals
⋮	⋮
’1’ & scalervalue (31)	Zählerstand des 95. Kanals
’1’ & time(14 → 0) & pattern(79 → 64)	Zeit- und Hitinformation (1)
’1’ & time(30 → 15) & pattern(95 → 80)	Zeit- und Hitinformation (2)
’01’ & geoID3 (10) & eventNr* (20)	Trailer (Block 3)
0x7bbbbbbb	Zustand „st_scdata_done“
0xcfed1200	Kontrollwort, beendet die Auslese

Tabelle A.3: *TDC-Datenformat für Pseudo-Randomtrigger-Daten (32 Bit). Mit (∗) markierte Signale kommen nicht vom TCS, sondern werden intern generiert.*

Datenwort	**Beschreibung**
0x00000000	Kontrollwort, leitet die Auslese ein
0x7ccccccc	Zustand „st_tdcdata“
'00' & eventNr* (6) & trigger time (9) & Ch	Header (Block 0)
'10' & channelID & Data (16) & Port	1. TDC Datenwort
⋮	⋮
'10' & channelID & Data (16) & Port	Letztes TDC Datenwort
'00' & eventNr* (6) & trigger time (9) & Ch	Trailer (Block 0)
'00' & eventNr* (6) & trigger time (9) & Ch	Header (Block 1)
'10' & channelID & Data (16) & Port	1. TDC Datenwort
⋮	⋮
'10' & channelID & Data (16) & Port	Letztes TDC Datenwort
'00' & eventNr* (6) & trigger time (9) & Ch	Trailer (Block 1)
⋮	⋮
'00' & eventNr* (6) & trigger time (9) & Ch	Header (Block 11)
'10' & channelID & Data (16) & Port	1. TDC Datenwort
⋮	⋮
'10' & channelID & Data (16) & Port	Letztes TDC Datenwort
'00' & eventNr* (6) & trigger time (9) & Ch	Trailer (Block 11)
0x7ddddddd	Zustand „st_tdcdata_done“
0xcfed1200	Kontrollwort, beendet die Auslese

Tabelle A.4: *Nach jedem Spill wird ein End-of-Spill-Marker in den Spy-Fifo geschrieben, womit die Rohdaten-Datei abgeschlossen wird.*

Datenwort	**Beschreibung**
0x00000000	Kontrollwort, leitet die Auslese ein
0x7eeeeeee	Zustand „st_spill_done“
0xcfed1200	Kontrollwort, beendet die Auslese

B. Datenformat der TCS-Daten

Im Folgenden wird das TCS-Datenformat erläutert. Für eine detailliertere Beschreibung sei auf [64] verwiesen.

31																											0

(SLink begin marker)

0x00000000

(SMUX header)

err(1)	ev. type(5)	SMUX source ID(10)	sum of event sizes +2(16)
stat(1)	spill no.(11)	event no.(20)	

(SLink header)

err(1)	ev. type(5)	TDC source ID(10)	TDC event size(16)
stat(1)	spill no.(11)	event no.(20)	
format(8)	#errorwords(8)	tcs error(8)	status(8)

(12x blocks)

00	event no.(6)	trigger time(9)	xor	ch/ch. ID(6)	port(4)	PLL locked
10	ch/ch. ID(6)	TDC-data(16)			port(4)	PLL locked

...

10	ch/ch. ID(6)	TDC-data(16)			port(4)	PLL locked
00	event no.(6)	trigger time(9)	xor	ch/ch. ID(6)	port(4)	PLL locked

(SLink Header)

err(1)	ev. type(5)	Scaler source ID(10)	Scaler event size(16)
stat(1)	spill no.(11)	event no.(20)	
format(8)	#errorwords(8)	tcs error(8)	status(8)

(3x blocks)

00	geo ID(10)	event no.(20)
1	Scaler value channel 0(31)	

...

1	Scaler value channel 31(31)	
1	time bit 14 . . . 0(15)	hit in scaler 15 . . . 0(16)
1	time bit 29 . . . 15(15)	hit in scaler 31 . . . 16(16)
01	geo ID(10)	event no.(20)

(SLink end marker)

0xcfed1200

C. Setzen der Parameter für die TDC-Scaler-Firmware

C.1 TDC- und Scaler-Hex-ID

VME-Adresse 0xe0{GANDALF Hex-ID}2804 [32 Bit]:

- SMUX-Source-ID (29 → 20)
- Scaler-Source-ID (19 → 10)
- TDC-Source-ID (9 → 0)

C.2 TDC

VME-Adresse 0xe0{GANDALF Hex-ID}2ba0 [32 Bit]:

- Triggerverzögerung (15 → 0) [TDC-LSB]

VME-Adresse 0xe0{GANDALF Hex-ID}2ba4 [32 Bit]:

- Trigger-Window (15 → 0) [TDC-LSB]

C.3 Scaler und Pseudo-Randomtrigger

VME-Adresse 0xe0{GANDALF Hex-ID}2b60 [32 Bit]:

- Triggerverzögerung (31 → 23) [25, 72 ns]
- Gateverzögerung (22 → 15) [25, 72 ns]

- Minimale Rate für die Begin- und End-Of-Spill-Erzeugung[1] ($5 \rightarrow 4$)
- Pseudo-Randomtrigger-Modus[2] (3)
- Frequenz des Pseudo-Randomtriggers[3] ($2 \rightarrow 0$)

[1] Überschreitet die Eingangsrate auf Kanal 45 einen einstellbaren Wert ν_{min}, so wird ein künstliches Begin-Of-Spill-Signal erzeugt ($\nu_{min} = 1,24\,\text{kHz}$ (00), $\nu_{min} = 9,95\,\text{kHz}$ (01), $\nu_{min} = 79,63\,\text{kHz}$ (10), $\nu_{min} = 1,27\,\text{MHz}$ (11)).

