Photoproduktion $\gamma\,\mathrm{p} \rightarrow \mathrm{p}\pi^0\pi^0$ Ergebnisse und Entwicklung einer schnellen FADC-Auslese für Doppelpolarisationsexperimente

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Torge Szczepanek

aus Wilhelmshaven

Bonn 2006

Diese Dissertation ist auf dem Hochschulschriftenserver der ULB Bonn
http://hss.ulb.uni-bonn.de/diss_online
elektronisch publiziert.

Angefertigt mit Genehmigung der
Mathematisch–Naturwissenschaftlichen Fakultät der
Rheinischen Friedrich–Wilhelms–Universität Bonn

1. Referent:	Prof. Dr. E. Klempt
2. Referent:	Prof. Dr. U. Thoma
Tag der Promotion:	12.7.2006

Inhalt

Zusammenfassung

Der Crystal-Barrel-Detektor ist ein am Bonner Elektronenbeschleuniger ELSA installiertes Detektorsystem zur Untersuchung von Baryonresonanzen mittels Photoproduktions-Experimenten. Mit einem abgedeckten Raumwinkelbereich von 98 % von 4 π und einer hohen Sensitivität auf neutrale Teilchen (Photonen) ist er besonders gut geeignet zur Suche nach bisher noch nicht nachgewiesenen Baryonresonanzen im Massenbereich bis zu 2,5 GeV/c^2. Mit dem Experiment wurden Messungen zur Doppelpionproduktion im Kanal $\gamma p \rightarrow p\pi^0\pi^0$ durchgeführt. Erste Ergebnisse einer Partialwellenanalyse der aufgenommenen Daten werden in dieser Arbeit präsentiert und mit bisherigen Messungen sowie mit theoretischen Vorhersagen verglichen. Zur Auflösung von Ambiguitäten in den Ergebnissen der Partialwellenanalyse werden Messungen mit Doppelpolarisation unter Verwendung eines polarisierten Photonenstrahls und eines polarisierten Butanol-Targets benötigt. Letzteres erzeugt jedoch einen hohen zusätzlichen elektromagnetischen Untergrund aufgrund von Paarbildung und Comptonstreuung primär in Vorwärtsrichtung. Dieser Untergrund muss stark reduziert werden, um die neuen Doppelpolarisations-Messungen im Kanal $\gamma p \rightarrow p\pi^0\pi^0$ durchführen zu könnnen.

Neben einem Tscherenkow-Detektor zur Reduktion der elektromagnetischen Ereignisse kann hierzu ein Flash-Analog-Digitalkonverter (Flash-ADC) zur Auslese der Kalorimeterkristalle und zur Erkennung von überlagerten Signalen im Vorwärtsbereich eingesetzt werden. Hiermit können nicht nur Startzeitpunkt und Amplitude der Signale aufgezeichnet werden, sondern es ist möglich, die gesamte Signalform zu digitalisieren, zu speichern und zu analysieren. Damit können elektromagnetische Ereignisse erkannt werden, die die Signale der hadronischen Ereignisse überlagern und die Energieinformation verfälschen. Mittels der Signalform kann die Energieinformation des hadronischen Ereignisses rekonstruiert werden.

Diese Arbeit beschreibt die Entwicklung und Erprobung einer sehr schnellen Schnittstellenkarte für die Steuerung und Auslese eines Struck DL300-Flash-ADC-Systems, welches die bisherige softwarebasierte Auslese und die Schnittstelle zu den nicht mehr verwendeten Motorola 68000er CPUs ersetzen soll. Um eine möglichst schnelle Verarbeitung und Speicherung der Daten zu ermöglichen, wurde eine Schnittstelle auf Basis eines feldprogrammierbaren integrierten Schaltkreises der Firma Xilinx entwickelt, welcher auf einer PCI/PMC-Steckkarte zusammen mit schnellem Speicher untergebracht ist und die Daten an den Ausleseprozessor über einen direkten Speichertransfer aus dem Speicher der Kontrollkarte an die eingesetzte CPU übertragen kann. Die Auslese des DL300-Systems erfolgt dabei autonom auf dem integrierten Schaltkreis. Erste Testmessungen im Labor zeigen, dass die Flash-ADC-Auslese funktionstüchtig ist und Datenraten von 1 kHz oder mehr erreicht werden können. Desweiteren werden die Erstellung einer graphischen Kontrollsoftware für das Datenerfassungssystem in Kombination mit einem System zur Verarbeitung, Speicherung und Anzeige von Statusinformationen der Datenerfassung, sowie ein Hardware-Synchronisationsmodul und ein Interruptmodul auf Basis einer seriellen RS-232-Schnittstelle für den Linux-Kernel vorgestellt, die ebenfalls im Rahmen dieser Arbeit entwickelt wurden.

1 Einleitung und physikalische Motivation

In den letzten 2500 Jahren hat sich das physikalische Verständnis der Menschheit grundlegend geändert. Bereits um 400 vor Christus ging der griechische Philosoph Demokrit davon aus, dass Materie aus sehr kleinen unteilbaren Teilchen, den Atomen[1], bestehen sollte. Diese Hypothese wurde erst wieder um 1803 von dem englischen Chemiker John Dalton aufgegriffen. Er postulierte, dass Materie aus kleinsten kugelförmigen Teilchen oder Atomen besteht, welche unteilbar sind. Die Entdeckungen zur Struktur der von Demokrit und Dalton postulierten Atome und deren Atomkerne sollen im Nachfolgenden in einem kurzen historischen Überblick beschrieben werden. Darauf folgend wird der Weg zum heute akzeptierten Standardmodell der Elementarteilchenphysik beschrieben und es wird näher auf die innere Struktur der Baryonen und Mesonen eingegangen, welche der starken Wechselwirkung unterliegen. Die starke Wechselwirkung wird durch die Quantenchromodynamik als eine der fundamentalen Wechselwirkungen im Standardmodell beschrieben. Zum Schluss sollen moderne Quarkmodelle vorgestellt werden.

1.1 Historische Entdeckung der Atome und ihrer Eigenschaften

Nach der Entdeckung des Elektrons durch J. J. Thomson im Jahre 1897 wurde das nach ihm benannte Thomsonsche Atommodell oder auch Rosinenkuchenmodell im Jahre 1903 von ihm vorgeschlagen. Es beschreibt das Atom als Kugel mit einer gleichförmig verteilten positiven Masse. Die von ihm entdeckten negativ geladenen Elektronen sind dem Modell zufolge über diese gleichmäßige Masse verteilt. Thomson schloss weiterhin aus Experimenten mit Röntgenstrahlung, dass die Anzahl der Elektronen in einem Atom in etwa der Massenzahl des Atoms entsprechen müsste. Schon etwa sechs Jahre später wurde durch einen von Hans Geiger, Ernest Marsden und Ernest Rutherford durchgeführten Streuversuch gezeigt, dass diese Vorstellung so nicht korrekt sein konnte. Im Streuversuch wurde eine Quelle für Alphateilchen vor einer sehr dünnen Goldfolie platziert. Die an der Goldfolie gestreuten Alphateilchen konnten dann mit einem Leuchtschirm sichtbar gemacht werden. Aufgrund der Verteilung der registrierten Alphateilchen und einem hohen Anteil an rückgestreuten Teilchen schloss Rutherford, dass es ein Massezentrum im Atom gibt, dessen Ladung dasselbe Vorzeichen besitzt, wie die Alphateilchen, mit denen es beschossen wurde. Insbesondere zeigte sich, dass dieses Massezentrum sehr klein im Vergleich zum Atomdurchmesser sein musste und dass zwischen den Atomkernen

Abbildung 1.1 Bild von Ernest Rutherford, Quelle: Wikipedia

[1] gr. atomos - unteilbar

in der Folie ein großer Freiraum bestand. Der Wirkungsquerschnitt aus Formel 1.1 beschreibt unter anderem die Streuung eines mit $Z_1 \cdot e$ geladenen Teilchens an einem mit $Z_2 \cdot e$ geladenen Atomkern. Diese Erkenntnis führte zum Rutherfordschen Atommodell, bei dem das Atom aus einem sehr kleinen positiv geladenen Kern besteht und die Elektronen sich auf Kreisbahnen um den Kern bewegen.

$$\frac{d\sigma}{d\Omega} = \left(\frac{1}{4\pi\varepsilon_0}\frac{Z_1 Z_2 e^2}{4E_0}\right)^2 \frac{1}{\sin^4\left(\frac{\vartheta}{2}\right)} \tag{1.1}$$

Nach der klassischen Vorstellung muss ein Elektron, das sich auf einer Kreisbahn um den Atomkern bewegt, Energie in Form elektromagnetischer Wellen abstrahlen. Durch den Energieverlust müsste sich das Elektron immer weiter auf den Atomkern zubewegen. Dies müsste nach der klassischen Vorstellung schon innerhalb von 10^{-8} Sekunden passieren, so dass es keine stabilen Atome geben dürfte, die man beobachten könnte.

Für die weiteren Erkenntnisse zum Aufbau der Atome entscheidend waren die spektroskopischen Entdeckungen, die unter anderem durch den Münchener Optiker und Physiker Joseph von Fraunhofer gemacht wurden. Er beobachtete und studierte 1814 die bereits 1802 von William Hyde Wollaston entdeckten dunklen Linien im Spektrum der Sonne. Er bestimmte durch sehr sorgfältige Messungen die Wellenlänge von über 570 Linien.

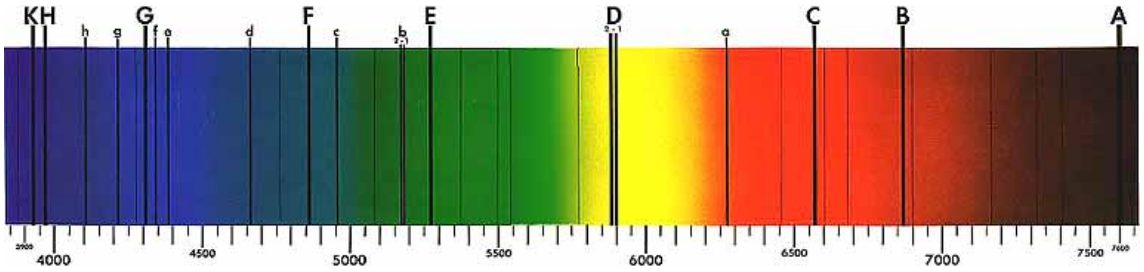


Abbildung 1.2 Fraunhoferlinien im Spektrum der Sonne. Quelle: Wikimedia Commons

Die Erklärung dieser Linien gelang, nachdem Gustav Kirchhoff und Robert Wilhelm Bunsen Elemente erhitzten und deren Strahlung spektroskopisch erforschten. Sie entdeckten, dass jedes Element bei Erhitzung ein charakteristisches Spektrum aussandte. Die hierbei entdeckten Emissionslinien konnten mit den von Fraunhofer im Sonnenlicht beobachteten Absorptionslinien in Verbindung gebracht werden. Die meisten der von Fraunhofer beobachteten Linien sind demnach Elementen der Sonnenatmosphäre zuzuordnen, die dort das von der Sonne in Richtung Erde emittierte Licht[2] absorbieren und isotrop wieder abstrahlen. Durch die isotrope Abstrahlung wird die Intensität des Schwarzkörperspektrums in den Emissionslinien der Elemente drastisch reduziert, so dass diese in Absorption beobachtet werden.

Niels Bohr formulierte nun zwei Postulate, die die Absorptionslinien im Sonnenspektrum bzw. das charakteristische Emissionsspektrum von Elementen erklären konnten, jedoch keine Erklärung im Rahmen der klassischen Elektrodynamik fanden und die Grundlage für das Bohrsche Atommodell bildeten:

[2] Die Sonne ist annähernd ein Schwarzkörperstrahler.

Ein atomares System

- hat stationäre Zustände mit bestimmten diskreten Energiewerten
- kann seine Energie nur ändern, indem es von einem stationären Zustand in einen anderen stationären Zustand übergeht. Bei einem Übergang zwischen zwei Zuständen wird Strahlung absorbiert oder emittiert.

Die Energie der absorbierten oder emittierten Strahlung ist hierbei über die Bedingung

$$\Delta E = h \cdot \Delta\nu \tag{1.2}$$

mit der Frequenz der Strahlung verknüpft und entspricht der Energiedifferenz der beiden stationären Zustände im Atom. Die Konstante h ist hierbei das sogenannte Plancksche Wirkungsquantum.

In einem von Stern und Gerlach 1921 durchgeführten Versuch wurde ein Strahl von Silberatomen durch ein inhomogenes Magnetfeld geschossen. Dieser spaltete im Magnetfeld in genau zwei Anteile auf. Stern und Gerlach konnten mit diesem Versuch zeigen, das die Elektronen in der Atomhülle einen Eigendrehimpuls besitzen, der in einer beliebig vorgegebenen Richtung nur die beiden diskreten Werte $\frac{\hbar}{2}$ und $-\frac{\hbar}{2}$ annehmen kann. Dieser Eigendrehimpuls wird als Spin[3] bezeichnet.

Die Vorhersage der stationären Zustände und die damit verbundene Quantelung der Energie sowie die Entdeckung des Spins haben im Weiteren zur Entwicklung der Quantenmechanik und zum Orbitalmodell der Atome geführt. Dies sind heute die Grundlagen für die Chemie und die Atomphysik.

Die Spektroskopie von chemischen Elementen, wie sie von Bunsen und Kirchhoff durchgeführt wurde und die Beobachtung der Absorptionslinien im Sonnenspektrum haben somit entscheidend zum Verständnis der Struktur der Atomhülle beigetragen.

1.2 Der Atomkern, Baryonen, Mesonen und Quarks

Über den positiv geladenen Atomkern war außer der durch Rutherford entdeckten Konzentration der Masse auf einen sehr kleinen räumlichen Bereich nicht viel bekannt. Durch die Entdeckung des Neutrons im Jahre 1932 durch James Chadwick änderte sich dies. Chadwick entdeckte das Neutron durch ein Experiment, bei dem er Alphateilchen auf Beryllium schoss. Als Reaktionsprodukt enstand hierbei Kohlenstoff und ein elektrisch neutrales Teilchen, das Neutron.

Neben diesen Kernumwandlungen, bei denen sich die Kernladungszahl maximal um zwei Einheiten ändert, wurden in den folgenden Jahren weitere Entdeckungen gemacht. Dazu gehört

[3] engl. spin - Drehung, Drall

insbesondere die von Otto Hahn und Fritz Straßmann 1938 entdeckte und von Lise Meitner und Otto Robert Frisch im Dezember 1938 als Kernspaltung erklärte Reaktion von Uran beim Beschuss mit Neutronen. Die daraus entstehenden Folgen für die Nutzung in Kernwaffen und die Entwicklung zur Kernenergie waren zu damaliger Zeit kaum absehbar.

Heute wissen wir, dass der Atomkern aus positiv geladenen Protonen und elektrisch neutralen Neutronen besteht, die durch die starke Wechselwirkung zusammengehalten werden. Um diesen Zusammenhalt zu erklären, schlug Hideki Yukawa vor, dass der Atomkern durch Austausch von Mesonen zwischen den Protonen und Neutronen eines Kerns zusammengehalten werden sollte. Bedingt durch die Masse der vom ihm postulierten Mesonen ist die Reichweite dieser Kräfte auf den Kern beschränkt. Die von ihm vorhergesagten Austauschteilchen sollten eine Masse von etwa 150 MeV besitzen, was etwa dem 300-fachen der Elektronmasse entspricht und etwa einem Sechstel der Neutron- bzw. Protonmasse. Hieraus erklärt sich auch der Name Meson[4]. Elektronen werden aufgrund ihrer geringen Masse zu den Leptonen[5] und Protonen und Neutronen wegen der vergleichsweise hohen Masse zu den Baryonen[6] gezählt. Powell gelang im Jahr 1947 der Nachweis des Pions in der kosmischen Höhenstrahlung, welches als das von Yukawa vorhergesagte Austauschteilchen aufgrund der Masse idenfiziert werden konnte.

Abbildung 1.3 Bild von Lise Meitner. Quelle: Wikimedia Commons

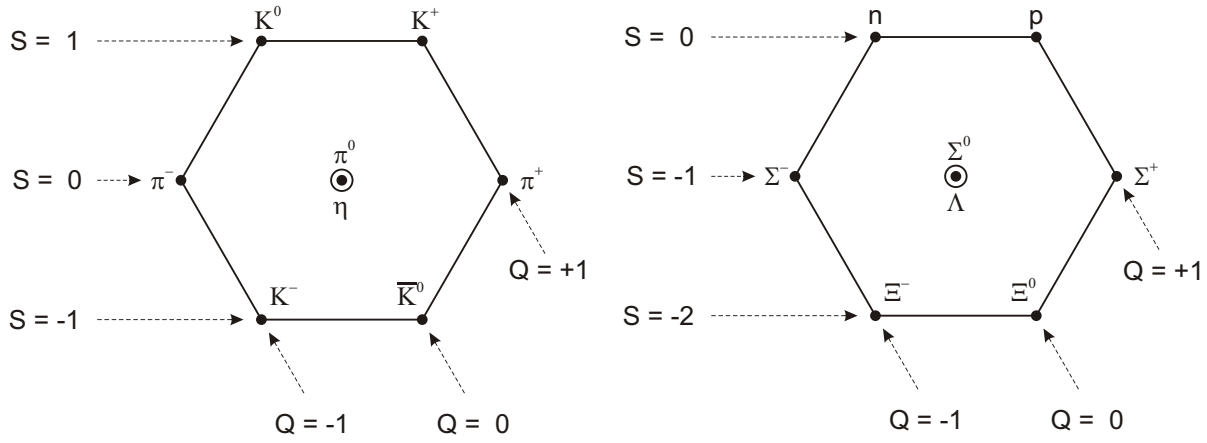
In den folgenden Jahren wurde insbesondere auch durch den Bau der ersten modernen Teilchenbeschleuniger ein wahrer Zoo an Teilchen entdeckt. Die Nutzung eines Teilchenbeschleunigers machte es erstmals möglich, Teilchen künstlich im Labor zu erzeugen, so dass die kosmische Höhenstrahlung nicht mehr die einzige verfügbare Teilchenquelle war.

Einige der entdeckten Baryonen hatten eine seltsame Eigenschaft: Sie ließen sich leicht erzeugen (in der kurzen Zeit von 10^{-23} s), zerfielen jedoch vergleichsweise langsam in einem Zeitraum von 10^{-10} s. Dieser seltsamen Eigenschaft ordnete man eine neue Quantenzahl S zu, die sogenannte Strangeness[7].

Gell-Mann gelang es 1961 Ordnung in diesem Teilchenzoo zu schaffen. Er ordnete Baryonen und Mesonen in geometrischen Figuren an. Die Positionen der Teilchen wurden aufgrund der Quantenzahlen geordnet, welche auf zwei Achsen aufgetragen waren. Auf der vertikalen Achse ist die schon erwähnte Strangeness S angeordnet, auf der horizontalen Achse die elektrische Ladung Q. Abbildung 1.4 zeigt zwei Oktetts für die acht leichtesten Mesonen und

[4] gr. mesos - mitten, mittleres, zwischen
[5] gr. leptos - dünn, mager
[6] gr. baryo - schwer
[7] engl. strangeness - Seltsamkeit

Baryonen. Aufgrund der Zahl acht wurde das Modell als Eightfold Way bekannt, entsprechend dem Achtfachen Pfad im Buddhismus, dem Weg zur Erkenntnis, zur Aufhebung allen Leids und zur Befreiung. Dieser Name zeigt, wie sehr dieses Modell im Weiteren zum Verständnis der scheinbar chaotischen Zustände des Teilchenzoos beigetragen hat, die in den 1960er Jahren vorherrschten.



Oktett der pseudoskalaren Mesonen $J^P = 0^-$.

Oktett der Baryonen $J^P = \frac{1}{2}^+$.

Abbildung 1.4 Anordnung der acht leichtesten Hadronen (Mesonen und Baryonen) in Oktetts.

Gell-Mann hatte nur ein Problem: Ihm fehlte eines der Teilchen im sogenannten Baryondekuplett. Aufgrund der Position im Dekuplett und der Strangeness S = -3 konnte er die Masse dieses Teilchens sehr präzise vorhersagen, und bereits 1964 gelang der Nachweis dieses im Dekuplett fehlenden Ω^- Teilchens. Vorhersage und Nachweis dieses Teilchens gehörten zu den ersten Erfolgen des Quarkmodells bei der Beschreibung von aus Quarks zusammengesetzten Mesonen und Baryonen.

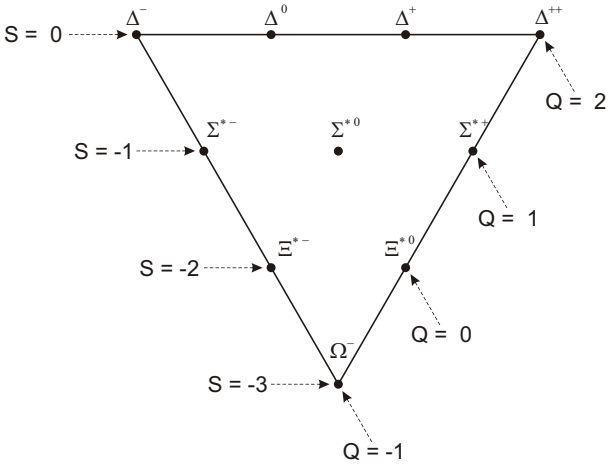


Abbildung 1.5 Anordnung von 10 Baryonen zu einem Dekuplett $J^P = \frac{3}{2}^+$.

Mit dem von Murray Gell-Mann 1961 eingeführten Eightfold Way war es möglich geworden, die Teilchen in einem geometrischen Schema anzuordnen. Die Frage nach dem Grund für diese Ordnung konnten Gell-Mann und Zweig 1964 mit Einführung der Quarks[8] ebenfalls beantworten. Gell-Mann sagte drei verschiedene Arten (sogenannte Flavors) von Quarks voraus, die in der Quantenchromodynamik mittels der SU(3)-Symmetriegruppe mathematisch beschrieben werden können (Flavor SU(3)). Heute wissen wir, dass insgesamt sechs verschiedenes Flavors existieren. Für die Beschreibung der von Gell-Mann angeordneten Hadronen reichten jedoch drei Quarks und deren Antiteilchen (drei Antiquarks) aus. Baryonen sind danach aus Kombinationen von drei Quarks zusammengesetzt (qqq), Mesonen aus einem Quark und einem Antiquark ($q\bar{q}$).

Allerdings gab es noch ein fundamentales Problem. Baryonen sind Fermionen[9], also Teilchen mit halbzahligem Spin. Die Gesamtwellenfunktion eines Fermions muss unter Vertauschung von zwei Teilchen antisymmetrisch bleiben. Ursache hierfür ist das Pauliprinzip, welches verbietet, dass zwei Teilchen mit halbzahligem Spin sich im gleichen Quantenzustand befinden.

Betrachtet man nun die Wellenfunktion eines Baryons als Kombination aus der Ortswellenfunktion, der Flavorwellenfunktion und der Spinwellenfunktion, so ergibt sich die Gesamtwellenfunktion als Produkt aus

$$\psi_{qqq} = \psi_{Ort} \cdot \psi_{Flavor} \cdot \psi_{Spin}. \tag{1.3}$$

Die Δ^{++}-Resonanz (siehe Abbildung 1.5) besteht aus drei u-Quarks. Nimmt man nun eine Wellenfunktion der Δ^{++} Resonanz an, bei der alle Spins in dieselbe Richtung zeigen:

$$\Delta^{++} = |u \uparrow u \uparrow u \uparrow\rangle = \psi_{uuu}, \tag{1.4}$$

so ergibt sich unter Vertauschung von zwei beliebigen Quarks dieselbe Gesamtwellenfunktion, da sich die Quarks voneinander nicht unterscheiden. Die Wellenfunktion ist somit symmetrisch in Spin und Flavor. Da das Δ^{++} der Grundzustand ist, vermutet man, dass alle drei Quarks sich in einem Zustand mit Drehimpuls L=0 befinden. Damit ist auch die Ortswellenfunktion symmetrisch gegenüber dem Austausch zweier Quarks und das Vorzeichen der Wellenfunktion ändert sich nicht. Dies steht im Widerspruch zum Pauliprinzip, das für eine Wellenfunktion eines

[8] Die Bezeichnung Quark entnahm Gell-Mann dem Buch Finnegans Wake von James Joyce. Zitat: „Three quarks for Muster Mark“.

[9] Benannt nach dem italienischen Kernphysiker Enrico Fermi.

Fermions eine Antisymmetrie und somit einen Vorzeichenwechsel bei Vertauschung der Quarks verlangt.

Als Lösung für dieses Problem wurde ein weiterer Freiheitsgrad vorgeschlagen. Hierbei werden den Quarks symbolisch die drei Farben rot, grün und blau zugeordnet. Analog zum sichtbaren Licht ergibt sich aus der Mischung der drei Farben die Farbe weiss. Jeder Zustand muss nun die Bedingung erfüllen, dass der Gesamtzustand farbneutral ist. Für Baryonen wird dies durch die drei verschiedenen Farben erreicht. Für Mesonen die aus Quark und Antiquark bestehen, wird zu jeder der drei Farben noch eine Antifarbe eingeführt. Ein Meson besteht aus Quark und Antiquark, sowie aus einer Farbe (z.B. rot) und der entsprechenden Antifarbe (z.B. antirot) und ist wiederum farbneutral.

Dadurch ergibt sich dann die Gesamtwellenfunktion zu:

$$\psi_{qqq} = \psi_{Ort} \cdot \psi_{Flavor} \cdot \psi_{Spin} \cdot \psi_{Farbe}. \tag{1.5}$$

Wählt man eine Wellenfunktion ψ_{Farbe}, die antisymmetrisch ist, ergibt sich durch das Produkt eine Gesamtwellenfunktion, die ebenfalls antisymmetrisch ist. Für gemischte Symmetrien ist dies nicht so schön anschaulich, wie im Fall des Δ^{++}. Es lässt sich jedoch auch hier mit deren Flavoranteil eine Gesamtwellenfunktion konstruieren, die unter Vertauschung zweier Quarks antisymmetrisch bleibt.

Mit der Einführung der Quarks und der Farbladung konnte eine Struktur in die entdeckten Teilchen gebracht werden. Viele Skeptiker in der damaligen Zeit akzeptierten das Quarkmodell dennoch nicht, da es nicht gelang, freie Quarks nachzuweisen, und die Einführung des Weiteren Freiheitsgrades Farbe zur Erhaltung des Pauli-Prinzips zu konstruiert erschien. Ebenso gab der Einschluss (Confinement)[10] von Quarks in Mesonen und Baryonen Rätsel auf.

Das Quarkmodell wurde allerdings nicht durch die Entdeckung von freien Quarks bestätigt, sondern durch eine Entdeckung eines neuen Teilchens, welche parallel von C.C. Ting und B. Richter im November 1974 publiziert wurde[11]. Ting nannte das Teilchen J und Richter ψ. Das heute J/ψ genannte Teilchen ist ein extrem schweres Meson, welches mehr als die dreifache Masse des Protons trägt. Wie wir heute wissen, besteht es aus einem damals noch nicht bekannten Quark-Flavor, dem Charm-Quark und dessen Antiquark ($c\bar{c}$). In den folgenden 3 bis 4 Jahren wurden dann weitere Baryonen und Mesonen mit Charm-Quark-Inhalt entdeckt (z.B. das D^0, das D^+ und das Σ_c^{++}).