[2] Bei '1' wird das Pseudo-Randomtrigger-Signal im FPGA direkt mit dem künstlichen End-Of-Spill-Signal verbunden, sodass nur einmal am Ende eines Spills getriggert wird. '0' ist die Standardeinstellung.

[3] Einstellungen sind von $n = 0$ (000) bis $n = 7$ (111) wählbar. Für $n = 0$ sind die Pseudo-Randomtrigger deaktiviert. Für alle anderen Fälle beträgt die Frequenz $\nu = 0,15 \cdot 2^{n-1}\,\text{kHz}$.

D. VHDL-Code Beispiele

D.1 1-Kanal Scaler-Design

Listing D.1: *VHDL-Code: Toplevel*

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  entity topmodul is
4      port (
5                      bos : in std_logic;
6                      eos : in std_logic;
7                      bos_k : in std_logic;
8                      eos_k : in std_logic;
9                      inscaler : in  std_logic;
10                     clk_38mhz : in std_logic;
11                     random_trigger : in std_logic;
12                     gate : in std_logic;
13                     data31_out_tcs : out std_logic_vector (30 downto 0);
14                     data31_out_r : out std_logic_vector (30 downto 0);
15                     hit_pattern : out std_logic;
16                     q1 : out std_logic_vector (3 downto 0);
17                     q_neu : in std_logic_vector (3 downto 0);
18                     reset : in std_logic;
19                     gate_delayed : in std_logic
20                          );
21 end topmodul;
22 architecture behavioral of topmodul is
23         component johnson_counter
24         port(
25                 inscaler : in std_logic;
26                 eos_bos : in std_logic;
27                 q : out std_logic_vector(7 downto 0)
28                 );
29         end component;
30         component reg_johnson
31         port(
32                 reset : in std_logic;
33                 clk_38mhz : in std_logic;
```

```
34                d_in : in std_logic_vector(7 downto 0);
35                q_out : out std_logic_vector(7 downto 0)
36                );
37        end component;
38        component johnson_to_binary
39        port(
40                q_undecoded : in std_logic_vector(7 downto 0);
41                q_decoded : out std_logic_vector(3 downto 0)
42                );
43        end component;
44        component reg
45        port(
46                reset : in std_logic;
47                clk_38mhz : in std_logic;
48                d_in : in std_logic_vector(3 downto 0);
49                q_out : out std_logic_vector(3 downto 0)
50                );
51        end component;
52        component diff
53        port(
54                zaehlerstand_gross : in std_logic_vector(3 downto 0);
55                zaehlerstand_klein : in std_logic_vector(3 downto 0);
56                differenz : out std_logic_vector(3 downto 0);
57                hit_pattern : out std_logic;
58                clk_38mhz : in std_logic
59                );
60        end component;
61        component adder_ohne_reg
62        port(
63                reset : in std_logic;
64                clk_38mhz : in std_logic;
65                gate : in std_logic;
66                a_b : in std_logic_vector(3 downto 0);
67                erg : inout std_logic_vector(30 downto 0)
68                );
69        end component;
70        component adder
71        port(
72                clk_38mhz : in std_logic;
73                a_b : in std_logic_vector(3 downto 0);
74                reset : in std_logic;
75                gate : in std_logic;
76                reg8_in : in std_logic_vector(7 downto 0);
77                reg4_in : in std_logic_vector(3 downto 0);
78                erg : inout std_logic_vector(30 downto 0);
79                reg8_out : out std_logic_vector(7 downto 0);
80                reg4_out : out std_logic_vector(3 downto 0)
81                );
82        end component;
83        component register31
84        port(
85                clk_38mhz : in std_logic;
86                reset : in std_logic;
87                inscaler : in std_logic;
88                data31_in : in std_logic_vector(30 downto 0);
89                data31_out : out std_logic_vector(30 downto 0)
90                );
```