Die Beobachtung eines neuen Quarks passte zur damals bekannten Anzahl von Leptonen. Insgesamt waren damit 4 Quarks und die 4 Leptonen Elektron e^-, Myon μ^-, Elektron-Neutrino ν_e und das Myon-Neutrino ν_μ bekannt. Glashow, Iliopoulos und Maiani schlugen bereits vor der Entdeckung des Charm-Quarks vor, dass es eine entsprechende Verbindung zwischen der Anzahl der Quarks und der Leptonen geben sollte [Gla70].

[10] engl. confinement - Einschluss

[11] Dieses Ereignis wird seither als Novemberrevolution bezeichnet.

Diese vermutete Symmetrie wurde allerdings sehr schnell gebrochen, als im Jahr 1975 von Martin L. Perl und seinen Mitarbeitern am SLAC bei Elektron-Positron-Kollisionen im SPEAR-Ring ein weiteres Lepton entdeckt wurde [Per75], das τ-Lepton. Weitere zwei Jahre später gelang es jedoch, noch ein weiteres Quark nachzuweisen: das b-Quark (Beauty oder Bottom) [Her77]. Am Fermilab wurden 1994 [Abe94] erste Anzeichen für die Existenz des Top-Quark gefunden und im Jahre 1995 bestätigt [Abe95]. Zusammen mit dem erfolgreichen Nachweis des ν_τ im Jahr 2000 durch das DONUT-Experiment[12] [Kod01] und dem bereits lange bekannten τ Lepton war somit die Symmetrie zwischen Leptonen und Quarks wiederhergestellt.

1.3 Das Standardmodell

Nach heutigem Kenntnisstand und Erkenntnissen aus e^+e^- Kollisionen am CERN wissen wir, dass es genau sechs Leptonen und sechs Quarks geben muss, welche entsprechend ihrer Masse drei verschiedenen Generationen zugeordnet werden.

Typ	Teilchen	Symbol	Ladung [e]	Masse [GeV]
Erste Generation				
Quarks	up	u	$+2/3$	$0,003$
	down	d	$-1/3$	$0,006$
Leptonen	Elektron	e^-	-1	$0,00051$
	Elektronneutrino	ν_e	0	$< 2,3 \cdot 10^{-9}$
Zweite Generation				
Quarks	charm	c	$+2/3$	$1,3$
	strange	s	$-1/3$	$0,14$
Leptonen	Myon	μ^-	-1	$0,106$
	Myonneutrino	ν_μ	0	$< 1,9 \cdot 10^{-4}$
Dritte Generation				
Quarks	top (truth)	t	$+2/3$	174
	bottom (beauty)	b	$-1/3$	$4,3$
Leptonen	Tauon	τ^-	-1	$1,777$
	Tauneutrino	ν_τ	0	$< 1,8 \cdot 10^{-2}$

Tabelle 1.1 Die drei Generationen von Leptonen und Quarks im Standardmodell.

Neben den elementaren Teilchen werden im Standardmodell drei fundamentale Kräfte vorhergesagt, die alle möglichen Reaktionen zwischen den in Tabelle 1.1 aufgeführten Teilchen

[12] DONUT - Direct Observation of the NU Tau

beschreiben. Jeder Wechselwirkung ist mindestens ein Austauschteilchen zugeordnet. In der Quantenfeldtheorie werden alle drei Kräfte auf den Austausch dieser Bosonen zurückgeführt.

1.3.1 Elektromagnetische Wechselwirkung

Die elektromagnetische Wechselwirkung wird durch das Photon vermittelt, welches ein Spin-1-Teilchen ist. Die Reichweite der Wechselwirkung ist unendlich, da das Photon als Austauschteilchen keine Masse trägt. Die elektromagnetische Wechselwirkung ist verantwortlich für die meisten der alltäglichen Phänomene. Hierzu gehören insbesondere Licht, Elektrizität, Magnetismus und die Eigenschaften von Festkörpern. Sie wirkt anziehend oder abstoßend, je nach Vorzeichen der beteiligten Ladungen. Die Stärke ist umgekehrt proportional zum Abstandsquadrat der beteiligten Ladungen. Die Quantenelektrodynamik (QED) ist die quantenfeldtheoretische Beschreibung des Elektromagnetismus. Als eine relativistische Eichtheorie wird die QED durch ihre Lagrangedichte

$$\mathcal{L} = -\frac{1}{4} F_{\mu\nu} F^{\mu\nu} + \sum_n \bar{\psi}_n (i\gamma_\mu D^\mu - m)\psi_n \tag{1.6}$$

definiert. $F^{\mu\nu}$ ist hierbei der Feldstärketensor, mit dessen Hilfe sich die vier Maxwellgleichungen formulieren lassen.

1.3.2 Schwache Wechselwirkung

Die schwache Wechselwirkung, die z.B. für radioaktive Zerfallsprozesse verantwortlich ist, besitzt drei Austauschbosonen. Diese wurden im Jahre 1983 am UA1- und UA2-Experiment am CERN durch Carlo Rubbia und seine Mitarbeiter nachgewiesen. Die beiden W-Bosonen tragen positive (W^+) bzw. negative elektrische Ladung (W^-), das Z-Boson trägt keine Ladung. Aufgrund der hohen Masse der Austauschbosonen von ca. 80 GeV für die beiden W-Bosonen und 91 GeV für das Z-Boson ist die Reichweite der Kräfte auf einen Bereich von etwa 10^{-18} m begrenzt, also auf subnukleare Abstände.

1.3.3 Starke Wechselwirkung

Die stärkste der fundamentalen Kräfte ist die starke Wechselwirkung. Grundlage für die Beschreibung der starken Wechselwirkung ist die Quantenchromodynamik (QCD). Hierbei erhält der Begriff Farbe eine theoretische Grundlage. Ähnlich wie die Quantenelektrodynamik ist die QCD eine Eichtheorie, die eine lokale Eichinvarianz unter Ort- und Zeittransformationen der Wellenfunktionen verlangt. Dies führt zu einem Vektorfeld mit Spin 1. Die Eichbosonen der QCD tragen jeweils eine Kombination aus einer Farb- und einem Antifarbladung. Mit den drei Farben und drei Antifarben ergeben sich neun mögliche Kombinationen, welche sich in einem Oktett und einem Singulett anordnen lassen. Das Singulett ist eine Linearkombination aus den jeweiligen Farb-Antifarb-Mischzuständen und ist in der QCD nicht relevant, da sich hiermit kein Farbaustausch zwischen zwei Quarks realisieren lässt. Somit ergeben sich acht Eichbosonen, die als

Gluonen[13] bezeichnet werden. Wichtigster Unterschied zur Quantenelektrodynamik ist hierbei, dass es eine Gluon-Gluon-Wechselwirkung gibt. Dadurch sind in Feynman-Diagrammen zusätzliche Terme zwischen den Reaktionspartnern (den Quarks) möglich, die die Berechnung von Reaktionen bei starken Wechselwirkungen maßgeblich erschweren. Insbesondere sind auch Vertices möglich, an die nur Gluonen koppeln (Selbstkopplung). Diese Selbstkopplung unterscheidet die starke Wechselwirkung von allen anderen fundamentalen Kräften.

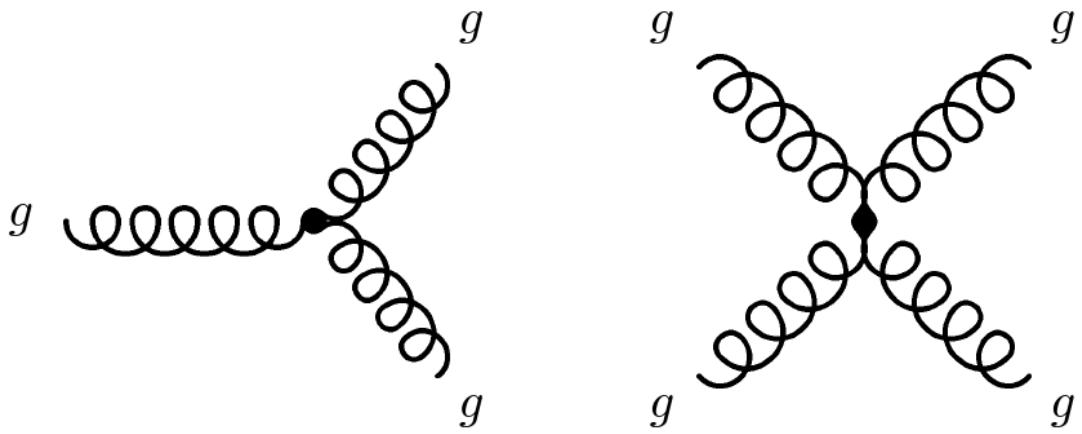


Abbildung 1.6 Gluonselbstkopplung

Das Potential der starken Wechselwirkung hat, ebenso wie die elektromagnetische Wechselwirkung, einen 1/r-Anteil, der allerdings durch einen linear ansteigenden Term ergänzt wird:

$$V_{qq} = -\frac{4}{3}\frac{\alpha_s(q^2)}{r} + \sigma r. \tag{1.7}$$

Die Kopplung α_s ist hierbei keine Konstante, sondern hängt vom Impulsübertrag q^2 wie folgt ab:

$$\alpha_s(q^2) = \frac{12 \cdot \pi}{(33 - 2n_f) \cdot ln\frac{q^2}{\Lambda^2}}. \tag{1.8}$$

Dabei ist n_f die Anzahl der Flavors und der Parameter Λ ein Skalenparameter, der im Bereich von 200 MeV liegt, was einem Abstand von ca. 1 fm entspricht.

Die Abhängigkeit vom Impulsübertrag lässt sich in drei Bereiche teilen:

- $Q^2 \ll \Lambda^2$ - chirale Symmetrie
- $Q^2 \approx \Lambda^2$ - Confinement
- $Q^2 \gg \Lambda^2$ - asymptotische Freiheit

Im Bereich der asymptotischen Freiheit, also bei hohen Impulsüberträgen ist eine störungstheoretische Behandlung der Wechselwirkungen möglich. In diesem Bereich ist die Kopplungskonstante α_s der starken Wechselwirkung klein. Bei kleineren Energien und insbesondere bei

[13] engl. glue - Kleber

gebundenen Zuständen wird α_s jedoch schnell größer. Die QCD weist bei sehr kleinen Energien eine besondere Eigenschaft auf, wenn man die Massen der Quarks vernachlässigt: die chirale Symmetrie.

Im nicht störungstheoretisch behandelbaren Bereich, also bei Impulsübertragungsquadraten in der Größenordnung des Quadrates des Skalenparameters Λ, müssen hierzu andere Modelle entwickelt werden, die gebundene Zustände wie das Nukleon und das Spektrum der angeregten Zustände erklären können.

1.4 Moderne Quarkmodelle

Als Modelle zur Erklärung der Interaktion zwischen Quarks gibt es verschiedene Ansätze, von denen einige im Weiteren kurz skizziert werden sollen; unter besonderer Berücksichtigung des in Bonn von U. Löring, B. C. Metsch und H. R. Petry entwickelten relativistischen Quarkmodells mit Instanton-induzierten Wechselwirkungen [Lö01].

1.4.1 Modell von Capstick und Isgur

Anfang bis Mitte der 1980er Jahre entwickelten S. Capstick und N. Isgur ein Quarkmodell mit relativistischen Korrekturen. Die Hamiltonfunktion ist hierbei durch

$$H = \sum i\sqrt{\vec{p}_i^{\,2} + m_i^2} \tag{1.9}$$

gegeben. Das Potential V setzt sich aus einem Confinement-Term und mehreren Restwechselwirkungen zusammen. Im nichtrelativistischen Grenzfall $p/m \to 0$ entspricht das Potential dem von nichtrelativistischen Quarkmodellen.

Das Potential enthält die folgenden Beiträge:

$$V = V_{string} + V_{coulomb} + V_{fein} + V_{hf} + V_{tp} \tag{1.10}$$

Das coulombähnliche und spinunabhängige Potential $V_{coulomb}$ ist bei kleinen Abständen durch den Austausch eines Gluons gegeben. Wie im Wasserstoffatom spielen weitere Potentiale eine Rolle. Die Feinstruktur (V_{fein}) wird durch eine Wechselwirkung zwischen Spin- und Bahndrehimplus bewirkt, die Hyperfeinstruktur (V_{hf}) analog durch eine Spin-Spin-Wechselwirkung. Zusätzlich gibt es noch einen Term V_{tp} der die Thomas-Präzession beschreibt.

Um nun mehr über die zugrundeliegende Physik zu erfahren, kann man ähnlich wie in der Atomphysik das Atom, das Nukleon in energetisch höhere Zustände anregen (Resonanzen). Dies erreicht man zum Beispiel dadurch, dass man Pionen und Nukleonen zur Kollision bringt (πN-Streuung) oder durch Anregung über hochenergetische Photonen. Die über Photonen angeregten Resonanzen zerfallen zumeist stark innerhalb der für die starke Wechselwirkung typischen Zeitdauer von 10^{-24} s in sekundäre Teilchen, die über Detektoren nachgewiesen werden können.

Beim Vergleich der vom Modell vorhergesagten mit experimentell gemessenen Resonanzen gelingt dem Modell eine Vorhersage der Massendifferenzen. Die absolute Vorhersage der Position ist allerdings nicht möglich. Das ist ein Problem vieler Quarkmodelle und zeigt wie schwierig es ist, ein entsprechend in allen Bereichen konsistentes Quarkmodell zu entwickeln. Beim Vergleich der Anzahl der vorhergesagten und bisher gefundenen Resonanzen fällt auf, das die Anzahl der vorhergesagten Resonanzen deutlich höher ist, als die der bisher experimentell nachgewiesenen. Hierauf soll später noch in Kapitel 1.4.3 eingegangen werden.

1.4.2 Bonn-Modell

Das bereits erwähnte Bonn-Modell versucht die Wechselwirkungen zu beschreiben, die zusätzlich zum Confinement existieren. Das Modell basiert auf der Bethe-Salpeter-Gleichung und baut, mit auf Instantonen basierenden Restwechselwirkungen, ein kovariantes Quarkmodell für die leichten Mesonen und Baryonen auf. Für die Berechnung der Spektren wird eine reduzierte Bethe-Salpeter-Gleichung verwendet. Diese wird numerisch gelöst, um die Massen der Baryonen als gebundene Drei-Quark-Zustände zu erhalten. Die Wechselwirkung wird durch zwei sogenannte Wechselwirkungskerne beschrieben, die zum einen die 3-Teilchen-Wechselwirkung enthalten, zum anderen die 2-Teilchen-Wechselwirkung. Für die Beschreibung des Potentials werden zwei Wechselwirkungen angenommen. Zum einen ein Confinementpotential, welches durch ein lineares Potential im Ruhesystem definiert wird und mittels Lorentztransformation in jedes System geboostet werden kann. Desweiteren eine Instanton-induzierte 2-Quark Wechselwirkung, die sich über die 't Hooftsche Kraft aus Instanton-Lösungen der klassischen QCD-Yang-Mills-Gleichungen bestimmt [t'H76]. Die Anzahl der freien Parameter der beiden Modelle ist in beiden Fällen gleich. Es gibt zwei Parameter für die Konstituentenquarkmassen, zwei für das Confinementpotential und drei Parameter zur Beschreibung der 't Hooftschen Kraft. Das sich ergebende Spektrum ist in Abbildung 1.7 gezeigt.

Die blauen Linien geben die Positionen der Massen des Bonn-Modells an. Im Vergleich dazu ist in grün das Modell nach Capstick, Roberts und Isgur zu sehen. Mit Fehlerbalken sind daneben die experimentellen Werte gezeigt, die von der Particle Data Group (PDG) gelistet wurden. Die experimentell gefundenen Resonanzen gelten allerdings nur etwa zur Hälfte als gesichert (sogenannte 3- oder 4-Stern-Resonanzen[14]).

Hierbei fällt wiederum auf, dass die Anzahl der vorhergesagten Resonanzen weitaus größer ist, als die Anzahl der bisher gefundenen Resonanzen. Hierfür gibt es unterschiedliche Erklärungsversuche.

1.4.3 Diskrepanzen zwischen Theorie und Experiment

Lichtenberg schlug bereits 1969 vor [Lic69], dass es möglicherweise eine Quark-Diquark Struktur im Baryon geben könnte.

[14] Die Anzahl der Sterne gibt an, wie gut die Resonanzen laut Particle Data Group etabliert sind. Die Skala geht dort von einem Stern (kaum etabliert) bis hin zu vier Sternen (etabliert).

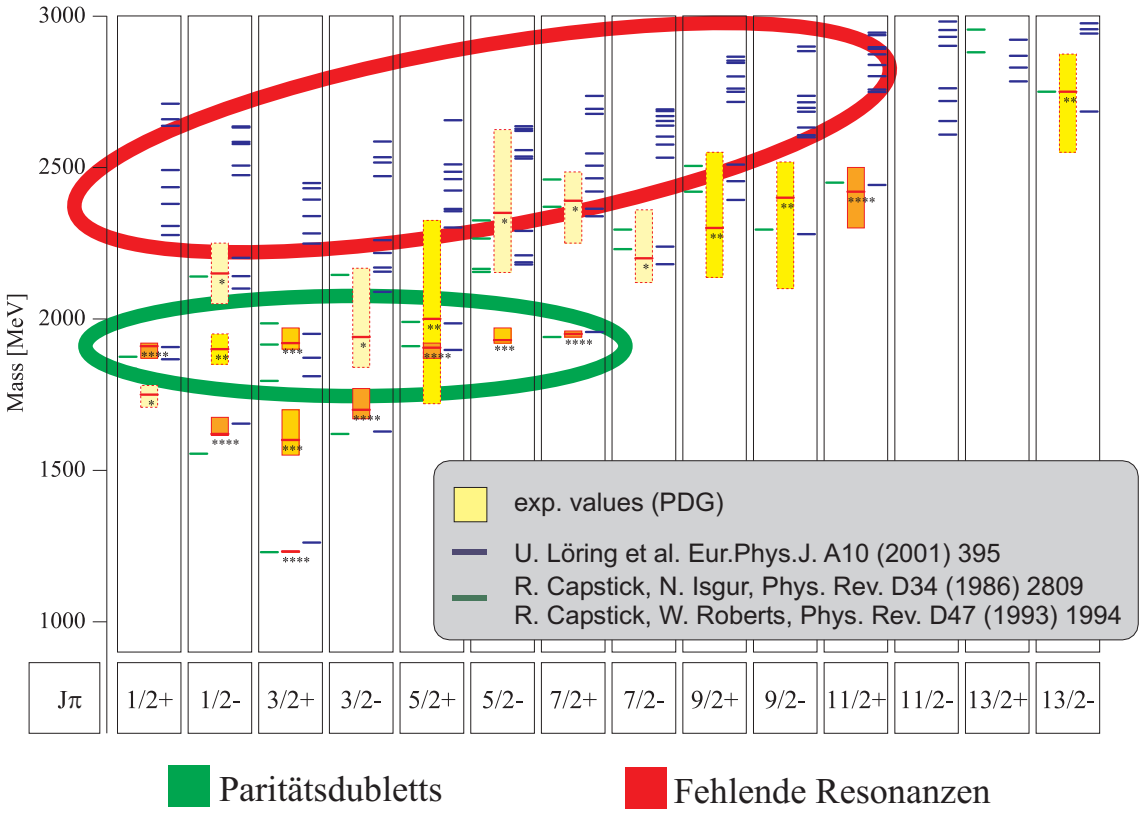
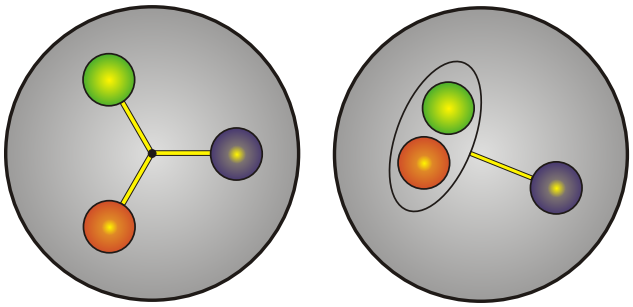


Abbildung 1.7 Vorhergesagte Δ^*-Resonanzen des Bonn-Modells und der Modelle von Capstick und Roberts sowie Capstick und Isgur. Vergleich zwischen experimentellen Daten und den Modellen.



a) Drei unabhängige Quarks

b) Quark-Diquark-Struktur

Abbildung 1.8 Quark-Diquark Struktur eines Nukleons.

Wie in Abbildung 1.8 gezeigt, ist dabei die Bewegung von zwei Quarks aneinander gekoppelt, so dass es nur eine Relativbewegung zwischen dem Quark und dem Diquark gibt. In Folge dessen verringert sich die Anzahl der Freiheitsgrade und die Anzahl der möglichen Resonanzen.

Eine mögliche weitere Erklärung gründet darauf, dass die bisher gefundenen experimentellen Daten aus früheren Streuexperimenten mit Pionen resultieren. Die fehlenden Resonanzen könnten gar nicht oder nur schwach an diesen Kanal koppeln, so dass die zusätzlichen Re-

Rating nach PDG	N^*	Δ^*	Σ^*	Λ^*	Ξ^*	Ω^*
****	11	7	6	9	2	1
***	3	3	4	5	4	1
**	6	6	7	1	2	2
*	2	6	8	3	3	0
Summe	12	22	26	18	11	4

Tabelle 1.2 Anzahl der experimentell entdeckten Baryonresonanzen mit Rating nach PDG.

sonanzen bisher noch nicht gefunden wurden. Um diese Resonanzen nachzuweisen, bieten sich photoninduzierte Kanäle an. Beispielrechnungen für den $\Delta\pi$-Kanal ergeben für viele der Resonanzen eine große Partialbreite in diesen Kanal. Das Crystal-Barrel-Experiment ist z.B. durch Untersuchung der Reaktion $\gamma p \rightarrow p\pi^0\pi^0$ ideal geeignet, um nach bisher noch nicht gefundene Resonanzen zu suchen und entscheidend zur Klärung der Natur der nicht etablierten Resonanzen beizutragen.

2 CB-ELSA-Experiment

Zur Untersuchung elektromagnetischer Anregungen subnuklearer Systeme, wie sie im Rahmen des transregionalen Sonderforschungsbereiches (SFB-TR16) durchgeführt werden sollen, wird ein Photonenstrahl hoher Intensität benötigt. Die hierfür verwendete Beschleunigeranlage in Bonn und das Detektorsystem des Crystal-Barrel-Experiments sollen im Folgenden ausführlich beschrieben werden.

2.1 Elektronenstretcheranlage ELSA

Die Elektronen-Stretcher-Anlage (ELSA) wurde im Jahr 1988 am Physikalischen Institut der Universität Bonn aufgebaut [Hus88]. Sie ergänzt das dort zuvor betriebene Synchrotron, welches nun als Vorbeschleuniger verwendet wird, um einen großen Speicherring zu speisen, welcher einen Elektronenstrahl mit einer Energie von bis zu 3,5 GeV liefern kann.

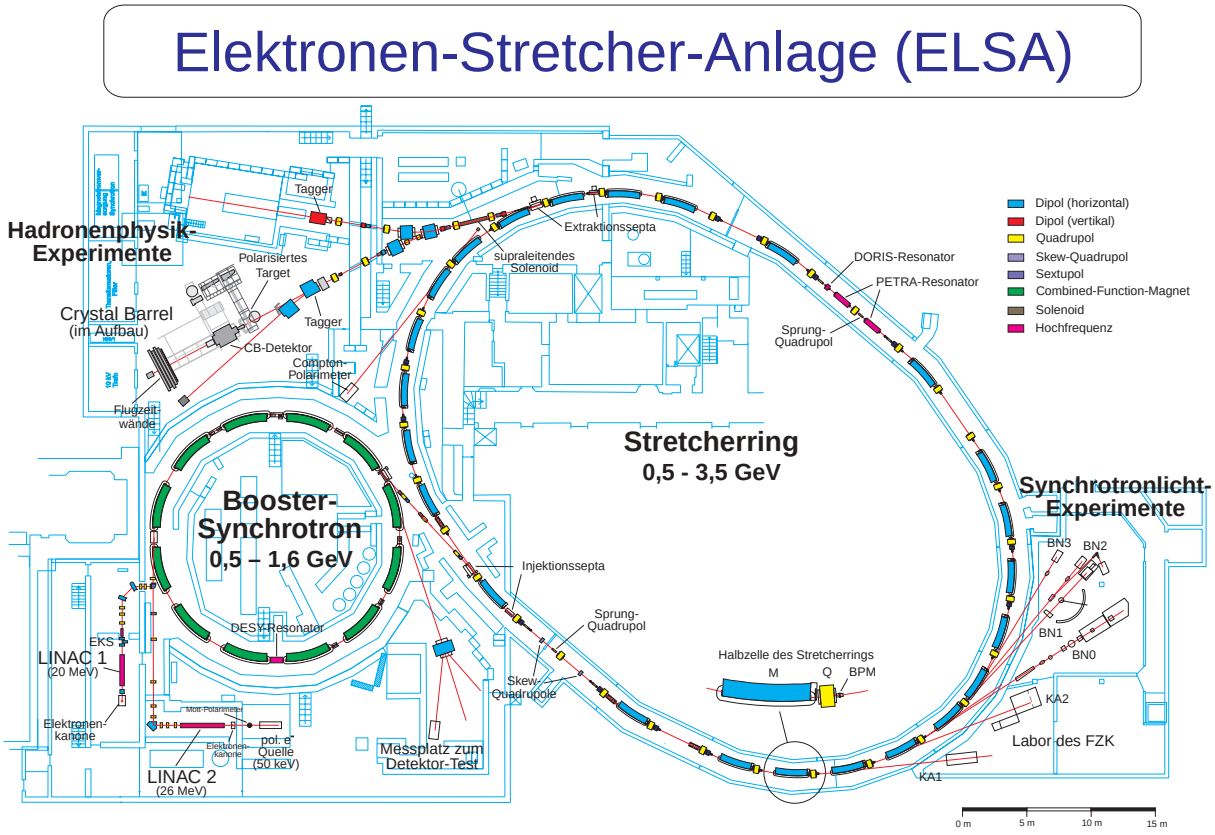


Abbildung 2.1 Die Elektronenstretcheranlage ELSA in Bonn.

Die beiden Linearbeschleuniger LINAC1 und LINAC2 erzeugen Elektronen aus einer Quelle und beschleunigen diese auf einer linearen Strecke auf eine Energie von 20 MeV (LINAC1) bzw. 26 Mev (LINAC2). Die im LINAC2 erzeugten Elektronen sind dabei mit etwa 80 % polarisiert, die im LINAC1 erzeugten Elektronen sind unpolarisiert. Die Beschleunigung der Teilchen erfolgt über schnell wechselnde Hochfrequenzspannungen. Dies wurde bereits im Jahre 1924 durch den Schweden Gustav Ising [Isi24] vorgeschlagen und drei Jahre später durch Wideröe erstmals

erfolgreich mit Driftröhren getestet. Die heute eingesetzten Linearbeschleuniger bestehen aus Hohlleiterstrukturen, in die Mikrowellen über Klystrons eingespeist werden.