```
91          end component;
92          component adder_as_31reg
93          port(
94                  trigger : in std_logic;
95                  clk     : in std_logic;
96                  a_in : in std_logic_vector(0 downto 0);
97                  b_in : in std_logic_vector(30 downto 0);
98                  reset : in std_logic;
99                  erg : out std_logic_vector(30 downto 0)
100                 );
101         end component;
102 signal q1_s : std_logic_vector (3 downto 0) := (others =>'0');
103 signal qout : std_logic_vector (3 downto 0) := (others =>'0');
104 signal qneu_s : std_logic_vector (3 downto 0) := (others =>'0');
105 signal qalt : std_logic_vector (3 downto 0) := (others =>'0');
106 signal differenz : std_logic_vector (3 downto 0) := (others =>'0');
107 signal value_tcs : std_logic_vector (30 downto 0) := (others =>'0');
108 signal value_r : std_logic_vector (30 downto 0) := (others =>'0');
109 signal reset_sig : std_logic := '0';
110 signal to_gate : std_logic := '0';
111 signal to_decode : std_logic_vector (3 downto 0);
112 signal qoutjohnson : std_logic_vector (7 downto 0):= x"00";
113 signal qoutjohnson2 : std_logic_vector (7 downto 0):= x"00";
114 signal eos_bos : std_logic;
115 signal eos_bos_k : std_logic;
116 begin
117 eos_bos <= eos or bos;
118 eos_bos_k <= eos_k or bos_k;
119 reset_sig <= reset;
120 to_gate <= gate_delayed;
121         johnsoncounter: johnson_counter port map(
122                 inscaler => inscaler,
123                 eos_bos => eos_bos,
124                 q => qoutjohnson
125         );
126         joh_bin_converter: johnson_to_binary port map(
127                 q_undecoded => qoutjohnson2,
128                 q_decoded => q1_s
129         );
130         q1 <= q1_s;
131         qneu_s <= q_neu;
132         differenz4bit: diff port map(
133                 zaehlerstand_gross => qneu_s,
134                 zaehlerstand_klein => qalt,
135                 differenz => differenz,
136                 hit_pattern => hit_pattern,
137                 clk_38mhz => clk_38mhz
138         );
139         adder_with_gate: adder port map(
140                 clk_38mhz => clk_38mhz,
141                 a_b => differenz,
142                 reset => reset_sig,
143                 gate => to_gate,
144                 erg => value_tcs,
145                 reg8_in => x"00",
146                 reg8_out => open,
147                 reg4_in => x"0",
```

```
148                 reg4_out => open
149         );
150         adder_without_gate: adder port map(
151                 clk_38mhz => clk_38mhz,
152                 a_b => differenz,
153                 reset => eos_bos_k,
154                 gate => '0',
155                 erg => value_r,
156                 reg8_in => qoutjohnson,
157                 reg8_out => qoutjohnson2,
158                 reg4_in => qneu_s,
159                 reg4_out => qalt
160         );
161         dsp31reg_tcs: adder_as_31reg port map(
162                 trigger => inscaler,
163                 clk => clk_38mhz,
164                 a_in => "0",
165                 b_in => value_tcs,
166                 reset => reset_sig,
167                 erg => data31_out_tcs
168         );
169         dsp31reg_random: adder_as_31reg port map(
170                 trigger => random_trigger,
171                 clk => clk_38mhz,
172                 a_in => "0",
173                 b_in => value_r,
174                 reset => eos_bos_k,
175                 erg => data31_out_r
176         );
177 end behavioral;
```

D.2 Komponenten

D.2.1 Johnson-Counter

Listing D.2: *VHDL-Code: Johnson-Counter*

```
1  entity johnson_counter is
2          port(
3                  inscaler: in std_logic;
4                  eos_bos : in std_logic;
5                  q : out std_logic_vector(7 downto 0)
6                          );
7  end johnson_counter;
8  architecture behavioral of johnson_counter is
9  signal q_out: std_logic_vector (7 downto 0) := x"00";
10 begin
11 q <= q_out;
12 count:process(inscaler, eos_bos)
13 begin
14         if eos_bos = '1' then    q_out <= x"00";
15         elsif rising_edge(inscaler) then
16         q_out <= (not q_out (0)) & q_out (7 downto 1);
17         end if;
18 end process;
19 end behavioral;
```

D.2.2 Dekoder

Listing D.3: *VHDL-Code: Dekoder*

```
entity johnson_to_binary is
    port (
                q_undecoded : in  std_logic_vector (7 downto 0);
                q_decoded : out  std_logic_vector (3 downto 0)
                   );
end johnson_to_binary;
architecture behavioral of johnson_to_binary is
signal to_decode : std_logic_vector (7 downto 0):= x"00";
begin
to_decode <= q_undecoded;
with to_decode select
                q_decoded <=
                        x"0" when "00000000",
                        x"1" when "10000000",
                        x"2" when "11000000",
                        x"3" when "11100000",
                        x"4" when "11110000",
                        x"5" when "11111000",
                        x"6" when "11111100",
                        x"7" when "11111110",
                        x"8" when "11111111",
                        x"9" when "01111111",
                        x"a" when "00111111",
                        x"b" when "00011111",
                        x"c" when "00001111",
                        x"d" when "00000111",
                        x"e" when "00000011",
                        x"f" when "00000001",
                        x"0" when others;
end behavioral;
```

D.2.3 Subtrahierer

Listing D.4: *VHDL-Code: Subtrahierer*

```
entity diff is
    port (
         zaehlerstand_gross : in  std_logic_vector (3 downto 0);
         zaehlerstand_klein : in  std_logic_vector (3 downto 0);
         differenz : out  std_logic_vector (3 downto 0);
         hit_pattern : out std_logic;
         clk_38mhz: in std_logic
                   );
end diff;
architecture behavioral of diff is
signal komplement_2er : std_logic_vector (4 downto 0):= '0' & x"0";
signal ergebnis : std_logic_vector (4 downto 0) := '0' & x"0";
signal hit_pattern_sig : std_logic := '0';
begin
hit_pattern_sig <= '0' when (ergebnis (3 downto 0) = x"0") else '1';
hit_pattern <= hit_pattern_sig;
komplement_2er <= std_logic_vector(to_unsigned(to_integer(unsigned(not
    zaehlerstand_klein)+1),5));
```