Die erzeugten und auf ca. 20 MeV beschleunigten Elektronen werden dann mit dem Booster-Synchrotron auf eine Energie zwischen 0,5 GeV und 1,6 GeV weiterbeschleunigt. Hierbei werden Strahlströme im Synchrotron von maximal 500 nA erreicht. Durch die Beschleunigung der Elektronen wird auch die Zentrifugalkraft größer. Um die Teilchen auf derselben Bahn zu halten müssen bei der Beschleunigung die Energie der Teilchen und die Magnetfelder der Dipolmagnete synchron geändert werden, damit sich der Radius der Kreisbahn nicht vergrößert (siehe Gleichung 2.1). Hieraus ergibt sich der Name Synchrotron [Wil02]. Das Prinzip des Synchrotrons wurde 1945 fast zeitgleich von E.M. McMillan an der Universität von Kalifornien [McM45] und V. Veksler [Vek45] in der ehemaligen Sowjetunion veröffentlicht.

$$R = \frac{E}{ecB} \tag{2.1}$$

Der nachgeschaltete Stretcherring sorgt für eine gleichmäßige Verteilung der Elektronen. Bei einer nachfolgenden langsamen Extraktion kann ein externer Elektronenstrahl mit einem Tastverhältnis[15] von bis zu 90 % erzeugt werden. Während der Extraktion erhält man aufgrund der vorherigen Verteilung im Stretcherring einen sehr gleichförmigen Strahl, der den Experimenten zur Verfügung steht. In diesem Stretchermodus sind Strahlströme zwischen 10 nA und 100 nA am externen Strahlplatz mit Energien von bis zu 1,6 GeV möglich. Um höhere Energien zu erreichen, kann im Stretcherring eine Nachbeschleunigung der Elektronen durchgeführt werden. Hierbei können Energien von 0,5 GeV bis hin zu 3,5 GeV am externen Strahlplatz mit Strahlströmen zwischen 1 pA und 20 nA erreicht werden.

Die aus ELSA extrahierten Elektronen lassen sich an zwei Experimentierplätzen nutzen. Abbildung 2.1 zeigt den Experimentierplatz, an dem das Crystal-Barrel-Experiment derzeit aufgebaut wird. Desweiteren ist ein zweiter Strahlplatz vorhanden, der für weitere Experimente zur Hadronenphysik verwendet werden kann. Dieser wurde zuvor durch das Crystal-Barrel-Experiment für Messungen mit einem unpolarisierten Photonenstrahl verwendet.

Neben der Extraktion besteht desweiteren noch die Möglichkeit der Speicherung der Elektronen im Ring. Durch die Ablenkung der hochenergetischen Elektronen im Magnetfeld wird Synchrotronstrahlung erzeugt, die an den sieben Synchrotronstrahlungsplätzen genutzt werden kann.

2.2 Übersicht über das Experiment

Im Rahmen des Sonderforschungsbereichs Transregio 16 wird das Crystal-Barrel-Experiment an einem neuen Strahlplatz an der Elektronenstretcheranlage ELSA aufgebaut. Der neue Strahlplatz ermöglicht die Nutzung eines polarisierten Elektronenstrahls sowie eines polarisierten Targets für die weitere Experimente im Rahmen des Sonderforschungsbereiches.

[15] Tastverhältnis - auch Tastgrad (engl. duty cycle) - gibt das Verhältnis der Länge des eingeschalteten Zustands (hier Extraktion eines Elektronenstrahls) zur Periodendauer bei einem Rechtecksignal an.

Das Crystal-Barrel-Experiment besteht aus mehreren Detektorkomponenten, die im Folgenden näher beschrieben werden sollen. Dazu gehören:

- Radiator-Target – Erzeugung von Bremsstrahlungsphotonen
- Møller-Polarimeter – Messung des Polarisationsgrades des polarisierten Elektronenstrahls
- Tagging-System – indirekte Messung der Photonenenergie
- Polarisiertes oder unpolarisiertes Target für die erzeugten Photonen
- Innendetektor – vertexnahe Messung von geladenen Teilchen
- Crystal-Barrel-Detektor – Kalorimeter zur Messung von ungeladenen Teilchen über die Messungen der Photonen im Endzustand
- Vorwärtsdetektor – Messung von geladenen und ungeladenen Teilchen mit hoher Rate in Richtung des Vorwärtsboosts unter Winkeln zwischen 30° und 12°
- Momo-Detektor – Detektierung von geladenen Teilchen in Vorwärtsrichtung
- Mini-TAPS-Detektor – Messung von geladenen und ungeladenen Teilchen im Winkelbereich zwischen 12° und 1,5°
- Flugzeitspektrometer – Detektor zur Messung der Flugzeit geladener Teilchen, insbesondere langsamer Protonen
- Photonenintensitätsmonitor – Messung des Photonenflusses hinter dem Experiment zur Bestimmung des Gesamtphotonenflusses

Aufgrund des experimentellen Aufbaus als Fixed-Target-Experiment ergibt sich mit einem auf das Target einlaufenden Photonenstrahl, dass die in der Photoproduktion erzeugten Teilchen in Vorwärtsrichtung gehäuft auftreten. Diese Häufung ergibt sich aus der Impulserhaltung z.B. in der zur untersuchenden Reaktion:

$$\gamma \mathrm{p} \rightarrow \mathrm{p}\,\pi^0\pi^0 \rightarrow \mathrm{p} + 4\gamma \tag{2.2}$$

In Abbildung 2.3 ist zu sehen, dass die Anzahl der Photonen im Vorwärtswinkelbereich kleiner 30° in der Doppelpionproduktion eine starke Häufung zeigt und vor allem die Protonenverteilung im Winkelbereich von kleiner 30° das Maximum erreicht. Das Crystal-Barrel-Kalorimeter deckt den Winkelbereich bis hinunter zu 30° in Vorwärtsrichtung ab. Für den Bereich kleinerer Winkel werden aufgrund der Akzeptanzlücke Vorwärtsdetektoren benötigt, die diesen Winkelbereich nach Möglichkeit vollständig abdecken. Hierzu wird im zukünftigen Aufbau der neue Vorwärtsdetektor in Kombination mit weiteren Detektoren zum Einsatz kommmen.

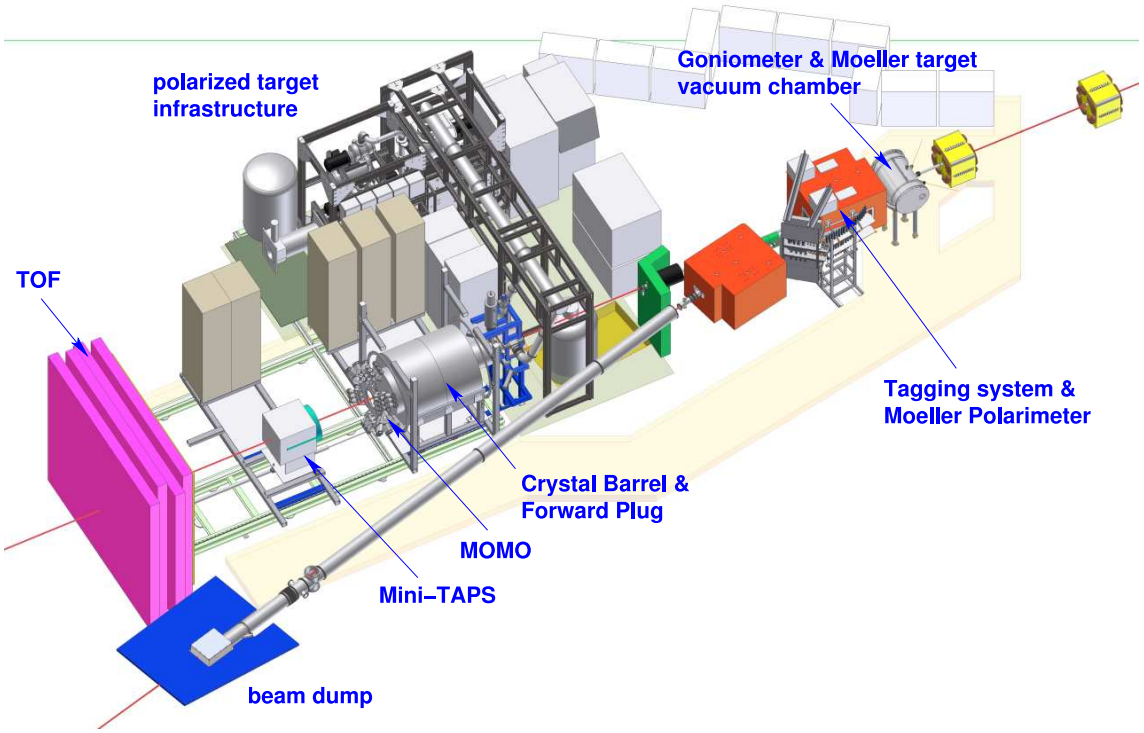


Abbildung 2.2 CAD-Zeichnung des zukünftigen Aufbaus mit Crystal-Barrel-Detektor, Tagging-System, Vorwärtsdetektoren, Target und Strahlvernichter.

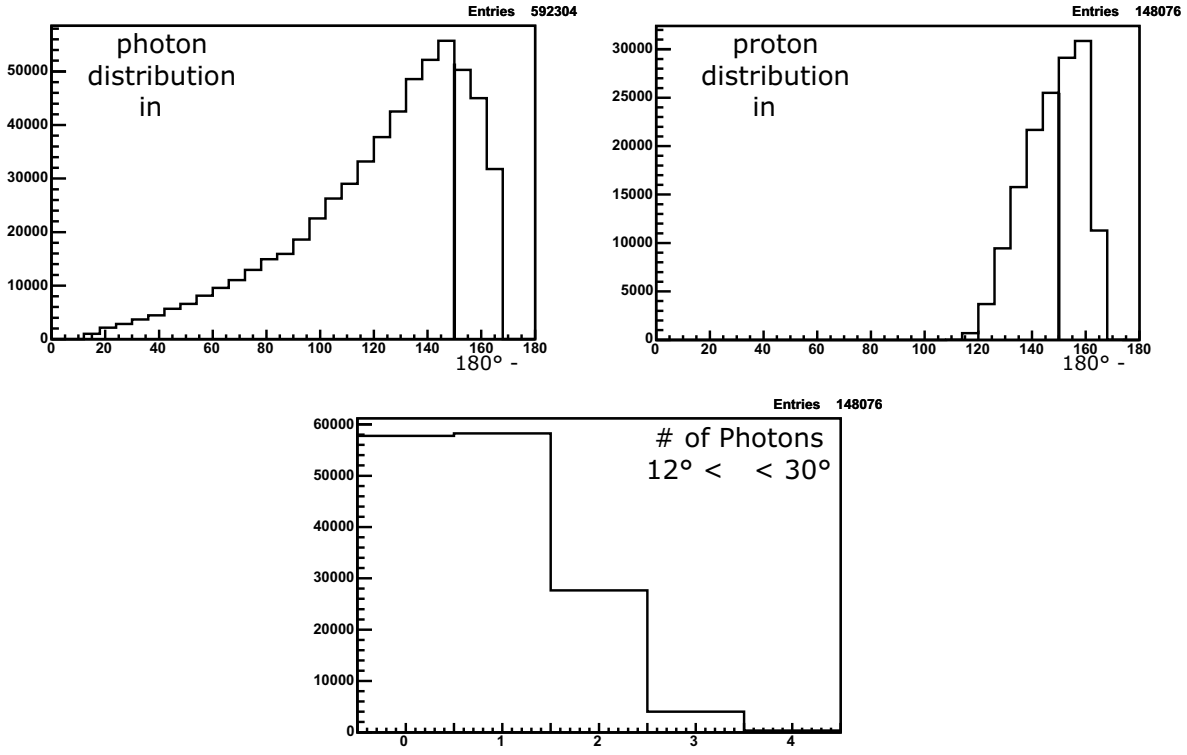


Abbildung 2.3 θ-Verteilung von Protonen und Photonen in der Doppelpionproduktion aus Monte-Carlo-Simulationen in der Reaktion $\gamma p \rightarrow p\pi^0\pi^0$.

2.3 Radiator-Target

Mit dem Radiator-Target wird aus dem von ELSA einlaufenden Elektronenstrahl ein Photonenstrahl erzeugt. Die hochenergetischen Elektronen mit Energien von bis zu 3,5 GeV treffen dabei auf einen Atomkern und werden im Coulombfeld des Kernes abgebremst, wobei sie Bremsstrahlung emittieren. Der Radiator verfügt über verschiedene amorphe Radiator-Targets, die über eine Drehscheibe in den Elektronenstrahl gefahren werden können. Die Radiatortargets unterscheiden sich in der Dicke, welche in Strahlungslängen[16] angegeben wird. Es stehen Targets mit 1/100, 3/100 und 1/1000 Strahlungslänge zur Verfügung. Im zentralen Bereich befindet sich ein Diamantkristall, mit dem linear polarisierte Photonen durch Braggstreuung erzeugt werden können. Der Diamant ist über die 5 Achsen des Goniometers hochpräzise positionierbar. Mit dem Kristall sind Linearpolarisationen von bis zu 50 % erreichbar, wobei der Grad der Polarisation energieabhängig ist.

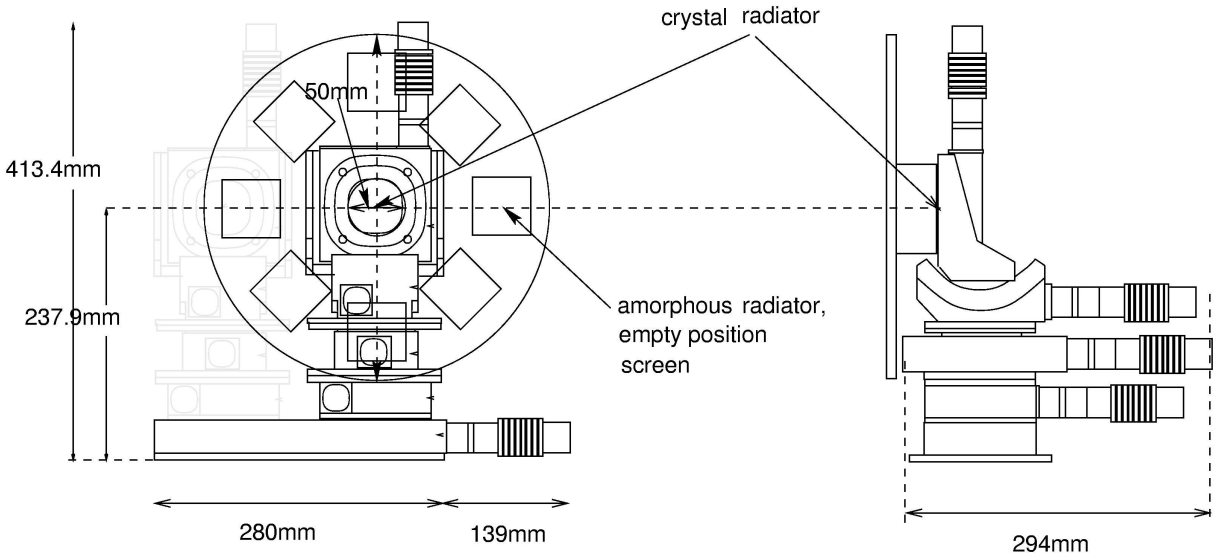


Abbildung 2.4 Das Goniometer

2.4 Tagging-System

Das Tagging-System dient der Energiemarkierung (tagging) der im Radiator erzeugten Photonen. Die durch Bremsstrahlungsprozesse erzeugten Photonen und die abgebremsten Elektronen erreichen das Magnetfeld des Taggingmagneten. Die Elektronen werden aufgrund ihrer Ladung im Magnetfeld abhängig vom Impuls abgelenkt. Hierbei gilt unter der Vorraussetzung, dass die Bewegungsrichtung senkrecht zum Magnetfeld B ist, für den Elektronenimpuls p_e

$$p_e = e \cdot B \cdot r \tag{2.3}$$

.

[16] Eine Strahlungslänge entspricht der Länge Material, bei der ein eintreffendes Photon die Hälfte seiner Energie in Bremsstrahlungs- oder Paarbildungsprozessen verloren hat.

Die Elektronen werden somit auf eine Kreisbahn mit dem Radius r abgelenkt.

Aufgrund des Energieverlustes der Elektronen beim Bremsstrahlungsprozess und mit Kenntnis der ursprünglichen Strahlenergie E_0 der Elektronen des Primärstrahls lässt sich über eine Positionsbestimmung die Energie der erzeugten Photonen über

$$E_\gamma = E_0 - E_e \tag{2.4}$$

bestimmen. Die Messung der Position des Elektronenstrahls erfolgt mit dem Tagging-Hodoskop. Das Tagging-Hodoskop wurde vom GDH-Experiment übernommen, ist allerdings aufgrund höherer Anforderungen an den Energiebereich deutlich erweitert [For04]worden. Der ursprünglich von den 65 Szintillationszählern abgedeckte Bereich betrug nur etwa 3,4 % bis 31,9 % der Elektronenenergie, welches einer Photonenenergie von 68,1 % bis 96,6 % der Primärstrahlenergie entspricht. Für den Einsatz am CB-TR-Experiment ist ein wesentlich größerer Energiebereich wünschenswert.

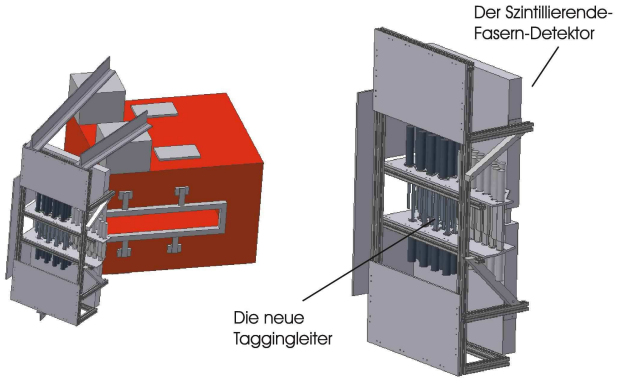


Abbildung 2.5 Das neue Tagging-System

Die unteren 35 Szintillationszähler des GDH-Tagging-Systems werden übernommen. Dies entspricht einem Bereich der niedrigen Elektronenenergien von 3,4 % bis 19 %. Der Bereich mittlerer und hoher Elektronenenergien von 19 % bis 82 % deckt der hochauflösende Szintillationsfaserdetektor zusammen mit alten und neu angefertigten Szintillatoren des GDH-Experiments ab. Der Szintillationsfaserdetektor wurde bereits im CB-ELSA-Experiment verwendet und besteht aus insgesamt 480 Fasern, die in zwei Lagen angeordnet sind. Jeweils 16 der Fasern sind in einem Modul zusammengefasst. Die Fasern haben einen Durchmesser von je 2 mm und sind so angeordnet, dass die Fasern der oberen Lage in der Breite um 1,33 mm mit Fasern der unteren Lage überlappen. Der Detektor verfügt über eine sehr hohe Effizienz von 98 %, eine sehr gute Ratenfestigkeit und eine Zeitauflösung von ca. 300 ps und deckt den gesamten Energiebereich zwischen 19 % und 82 % der Primärstrahlenergie ab. Somit deckt das neue Tagging-System einen wesentlich größeren Energiebereich ab als das System, welches zuvor am GDH-Experiment genutzt wurde und ermöglicht so eine sinnvolle Nutzung am CB-TR-Experiment.

2.5 Møller-Polarimeter

Für ein Doppelpolarisationsexperiment ist es notwendig, die Polarisation des Photonenstrahls und des Targets zu bestimmen. Bei der Erzeugung von Bremsstrahlung an einem Radiator-Target werden aus den von ELSA zur Verfügung gestellten longitudinal polarisierten Elektronen zirkular polarisierte Elektronen erzeugt. Der Helizitätstransfer hängt stark von der Energie der erzeugten Photonen ab [Ols59], sowie vom absoluten Polarisationsgrad des Elektronenstrahls. Der Helizitätstransfer ist durch Formel 2.5 gegeben, wobei der Parameter k dem Verhältnis der Energie der erzeugten Photonen zur Energie der Elektronen $k = \frac{E_\gamma}{E_e}$ entspricht.

$$\frac{p_\gamma}{p_e} = \frac{k(3 + (1 - k))}{3 - (2(1 - k)) + 3(1 - k)^2} \tag{2.5}$$

Die Messung des Polarisationsgrades kann mittels Møllerstreuung [Moe32] erfolgen. Ein Entwurf eines entsprechenden Møller-Polarimeters für CB-Transregio wurde in der Diplomarbeit von Kathrin Fornet-Ponse erstellt [For04]. Dabei werden zwei Targetfolien verwendet. Die Folien sollen aus leicht zu magnetisierenden Materialien bestehen. Zum Einsatz kommt hierbei Reineisen oder Vacoflux. Für die Magnetisierung kann aus Platzgründen kein Helmholtzspulenpaar verwendet werden. Stattdessen wird ein Solenoid genutzt.

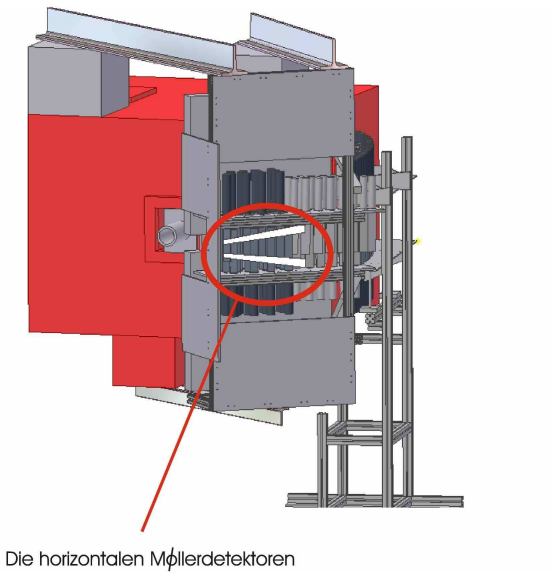


Abbildung 2.6 Møllerdetektoren

Für den Nachweis der gestreuten Møllerelektronen kommen zwei Szintillatorstreifen zum Einsatz, welche zum koinzidenten Nachweis von vertikal gestreuten Møllerelektronen verwendet werden. Die beiden Møllerdetektoren und deren Positionierung sind in Abbildung 2.6 zu sehen. Mittels der Zählratenasymmetrie der beiden Detektoren, der Kenntnis der geometrischen Anordnung der Detektoren und der Targetpolarisation lässt sich die Elektronenstrahlpolarisation bestimmen.

2.6 Target

Für die Experimente am Crystal-Barrel stehen zwei unterschiedliche Targets zur Verfügung. Zum einen ein polarisiertes Target, zum anderen ein unpolarisiertes Flüssigwasserstoff-Target, welches bereits zuvor im CB-ELSA-Experiment und am CB-LEAR-Experiment erfolgreich eingesetzt wurde. Mit dem polarisierten Target und dem bereits erwähnten polarisierten Photonenstrahl (Kapitel 2.3) wird es möglich, Doppelpolarisationsexperimente durchzuführen.

2.6.1 Polarisiertes Target

Um Doppelpolarisationsexperimente durchführen zu können, wird neben dem polarisierten Photonenstrahl ein polarisiertes Target benötigt. Die Arbeitsgruppe um H. Dutz hat hierfür ein Frozen-Spin-Target [Nii76] entwickelt, welches mit flüssigem Helium arbeitet und das Butanol-Target (C_4H_9OH) auf eine Temperatur von 85 mK – nahe das absoluten Nullpunktes – abkühlt. Das Target hat eine Länge von 18,8 mm und einen Durchmesser von 2 cm. Die Dichte bei 1 K beträgt 0,94 g/cm^3.

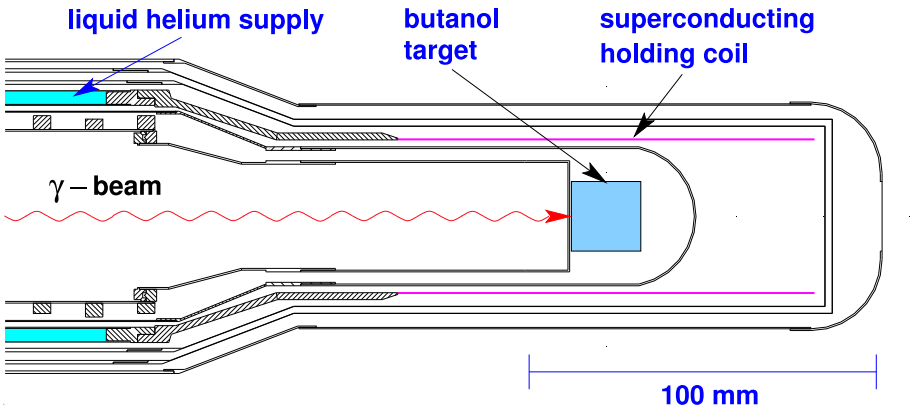


Abbildung 2.7 Polarisiertes Frozen-Spin-Target

Das Target wird durch einen supraleitenden Magneten mit einer Feldstärke von bis zu 2,5 T polarisiert. Es lässt sich ein Polarisationsgrad von über 70 % erreichen. Die Nukleonen beginnen nach der Polarisierung zu relaxieren, wobei bedingt durch die niedrige Temperatur und durch das von einem Haltemagneten [Dut95] erzeugten Feld von 0,3 - 0,6 Tesla diese Relaxationszeit relativ hoch ist. Sie beträgt etwa 200 Stunden für ein Butanol-Target. Die Datennahme muss alle zwei Tage unterbrochen werden, um in einem Zeitraum von zwei Stunden den Polarisationsvorgang mit dem supraleitenden Magneten durchzuführen.

Für die Zukunft ist hier geplant, eine Polarisationsspule direkt in das Experiment einzubauen. Da dies eine sehr hohe technische Anforderung darstellt, konnte eine interne Spule bisher noch nicht realisiert werden.

2.6.2 Unpolarisiertes Target

Das unpolarisierte Flüssigwasserstofftarget ist aus zwei Kühlkreisläufen aufgebaut. Beide Kühlkreisläufe sind über einen Wärmetauscher miteinander verbunden. Im Sekundärkreislauf wird

dem Wärmetauscher, der sich in unmittelbarer Nähe der Targetzelle befindet, gasförmiger Wasserstoff zugeführt. Die Targetzelle selbst besteht aus Kaptonfolie und hat eine Länge von 52,75 mm und einen Durchmesser von 30 mm. Die Folie des Zylinders ist 125 μm dünn. Die Endkappen bestehen aus 80 μm dünnen Kaptonfolien, da diese die Ein- und Austrittsfenster für den Photonenstrahl bilden und man Reaktionen mit der Kaptonfolie nach Möglichkeit vermeiden möchte. Die Strahlungslänge von flüssigem Wasserstoff $X_0(LH_2)$ beträgt 890 cm. Eine genaue und detaillierte Beschreibung des Flüssigwasserstofftargets findet sich in [Kop02].

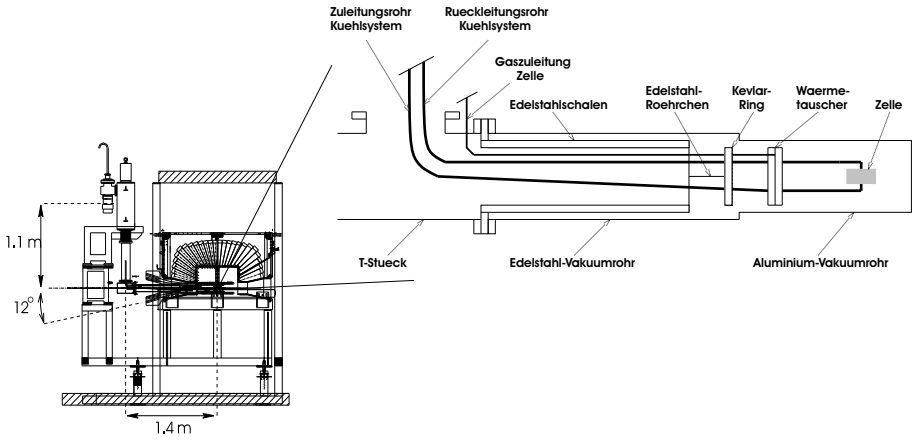


Abbildung 2.8 Flüssigwasserstofftarget

2.7 Innendetektor

Der Innendetektor [Fö00] umschließt zylinderförmig das Target. Er dient dem Nachweis von geladenen Teilchen, die aus dem Target austreten, wobei eine dreidimensionale Rekonstruktion des Durchstoßpunktes durch die spezielle Detektoranordnung möglich wird. Der Detektor besteht aus drei Lagen von insgesamt 513 szintillierenden Fasern mit einem Durchmesser von 2 mm je Faser. Die Fasern verlaufen in der äußersten Lage parallel zur Strahlachse.

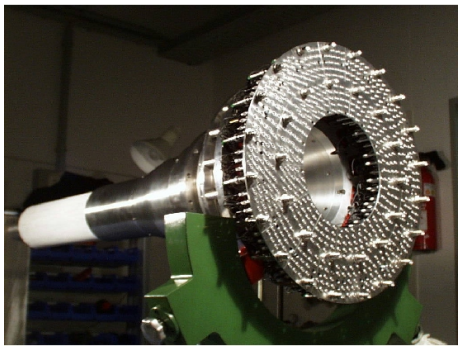


Abbildung 2.9 Bild des Innendetektors vor dem Einbau.

Die beiden darunterliegenden Lagen sind jeweils um etwa 25° zur Strahlachse gekippt, die innere Lage um +25 ° und die mittlere um -25°. Der Abstand von der Strahlachse beträgt für die drei Lagen von innen nach außen: 5,8 cm, 6,1 cm und 6,5 cm. Die Fasern werden über 16 Vielsegment-Photomultiplier[17] ausgelesen. Die Geometrie des Detektors ist so gewählt, dass die im Winkel zur Strahlachse verlaufenden Fasern genau einen Halbkreis beschreiben. Hierdurch werden Mehrdeutigkeiten bei Faseransprechern vermieden und es kann bei drei Ansprechern in getrennten Lagen, beim Durchgang eines Teilchens, der Durchstoßpunkt ohne Mehrdeutigkeiten ermit-

[17] engl. Photomultiplier - Sekundärelektronenvervielfacher - Elektronenröhre mit Photokathode, um schwache Lichtsignale zu verstärken und in ein elektrisches Signal zu wandeln.

telt werden [Bog01]. Mit der Annahme, dass das Teilchen aus dem Target stammt und den Ansprechern im äußeren Kalorimeter, können Cluster im Barrel, nach einer geeigneten projektiven Rekonstruktion, einem geladenen Endzustandsteilchen zugeordnet werden.

2.8 Crystal-Barrel-Detektor

Kernstück des gesamten Experiments ist das Crystal-Barrel-Kalorimeter. Ursprünglich wurde das Kalorimeter am Antiprotonenring LEAR am CERN eingesetzt. Hier wurde die Annihilation von Antiprotonen mit Protonen untersucht. Der ursprüngliche Aufbau des Crystal-Barrel besteht aus 1380 CsI(Tl) Kristallen, die fassförmig um ein zentrales Target angeordnet sind und nahezu den gesamten Raumwinkelbereich von 4 π abdecken (97,8 % des gesamten Raumwinkelbereichs). Die physikalischen Eigenschaften der verwendeten CsI(Tl) Kristalle findet sich in Tabelle 2.1.

Eigenschaft	**Wert CsI(Tl)**
Dichte	4,53 g/cm^3
Strahlungslänge	1,85 cm
Molièreradius	3,8 cm
dE/dx	5,6 MeV/cm
Maximale Emission	550 nm
Abklingzeiten	0,7 μs (dominant) und 7 μs

Tabelle 2.1 Die physikalischen Eigenschaften von CsI(Tl).

Das Kalorimeter wurde so konzipiert, dass Photonen als Zerfallsprodukte der neutralen Mesonen (z.B. π^0 und η) besonders gut detektiert werden können. Die Photonen erzeugen in den Barrelkristallen einen elektromagnetischen Schauer, dessen Energie nahezu vollständig im Kalorimeter deponiert wird, was eine sehr präzise Energiemessung ermöglicht.

Jeder Kristall ist ca. 30 cm lang, was etwa 16,1 Strahlungslängen entspricht. Die Kristalle sind über einen Wellenlängenschieber aus Plexiglas mit einer Photodiode verbunden. Der Wellenlängenschieber passt die größere Grundfläche des Kristalls auf die kleinere Fläche der Photodiode an. Desweiteren wird durch den Wellenlängenschieber das Emissionsmaximum der Kristalle von 550 nm in den Wellenlängenbereich größerer Empfindlichkeit der Photodiode verschoben. Um den leicht hygroskopischen Kristall vor Feuchtigkeit und Licht zu schützen und um zusätzliche Stabilität zu erhalten, sind die Kristalle in dünne Titanhüllen verpackt.

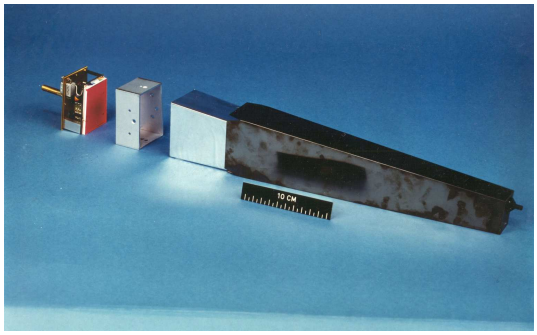


Abbildung 2.10 CsI(Tl) Modul des Crystal-Barrel-Detektors.

Die Photodiode erzeugt aus dem einfallenden Licht einen elektrischen Impuls. Nach einem Vorverstärker erhält man Pulse mit einer Pulshöhe von 1,5 V pro GeV deponierter Energie. Die Anstiegszeit des Signals nach dem Vorverstärker ist relativ groß und beträgt ca. 10 - 15 μs. Die Abklingzeit des Signals ist mit 100 μs etwa um den Faktor 5-10 höher als die Anstiegszeit. Da die relevante Energieinformation nur in der ansteigenden Flanke des Signals liegt, kann über einen elektronischen Shaper das Signal in ein 6 μs langes Signal umgewandelt werden, dessen integrierte Fläche der vollen Energieinformation entspricht. Die minimal detektierbare Energie beträgt etwa 2 MeV und wird durch das elektronische Rauschen der Elektronik bestimmt. Die Signale der Photodioden werden über Fastbus Analog-Digital-Konverter des Typs 1885F der Firma LeCroy ausgelesen.

Die Kristalle sind so geformt und angeordnet, dass sich eine Konfiguration von insgesamt 26 Kristallringen ergibt, die um die Strahlachse angeordnet und radial auf das Target ausgerichtet sind. Von diesen 26 Kristallringen bestehen die mittleren 20 Ringe aus 60 Kristallen, welche jeweils einen Winkelbereich von 6° in ϕ-Richtung und 6° in θ-Richtung überdecken. Die äußeren drei Kristallringe in Vorwärts- und Rückwärtsrichtung verfügen über je 30 Kristalle. Die Kristalle decken hier einen Winkel von 12° in ϕ-Richtung ab, so dass sich die vollen 360° ergeben. Die Öffnungswinkel an den beiden Stirnseiten des Kalorimeters betragen in der ursprünglichen Konfiguration jeweils 12°.

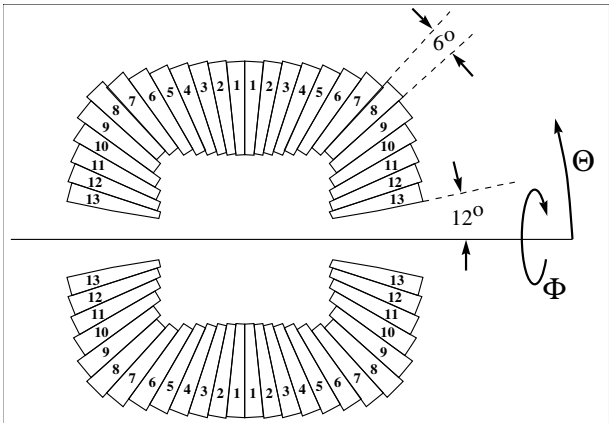


Abbildung 2.11 Schemazeichnung des Crystal-Barrel-Kalorimeters in seiner ursprünglichen Form.

Für die Verwendung am CB-LEAR-Experiment war dieser Aufbau optimal, da hier Proton-Antiproton-Annihilationen in Ruhe untersucht wurden und sich hierbei die Reaktionsprodukte homogen über die zwei Kalorimeterhälften verteilten. Für das Crystal-Barrel-Experiment an ELSA ist dies aufgrund des einlaufenden Photonenstrahls, des ruhenden Targets und der damit verbundenen Kinematik (Lorentz-Boost) so nicht gegeben. Hier werden ca. 70 % der Ansprecher im Bereich kleiner 30° in Vorwärtsrichtung erwartet (siehe Abbildung 2.3).

Aus diesem Grund wurde in den Strahlzeiten von Ende 2001 bis Ende 2003 der TAPS-Detektor im Vorwärtsbereich verwendet. Hierzu wurden die drei äußeren Kristallringe in Strahlrichtung entfernt und der freiwerdende Winkelbereich vom TAPS-Detektor abgedeckt. Da der Detektor seit Ende 2003 nicht mehr zur Verfügung steht, musste der Winkelbereich durch andere

Detektoren ergänzt werden. Hierzu gehören die in Kapitel 2.9 beschriebenen Detektoren in Vorwärtsrichtung.

2.9 Vorwärtsdetektoren

Für den Nachweis von Teilchen unter Vorwärtswinkeln im Bereich kleiner 30° stehen mehrere Vorwärtsdetektoren zur Verfügung, die hier im Weiteren beschrieben werden sollen.

2.9.1 Forward-Plug

Der neue Vorwärtsdetektor besteht aus den 90 Kristallen, die für die Strahlzeit des CB-TAPS Experiments aus der ursprünglichen Kristallanordnung des Crystal-Barrel-Detektors entfernt wurden. Zur ortsauflösenden Detektion von geladenen Teilchen wird vor dem Detektor ein zweilagiger segmentierter Szintillationszähler eingesetzt, der mit den Kristallen eine Detektoreinheit bildet. Der Detektor überdeckt somit den Winkelbereich von etwa 30° bis etwa 12°.

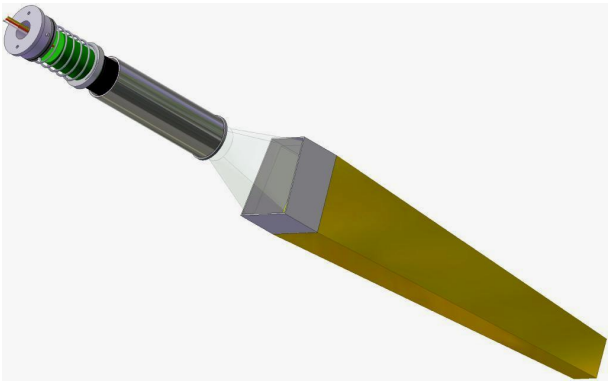
Aufgrund einer zusätzlichen Haltestruktur, welche beim Umbau für die TAPS-Messperiode hinzugefügt werden musste, können die Kristalle nicht mehr an der ursprünglichen Position montiert werden. Hierdurch ist der neue Detektor nicht mehr auf das Target fokussiert. Der Fokus verschiebt sich in Folge dessen um etwa 3 cm und es entsteht ein toter Winkelbereich zwischen 30° und 27,54°.

Die Anforderungen an den neuen Detektor im Vorwärtsbereich sind:

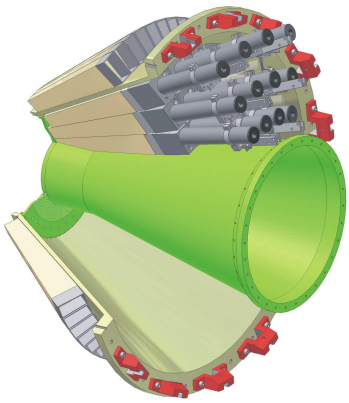
- eine Unterscheidung zwischen geladenen und ungeladenen Teilchen
- hohe Ratenfestigkeit des Detektors bis 1 MHz
- Triggermöglichkeit in der ersten Triggerstufe für ungeladene und geladene Teilchen
- Bestimmung der Multiplizität von geladenen Teilchen zum Zeitpunkt der Triggerentscheidung.

Um diese Ziele zu erreichen kann die ursprüngliche Auslese der Signale nicht mehr mittels Photodioden erfolgen. Wie bereits in Kapitel 2.8 beschrieben, ist hierfür die Anstiegs- und Abklingzeit der elektrischen Signale zu lang. Hier bietet sich die Auslese mittels Photomultiplier an, die für den neuen Vorwärtsdetektor auch realisiert wurde. Bei Photomultipliern liegt die Anstiegszeit des Signals im Bereich von wenigen Nanosekunden und ist nur durch die Kristalleigenschaften limitiert. In Abbildung 2.12 links ist eine Schemazeichnung eines Kristalls mit Auslese über einen Photomultiplier gezeigt.

Für die Detektion von geladenen Teilchen wurde ein neuer Teil-Detektor entworfen [Wen04] und gebaut. Hierfür wurden dünne Szintillatoren vor den Stirnflächen der Kristalle montiert, welche über wellenlängenschiebende Fasern und Vielkanal-Photomultiplier vom Typ Hamamatsu H6568-MOD4 ausgelesen werden. Der Vielfachphotomultiplier zeichnet sich durch ein sehr hohes zeitliches Auflösungsvermögen aus. Durch die Verwendung einer Boosterbasis bleibt die



Schemazeichnung eines Kristalls mit Photomultiplier.



Schemazeichnung des Forward-Plug mit einzelnen Kristallen und Photomultipliern.

Abbildung 2.12 Schemazeichnung Vorwärtsdetektor.

Linearität zwischen der Anzahl der detektierten Photonen und elektrischem Ausgangssignal auch bei hohen elektrischen Ausgangspulsen und großen Raten erhalten [Wen04]. Als Szintillator kommt der Plastikszintillator BC 408 der Firma Bicron zum Einsatz, der eine Zeitkonstante von 2,1 ns besitzt und somit ebenfalls ein sehr schnelles Signal liefert.

Die 3 mm dicken Szintillatorplättchen werden in zwei Lagen vor den 90 Kristallstirnflächen montiert. In den beiden Ebenen befinden sich somit 90 Einzelszintillatoren, wobei die innere Lage der Szintillatoren exakt auf die Stirnflächen der Kristalle passen und die zweite Ebene um die Hälfte einer Kristallbreite gedreht sind. Dadurch wird eine höhere Ortsauflösung erreicht. Jedes der Szintillatorplättchen wird mit zwei Fasern ausgelesen. Dadurch ergeben sich insgesamt 360 Auslesefasern. Die zwei Auslesefasern eines Plättchens werden auf einem Photomultiplierkanal wieder zusammengefasst. Der tote Winkelbereich zwischen 30° und 27,54° wird zur Führung der Auslesefasern der Szintillatordetektoren genutzt. Zur Detektion der geladenen Teilchen wird eine Koinzidenz zwischen den beiden sich überlagernden Ebenen gefordert. Dadurch wird eine höhere Ortsauflösung erreicht als mit einer Ebene alleine. Die Elektronik liefert dann eine getrennte Multiplizität für jeden der von den Szintillatoren überdeckten drei Kristallringe.

Für die Aufbereitung der analogen elektrischen Signale der Photomultiplier von den Vorwärtsdetektorkristallen musste eine neue Analogelektronik entwickelt werden, die die Signale möglichst störungsfrei über eine Distanz von etwa 50 m in die Messhalle überträgt. Dies ist notwendig, da die Elektronik für das Experiment in einer separaten Halle untergebracht ist, um auch während des Betriebs des Experiments Zugang zu den Elektronikkomponenten zu erhalten. Die notwendige Analogelektronik wurde im Rahmen einer Diplomarbeit [Hof04] entwickelt. Die Übertragung erfolgt über Standardnetzwerkkabel der Kategorie 7, die für eine Signalübertragung von Frequenzen bis 600 MHz spezifiziert sind. Die gedrillten Adernpaare sind in diesen Kabeln doppelt geschirmt. Die Analogelektronik besteht aus einem Treibermodul, das direkt am Experiment steht und das Analogsignal zur Übertragung aufbereitet und gleichzeitig einen schnellen Ausgang für den Trigger liefert. Ein zweites Shaper-Modul in der Messhalle empfängt die Analogsignale und bereitet sie mittels einer CR-RC-Formung auf. Die Anstiegszeit der Signale mit einem Prototypen des Shaper-Moduls beträgt zwischen etwa 100 - 300 ns und die

Abklingzeit beträgt zwischen 400 - 600 ns. Damit ergibt sich eine Gesamtpulsdauer von etwa 1 μs. Der Shaper verfügt über drei parallele Ausgänge. Einer der Ausgänge wird auf einen integrierenden ADC gegeben, so wie dies auch für die restlichen 1230 Kristalle geschieht. Ein weiterer Ausgang wird für eine Clusterlogik benötigt, die die Anzahl der detektierten Cluster ermittelt und die Anzahl der Cluster der zweiten Triggerstufe zur Verfügung stellt, so dass es möglich wird, auf eine bestimmte Anzahl an Clustern im gesamten Crystal-Barrel zu triggern. Der dritte Ausgang soll einem Flash-ADC zur Verfügung gestellt werden, der im Gegensatz zu einem integrierenden ADC die Signalform eines Pulses digitalisieren und speichern kann. Diese Arbeit beschreibt die Entwicklung der Ausleseelektronik auf Basis eines Interface mit programmierbaren Chips (FPGAs) und einer PCI-Schnittstelle.

2.9.2 Mini-TAPS-Detektor-Array

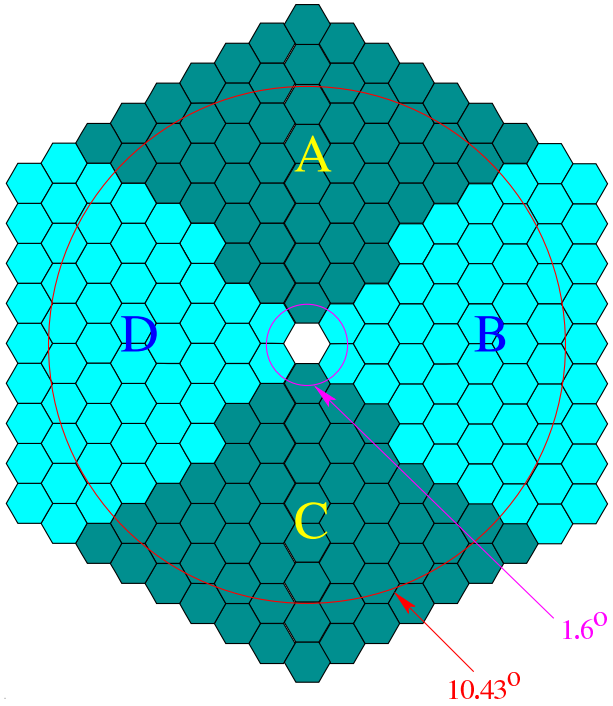


Abbildung 2.13 Die Mini-TAPS-Detektor Kristallanordnung.

Der Mini-TAPS Detektor wird an der Universität Gießen entwickelt und soll den Winkelbereich zwischen 12° bis hinunter zu 1° abdecken. Der Detektor besteht aus 216 Modulen des TAPS-Detektors, der bereits im vollen Aufbau zuvor in einer Messperiode im Einsatz war. Die Module haben eine hexagonale Form und sind wabenförmig angeordnet. Sie bestehen aus Bariumfluorid-Kristallen, welche über Photomultiplier ausgelesen werden. Vor jedem Modul ist ein Plastikszintillator angebracht, der als Vetodetektor zur Erkennung von geladenen Teilchen dient. Die Auslese erfolgt hier, wie im Forward-Plug, über eine wellenlängenschiebende Faser, die beidseitig ausgelesen wird.

2.9.3 MOMO-Detektor

Der MOMO-Detektor ist ein weiteres Detektorsystem in Vorwärtsrichtung zur Messung von geladenen Teilchen. Der Detektor deckt einen Vorwärtswinkelbereich von $\pm$13,5° bis zu $\pm$1,3° ab. Er besteht aus 6 Modulen, die jeweils über 112 szintillierende Fasern vom Typ SCSF-81 der Fa. Kuraray mit einem Durchmesser von 2,5 mm verfügen. Die Eigenschaften der Fasern sind in Tabelle 2.2 aufgeführt. Die Fasern werden über Vielkanal-Photomultiplier vom Typ Hamamatsu R4760 ausgelesen.

Eigenschaft	Wert
Brechungsindex Core	1,59
Brechungsindex Cladding	1,42
Grenzwinkel Θ	63,3°
Abschwächungslänge λ	227 $\pm$ 6 cm
Maximum des Emissionsspektrums	437 nm
Claddingdicke	75 μm

Tabelle 2.2 Die physikalischen Eigenschaften der SCSF-81 Fasern der Fa. Kuraray.

2.9.4 Flugzeitwand

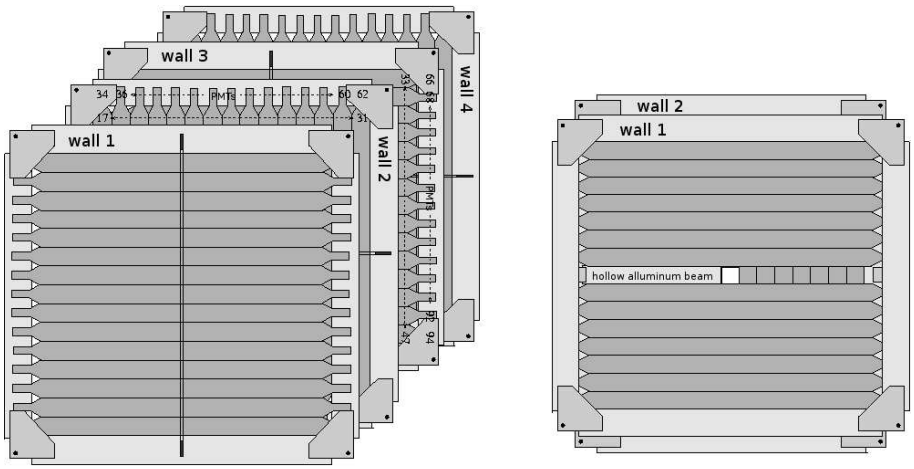


Abbildung 2.14 Schemazeichnung der Flugzeitwand.

Mit der Flugzeitwand [Hö00] lassen sich geladene Reaktionsprodukte und deren Trajektorien in Vorwärtsrichtung erfassen. Jede der vier Flugzeitwände besteht aus 14 großflächigen Szintillatoren, die auf beiden Seiten mit Photomultipliern ausgelesen werden. Die Szintillatoren sind so angeordnet, dass die Szintillatoren der nachfolgenden Wand um 90° gedreht sind. Bei einem Durchflug eines Teilchens durch die vier Wände lässt sich somit die Teilchenbahn sehr genau verfolgen. Jeder Szintillator ist 3 m lang, 20 cm breit und 5 cm dick. Durch die beidseitige Auslese mit Photomultipliern und Laufzeitmessungen der Lichtimpulse zu den beiden Enden

lässt sich die Position eines detektierten Teilchens auf bis zu 5 cm genau bestimmen. In der Mitte jeder Wand ist ein Freiraum von 20 cm zwischen den Szintillatoren freigelassen. Durch das sich ergebende Loch von 20 cm x 20 cm kann der primäre Photonenstrahl diesen Bereich ungehindert passieren.

2.9.5 Spurverfolgungsdetektor

Im Rahmen des Teilprojektes B1 des Sonderforschungsbereiches Transregio 16 soll ein hochauflösender Spurverfolgungsdetektor in der zweiten Förderperiode realisiert werden. Alternativ zum Mini-TAPS-Detektor soll hier der bereits erwähnte Forward-Plug in Kombination mit einem szintillierenden Faserdetektor und zwei Strawtube-Kammern zum Einsatz kommen. In Vorwärtsrichtung ist hier ebenfalls die TOF-Wand im Einsatz. Der zukünftige Aufbau ist in Abbildung 2.15 gezeigt. Dieser Aufbau eignet sich ideal für die Untersuchung von Reaktionen, in denen Kaonen in Vorwärtsrichtung auftreten.

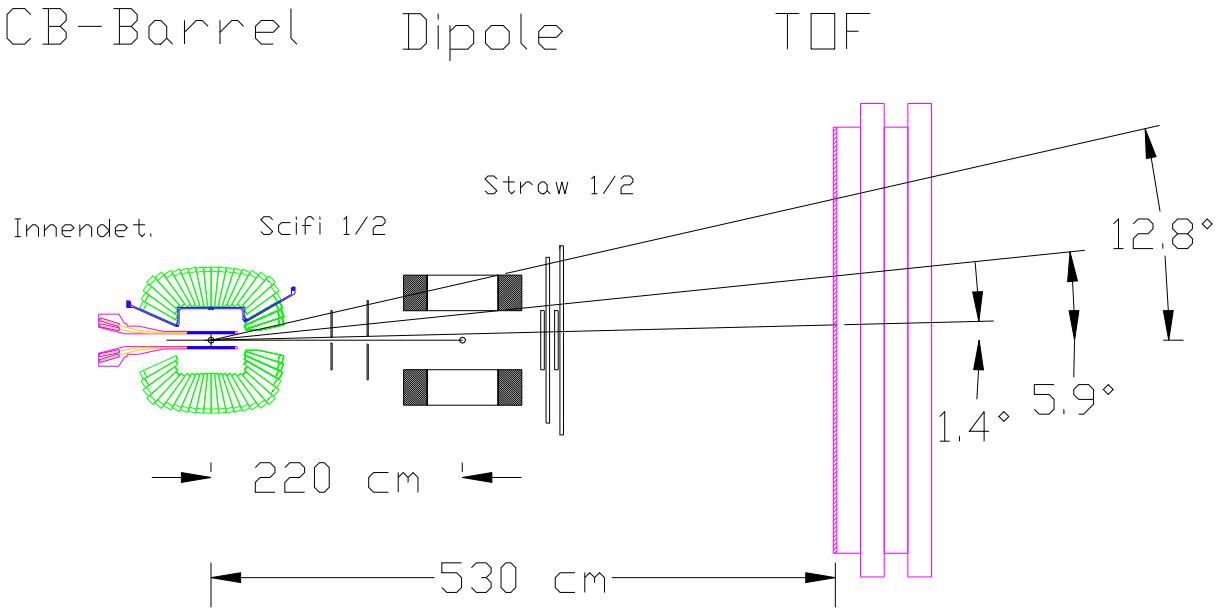


Abbildung 2.15 Alternativer Aufbau des Crystal-Barrel-Experiments mit einem Detektor zur Spurverfolgung geladener Teilchen im Rahmen des B1-Projektes.