```
18 ergebnis <= std_logic_vector(to_unsigned(to_integer(unsigned(
       komplement_2er) + unsigned(zaehlerstand_gross)),5));
19 differenz <= ergebnis (3 downto 0);
20 end behavioral;
```

D.2.4 Addierer

Listing D.5: *VHDL-Code: Addierer*

```
1  entity adder is
2      port (
3                  clk_38mhz : in  std_logic;
4                  a_b : in  std_logic_vector (3 downto 0);
5                  reset : in std_logic;
6                  gate : in std_logic;
7                  erg : inout  std_logic_vector (30 downto 0);
8                  reg8_in : in std_logic_vector (7 downto 0);
9                  reg8_out : out std_logic_vector (7 downto 0);
10                 reg4_in : in std_logic_vector (3 downto 0);
11                 reg4_out : out std_logic_vector (3 downto 0)
12                    );
13 end adder;
14 architecture behavioral of adder is
15 component adder_dsp_z4_31
16   port (
17     a : in std_logic_vector(3 downto 0);
18     b : in std_logic_vector(43 downto 0);
19     clk : in std_logic;
20     ce : in std_logic;
21     sclr : in std_logic;
22     s : out std_logic_vector(43 downto 0)
23   );
24 end component;
25 signal a : std_logic_vector (3 downto 0) := x"0";
26 signal b : std_logic_vector (43 downto 0) := x"00000000000";
27 signal inv_gate : std_logic := '1';
28 signal res : std_logic := '0';
29 signal temp : std_logic_vector (43 downto 0) := x"00000000000";
30 begin
31 a <= a_b;
32 b <= reg8_in & reg4_in & '0' & erg;
33 res <= reset;
34 inv_gate <= not gate;
35 erg <= temp (30 downto 0);
36 reg8_out <= temp (43 downto 36);
37 reg4_out <= temp (35 downto 32);
38 your_instance_name : adder_dsp_z4_31
39   port map (
40     a => a,
41     b => b,
42     clk => clk_38mhz,
43     ce => inv_gate,
44     sclr => res,
45     s => temp
46   );
47 end behavioral;
```

D.2.5 31 Bit-Register

Listing D.6: *VHDL-Code: 31 Bit-Register (TCS-Trigger & Randomtrigger)*