2.10 Photonenintensitätsmonitor

Nicht alle am Radiator erzeugten Photonen lösen eine Reaktion im Flüssigwasserstofftarget aus. Die Anzahl der Photonen, die eine hadronische Reaktion im Target auslösen beträgt nur etwa 0,1 ‰, aufgrund der Targetdichte und der Größe der Reaktions-Wirkungsquerschnitte. Der überwiegende Anteil der Photonen passiert das Target und trifft am Ende des Experiments auf einen Strahlvernichter. Um den Photonenfluss genau zu bestimmen, ist es notwendig, die Anzahl der Photonen zu bestimmen, die das Target ohne Reaktion wieder verlassen. Hierzu dient der Photonenintensitätsmonitor [Kon01], der vor dem Strahlvernichter platziert ist. Dieser muss

auch bei sehr hohen Photonenraten entsprechend korrekt arbeiten und die Nachweiseffizienz darf sich auch über die Betriebszeit nicht ändern.

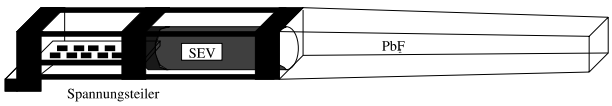


Abbildung 2.16 Modul des Photonenintensitätsmonitors.

Der eingesetzte Detektor besteht aus 16 Modulen, die in einer 4x4-Matrix angeordnet sind. Jedes Modul enthält einen Bleifluorid-Kristall, der über einen Photomultiplier ausgelesen wird. Bei der Verwendung von Bleifluorid entsteht im Kristall Tscherenkowlicht. Bei der Kombination des Kristalls mit einem Photomultiplier können elektronische Signalbreiten von 20 ns erreicht werden, so dass eine maximale Datenrate von 50 MHz je Einzelkristall möglich wird. Durch die Anordnung der Kristalle in Matrixform kann auch eine Positionsbestimmung des Photonenstrahls durchgeführt werden.

Bei dauerhaft hohen Photonenraten verliert der Kristall an Nachweiseffizienz. Der Effizienzverlust beginnt allerdings erst nach einigen Wochen kontinuierlichen Betriebs im Photonenstrahl und kann durch Ausheilung mit Halogenlampen relativ schnell rückgängig gemacht werden. Um das Experiment ohne Unterbrechung betreiben zu können, stehen die 16 Kristalle in doppelter Ausführung zur Verfügung, so dass eine Kristallanordnung ausheilen kann, während die andere in Betrieb ist.

2.11 Datenakquisitionssystem

Die Datenerfassung des Experimentes sorgt dafür, dass mehrere tausend elektronische Kanäle der Subdetektoren digitalisiert, ausgelesen und gespeichert werden. Zentraler Bestandteil der Auslese eines jeden Subdetektors ist eine VME-Bus-CPU[18] (sogenannter lokaler Eventbuilder), die über den VME-Bus alle elektronischen Kanäle des Subdetektors auslesen kann. Für die Auslese von Elektronikkomponenten, die nicht auf VME basieren, sind entsprechende Interfacemodule vorhanden, die den Zugriff über den VME-Bus erlauben. Hierzu gehören zum Beispiel Interfacemodule für den Zugriff auf den CAMAC-Bus oder Fastbus-Komponenten. Die Synchronisation zwischen den autonomen VME-CPUs geschieht über ein selbstentwickeltes VME-Synchronisations-Modul (siehe Anhang C). Die Steuerung der Synchronisation erfolgt dabei durch eine separate CPU (globaler Eventbuilder), die neben der Kontrolle der Synchronisation auch die Triggerverwaltung und Triggersteuerung übernimmt. Die Daten der lokalen Eventbuilder-CPUs werden über Ethernet[19] an eine zentrale Doppelprozessormaschine (Eventsaver) übergeben, dort zu Ereignissen zusammengefasst und zur späteren Analyse auf einem Datenspeichersystem abgelegt. Die Struktur der Datenerfassung am CB-ELSA-Experiment ist in Abbildung 2.17 zu sehen.

[18] VME bezeichnet einen Busstandard, der 1980 als prozessorunabhängiger Busstandard definiert wurde und aus dem VERSAbus der Firma Motorola hervorging.

[19] 100-MBit-Ethernet-Netzwerk

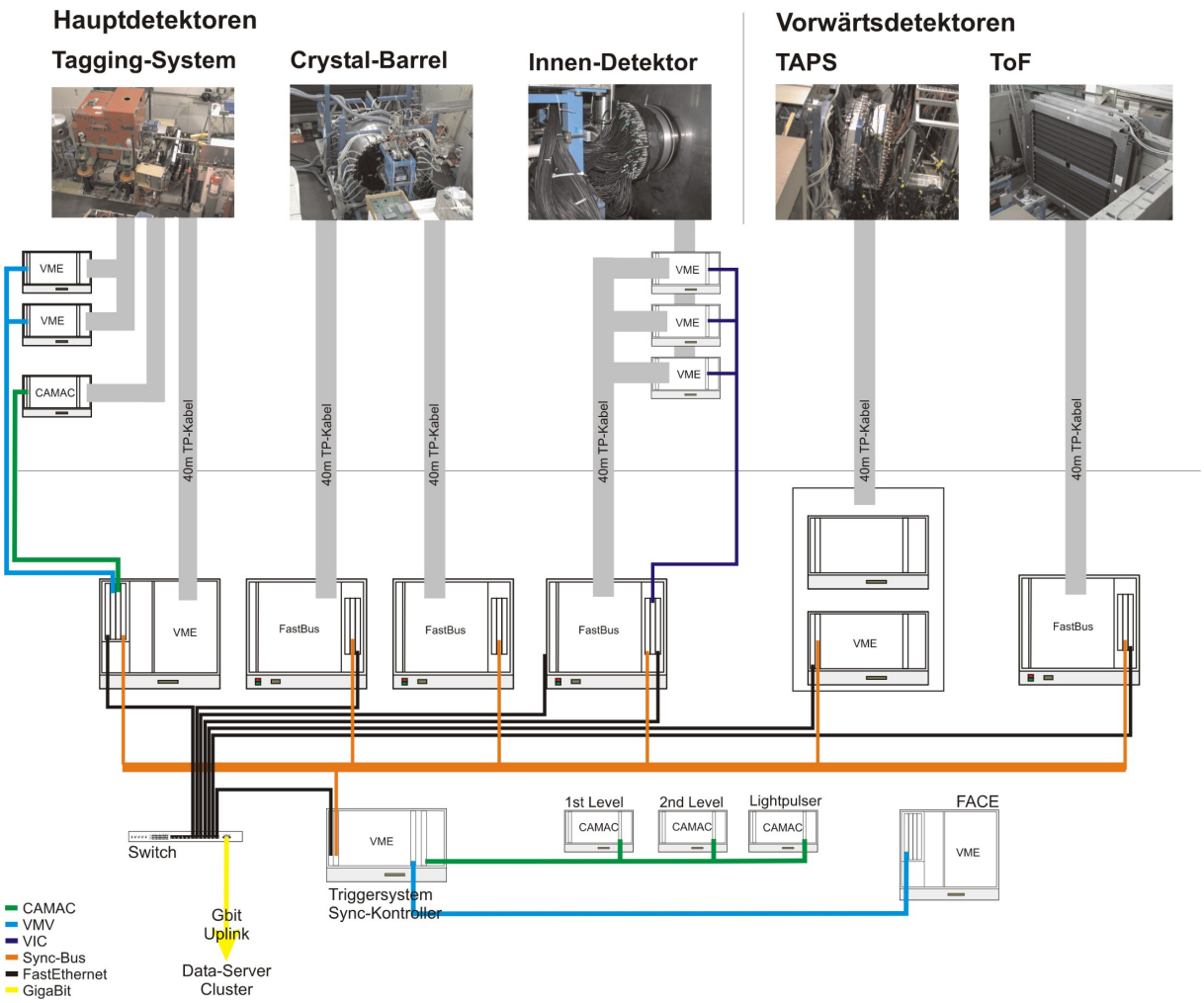


Abbildung 2.17 Schematische Darstellung des Datenauslesesystems des CB-ELSA Experiments.

Für das Datenerfassungssystem werden, soweit dies möglich ist, Standardkomponenten und Industrie-Technologien genutzt. Zum Einsatz kommt hierbei für die Datenübertragung ein Netz auf Basis von Ethernet, welches sich in den letzten Jahren zu einer Standardtechnologie im Bereich von lokalen und mittlerweile zunehmend auch von Stadt- oder Weitverkehrsnetzen entwickelt hat. Die verwendeten Übertragungsprotokolle basieren auf den Standards TCP/IP. Softwareseitig ist durch die Implementierung in objektorientierten Programmiersprachen wie C++ ein sehr modulares Softwarekonzept realisiert worden [Sch04]. Hierdurch sind Anpassungen an zukünftige Entwicklungen des Detektorsystems sehr einfach und effizient möglich. Die Software wurde im Rahmen dieser Dissertation an die neue Compilergeneration GCC 3.x angepasst und begleitend zur Programmierung der DAQ-Software wurde der Einsatz der GNU-Entwicklungswerkzeuge autoconf und automake eingeführt.

Die verwendeten VME-Prozessoren basieren auf Intel-CPUs und werden, wie die anderen Datenerfassungsmaschinen, unter Debian Linux[20] betrieben. Die langen Veröffentlichungszyklen von Debian sind von Vorteil, da hierdurch Änderungen nur relativ selten auftreten (Releasezy-

[20] Die eingesetzte Distribution ist derzeit Debian Sarge 3.1.

klen zwischen ca. 2 und 3 Jahren) und im Normalfall das Debian-Projekt Rückwärtskompatibilität zu alten Softwarestandards beibehält. Dies ermöglicht eine nahezu reibungslose Migration zwischen aufeinanderfolgenden Versionen. Die Umstellung der Datenerfassungsoftware von SuSE Linux auf Debian Linux und auf die neue GNU-Compiler-Generation wurden im Rahmen dieser Arbeit durchgeführt. Zum Einsatz kommt der Linux Kernel 2.4, da hierfür der entsprechende Treiber für die VME-Bridge bereits vorhanden war und im Produktivbetrieb bereits unter dieser Kernelversion eingesetzt wurde. Desweiteren wurde im Rahmen dieser Arbeit ein Synchronisations- und Interruptmodul sowie eine graphische Kontrollsoftware entwickelt, welche als zentrale Steuerungs- und Kontrollelemente in der Datenerfassung eingesetzt werden. Eine kurze Dokumentation der wichtigsten Eigenschaften und Funktionen ist im Anhang B, Anhang C und Anhang D zu finden.

Mit dem hier beschriebenen Aufbau und der Datenerfassung wurden in den vergangenen Jahren Messungen, unter anderem in dem Kanal $\gamma p \rightarrow p\pi^0\pi^0$, durchgeführt. Die Ergebnisse der Doppelpionproduktion und einer Partialwellenanalyse der aufgezeichneten Daten sollen im Folgenden im Vergleich zu theoretischen Vorhersagen und anderen Experimenten vorgestellt werden.

3 Photoproduktion $\gamma\mathrm{p} \rightarrow \mathrm{p}\pi^0\pi^0$

Für die Suche nach bisher noch nicht gefundenen Baryonresonanzen eignen sich verschiedene Reaktionskanäle. Bisher untersucht wurden allerdings zumeist Reaktionen, in denen einzelne Mesonen produziert werden, wie z.B. die Reaktion $\gamma p \rightarrow p\pi^0$, $n\pi^+$, $p\eta$ oder Reaktionen mit Strangeness ΛK^+, $\Sigma^0 K^+$, $\Sigma^+ K^0$. Im Rahmen des Crystal-Barrel-Experiments wurden bisher die Photoproduktion neutraler Pionen [Bar05] und von η-Mesonen [Cre05] untersucht. Mittels einer Partialwellenanalyse dieser Daten unter Berücksichtigung weiterer Datensätze anderer Kollaborationen konnten die Eigenschaften vieler N^*- und Δ^*-Resonanzen [Ani05, Sar05] und ihre Kopplungen an Nπ, Nη, ΛK und ΣK bestimmt werden. Evidenz für 4 neue Resonanzen $D_{15}(2070)$, $P_{11}(1840)$, $D_{13}(1875)$ und mit schwächerer Evidenz $D_{13}(2170)$ wurde ebenfalls gefunden. Bei höheren Energien werden Multimeson-Endzustände immer wichtiger. Da im Bereich höherer Energie ebenfalls die „fehlenden“ Resonanzen liegen, sind diese von besonderem Interesse. Nach Vorhersagen sollen zudem viele dieser Resonanzen an den Kanal $\Delta\pi$ koppeln, was Photoproduktion von zwei Pionen besonders interessant macht. Zum einen kann man hoffen, noch weitere Resonanzen zu entdecken, zum anderen ist die Dynamik der Zerfallsprozesse angeregter Baryonen weitgehend unbekannt. Einen guten Einstieg in die Untersuchung der Doppelpionproduktion stellt die Reaktion $\gamma p \rightarrow p\pi^0\pi^0$ da. Sie bietet gegenüber der Reaktion $\gamma p \rightarrow p\pi^+\pi^-$ entscheidende Vorteile: ein großer Beitrag zur Photoproduktion von $\pi^+\pi^-$-Paaren im höheren Energiebereich stammt von der diffraktiven ρ-Produktion, die nicht über Bildung baryonischer Resonanzen abläuft. Da $\rho \rightarrow \pi^0\pi^0$ Zerfälle wegen Erhaltung der Ladungskonjugation verboten sind, trägt dieser Untergrund zu $\gamma p \rightarrow p\pi^0\pi^0$ nicht bei. Ein weiterer Untergrund stammt von der Ladungstrennung $\gamma p \rightarrow \Delta^{++}\pi^-$, dem sogenannten Kroll-Ruderman-Graphen, der bei kleineren Energien dominant ist. Dieser Prozess ist in der $\pi^0\pi^0$-Produktion ebenfalls unterdrückt. Zudem tragen auch t-Kanal-Prozesse oder Born-Terme nur in einem erheblich reduzierten Maß bei. So tritt z.B. der π-t-Kanal-Austausch im $p\pi^0\pi^0$ -Kanal anders als in den Kanälen mit geladenen Pionen nicht bei. Damit ist die Reaktion $\gamma p \rightarrow p\pi^0\pi^0$ für die Untersuchung baryonischer Resonanzen empfindlicher und es existiert hier ein viel größerer Beitrag von resonanten Amplituden. Im Folgenden soll der Kanal der Doppelpionproduktion $\gamma p \rightarrow p\pi^0\pi^0$ vorgestellt werden, der bereits durch Messungen am Mainzer Microtron (MAMI) mit dem TAPS-Detektor und an GRAAL in Grenoble mit dem LAGRANGE-Detektor im unteren Energiebereich mit Photonenenergien bis 820 MeV und 1,5 GeV untersucht wurde. Diese Ergebnisse werden durch zwei theoretische Modelle beschrieben, die ebenfalls vorgestellt werden. An CB-ELSA konnten mit Elektronenenergien von 1,4 GeV und 3,2 GeV Photonen mit Energien bis zu 1,3 GeV und 3,0 GeV erzeugt und für Messungen verwendet werden. Die Analyse der Daten im Hinblick auf die Erzeugung von zwei neutralen Pionen im Endzustand erlaubt die Bestimmung von Zerfallsmoden von Baryonresonanzen über den Zwischenzustand $\Delta\pi^0$ oder andere Zwischenzustände. Die Daten von CB-ELSA erweitern die bisherige Datenbasis um den Bereich höherer Energien, insbesondere um Daten in der dritten und vierten Resonanzregion. Darauf aufbauend kann eine eventbasierte Partialwellenanalyse durchgeführt werden, deren Ergebnisse hier erstmalig vorgestellt werden. Als Konsequenz aus den bisher erzielten Ergebnissen

ergeben sich Anforderungen für neue Experimente.

3.1 Theoretische Modelle

Im Weiteren werden theoretische Modelle für die Doppelpionproduktion erläutert, die die Daten der Experimente beschreiben sollen. Hierbei wurden von zwei verschiedenen Gruppen Modelle erstellt, die beide auf dem Modell vom Lüke und Söding basieren, welches als erstes beschrieben werden soll.

3.1.1 Lüke und Söding

Das Modell von Lüke und Söding [Lü71] beschrieb Reaktionen mit zwei geladenen Pionen im Endzustand. Hierzu verwendet das Modell eine effektive Lagrangefunktion. In einem solchen Modell werden die Hadronen durch elementare Felder beschrieben. Der Zwischenzustand $\Delta\pi$ kann durch den Zerfall einer Resonanz erzeugt werden. In der Analyse von Lüke und Söding zeigt sich jedoch, dass die Reaktion durch den Δ-Kroll-Ruderman-Term (Kontaktwechselwirkung) und einen Pion-Pol-Term dominiert wird. Im Modell werden die in Abbildung 3.1 gezeigten Feynmandiagramme berücksichtigt.

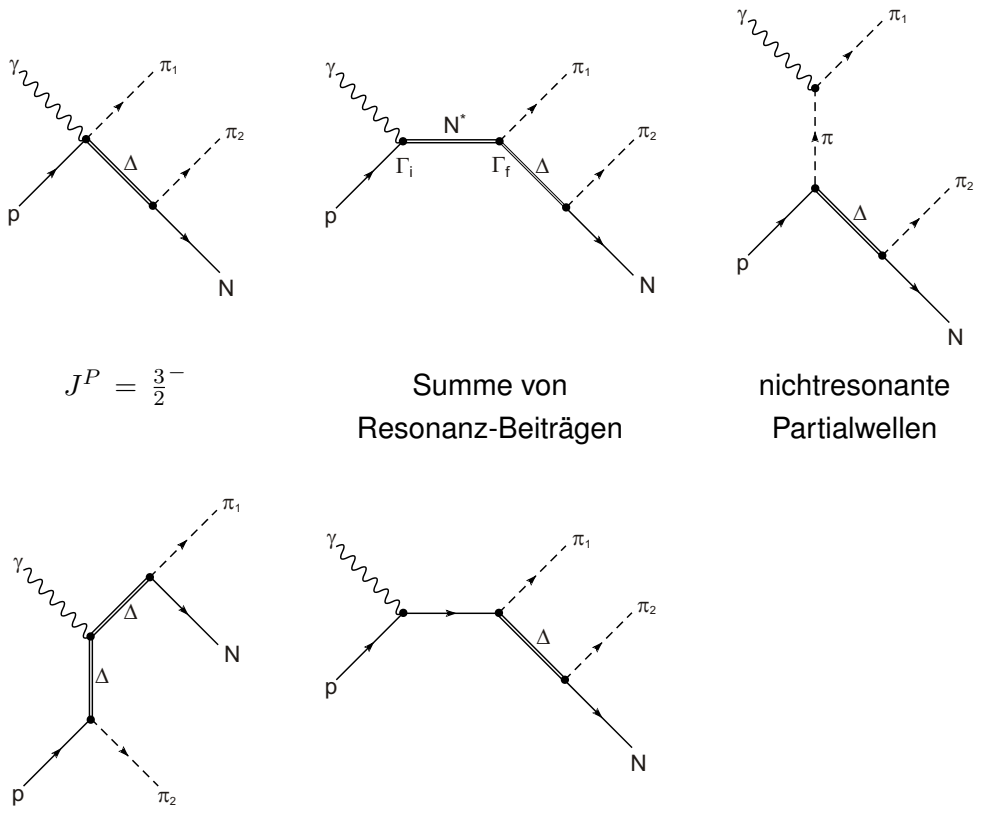


Abbildung 3.1 Im Modell von Lüke und Söding berücksichtigte Feynmandiagramme für Reaktionen vom Typ $\gamma p \rightarrow p\pi^+\pi^-$ [Lü71].

Experimente, welche zwei neutrale Pionen im Endzustand untersuchen, werden erst seit den 1990er Jahren durchgeführt. Hierfür wurden zwei Erweiterungen des Modells von Lüke und Söding entwickelt, welche alle Isospinkanäle berücksichtigten.

3.1.2 Modell von Gómez-Tejedor und Oset

Das Modell von Lüke und Söding wurde in Valencia durch Gómez-Tejedor und Oset erweitert. Hierbei wurden zunächst in einer ersten Version [Gó94] 67 Feynmandiagramme verwendet, und ebenfalls nur die Doppelpionproduktion mit geladenen Pionen untersucht. In einer zweiten Erweiterung des Modells [Gó96] wurden dann alle Isospinkanäle berücksichtigt. Von 67 Diagrammen trägt der überwiegende Teil nicht bei. Die Anzahl der zu berücksichtigenden Diagramme reduziert sich auf 20. Die für die Erzeugung von neutralen Pionen wichtigen Diagramme sind in Abbildung 3.2 gezeigt.

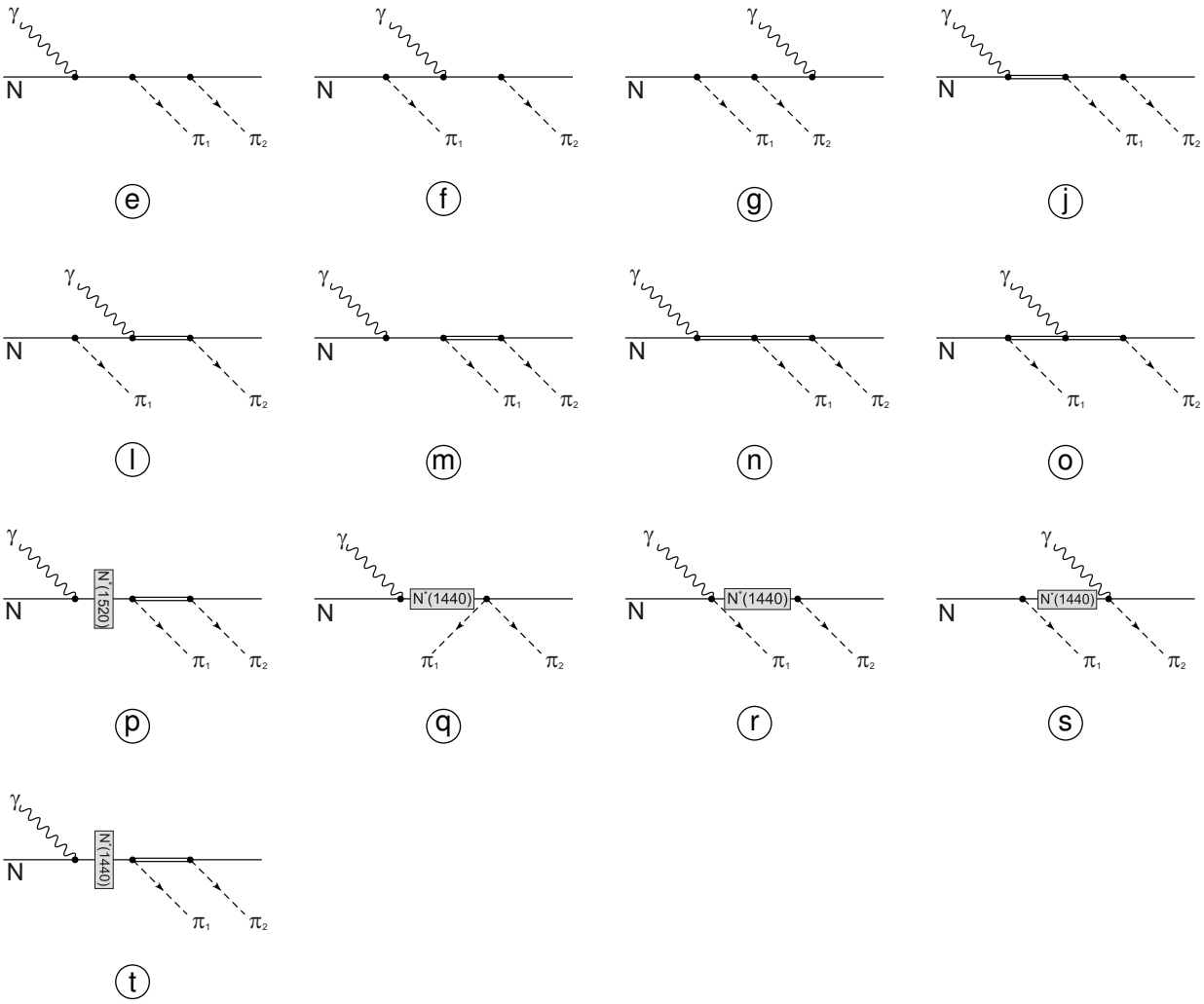


Abbildung 3.2 Für die Doppelpionproduktion von neutralen Pionen relevante Feynmandiagramme im Modell von Gómez-Tejedor und Oset [Gó96]. Bezeichnung nach Gómez-Tejedor und Oset.

Für die Doppelpionproduktion von neutralen Pionen macht das Modell die Vorhersage, dass ein sequentieller Zerfall bei niedrigen Energien (< 800 MeV) zu einem großen Anteil über die $N(1520)D_{13}$ als Zwischenzustand stattfindet. Dieser Zwischenzustand dominiert den totalen Wirkungsquerschnitt im Bereich von 720 MeV. Der Anteil beträgt laut dem Modell 6 μb am berechneten Gesamtwirkungsquerschnitt von etwa 12 μb im Energiebereich um 720 MeV. Die nicht resonanten Amplituden, die die Δ-Resonanz enthalten, tragen bei diesen Energien ebenfalls bei und der Anteil am totalen Wirkungsquerschnitt beträgt hier etwa 25 %. Zu höheren Energien steigt der Anteil der $\Delta(1232)$ moderat an. Der berechnete totale Wirkungsquerschnitt der Reaktion und die Anteile der einzelnen Resonanzen ist in Abbildung 3.3 zu sehen. Um 600 MeV tragen alle Resonanzen in etwa mit gleicher Stärke zum Wirkungsquerschnitt bei. Für sehr kleine Energien liefert die $N(1440)P_{11}$-Resonanz den größten Beitrag.