```
entity adder_as_31reg is
    port (
                trigger : in  std_logic;
                clk : in std_logic;
                a_in : in std_logic_vector (0 downto 0);
                b_in : in std_logic_vector (30 downto 0);
                reset : in std_logic;
                erg : out  std_logic_vector (30 downto 0)
        );
end adder_as_31reg;
architecture behavioral of adder_as_31reg is
  component adder_dsp_z
  port (
    a : in std_logic_vector(0 downto 0);
    b : in std_logic_vector(30 downto 0);
    clk : in std_logic;
    ce : in std_logic;
    sclr : in std_logic;
    s : out std_logic_vector(30 downto 0)
  );
end component;
begin
your_instance_name : adder_dsp_z
  port map (
    a => a_in,
    b => b_in,
    clk => clk,
    ce => trigger,
    sclr => reset,
    s => erg
  );
end behavioral;
```

D.3 Pseudo-Randomtrigger

Listing D.7: *VHDL-Code: Pseudo-Randomtrigger*

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity randomtrigger is
  port (
        clk     :       in std_logic;
        bos     :       in std_logic;
        eos     :       in std_logic;
        freq    :       in std_logic_vector (2 downto 0);
        random  :       out std_logic
);
end randomtrigger;
architecture behavioral of randomtrigger is
type random31_arr is array (1 to 18) of std_logic_vector (1 to 31);
  signal random31 : random31_arr :=
        (
```

```
17          b"010"& x"bcdef01",
18          b"010"& x"8472830",
19          b"101"& x"cef8321",
20          b"011"& x"948bc63",
21          b"110"& x"037ef72",
22          b"000"& x"67838bc",
23          b"011"& x"73840ac",
24          b"101"& x"b472839",
25          b"110"& x"f930193",
26          b"000"& x"0578493",
27          b"011"& x"b4738c5",
28          b"101"& x"83920bf",
29          b"110"& x"8394038",
30          b"000"& x"0cd3839",
31          b"011"& x"383920c",
32          b"101"& x"94028bc",
33          b"110"& x"9cdef78",
34          b"000"& x"cf47389"
35          );
36  signal int_deadtime : integer range 0 to 40000 := 0;
37  signal trg_state : std_logic_vector (2 downto 0) := "000";
38  signal n : integer range 0 to 31;
39  signal sig_random: std_logic := '0';
40  signal trg : std_logic_vector (31 downto 0):= x"00000000";
41  attribute equivalent_register_removal : string;
42  attribute equivalent_register_removal of random31 : signal is "no";
43  attribute equivalent_register_removal of trg : signal is "no";
44  attribute keep : string;
45  attribute keep of random31: signal is "true";
46  attribute keep of trg: signal is "true";
47  begin
48  random <= sig_random;
49  proc_deadtime: process(clk)
50  begin
51  if rising_edge(clk) then
52          sig_random <= '0';
53          case trg_state is
54          when "000" =>
55                          if bos = '1' then trg_state <= "001";
56                          end if;
57                          n <= 19 - to_integer(unsigned(freq));
58          when "001" =>
59                          if int_deadtime = 40000 then
60                          int_deadtime <= 0;
61                          trg_state <= "010";
62                          else    int_deadtime <= int_deadtime + 1;
63                          end if;
64          when "010" =>
65                          if eos = '1' then
66                          trg_state <= "101";
67                          else
68                                  if  trg(n) = '1' then
69                                  trg_state <= "011";
70                                  end if;
71                          end if;
72          when "011" =>
73                          if eos = '1' then
```

```
74                          trg_state <= "101";
75                          else
76                          sig_random <= '1';
77                          trg_state <= "100";
78                          end if;
79          when "100" =>
80                          sig_random <= '0';
81                          if eos = '1' then trg_state <= "101";
82                          elsif   int_deadtime = 40 then
83                          int_deadtime <= 0;
84                          trg_state <= "010";
85                          else int_deadtime <= int_deadtime + 1;
86                          end if;
87          when "101" =>
88                          int_deadtime <= 0;
89                          trg_state <= "110";
90          when others =>
91                          trg_state <= "000";
92                          sig_random <= '0';
93          end case;
94  end if;
95  end process;
96  process(clk)
97  begin
98  if rising_edge(clk) then
99          for i in 1 to 18 loop
100                 random31(i) <= (random31(i)(31)xor random31(i)(28))
101                 & random31(i)(1 to 30);
102         end loop;
103 end if;
104 end process;
105 trg(18) <= '1' when
106         (random31(1)(1) = '1' and random31(2)(1) = '1' and
107          random31(3)(1) = '1' and random31(4)(1) = '1' and
108          random31(5)(1) = '1' and random31(6)(1) = '1' and
109          random31(7)(1) = '1' and random31(8)(1) = '1' and
110          random31(9)(1) = '1' and random31(10)(1) = '1' and
111          random31(11)(1) = '1' and random31(12)(1) = '1' and
112          random31(13)(1) = '1' and random31(14)(1) = '1' and
113          random31(15)(1) = '1' and random31(16)(1) = '1' and
114          random31(17)(1) = '1' and random31(18)(1) = '1'
115          )
116          else '0';
117 trg(17) <= '1' when
118         (random31(1)(1) = '1' and random31(2)(1) = '1' and
119          random31(3)(1) = '1' and random31(4)(1) = '1' and
120          random31(5)(1) = '1' and random31(6)(1) = '1' and
121          random31(7)(1) = '1' and random31(8)(1) = '1' and
122          random31(9)(1) = '1' and random31(10)(1) = '1' and
123          random31(11)(1) = '1' and random31(12)(1) = '1' and
124          random31(13)(1) = '1' and random31(14)(1) = '1' and
125          random31(15)(1) = '1' and random31(16)(1) = '1' and
126          random31(17)(1) = '1'
127          )
128          else '0';
129 trg(16) <= '1' when
130         (random31(1)(1) = '1' and random31(2)(1) = '1' and
```

```
131             random31(3)(1) = '1' and random31(4)(1) = '1' and
132             random31(5)(1) = '1' and random31(6)(1) = '1' and
133             random31(7)(1) = '1' and random31(8)(1) = '1' and
134             random31(9)(1) = '1' and random31(10)(1) = '1' and
135             random31(11)(1) = '1' and random31(12)(1) = '1' and
136             random31(13)(1) = '1' and random31(14)(1) = '1' and
137             random31(15)(1) = '1' and random31(16)(1) = '1'
138             )
139             else '0';
140     trg(15) <= '1' when
141             (random31(1)(1) = '1' and random31(2)(1) = '1' and
142              random31(3)(1) = '1' and random31(4)(1) = '1' and
143              random31(5)(1) = '1' and random31(6)(1) = '1' and
144              random31(7)(1) = '1' and random31(8)(1) = '1' and
145              random31(9)(1) = '1' and random31(10)(1) = '1' and
146              random31(11)(1) = '1' and random31(12)(1) = '1' and
147              random31(13)(1) = '1' and random31(14)(1) = '1' and
148              random31(15)(1) = '1'
149              )
150              else '0';
151     trg(14) <= '1' when
152             (random31(1)(1) = '1' and random31(2)(1) = '1' and
153              random31(3)(1) = '1' and random31(4)(1) = '1' and
154              random31(5)(1) = '1' and random31(6)(1) = '1' and
155              random31(7)(1) = '1' and random31(8)(1) = '1' and
156              random31(9)(1) = '1' and random31(10)(1) = '1' and
157              random31(11)(1) = '1' and random31(12)(1) = '1' and
158              random31(13)(1) = '1' and random31(14)(1) = '1'
159              )
160              else '0';
161     trg(13) <= '1' when
162             (random31(1)(1) = '1' and random31(2)(1) = '1' and
163              random31(3)(1) = '1' and random31(4)(1) = '1' and
164              random31(5)(1) = '1' and random31(6)(1) = '1' and
165              random31(7)(1) = '1' and random31(8)(1) = '1' and
166              random31(9)(1) = '1' and random31(10)(1) = '1' and
167              random31(11)(1) = '1' and random31(12)(1) = '1' and
168              random31(13)(1) = '1'
169              )
170              else '0';
171     trg(12) <= '1' when
172             (random31(1)(1) = '1' and random31(2)(1) = '1' and
173              random31(3)(1) = '1' and random31(4)(1) = '1' and
174              random31(5)(1) = '1' and random31(6)(1) = '1' and
175              random31(7)(1) = '1' and random31(8)(1) = '1' and
176              random31(9)(1) = '1' and random31(10)(1) = '1' and
177              random31(11)(1) = '1' and random31(12)(1) = '1'
178              )
179              else '0';
180     end behavioral;
```

Literaturverzeichnis

[1] M. Hedenus, "Der Komet in der Entladungsröhre: Eugen Goldstein, Wilhelm Foerster und die Elektrizität im Weltraum", 1. Auflage, GNT Verlag, 2007.

[2] J. Thomson, "Cathode rays", *Phil.Mag.* **44** (1897) 293–316.

[3] E. Rutherford, "The Structure of the Atom", *Philosophical Magazine Series 6* **27** (1914) 488–498.

[4] R. Frisch and O. Stern, "Über die magnetische Ablenkung von Wasserstoffmolekülen und das magnetische Moment des Protons.", *Zeitschrift für Physik* **85** (1933) 4–16. doi:10.1007/BF01330773.

[5] L. W. Alvarez and F. Bloch, "A Quantitative Determination of the Neutron Moment in Absolute Nuclear Magnetons",*Phys. Rev.* **57** (Jan, 1940) 111–122.

[6] K. Kleinknecht, "Detektoren für Teilchenstrahlung", 4. Auflage, Teubner, 2005.

[7] M. Gell-Mann, "A Schematic Model of Baryons and Mesons", *Phys. Lett.* **8** (1964) 214–215.

[8] G. Zweig, "An SU_3 model for strong interaction symmetry and its breaking", *CERN preprints* **TH-401, TH-412** (1964) .

[9] D. Griffiths, "Einführung in die Elementarteilchenphysik", Akademie Verlag, 1996.

[10] S. Kullander, "Highlights of the European muon collaboration", *Nuclear Physics A* **518** (1990) no. 1-2, 262 – 296. doi:10.1016/0375-9474(90)90549-2.

[11] **European Muon** Collaboration, J. Ashman *et al.*, "A measurement of the spin asymmetry and determination of the structure function g(1) in deep inelastic muon proton scattering", *Phys. Lett.* **B206** (1988) 364–370.

[12] J. Reichardt, "Lehrbuch Digitaltechnik: Eine Einführung mit VHDL", 2. Auflage, Oldenbourg, Mai, 2011.

[13] Xilinx Inc., **DS100**, "Virtex-5 Family Overview", v5.0 ed., Feb, 2009. http://www.xilinx.com.

[14] T. Fliessbach, "Quantenmechanik", BI-Wissenschafts-Verlag, 1991.

[15] R. L. Jaffe und A. Manohar, "The g_1 Problem: Fact and Fantasy on the Spin of the Proton", *Nucl. Phys.* **B337** (1990) 509–546. doi:10.1016/0550-3213(90)90506-9.

[16] The COMPASS Collaboration, "The deuteron spin-dependent structure function g_1^d and its first moment", *Phys. Lett.* **B647** (2007) 8–17. doi:10.1016/j.physletb.2006.12.076.

[17] C. Adolph *et al.*, "Leading order determination of the gluon polarisation from DIS events with high-p_T hadron pairs", *Phys.Lett.* **B718** (2013) 922–930, `arXiv:1202.4064 [hep-ex]`.

[18] X. Ji, "Gauge-Invariant Decomposition of Nucleon Spin", *Phys. Rev. Lett.* **78** (Jan, 1997) 610–613. doi:10.1103/PhysRevLett.78.610.

[19] M. Burkardt, A. Miller, and W. D. Nowak, "Spin-polarized high-energy scattering of charged leptons on nucleons", *Rep. Prog. Phys.* **73** (2009) 016201, `arXiv:0812.2208 [hep-ph]`.

[20] F. Herrmann, "Development and Verification of a High Performance Electronic Readout Framework for High Energy Physics", Dissertation, Universität Freiburg, August, 2011.

[21] B. Povh, K. Rith, C. Scholz, and F. Zetsche, "Teilchen und Kerne: Eine Einführung in die physikalischen Konzepte.", Springer-Lehrbuch, Springer, 2004.

[22] R. Field, "Applications of perturbative QCD", Frontiers in physics, Addison-Wesley, The Advanced Book Program, 1995.

[23] F. Halzen and A. Martin, "Quarks & Leptons: An Introductory Course in Modern Particle Physics", Wiley, 1984.

[24] C. Amsler *et al. Phys. Lett.* **B667** (2008) 194–201. http://pdg.lbl.gov.

[25] U. Elschenbroich, "Diagram 01-105", http://www-hermes.desy.de.

[26] D. Müller *et al.*, "Wave Functions, Evolution Equations and Evolution Kernels from Light-Ray Operators of QCD", *Fortsch. Phys.* **42** (1998) 101, `arXiv:hep-ph/9812448`.

[27] A. V. Radyushkin, "Nonforward parton distributions", *Phys. Rev. D* **56** (Nov, 1997) 5524–5557. doi:10.1103/PhysRevD.56.5524.

[28] A. V. Belitsky, D. Mueller, and A. Kirchner, "Theory of deeply virtual Compton scattering on the nucleon", *Nucl.Phys.* **B629** (2002) 323–392, `arXiv:hep-ph/0112108 [hep-ph]`.

[29] F. Gautheron, "COMPASS-II Proposal", Tech. Rep. CERN-SPSC-2010-014. SPSC-P-340, CERN, Geneva, May, 2010.