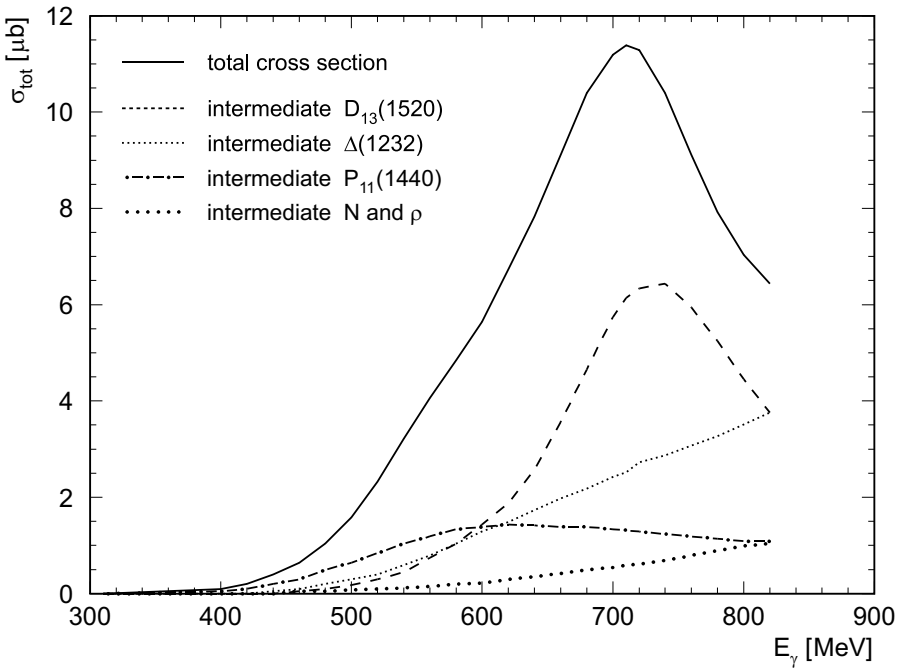


Abbildung 3.3 Totaler Wirkungsquerschnitt der Reaktion $\gamma p \rightarrow p\pi^0\pi^0$ nach Gómez-Tejedor und Oset. Im Modell dominierend ist die Erzeugung einer $N(1520)D_{13}$-Resonanz [Gó94].

3.1.3 Murphy- und Laget-Modell / Laget-Modell

Das Modell von Murphy und Laget [Mur95] erweitert ebenfalls das Modell von Lüke und Söding und dient zur Beschreibung aller Isospin-Kanäle. In Abbildung 3.4 sind die Feynmandiagramme gezeigt, welche für die Produktion von zwei neutralen Pionen relevant sind. Hierbei sind im Wesentlichen die drei Diagramme II a, III a und III b wichtig, da die anderen Terme unterdrückt sind oder keine Rolle für die Produktion von neutralen Pionen spielen (I a, IV a, I c, I d).

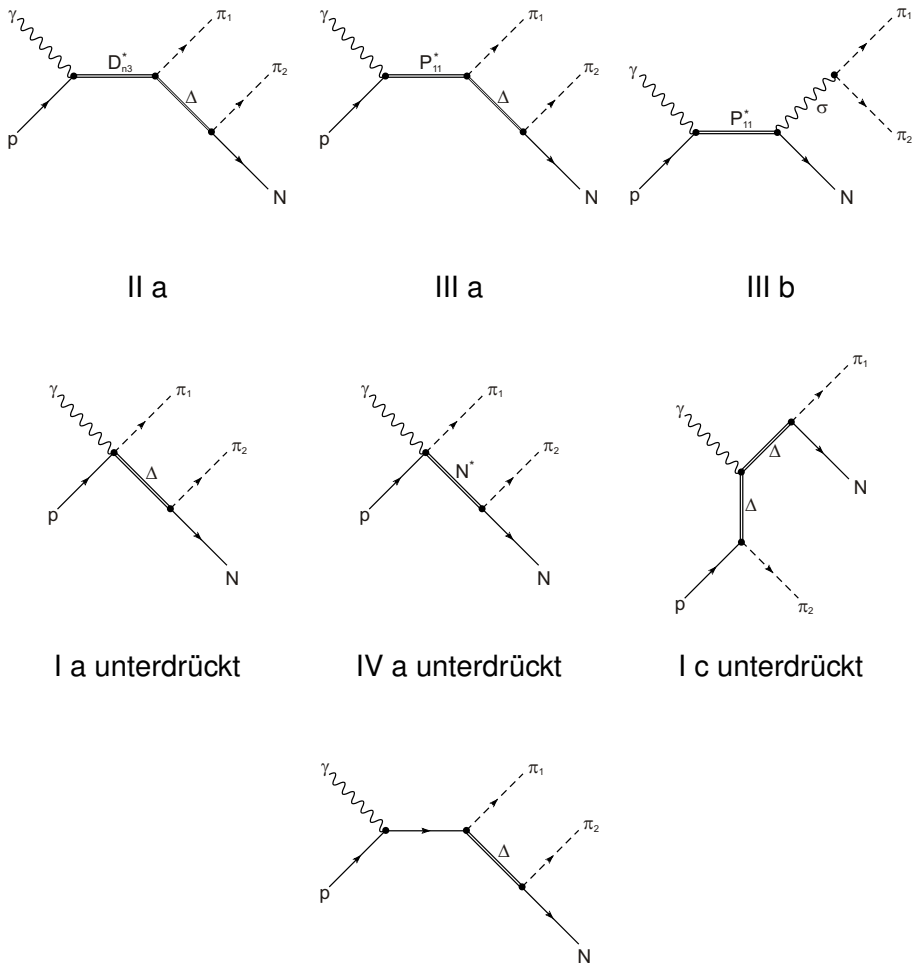


Abbildung 3.4 Feynmangraphen, die im Modell von Murphy und Laget [Mur95] für die Produktion von neutralen Pionen relevant sind. Benennung nach Murphy und Laget Fig. 1.

Das Modell wurde zunächst für die Beschreibung der MAMI-Daten entwickelt. Aus diesem Grund sind in der ersten Version des Modells nur Produktionsmechanismen für Photonenenergien bis 800 MeV berücksichtigt, welches einer Schwerpunktsenergie $\sqrt{s}$ von 1540 MeV entspricht. Im Modell sind somit nur die Resonanzen $N(1440)P_{11}$, $N(1520)D_{13}$ und $N(1700)D_{33}$ berücksichtigt. In Abbildung 3.4 sind die beitragenden Feynmandiagramme zu sehen.

Um auch die Daten von Messungen an GRAAL beschreiben zu können, sind Modifikationen an dem Modell vorgenommen worden (Laget-Modell) [Ass03]. Die Änderungen am Modell berücksichtigen zusätzlich die Anregungen der Resonanzen $N(1710)P_{11}$ und $N(1700)D_{13}$, die in $\Delta^+\pi^0$ zerfallen, wobei das Δ^+ nachfolgend in ein weiteres π^0 zerfällt. Desweiteren wird die Anregung der Resonanz $N(1710)P_{11}$ verwendet, die direkt in ein Proton unter Aussendung eines σ-Mesons zerfällt, welches in $\pi^0\pi^0$ zerfällt, sowie die direkte Emission eines σ-Mesons über einen ρ-Austausch zwischen dem einlaufenden Photon und dem Nukleon des Targets.

Betrachtet man den totalen Wirkungsquerschnitt der Reaktion in Abbildung 3.5, so sieht man, dass im Modell von Murphy und Laget die Anregung der $N(1440)P_{11}$ und der direkte Zerfall in p σ dominierend ist.

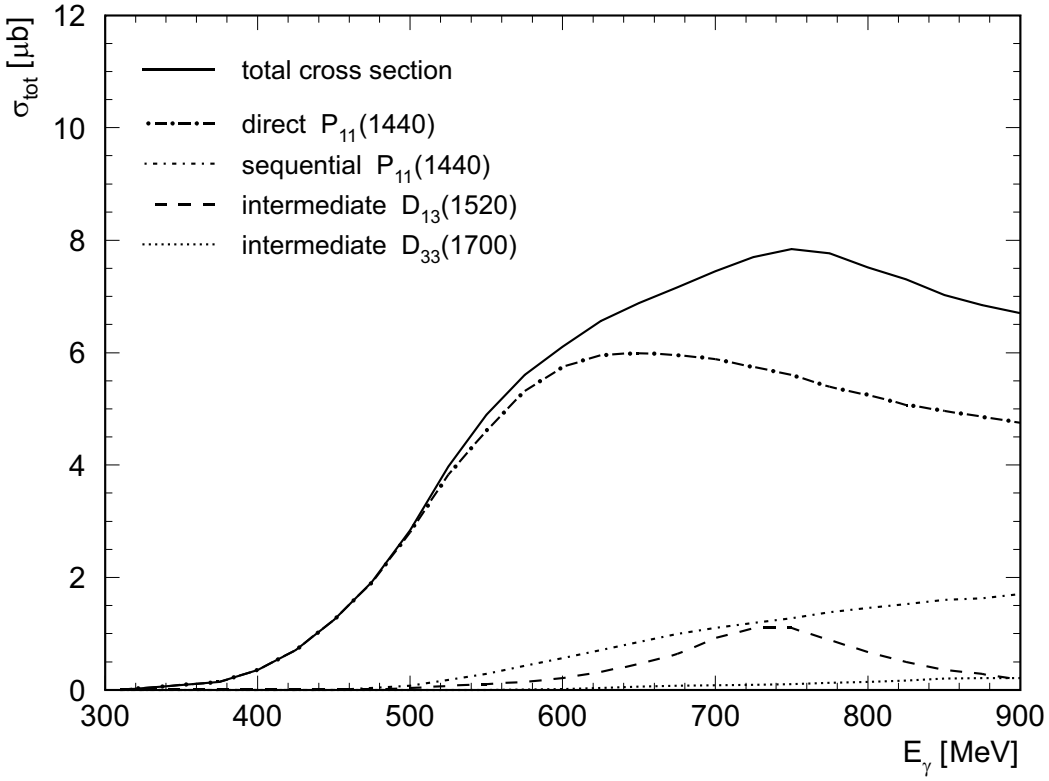


Abbildung 3.5 Totaler Wirkungsquerschnitt der Reaktion $\gamma p \rightarrow p\pi^0\pi^0$ nach Murphy und Laget. Im Modell dominierend ist die Erzeugung einer $N(1440)P_{11}$-Resonanz [Mur95]. Die Ergebnisse des neueren Modells sind in Abbildung 3.11 gezeigt.

3.1.4 Vorhersagen der beiden Modelle

Die beiden Modelle beschreiben den totalen Wirkungsquerschnitt der Reaktion $\gamma p \rightarrow p\pi^0\pi^0$. Die Form des totalen Wirkungsquerschnitts beider Modelle bei niedrigen Energien ist ähnlich. Unterschiedlich ist allerdings insbesondere der Anteil der Resonanzen, die zum Wirkungsquerschnitt beitragen. Das Modell von Gómez-Tejedor und Oset verwendet als dominierenden Anteil den Zerfall über die Baryonkaskade $N(1520)D_{13} \rightarrow \Delta\pi$, das Modell von Murphy und Laget jedoch die Amplitude $N(1440)P_{11} \rightarrow p\sigma$. Hierdurch ergibt sich ein Widerspruch in der Interpretation der Daten. Die Modelle wurden zur Beschreibung der Daten von TAPS an MAMI und LAGRANGE an GRAAL entwickelt. Diese bisherigen Messungen sollen nun vorgestellt werden. Danach werden neue Ergebnisse zur Produktion von neutralen Pionen und eine Partialwellenanalyse dieser Daten vorgestellt, die diesen Widerspruch der Modelle aufzulösen helfen.

3.2 Bisherige Messungen zur Doppelpionproduktion

Durch die Entwicklungen im Bereich der Beschleunigeranlagen und der Weiterentwicklung von Experimenten an diesen, ist es in den letzten Jahren möglichen geworden, neben der klassischen πN-Streuung auch Messungen mit zwei Pionen im Endzustand mit einem großen abgedeckten Raumwinkelbereich mittels Photoproduktion an einem Protonentarget durchzuführen. Hierbei ist insbesondere der neutrale Kanal $\gamma p \rightarrow p\pi^0\pi^0$ von Interesse. Im Folgenden sollen die Experimente vorgestellt werden, die in den letzten Jahren Ergebnisse zu diesem Kanal beigetragen haben.

3.2.1 Messungen an MAMI

Am Mainzer Microtron (MAMI) wurde in den 1990er Jahren aus dem kontinuierlichen Elektronenstrahl des Microtrons mit einer Elektronenenergie von 880 MeV über eine Nickelfolie quasi-monochromatische Bremsstrahlung erzeugt, die Energien im Bereich zwischen 285 MeV und 820 MeV erreichte. Die Energiebestimmung der Photonen erfolgt über ein Tagging-System, welches die Energie der gestreuten Elektronen bestimmte. Als Target für die Photonen dient ein Target mit flüssigem Wasserstoff, an dem die Mesonen erzeugt werden. Das Target hatte hierbei eine Länge von 10 cm und einen Durchmesser von 4 cm. Mit dem TAPS-Photonenspektrometer wurden dann die Reaktionsprodukte gemessen [Hä97]. Der TAPS-Detektor besteht aus 6 Blöcken mit je 62 hexagonalförmigen BaF_2 Kristallen, die in einer 8x8-Matrix angeordnet sind. Die 6 Blöcke waren in einer horizontalen Ebene um das Produktionstarget herum angeordnet. Hierbei bildeten die Blöcke einen Winkel von $\pm$ 54°, $\pm$ 103° und $\pm$ 153° bezüglich der Strahlachse. Der Abstand zum Target betrug 55 cm. Zusätzlich wurde in Vorwärtsrichtung eine Wand mit 138 BaF_2 Kristallen genutzt, die sich im Abstand von 60 cm vom Produktionstarget entfernt befand. Mit diesem Aufbau wurde eine Raumwinkelabdeckung von etwa 40 % erreicht. Der Aufbau an MAMI ist in Abbildung 3.6 zu sehen.

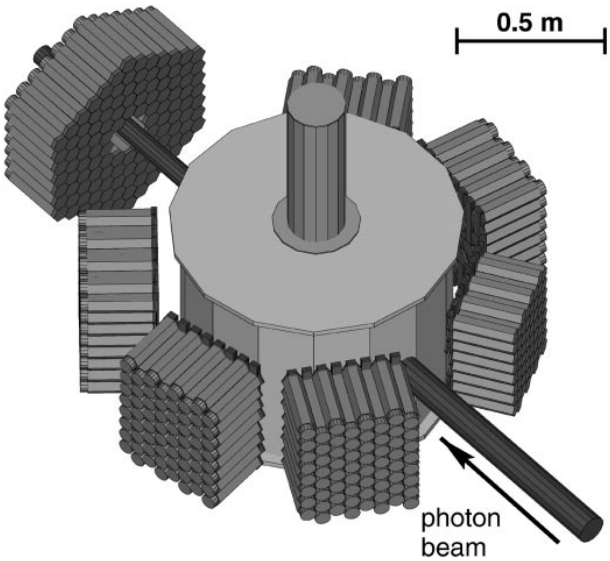


Abbildung 3.6 Aufbau des TAPS-Experiments am Mainzer Microtron (MAMI).

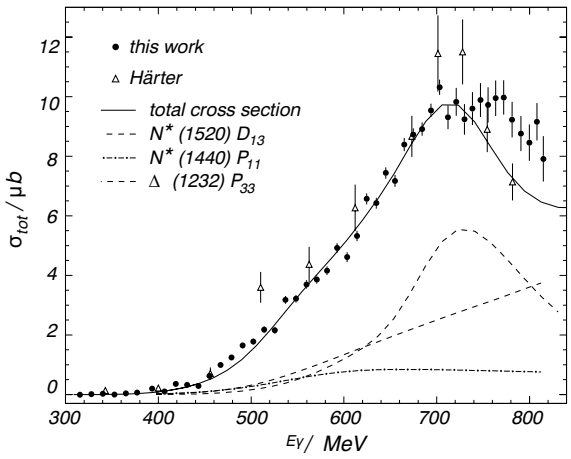


Abbildung 3.7 Totaler Wirkungsquerschnitt der $2\pi^0$ Produktion am Proton, gemessen durch TAPS an MAMI [Wol00].

Zur Untersuchung wurden die 3γ- und 4γ-Endzustände verwendet. Hierbei wurden die 3γ-Endzustände selektiert, falls die Photonen zeitgleich im Detektor detektiert wurden, sich zwei der drei Photonen zu einer invarianten Masse eines π^0 kombinieren ließen und die Energie des einlaufenden Photons weniger als 440 MeV betrug. In der Nähe der Produktionsschwelle für zwei Pionen konnte durch die zusätzliche Verwendung der 3γ-Endzustände die Statistik verbessert werden. Der konkurrierende Prozess $\gamma p \rightarrow p\pi^0\gamma'$ trägt in diesem Bereich mit weniger als 5 nb zum Gesamtwirkungsquerschnitt bei. Im Bereich zwischen 440 MeV und 707,9 MeV wurden nur 4γ-Ereignisse verwendet. Hierbei wurden alle möglichen Kombinationen zwischen den Energien der 4 Photonen gebildet, um über die invariante Masse der Photonen die beiden Pionen zu identifizieren. Oberhalb der Schwelle von 707,9 MeV werden η-Mesonen produziert, die zu etwa 32 % in 3 π^0 zerfallen und bei Verlust eines π^0 aufgrund der Raumwinkelabdeckung von 40 % ebenfalls als 2 π^0-Ereignisse registriert werden können. Der Zerfall der η-Mesonen trägt daher zum Untergrund der zu untersuchenden $2\pi^0$-Reaktion mit bei. Oberhalb der η-Produktionsschwelle wurden aus diesem Grund nur die 4γ-Ereignisse verwendet. In diesem Energiebereich werden die Pionen über die invariante Masse von zwei Photonen identifiziert und die „missing mass“[21] analysiert, die sich in der gewünschten Reaktion

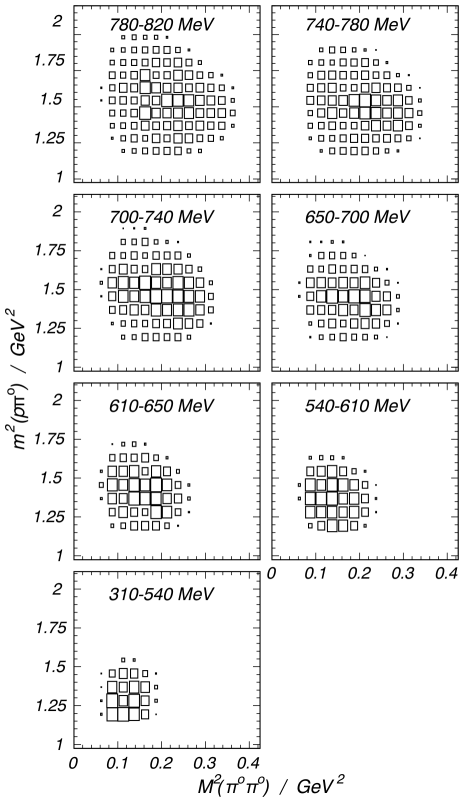


Abbildung 3.8 Akzeptanzkorrigierte Dalitz-Plots. Aufgetragen ist die quadrierte invariante Masse $m^2(\pi^0\pi^0)$ gegen die quadrierte invariante Masse $m^2(\pi^0 p)$ [Wol00].

[21] engl. missing mass - fehlende Masse

zur Masse des Protons ergeben muss. In Abbildung 3.7 ist der bestimmte Wirkungsquerschnitt [Hä97], sowie die neueren Ergebnisse im Vergleich hierzu [Wol00] zu sehen. Der Wirkungsquerschnitt beträgt im Bereich zwischen der Schwelle und 450 MeV weniger als 300 nb. Mit zunehmender Photonenenergie steigt der Wirkungsquerschnitt an, erreicht für 750 MeV ein Maximum von etwa 11 μb und fällt danach wieder ab. Der Wirkungsquerschnitt wird laut den Autoren am besten durch das Modell von Gómez-Tejedor und Oset beschrieben, mit geringen Abweichungen bei den höchsten Photonenenergien.

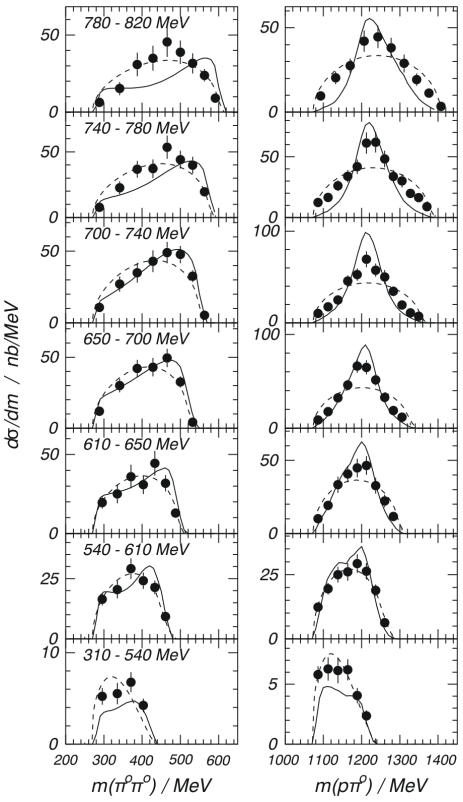


Abbildung 3.9 Invariante Massen $m^2(\pi^0\pi^0)$ und $m^2(\pi^0 p)$ für verschiedene Energiebereiche [Wol00].

Bei der Analyse mittels Dalitz-Plots ist es möglich, Abweichungen der Impulsverteilungen vom normalen Phasenraum „sichtbar“ zu machen. Hierbei wurden die Variablen $m^2(p,\pi^0)$ und $m^2(\pi_1^0,\pi_2^0)$ gegeneinander aufgetragen. Die Dalitz-Plots sind in Abbildung 3.8 zu sehen und zeigen eine Abweichung von einer flachen Verteilung für Photonenenergien größer 610 MeV.

In Abbildung 3.9 sind die invarianten Massen $m(\pi_1^0,\pi_2^0)$ und $m(p,\pi^0)$ dargestellt. Die durchgezogene Linie entspricht dabei dem Modell von Gómez-Tejedor und Oset [Gó96], die gestrichelte Linie markiert den Phasenraum. Die gemessenen Datenpunkte für die invariante Masse der beiden Pionen sind in Übereinstimmung mit dem Phasenraum und auch mit dem Modell. Im Bereich kleinerer Energien gilt dies auch für die invariante Masse des Protons und eines Pions. Bei Energien größer 610 MeV und 650 MeV erkennt man eine Abweichung vom Phasenraum. Diese Abweichung lässt sich durch den Beitrag der $N(1520)D_{13}$ über den Zerfall nach $\Delta\pi^0$ in den Endzustand $p\pi^0\pi^0$ erklären, der auch der dominante Zerfallsmodus im Modell von Gómez-Tejedor und Oset ist.

3.2.2 Messungen an GRAAL

Im Jahr 2003 wurden Ergebnisse von Messungen des LAGRANGE-Detektors veröffentlicht [Ass03], die im γ-Energiebereich zwischen 650 MeV und 1,5 GeV an der Photonenrückstreuanlage GRAAL in Grenoble durchgeführt wurden. Durch die Rückstreuung von Laserlicht an einem Elektronenstrahl kann ein polarisierter Photonenstrahl erzeugt werden. Hierbei steht einmal die Compton-Streuung mit einer grünen Laserlinie für Photonenenergien zwischen 0,6 GeV und 1,1 GeV zur Verfügung, sowie eine UV-Linie des Lasers, die polarisierte Photonen im Energiebereich von 0,8 GeV bis 1,5 GeV erzeugt. Der Polarisationsgrad liegt zwischen 60 % und 98 %.

Die erzeugten Photonen werden energiemarkiert und treffen auf ein Flüssigwasserstoff-Target mit einer Länge von 6 cm. Der Detektor besteht aus einer Kombination von mehreren Subdetektoren. Vertexnah sind zwei zylinderförmige Drahtkammern installiert, die das Produktionstarget umschließen. Danach folgen Plastikszintillatoren und ein rugbyförmiges Kalorimeter umschließt die bisher genannten Detektoren. Das zentrale Kalorimeter deckt 90 % des Raumwinkelbereiches von 4 π ab. Es besteht aus 480 BGO-Kristallen mit einer Länge von 24 cm (entsprechend 21 Strahlungslängen) und detektiert den Zerfall der neutralen Mesonen π^0 und η in zwei Photonen. Die Öffnungswinkel zur Strahlachse betragen je 25°. Der Vorwärtswinkelbereich von 28° wird durch mehrere Vorwärtsdetektoren abgedeckt. Hierbei kommen zwei planare Drahtkammern, sowie in einem Abstand eine doppelte Szintillatorwand und ein Schauerdetektor zum Einsatz, so dass Flugzeitbestimmungen mit den Szintillatorwänden für Neutronen und Protonen durchgeführt werden können. Der Aufbau ist schematisch in Abbildung 3.10 gezeigt.

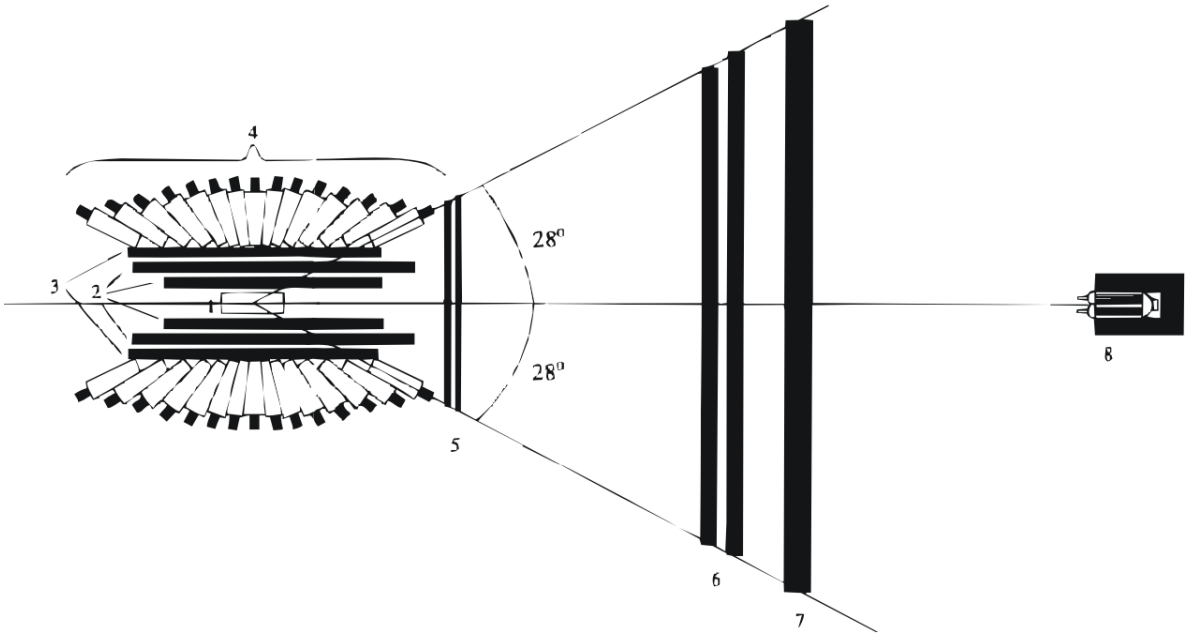


Abbildung 3.10 Aufbau des LAGRANGE Detektors an der Photonenrückstreuanlage GRAAL.

Zur Analyse der Reaktion $\gamma p \rightarrow p\pi^0\pi^0$ wurden die Winkel θ und ϕ des Protons durch das BGO-Kalorimeter gemessen. In Vorwärtsrichtung erfolgt die Messung durch die Szintillatorwände. Als Selektion dienten zum einen die Ereignisse, bei denen 4 Photonen im BGO-Kalorimeter nachgewiesen wurden, sowie alternativ mit einem der Photonen in der Schauerwand in Vorwärtsrichtung. Durch eine fünffache Koinzidenz (4 Photonen und ein Proton) konnte der Untergrund stark reduziert werden. Die zufällige Rate wurde durch die Autoren auf weniger als 1,5 % abgeschätzt. Die Akzeptanzkorrekturen wurden über eine GEANT3-basierte Software durchgeführt. Bei der Korrektur wurden drei kinematische Variablen verwendet: Die Energie des einlaufenden Photons, der Impuls und der Winkel des 2 π^0-Systems. Die Bestimmung ergab eine Effizienz von 30 % im Mittel und eine Extrapolation im nicht abgedecktem Phasenraum von weniger als 3 %.

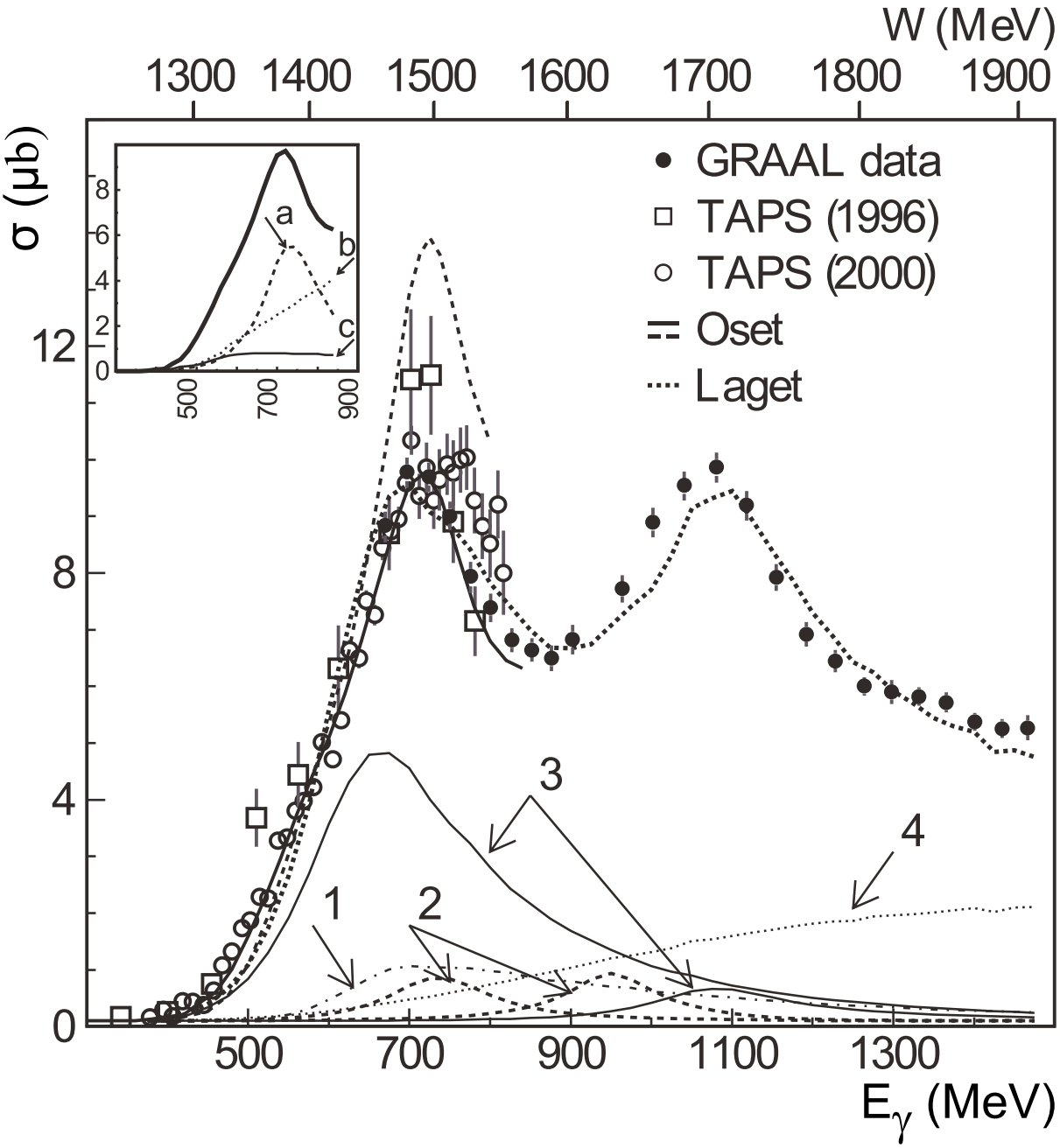


Abbildung 3.11 Totaler Wirkungsquerschnitt der Reaktion $\gamma p \to p\pi^0\pi^0$. Gezeigt sind die Daten von GRAAL in Kombination mit früheren Messungen an MAMI und theoretische Berechnungen von Murphy/Laget und Gómez-Tejedor/Oset. Links oben ist das Modell von Oset abgebildet. Hierbei sind die Zwischenzustände $D_{13}(1520)$, Δ und $P_{11}(1440)$ mit a, b und c beschriftet. Im Hauptteil der Grafik sind die vom Laget-Modell vorhergesagten Resonanzen aufgeführt mit den Bezeichnungen 1: $\gamma p \to P_{11}(1440) \to \Delta\pi$, 2: $\gamma p \to D_{13}(1520)$, $D_{13}(1700) \to \Delta\pi$, 3: $\gamma p \to P_{11}(1440), P_{11}(1710) \to \sigma p$ und 4: $\gamma p \to \sigma p$.

Der totale Wirkungsquerschnitt ist in Abbildung 3.11 in Kombination mit den Daten von TAPS und den Vorhersagen des Modells von Murphy/Laget und Gómez-Tejedor/Oset (2 Parametrisierungen) zu sehen. Deutlich zu erkennen ist der bereits durch TAPS beobachtete Peak bei einer Energie von etwa 700 MeV. Die Daten von GRAAL zeigen bei E_γ = 700 MeV eine Struktur. Bei Energien zwischen 700 MeV und 800 MeV liegt der von GRAAL gemessene Wirkungsquerschnitt systematisch unter dem von TAPS gemessenen.

Das Modell von Murphy und Laget wurde mit den in der Veröffentlichung angegebenen Parametern für die im Modell verwendeten Resonanzen angepasst, so dass der totale Wirkungsquerschnitt gut beschrieben wird. Die Parameter liegen im Bereich der zuvor bestimmten Werte [Ass03]. Beide Modelle beschreiben den totalen Wirkungsquerschnitt hinreichend gut. Die Interpretation der Daten unterscheidet sich jedoch deutlich voneinander, da unterschiedliche Beiträge in der Modelle für die Form des totalen Wirkungsquerschnitts beitragen.

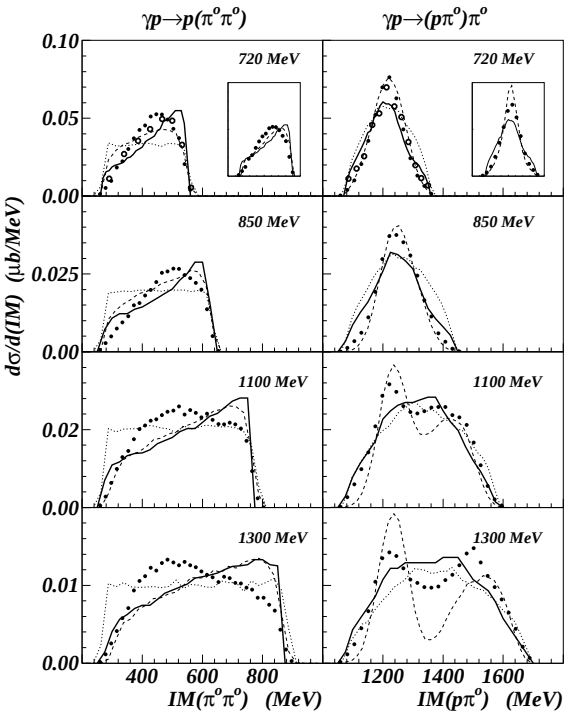


Abbildung 3.12 Invariante Masse in der Reaktion $\gamma p \rightarrow p\pi^0\pi^0$ gemessen an GRAAL (Punkte) bei vier verschiedenen Strahlenerigien in Kombination mit den theoretischen Vorhersagen des Laget- und des Oset-Modells (durchgezogene bzw. gestrichelte Linie).

In Abbildung 3.12 ist der differentielle Wirkungsquerschnitt in der Doppelpionproduktion gezeigt, wie er an GRAAL gemessen wurde (Punkte). Zusätzlich sind die Vorhersagen der Modelle von Laget und Oset angegeben (durchgezogene bzw. gestrichelte Linien). Die Daten werden weder durch das Modell von Laget, noch durch das Modell von Oset gut beschrieben.

3.2.3 Messungen an ELSA

In den letzten Jahren wurden Messungen an der Beschleunigeranlage ELSA in Bonn durchgeführt. Der experimentelle Aufbau und die zugehörigen Subdetektoren wurden bereits ausführlich in Kapitel 2 beschrieben. Die Messung erfolgte am damaligen SAPHIR-Strahlplatz. Das Crystal-Barrel-Kalorimeter wurde hierbei in der ursprünglichen Konfiguration mit 1380 CsI-Kristallen verwendet, so dass ein Raumwinkelbereich von 12° bis 168° in θ vom Kalorimeter abgedeckt wurde. Die Messung wurde unter Verwendung eines unpolarisierten Flüssigwasserstofftargets in Kombination mit dem modifizierten SAPHIR-Taggingsystem und der Flugzeitwand in Vorwärtsrichtung durchgeführt. Der verwendete Aufbau ist in Abbildung 3.13 gezeigt.

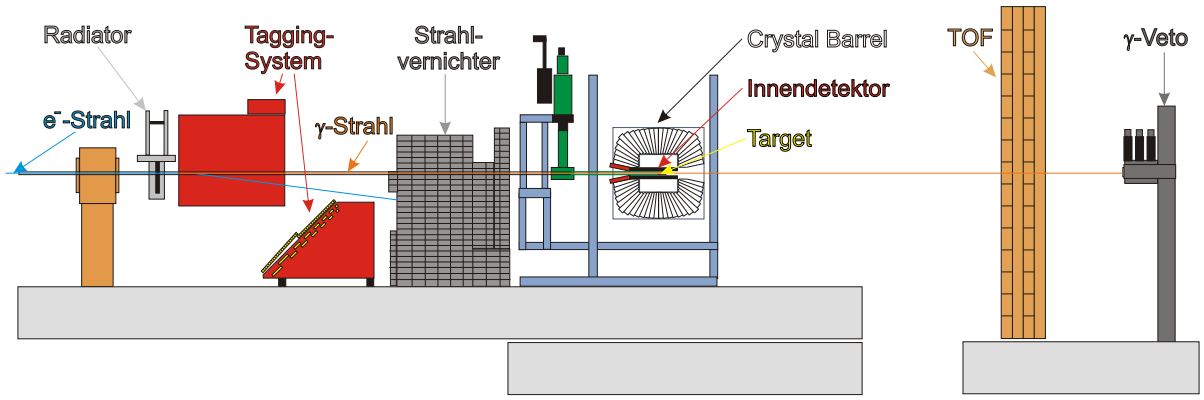


Abbildung 3.13 Aufbau des Crystal-Barrel-Experiments an ELSA.

Die Daten zur Doppelpionproduktion wurden grundlegend in einer Dissertation [Fuc05] analysiert und für eine Partialwellenanalyse vorbereitet. Im Weiteren sollen nun diese Ergebnisse vorgestellt werden.

Für die Selektion der Daten der Doppelpionproduktion stehen bei dem gezeigten Aufbau die Informationen des Innendetektors zur Identifikation von Protonen, sowie die Energieinformationen aus dem Crystal-Barrel-Kalorimeter zur Verfügung. Die erzeugten neutralen Pionen zerfallen zu 98,8 % in zwei Photonen und können über die Energiedeposition im Kalorimeter (PED) nachgewiesen werden. Die an der Reaktion beteiligten Protonen können ebenfalls ein Signal im Kalorimeter erzeugen. Mittels des Innendetektors zur Identifikation von geladenen Teilchen und einer Korrelation der Ansprecher im Kalorimeter und im Innendetektor kann das Proton identifiziert werden. Aufgrund von Akzeptanzlücken im Vorwärts- oder Rückwärtsbereich des Crystal-Barrel-Kalorimeters (12° Öffnungswinkel) kann es vorkommen, dass nicht in allen Fällen alle 5 Teilchen im Endzustand nachgewiesen werden können. Desweiteren kann bei niederenergetischen Protonen das Proton bereits im Innendetektor oder in der Haltestruktur des Kalorimeters stecken bleiben. Aus diesem Grund werden für die Auswahl der Ereignisse für die Reaktion $\gamma p \rightarrow p\pi^0\pi^0$ 4 Ansprecher im Kalorimeter und ein Ansprecher im Innendetektor oder alternativ 5 Ansprecher im Kalorimeter verlangt, wobei im Falle von 5 PEDs im Kalorimeter einer der Ansprecher über den Innendetektor als Proton identifiziert werden muss. Mittels eines kinematischen Fits, der im Rahmen einer Dissertation [van03] entwickelt wurde, werden Energie- und Impulserhaltung innerhalb eines Ereignisses überprüft. Hierbei wird das Proton

als fehlendes Teilchen behandelt und es werden alle Ereignisse verworfen, bei denen die Richtung des über den kinematischen Fit ermittelte Protons nicht innerhalb von 20° mit der Richtung des nachgewiesenen Protons aus dem Innendetektor übereinstimmen. Desweiteren wird ein sogenannter z-Cut durchgeführt. Diese Schnittbedingung ermöglicht es, nur die Reaktionen zu berücksichtigen, bei denen das ermittelte Proton aus dem Bereich der Targetzelle des Flüssigwasserstofftargets stammt. Hierbei wird die Winkelinformation aus der Rekonstruktion des Innendetektors für die Bestimmung einer minimalen und maximalen z-Koordinate verwendet, aus deren Zwischenbereich ein Proton stammen muss, um im Flüssigwasserstofftarget erzeugt worden zu sein.

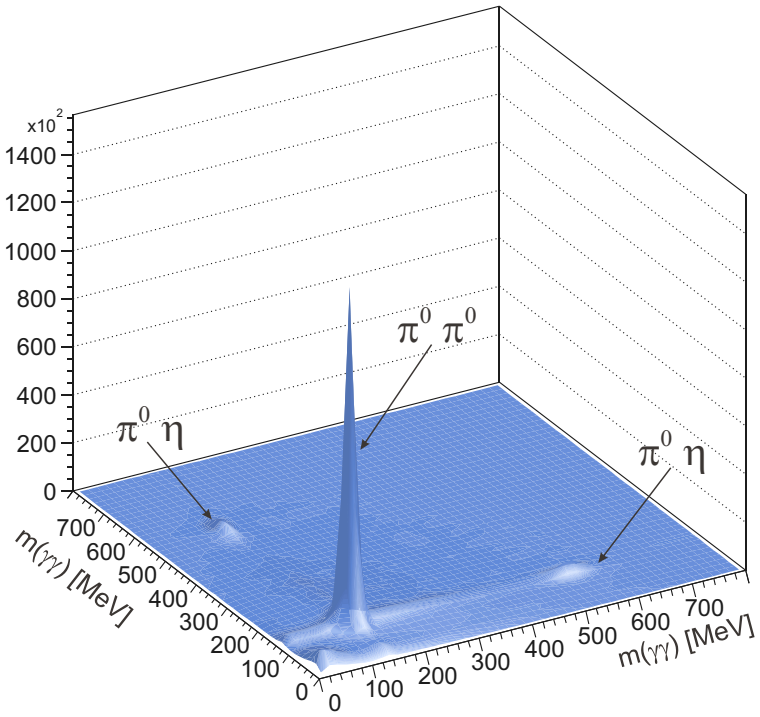


Abbildung 3.14 $\gamma\gamma$-invariante Masse eines Photonenpaares aufgetragen gegen die $\gamma\gamma$-invariante Masse des zweiten Photonenpaares für die 4- und 5-PED Ereignisse.

Die im Kalorimeter gefundenen Ansprecher müssen zur weiteren Analyse zu Pionen kombiniert werden, aus deren Zerfall die detektierten Photonen hervorgehen. Bei 4 detektieren Photonen sind hier insgesamt 6 Kombinationen möglich, von denen zwei Kombinationen die richtige π^0 Masse ergeben müssen, sofern es sich um eine Reaktion mit zwei neutralen Pionen im Endzustand handelt. In Abbildung 3.14 zu sehen ist die $\gamma\gamma$-invariante Masse eines Photonenpaares aufgetragen gegen die $\gamma\gamma$-invariante Masse des zweiten Photonenpaares nach einem kinematischen Fit mit der Hypothese $\gamma p \rightarrow p4\gamma$. Deutlich zu sehen sind die Ereignisse, bei denen zwei Pionen im Endzustand auftreten, sowie der Prozess $\gamma p \rightarrow p\pi^0\eta$, bei dem das η ebenfalls in zwei Photonen zerfällt (das Verzweigungsverhältnis $\eta \rightarrow \gamma\gamma$ beträgt 39,4 %). Zur Selektion wird im Graphen der $\gamma\gamma$ invarianten Masse eine Gaußfunktion angepasst, die eine Standardabweichung von σ= 8 MeV liefert. Nachfolgend wird dann ein Schnitt auf die Pionmasse mit einer Breite von 2 σ durchgeführt, was einem Intervall von 119 MeV bis 151 MeV entspricht und nach Regeln der Statistik 95 % der Ereignisse enthält. Um den Untergrund weiter zu reduzie-

ren, wurden weitere kinematische Anpassungen mittels der Hypothesen $\gamma p \rightarrow p\pi^0\pi^0 \rightarrow p4\gamma$, sowie $\gamma p \rightarrow p\pi^0\eta \rightarrow p4\gamma$ durchgeführt und es wurde verlangt, dass das Vertrauens- oder Konfidenzniveau (engl. Confidence Level) der Hypothese mit zwei neutralen Pionen größer als das Vertrauensniveau mit $p\pi^0\eta$ sein sollte. Mit den verwendeten Daten mit einer maximalen Photonenenergie von ca. 1,4 GeV (Niederenergiebereich) und insgesamt 5.930.496 4/5 PED-Ereignissen ergeben sich 125.555 Ereignisse nach Anwendung der beschriebenen Schnitte. Der Untergrund wurde zu weniger als 1 % abgeschätzt. Die Akzeptanz des Detektors wurde über GEANT-basierte Monte-Carlo Simulationen bestimmt. Der sich ergebende totale Wirkungsquerschnitt für die Reaktion $\gamma p \rightarrow p\pi^0\pi^0$ ist in Abbildung 3.15 im Vergleich zu den Daten der Messungen mit TAPS an MAMI, sowie LAGRANGE an GRAAL zu sehen. Zusätzlich sind die Daten für den Energiebereich mit Elektronenenergien bis 3,2 GeV angegeben.

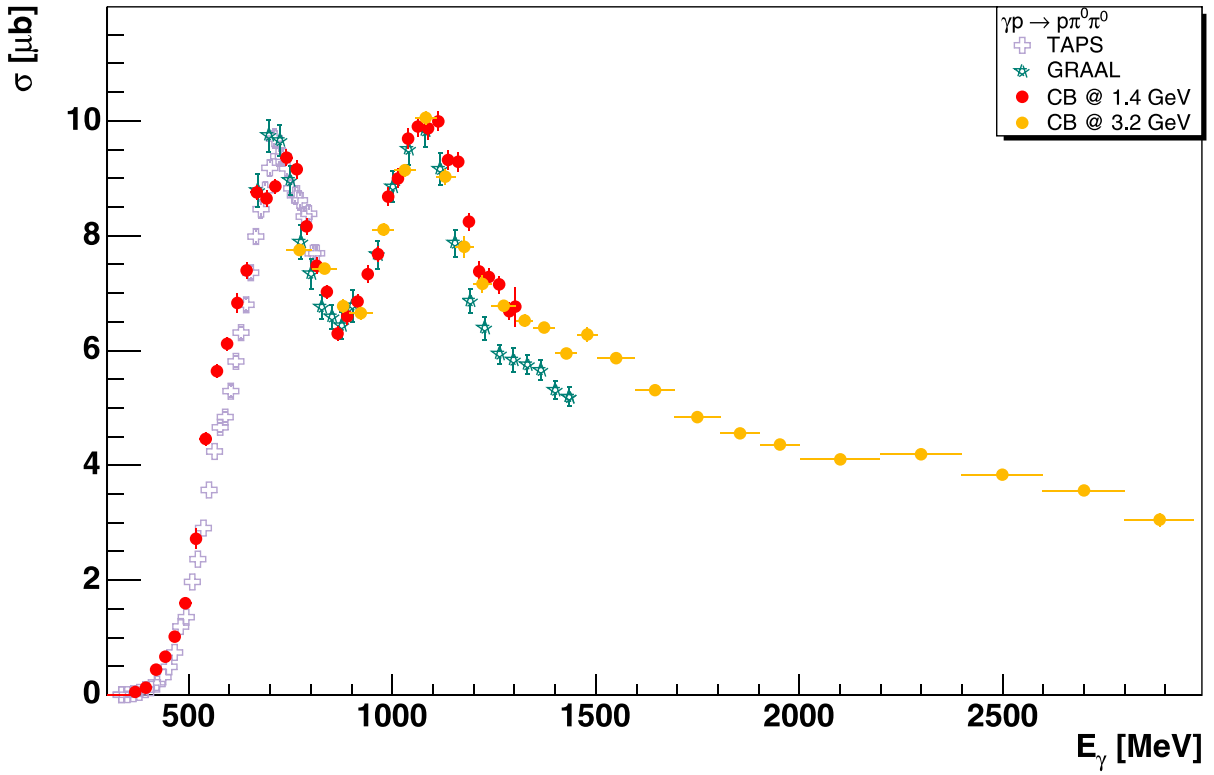


Abbildung 3.15 Totaler Wirkungsquerschnitt der Reaktion $\gamma p \rightarrow p\pi^0\pi^0$ bis 3 GeVPhotonenenergie. Dargestellt sind die Messungen der Daten von TAPS, GRAAL und die neuen Ergebnisse von CB-ELSA. Die systematischen Fehler betragen 6 % für die Akzeptanzbestimmung, 5 % für die Rekonstruktion und 5 % bzw. 15 % für die Flussbestimmung bei einer maximalen Photonenenergie von 1,3 GeV bzw. 3,0 GeV.

Deutlich zu erkennen ist die gute Übereinstimmung der Daten im Energiebereich zwischen 700 und 1200 MeV mit den neuesten Daten der Messungen an MAMI [Kot04] und GRAAL [Ass03]. Im Bereich zwischen 550 MeV und 700 MeV Photonenenergie ist eine Abweichung zu erkennen, bei denen der gemessene Wirkungsquerschnitt über dem der bisherigen Messungen liegt. Im Bereich größer 1200 MeV ist eine Schulter zu erkennen, die der abfallende Flanke des Peaks bei 1100 MeV überlagert ist. Die gewonnenen Daten erweitern den Energiebereich für die Photoproduktion neutraler Pionpaare bis zu einer Energie von 3,0 GeV. Im Bereich um 2300 MeV ist

im neu gemessenen Energiebereich eine leichte Erhöhung des totalen Wirkungsquerschnitts zu erkennen.

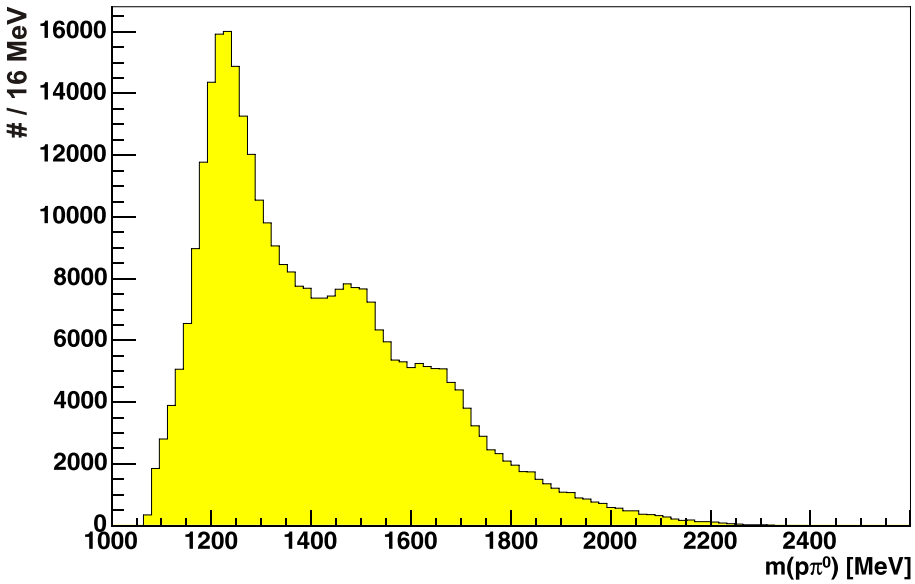


Abbildung 3.16 Invariante Masse $m(\mathrm{p}\,\pi^0)$.

In Abbildung 3.16 ist die invariante Masse $m(\mathrm{p}\,\pi^0)$ für selektierte Ereignisse der Daten bis 3,2 GeV Elektronenenergie zu sehen. Deutlich erkennbar ist der bereits bekannte Zerfall von Baryonresonanzen über $\Delta\pi$ bei einer Energie von 1232 MeV. Desweiteren sind zwei weitere Peaks zu erkennen, die bei einer Masse von 1520 MeV und 1660 MeV liegen. Hierbei handelt es sich um Zerfälle der Baryonresonanzen über die Zerfallsmoden $N(1520)D_{13}\pi$ und $X(1660)\pi$.

3.3 Partialwellenanalyse

In Kapitel 3.2 wurden Ergebnisse zur Photoproduktion im Kanal $\gamma p \rightarrow p\pi^0\pi^0$ und der totale Wirkungsquerschnitt vorgestellt, wie er an den Beschleunigeranlagen MAMI, GRAAL und ELSA gemessen wurde. Für die eingehende Analyse der Daten bietet sich die Technik der Partialwellenanalyse (PWA) an. Hiermit können zum Endzustand beitragende Resonanzen aus den experimentellen Daten extrahiert und in die beitragenden Amplituden zerlegt werden, unter besonderer Berücksichtigung von nicht-resonanten Beiträgen, wie z.B. dem t-Kanal-Austausch. Der totale Wirkunsquerschnitt der Reaktion ergibt sich dann als Quadrat der Summe über alle beteiligten resonanten und nicht-resonanten Amplituden. Die Analyse wurde im Rahmen des Isobarenmodells (sukzessive Zweikörperzerfälle) durchgeführt.