[30] X. Ji, "Deeply virtual Compton scattering",*Phys. Rev. D* **55** (June, 1997) 7114–7125. doi:10.1103/PhysRevD.55.7114.

[31] K. Göke, M. V. Polyakov, and M. Vanderhaeghen, "Hard Exclusive Reactions and the Structure of Hadrons", *Prog. Part. Nucl. Phys.* **47** (2001) no. hep-ph/0106012, 401–515.

[32] M. Burkardt, "Impact Parameter Space Interpretation for Generalized Parton Distributions", *International Journal of Modern Physics A* **18** (2003) 173. doi:10.1142/S0217751X03012370.

[33] Jefferson Lab, "Conceptual Design Report (CDR) for The Science and Experimental Equipment for The 12 GeV Upgrade of CEBAF", Prepared for the DOE Science Review, März, 2005. http://www.jlab.org.

[34] P. Abbon *et al.*, "The COMPASS experiment at CERN", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **577** (2007) no. 3, 455 – 518, `arXiv:hep-ex/0703049`.

[35] Forthommel, "Map of the CERN accelerator complex", http://public.web.cern.ch.

[36] T. Szameitat, "Entwicklung einer Monte-Carlo-Simulation für das COMPASS-II-Experiment", Diplomarbeit, Universität Freiburg, November, 2012.

[37] T. Kunz, "Entwicklung einer Simulationsumgebung für das COMPASS-II-Experiment mit Geant4", Diplomarbeit, Universität Freiburg, November, 2012.

[38] P. Abbon *et al.*, "Particle identification with COMPASS RICH-1", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **631** (2011) no. 1, 26 – 39.

[39] H. Kolanoski, "Einführung in die Astroteilchenphysik", Vorlesungsskript, Institut für Physik, Humbold-Universität Berlin, 2007.

[40] S. Trippel, "Aufbau einer Messeinrichtung zur Bestimmung des Myonenflusses bei COMPASS", Diplomarbeit, Universität Freiburg, März, 2005.

[41] M. Gorzellik, "Entwicklung eines digitalen Triggersystems für Rückstoßproton-Detektoren", Diplomarbeit, Universität Freiburg, März, 2013.

[42] J. Barth, "Private communications", (2012) .

[43] C. Bernet *et al.*, "The COMPASS trigger system for muon scattering", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **550** (2005) no. 1–2, 217 – 240. doi:10.1016/j.nima.2005.05.043.

[44] S. Schopferer, Dissertation, Universität Freiburg, bisher unveröffentlicht, 2013.

[45] B. Grube, "A Trigger Control System for COMPASS and a Measurement of the Transverse Polarization of Λ and Ξ Hyperons from Quasi-Real Photo-Production.", Dissertation, Technische Universität München, 2006.

[46] T. Baumann, “Entwicklung einer Schnittstelle zur Übertragung von Pulsinformationen eines Rückstoßdetektors an ein digitales Triggersystem”, Diplomarbeit, Universität Freiburg, Januar, 2013.

[47] S. Schopferer, “Entwicklung eines hochauflösenden Transientenrekorders”, Diplomarbeit, Universität Freiburg, Februar, 2009.

[48] L. Lauser, “Einbindung eines Transientenrekorders in das COMPASS-Datennahmesystem”, Diplomarbeit, Universität Freiburg, April, 2009.

[49] M. Büchele, “Entwicklung eines FPGA-basierten 128-Kanal Time-to-Digital Converter für Teilchenphysik-Experimente”, Diplomarbeit, Universität Freiburg, Januar, 2012.

[50] J. Bieling, “Entwicklung eines ungetakteten 64-Kanal-Meantimers und einer Koinzidenzschaltung auf einem FPGA”, Diplomarbeit, Universität Bonn, Februar, 2010.

[51] American National Standards Institute, Inc., “American National Standard for VME64 Extensions”, 1997. http://www.vita.com.

[52] Xilinx Inc., **SP002**, “Aurora Protocol Specification”, v2.0 ed., September, 2007. http://www.xilinx.com.

[53] “Virtex-5 FPGA Packaging and Pinout Specification”. **UG195**, v4.8 ed., December, 2010. http://www.xilinx.com.

[54] Xilinx Inc., **DS080**, “System ACE CompactFlash Solution”. http://www.xilinx.com.

[55] American National Standards Institute, Inc., “American National Standard for VXS VMEbus Switched Serial Standard”, 2006. http://www.vita.com.

[56] R. M. Owen Boyle and E. van der Bij, “The S-LINK Interface Specification”. ECP Division, CERN, March, 1997. http://www.cern.ch/HSI/s-link/.

[57] F. Herrmann, “The GANDALF framework”, November, 2012. COMPASS collaboration meeting, Vortrag.

[58] Silicon Labs, **SI5326**, “Any-frequency precision clock multiplier/jitter attenuator”, rev. 1.0 9/10 ed., 2010. http://www.silabs.com/.

[59] Texas Instruments, **CDCE949**, “Programmable 4-PLL VCXO Clock Synthesizer with 1.8V, 2.5V and 3.3V LVCMOS Outputs”, August, 2007. http://www.ti.com/.

[60] National Semiconductor, **CLC016**, “CLC016 Data Retiming PLL with Automatic Rate Selection”, 2011. http://www.national.com/.

[61] HONDA TSUSHIN KOGYO Co., Ltd., **HDRA-ED136LFZG**T, “Right-angle SMT type female connector for PCBs”. http://www.hondaconnectorshk.com.

[62] ON semiconductor, **NB4N855S**, "3.3 V, 1.5 Gb/s Dual AnyLevel^{TM}to LVDS Receiver/Driver/Buffer/ Translator", August, 2007. http://www.onsemi.com/.

[63] B. G. Taylor, "Timing distribution at the LHC", *8th Workshop on Electronics for the LHC Experiments* (September, 2002) .

[64] H. Fischer *et al.*, "The COMPASS Online Data Format - Version 3", COMPASS note 2002-8, July, 2002.

[65] Xilinx Inc., "Dual-Port Block Memory Core v6.3", August, 2005. http://www.xilinx.com.

[66] "GANDALF Framework User Guide v. 1.1". http://hadron.physik.uni-freiburg.de/gandalf.

[67] P. Ashenden, "The Designer's Guide to VHDL", 3rd Edition, Morgan Kaufmann, 2008.

[68] B. Hoppe, "Verilog", Oldenbourg, 2006.

[69] Mentor Graphics, "ModelSim SE V.10.0b". http://model.com.

[70] Xilinx Inc., "ISE-Design Suite v.14.2". http://www.xilinx.com.

[71] Xilinx Inc., **UG190**, "Virtex-5 FPGA User Guide", v5.3 ed., May, 2010. http://www.xilinx.com.

[72] Xilinx Inc., **UG175**, "LogiCORE IP FIFO Generator v8.1 - User Guide", März, 2011. http://www.xilinx.com.

[73] M. Niebuhr, "Entwicklung eines 250 MHz Zählers mit totzeitfreier Auslese für das COMPASS Experiment", Diplomarbeit, Universität Freiburg, November, 2000.

[74] H. Wollny, "Bestimmung des Myonenflusses am COMPASS-Experiment", Diplomarbeit, Universität Freiburg, März, 2007.

[75] Xilinx Inc., **DS202**, "Virtex-5 FPGA Data Sheet: DC and Switching Characteristics", v5.3 ed., May, 2010. http://www.xilinx.com.

[76] M. Büchele *et al.*, "The GANDALF 128-Channel Time-to-Digital Converter", *Physics Procedia* (2011) , `arXiv:1112.4291`.

[77] M. Büchele *et al.*, "A 128-channel Time-to-Digital Converter (TDC) inside a Virtex-5 FPGA on the GANDALF module", (2012) .

[78] D. Rabe, "FPGA: Pseudo Random Generator (PRNG)". http://www.technik-emden.de/ rabe/.

[79] W. Geiselmann, "Datensicherheitstechnik", Vorlesungsskript, Universität Karlsruhe, Januar, 2009.

[80] B. Langer, "Stromchiffren – Entwurf, Einsatz und Schwächen", Diplomarbeit, Technische Universität Darmstadt, Juli, 2006.

[81] P. Alfke, "Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators". Xilinx Inc., **XAPP 052**, v1.1 ed., July, 1996. http://www.xilinx.com.

[82] A. C. Melissinos, "Experiments in modern physics", Academic Press, 1966.

[83] S. Schopferer, "Private communications", (2013) .

[84] Hamamatsu Photonics K.K. , 14-5 Shimokanzo, Toyooka village, Shizouka-ken, Japan. http://www.hamamatsu.com.

[85] A. Gorin, "Nuclear Instruments and Methods", 2000.

[86] R. Brun, F. Rademakers, "ROOT webpage". http://root.cern.ch.

[87] "Qt webpage". http://qt.digia.com/.

[88] "Qt-ROOT webpage". http://root.bnl.gov/QtRoot/QtRoot.html.

[89] C. Michalski, V. Frolov, "Beam Monitoring Page". http://wwwcompass.cern.ch/spill/spill.html.

[90] S. Horikawa *et al.*, "A Scintillating Fiber Tracker with High Time Resolution for High-Rate Experiments", IEEE TRANSACTIONS ON NUCLEAR SCIENCE, 2002.

Danksagung

Hiermit möchte ich mich bei allen Personen bedanken, die zum Gelingen dieser Arbeit beigetragen haben:

- Prof. Horst Fischer für die Vergabe des interessanten Themas, sowie seine umfassende Betreuung. Seine konstruktiven Anregungen waren unverzichtbar und seine Tür stand immer sehr weit offen.
- Prof. Kay Königsmann für die freundliche Aufnahme in seiner Abteilung und für die Möglichkeit meine Diplomarbeit als Mitglied der COMPASS Kollaboration verfassen zu dürfen.
- Den drei „Hardware'lern" Maximilian Büchele, Sebastian Schopferer und Dr. Florian Herrmann für die Hilfe rund um die Themen VHDL und FPGA und das Korrekturlesen meiner Diplomarbeit.
- Tobias Szameitat und Katharina Schmidt als weiteren Korrekturlesern.
- Der gesamten Abteilung für die angenehme Atmosphäre.
- Meiner Familie und insbesondere meinen Eltern, ohne deren liebevolle Unterstützung mein Studium in dieser Form nicht möglich gewesen wäre.
- Kerstin für ihre Unterstützung und ihr Verständnis, wenn durch CERN-Fahrten oder längere Arbeitstage manchmal wenig Zeit für das Leben außerhalb der Physik blieb.

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Freiburg, den 1. März 2013

.....