Im Falle der resonanten Produktion von zwei neutralen Mesonen im Isobarenmodell erfolgt die Erzeugung des Endzustandes $\pi^0\pi^0$ über sukzessive Zweikörperzerfälle, bei dem jeweils ein π-Meson produziert wird. Das zugehörige Feynmandiagramm ist in Abbildung 3.17 gezeigt.

Die Schwierigkeit bei der Analyse der Daten sowie bei der Extraktion der beteiligten Resonanzen und deren Polstellen (Masse, Breite) entsteht durch stark überlappende Resonanzen. In diesem Falle können die Resonanzen nicht mehr durch relativistische Breit-Wigner-Amplituden

beschrieben werden. Eine Möglichkeit zur Behandlung von stark überlappenden Resonanzen ist der K-Matrix Formalismus [Chu95] um die Unitarität auch bei nahe beieinanderliegenden Resonanzen zu erhalten. Das Verfahren wird für die Analyse der Ein- und Zweikörper-Photoproduktionsdaten angewandt. Die Berechnung der Winkelverteilungen erfolgt über einen Operatorformalismus und ist in einer Veröffentlichung [Ani05] im Detail beschrieben. Eine Besonderheit der Analyse ist, dass diese ereignisbasiert arbeitet und somit alle Korrelationen zwischen den vorhandenen fünf freien Variablen, durch die das Ereignis bestimmt ist, berücksichtigt. Die Anpassung der Daten an das Modell erfolgt hierbei über einen Maximum-Likelihood-Fit.

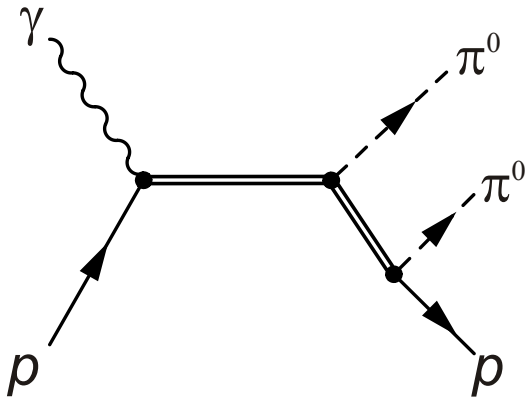


Abbildung 3.17 Feynmandiagramm der Produktion zweier neutraler Mesonen im Isobarenmodell.

Bei anderen Analysen erfolgt nur eine Anpassung der differentiellen Wirkungsquerschnitte in Summe über alle Ereignisse an die gemessenen Daten. Die differentiellen Wirkungsquerschnitte stellen jedoch lediglich eindimensionale Projektionen eines 5 dimensionalen Parameter-Raumes dar, so dass die ereignisbasierte Methodik klare Vorteile gegenüber anderen Methoden bietet.

3.4 Ergebnisse der Partialwellenanalyse der Crystal-Barrel-Daten

Für die Analyse der an ELSA mit dem Crystal-Barrel-Experiment gewonnenen Daten kann die zuvor beschriebene Methode der Partialwellenanalyse mit relativistischen Breit-Wigner-Amplituden und dem K-Matrix-Formalismus für sich überlagernde und interferierenden Resonanzen angewendet werden. Im Weiteren sollen hier die Ergebnisse dieser Partialwellenanalyse vorgestellt werden. Eine entsprechende Veröffentlichung [Tho06] wird derzeit vorbereitet. In einem ersten Schritt werden zunächst die Daten analysiert, die bei einer extrahierten Elektronenenergie von 1,4 GeV aufgezeichnet wurden. Um Vieldeutigkeiten der Lösungen zu vermeiden, wurden neben den hier diskutierten Daten auch die Ergebnisse zu den Reaktionen $\gamma p \to p\pi^0$, $p\eta$, ΛK, ΣK (einschließlich gemessener Polarisationsvariablen) in den Fit mit einbezogen (gekoppelte Partialwellenanalyse).

Die Partialwellenanalyse basiert auf den in Kapitel 3.2.3 vorgestellten Daten im Bereich bis zu einer Elektronenenergie von 1,4 GeV. In den Anpassungen der Daten werden Gewichtungen für die verschiedenen Reaktionskanäle im Bereich zwischen 1 und 25 eingeführt. Die Daten zur

Doppelpionproduktion gehen mit einem Gewichtungsfaktor 4 ein. Als Startparameter werden die Lösungen der vorangegangenen Analysen [Ani05, Sar05] verwendet. Durch die Verwendung der neuen Daten ergeben sich nur geringfügige Änderungen bei den bisher ermittelten Parametern. Die beiden S_{11} Resonanzen und die P_{11} werden über K-Matrizen behandelt, bei denen die Kanäle $N\pi$, $N\eta$, $K\Lambda$, $K\Sigma$, $\Delta\pi$ und $N\Sigma$ bzw. $N\pi$, $N\Sigma$ und $\Delta\pi$ als mögliche Zerfallskanäle berücksichtigt werden. Alle anderen Resonanzen werden über Vielkanal-Breit-Wigner-Amplituden modelliert [Fla76].

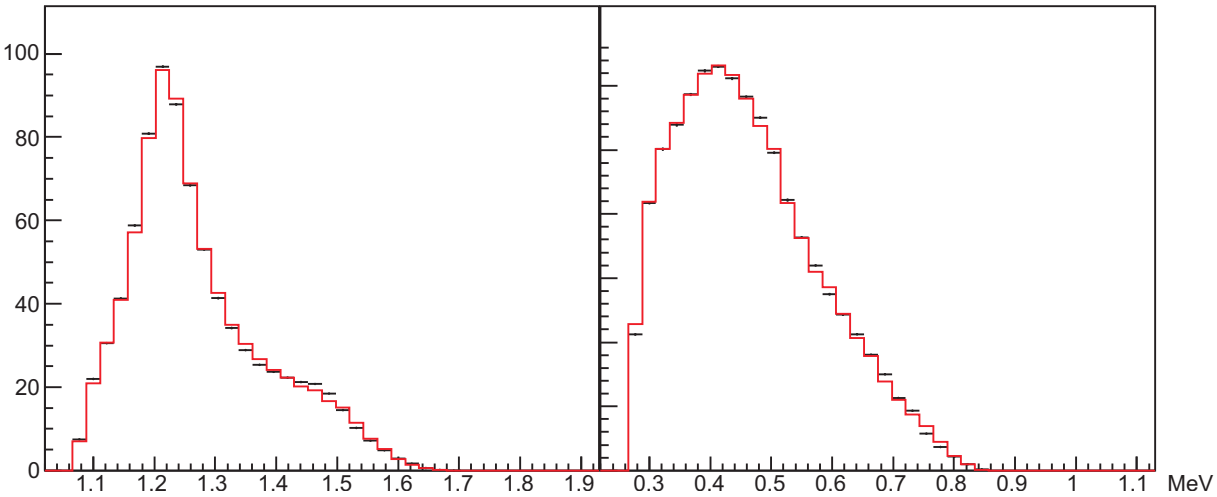


Abbildung 3.18 Invariante Masse $p\pi^0$ und $p\pi^0\pi^0$ im Kanal $\gamma p \rightarrow p\pi^0\pi^0$. Daten als Kreuze, Fit als durchgezogene rote Linie.

In Abbildung 3.18 sind die invarianten Massen $p\pi^0$ und $\pi^0\pi^0$ im Kanal $\gamma p \rightarrow p\pi^0\pi^0$. Deutlich zu sehen ist die gute Übereinstimmung der Daten (Kreuze) mit der PWA-Analyse (rote durchgezogene Linie). Die Partialwellenanalyse ergibt im Bereich des ersten Peaks als dominanten Anteil die Anregung und den nachfolgenden Zerfall $\gamma p \rightarrow N(1520)D_{13} \rightarrow \Delta\pi$. Auch die Winkelverteilungen sind durch die PWA gut beschrieben. Das Ergebnis widerspricht dem Modell von Murphy und Laget, bei dem in diesem Energiebereich die $P_{11}(1440)$ mit dem Zerfall nach σ dominieren sollte. Desweiteren finden sich in den analysierten Daten keine Hinweise auf die Resonanz $N(1710)P_{11}$, so wie dies auch in vorangegangenen Analysen [Ani05, Sar05] bereits der Fall war. Die Lösungen der PWA liefern weitere so nicht zu erwartetende Resultate. Die $N(1700)D_{13}$ Resonanz zerfällt dominant in einer D-Welle in $\Delta\pi$ oder alternativ in die Roper-Resonanz $N(1440)P_{11}\pi$ und die Resonanz $N(1720)P_{13}$ zerfällt nach $\Delta\pi$ via F-Welle und nicht über eine P-Welle. Diese Zerfälle in die höheren Partialwellen sollten durch die Drehimpulsbarriere unterdrückt und der Zerfall in die niedrigeren Drehimpulse bevorzugt sein. Für die Resonanzen $N(1675)D_{15}$, $N(1680)F_{15}$ und $\Delta(1700)D_{33}$ erfolgt der Zerfall in die tiefer liegende Partialwelle. Zudem koppeln die Resonanzen $N(1675)D_{15}$, $N(1720)P_{13}$ und $\Delta(1700)D_{33}$ stark an den Kanal $N(1520)D_{13}\pi$. Die vollständigen Ergebnisse der Partialwellenanalyse sind der Tabelle 3.1 zu entnehmen. Die Partialwellenanalyse liefert insbesondere bei hohen Energien nicht immer eindeutige Ergebnisse, da die Experimente mit einem unpolarisierten Strahl und einem unpolarisierten Target durchgeführt wurden. Hierdurch ergeben sich Mehrdeutigkeiten insbesondere bei nahe aneinanderliegenden Resonanzen. Als problematisch erweisen sich bei der zuvor beschriebenen Partialwellenanalyse vor allem die Zerfälle in die $\Delta\pi$ S-Welle, die sich

aus Zerfällen der Resonanzen $N(1700)D_{13}$, $\Delta(1700)D_{33}$, sowie $N(1520)D_{13}$ zusammensetzen. Hier existieren mehrere Lösungen mit nahezu gleichen Wahrscheinlichkeiten.

Resonanz	$N(1520)D_{13}$	$N(1700)D_{13}$	$N(1675)D_{15}$	$N(1720)P_{13}$	$N(1680)F_{15}$	$\Delta(1620)S_{31}$	$\Delta(1700)D_{33}$
Masse	1522 ± 5	1740 ± 15	1690 ± 5	1770 ± 30	1678 ± 5	1645 ± 15	1750 ± 30
PDG	1515-1530	1650-1750	1670-1685	1650-1750	1675-1690	1615-1675	1670-1770
Γ_{tot}	114 ± 10	120 ± 20	180 ± 25	450 ± 50	100 ± 10	150 ± 30	500 ± 150
PDG	110-135	50-150	140-180	100-200	120-140	120-180	200-400
$A^{1/2}/A^{3/2}$	$-0{,}01 \pm 0{,}07$	$0{,}35 \pm 0{,}09$	$0{,}50 \pm 0{,}25$	$1{,}1 \pm 0{,}4$	$-0{,}13 \pm 0{,}04$	-	$0{,}8 \pm 0{,}3$
PDG	$-0{,}15 \pm 0{,}06$	undef	$1{,}3 \pm 0{,}9$	$-0{,}9 \pm 1{,}9$	$-0{,}11 \pm 0{,}05$		$1{,}2 \pm 0{,}4$
Γ_{miss}	20 ± 15	8 ± 8	70 ± 25	180 ± 30	2 ± 3	40 ± 25	40 ± 30
PDG	16-32	≤ 50	≤ 6	70-170	4-20	10-45	60-200
$\Gamma_{\pi N}$	63 ± 6	10 ± 8	45 ± 25	100 ± 20	75 ± 10	40 ± 15	20 ± 10
PDG	55-100	3-22	56-90	10-40	70-100	25-60	30-80
$\Gamma_{\eta N}$	$0{,}1 \pm 0{,}1$	15 ± 10	1 ± 1	120 ± 50	$0{,}1 \pm 0{,}4$	-	-
PDG	$\approx 0{,}3$	≤ 1	≤ 2	≈ 6	≤ 2		
$N\sigma$	$0{,}6 \pm 0{,}4$	10 ± 4	15 ± 10	30 ± 25	14 ± 4	-	-
$\Gamma_{K\Lambda}$	-	$1{,}0 \pm 0{,}5$	5 ± 1	20 ± 10	-	-	-
$\Gamma_{K\Sigma}$	-	$1{,}5 \pm 1$	0 ± 1	3 ± 2	-	-	$0{,}4 \pm 1$
$\Gamma_{\Delta\pi(L<J)}$	9 ± 2	6 ± 3	25 ± 8	1 ± 2	8 ± 3	20 ± 10	350 ± 100
PDG	≈ 10	≈ 11 *	≈ 95 *	-	≈ 13	-	≈ 112
$\Gamma_{\Delta\pi(L>J)}$	10 ± 2	40 ± 8	0 ± 5	35 ± 10	0 ± 1	-	45 ± 20
PDG	≈ 15	≈ 79 *	-	-	≤ 1		≈ 12
$\Gamma_{P_{11}\pi}$	$0{,}2 \pm 1$	40 ± 8	0 ± 2	0 ± 4	-	10 ± 3	5 ± 8
$\Gamma_{D_{13}\pi(L<J)}$	-	-	30 ± 20	20 ± 10	-	-	40 ± 20

Tabelle 3.1 Vorläufige Ergebnisse der Partialwellenanalyse in der dritten Resonanzregion mit Massen und Partialbreiten [Tho06] mit Angaben der Werte der Particle Data Group (PDG) [Par04] bzw. mit * markiert nach [Vra00].

Die vorgestellte Analyse beschränkt sich im Augenblick auf einen niedrigen Energiebereich bis 1,3 GeV Photonenenergie, da im Bereich höherer Energie und insbesondere im Bereich der dritten Resonanzregion noch mehr Zustände annähernd gleicher Energie existieren, die eine Partialwellenanalyse mit unpolarisierten Messdaten erheblich erschweren. Um dennoch eine Partialwellenanalyse zu ermöglichen werden dringend Experimente zur Doppelpolarisation benötigt, die im nächsten Abschnitt beschrieben werden. Polarisationsexperimente stellen wichtige zusätzliche Informationen zur Verfügung, die es erlauben, die auftretenden Lösungen in der Partialwellenanalyse weiter einzuschränken.

3.5 Neue Doppelpolarisationsexperimente

Um weitere Information für die Partialwellenanalyse zu gewinnen, die dann zu eindeutigeren Ergebnissen führen, ist die Messung von Polarisationsobservablen notwendig [Cry05]. Geplant sind Doppelpolarisationsexperimente mit longitudinal polarisiertem Target und einem polarisiertem Photonenstrahl. Hierdurch lassen sich die existierenden Ambiguitäten der Lösungen der Partialwellenanalyse deutlich reduzieren. Neben den differentiellen Wirkungsquerschnitten sind insbesondere auch die Polarisationsobservablen wichtig, da diese stark von Interferenzeffekten beeinflusst werden. Mittels der Polarisationsobservablen lassen sich auch sehr kleine Effekte beobachten.

Der differentielle Wirkungsquerschnitt für die Photoproduktion eines Mesons mit einem longitudinal polarisierten Target und linear oder zirkular polarisierten Photonen lässt sich mittels des differentiellen Wirkungsquerschnittes σ_0 (ohne Polarisation) als

$$\frac{d\sigma}{d\Omega} = \sigma_0\{1 - \delta_l \mathbf{\Sigma}\cos(2\phi) - \Lambda_z(-\delta_l \mathbf{G}\sin(2\phi) + \delta_\odot \mathbf{E})\} \tag{3.1}$$

schreiben [Kno95]. δ_l, $\delta_\odot$ und Λ_z sind hierbei der Polarisationsgrad der linear bzw. zirkular polarisierten Photonen bzw. die longitudinale Polarisation des Targets und ϕ der Polarisationswinkel. Σ, G und E sind die bereits erwähnten Polarisationsobservablen. Im Falle von zwei Mesonen im Endzustand erfolgt die Produktion der Mesonen nicht mehr in einer einzigen Ebene. In diesem Fall lässt sich, ohne Messung der Polarisation des Protons, die Reaktionsrate als

$$\frac{d\sigma}{dx_i} = \sigma_0\{(1 - \Lambda_z \cdot \mathbf{P_z}) + \delta_\odot(\mathbf{I^\odot} - \Lambda_z \cdot \mathbf{E}) - \delta_l[\sin 2\phi(\mathbf{I^s} - \Lambda_z \cdot \mathbf{G}) + \cos(2\phi)(\mathbf{\Sigma} - \Lambda_z \cdot \mathbf{P_z^c})]\} \tag{3.2}$$

schreiben [Rob05]. Es ergeben sich die zusätzlichen Polarisationsobservablen P_z^c, $I^\odot$ und I^s. Die Messung der Polarisationsvariablen P_z, G , E und P_z^c kann durch Messung mit verschiedenen Polarisationsrichtungen erfolgen, so dass der unpolarisierte Untergrund von Reaktionen der Photonen mit unpolarisiertem C_4O im Butanol-Target sich aufhebt. Für die Messung der Observablen $I^\odot$, I^s und Σ bei Einfachpolarisation kann zur Vermeidung des C_4O Untergrunds das unpolarisierte Flüssigwasserstofftargets in Kombination mit zirkular oder linear polarisierten Photonen verwendet werden.

Um die Sensitivität auf Doppelpolarisationsvariablen abzuschätzen wurden Studien durchgeführt [Cry05]. Hierzu wurden die Daten ohne Polarisation verwendet und mittels einer Partialwellenanalyse wurde eine Lösung an die Daten angepasst. Basierend auf dem Fit-Ergebnis wurden unter anderem die Helizitätsdifferenzen

$$\frac{d\sigma_{3/2}}{dx_i} - \frac{d\sigma_{1/2}}{dx_i} \tag{3.3}$$

berechnet und gegen den Winkel zwischen den beiden neutralen Pionen im CMS-System bzw. zwischen Proton und Pion im $p\pi^0$ System aufgetragen. Nachfolgend wurden dann einzelne Resonanzen aus dem Fit der Partialwellenanalyse herausgenommen, die verbleibenden Amplituden erneut an die Daten angepasst und dann wiederum die resultierende Helizitätsdifferenz berechnet. Teile der Ergebnisse hierzu sind Abbildung 3.3 zu sehen. Die Fehler entsprechen

einer Strahlzeit von 800 Stunden. Das Ergebnis basierend auf der besten Anpassung an die unpolarisierten Daten ist in schwarz angegeben. In blau ist die Lösung ohne die Resonanzen $D_{33}(1940)$ und in grün die Lösung ohne die $P_{33}(1920)$ bei einer Elektronenstrahlenergie von $E_{e^-} = 3,2GeV$ in der Abbildung zu sehen. Deutlich zu erkennen ist der Unterschied in $d\sigma_{3/2} - d\sigma_{1/2}$ berechnet aus dem Fit mit und ohne die $D_{33}(1940)$, so dass mittels der Doppelpolarisationsmessungen entschieden werden kann, ob die $D_{33}(1940)$ existiert und zur $2\pi^0$ Photoproduktion beiträgt.

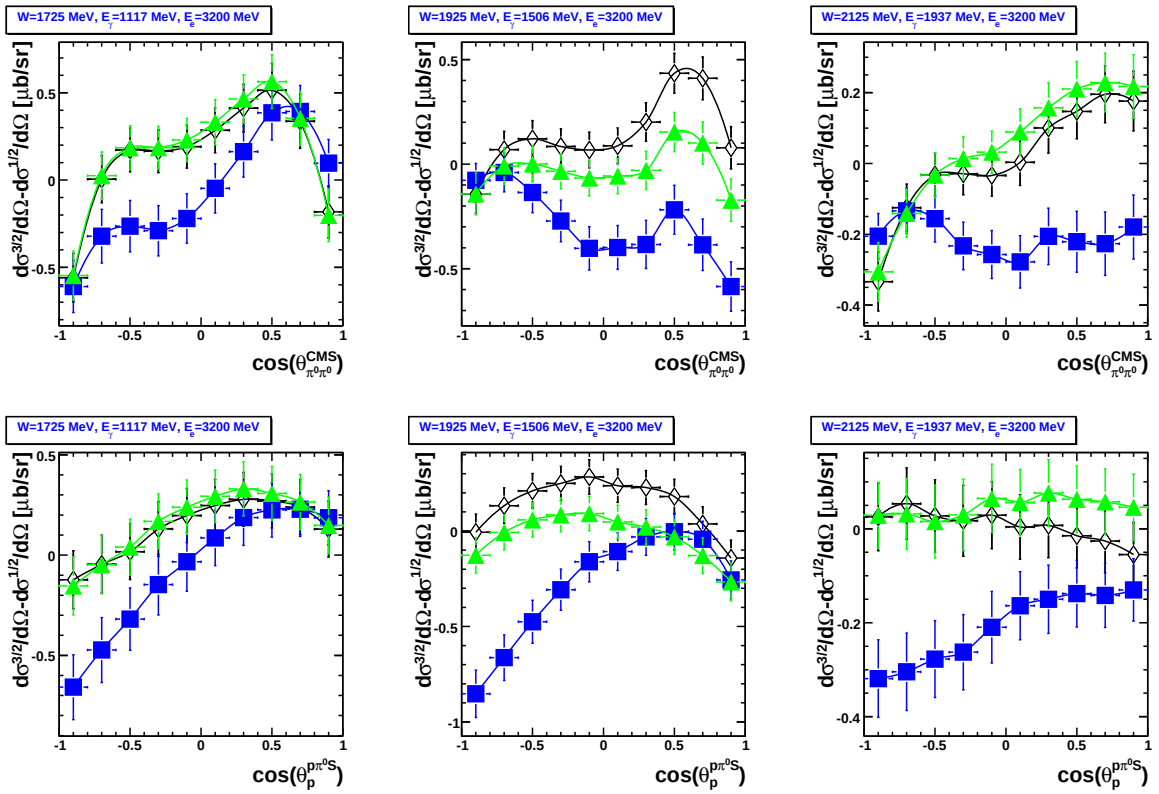
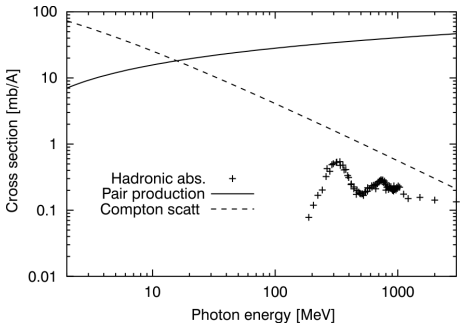


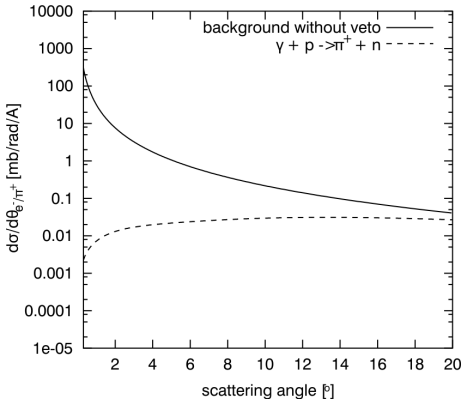
Abbildung 3.19 Sensitivitätsstudien zu Messungen mit Doppelpolarisation. In schwarz: beste Lösung, in blau: keine $D_{33}(1940)$ und in grün: keine $P_{33}(1920)$.

Für die Messungen im Reaktionskanal $\gamma p \to p\pi^0\pi^0$ mit einem unpolarisiertem Flüssigwasserstofftarget und einem unpolarisierten Photonenstrahl wurde der Untergrund in der Reaktion zu 1 % für den Energiebereich der Photonen von 350 bis 1300 MeV abgeschätzt, sowie zu 2 % für den Energiebereich von 0,8 bis 3,0 GeV.

Bei Messungen mit einem Frozen-Spin-Butanol-Target ist ein wesentlich höherer Untergrund zu erwarten, da hier nicht nur Reaktionen an den Protonen im Target auftreten, sondern auch Produktion von 2 π^0 an gebundenen Nukleonen des Targetmaterials (C_4H_9OH) stattfinden kann. Desweiteren gibt es einen stark erhöhten elektromagnetischen Untergrund durch Compton-Streuung und Paarproduktion. Der Wirkungsquerschnitt für diese elektromagnetischen Reaktionen ist um mehrere Größenordnungen höher als der hadronische Wirkungsquerschnitt. In Abbildung 3.20 ist die Energie- und Winkelabhängigkeit der elektromagnetischen Wirkungsquerschnitte im Vergleich zum hadronischen Wirkungsquerschnitt der Reaktion $\gamma p \to \pi^+ n$ zu sehen, wie er von der GDH-Collaboration berechnet wurde [Hel02]. Die elektromagnetischen



Elektromagnetischer Untergrund im Vergleich zum totalen hadronischen Wirkungsquerschnitt an Wasserstoff [Hel02]



Winkelverteilung der Elektronen aus dem elektromagnetischen Untergrund im Vergleich zu einem typischen hadronischen Prozess [Hel02]

Abbildung 3.20 Elektromagnetischer Untergrund verursacht durch das Butanol-Target.

Ereignisse werden aufgrund der Kinematik primär in Vorwärtsrichtung erzeugt. Diese ungewünschten korrelierten Untergrundereignisse sollen durch Triggerbedingungen der beteiligten Subdetektoren deutlich reduziert werden. Unter Verwendung eines Tscherenkow-Detektors im Vorwärtsbereich als Veto-Detektor können diese korrelierten Ereignisse erkannt und verworfen werden. Allerdings ist dies technisch bedingt nur innerhalb der Breite des Koinzidenzfensters der ersten Triggerstufe von voraussichtlich 40 ns möglich. Die unkorrelierten elektromagnetischen Ereignisse können jedoch die hadronischen Ereignisse überlagern, so dass bei einem Signal eines Kristalls im Vorwärtsdetektor die Information über die im Kristall deponierte Energie, durch die Überlagerung eines elektromagnetischen Signals, verfälscht wird. Diese Ereignisse, die 40 ns nach dem Auslösen der ersten Triggerstufe erzeugt werden, können die Signale der Kristalle überlagern und eine fehlerhafte Energieinformation zur Folge haben. Durch eine Digitalisierung des Signals mittels integrierender Analog-Digital-Konverter kann im Nachhinein die Überlagerung von Signalen aus dem hier erwähnten elektromagnetischen Untergrund nicht mehr erkannt oder korrigiert werden. In Abbildung 3.21 ist eine solche Überlagerung von Signalen auf dem Bildschirm des Oszilloskops zu sehen. Die Aufnahme wurde bei ersten Testmessungen mit einem CsI-Kristall im Bereich des neuen Vorwärtsdetektors am GDH-Strahlplatz und einer Auslese mit einem Photomultiplier gewonnen. Deutlich zu sehen ist ein Signal auf dessen abfallender Flanke weitere Signale überlagert sind.

Zur Erkennung von solchen Doppel- oder Mehrfachansprechern bietet sich die Verwendung von Flash-Analog-Digital-Konvertern (Flash-ADCs) an, die die gesamte Signalform aufzeichnen und digitalisieren. Die Entwicklung einer Auslese mittels Flash-ADCs und erste Ergebnisse, die im Rahmen dieser Arbeit durchgeführt bzw. erlangt wurden, sollen in den folgenden Kapiteln vorgestellt werden.

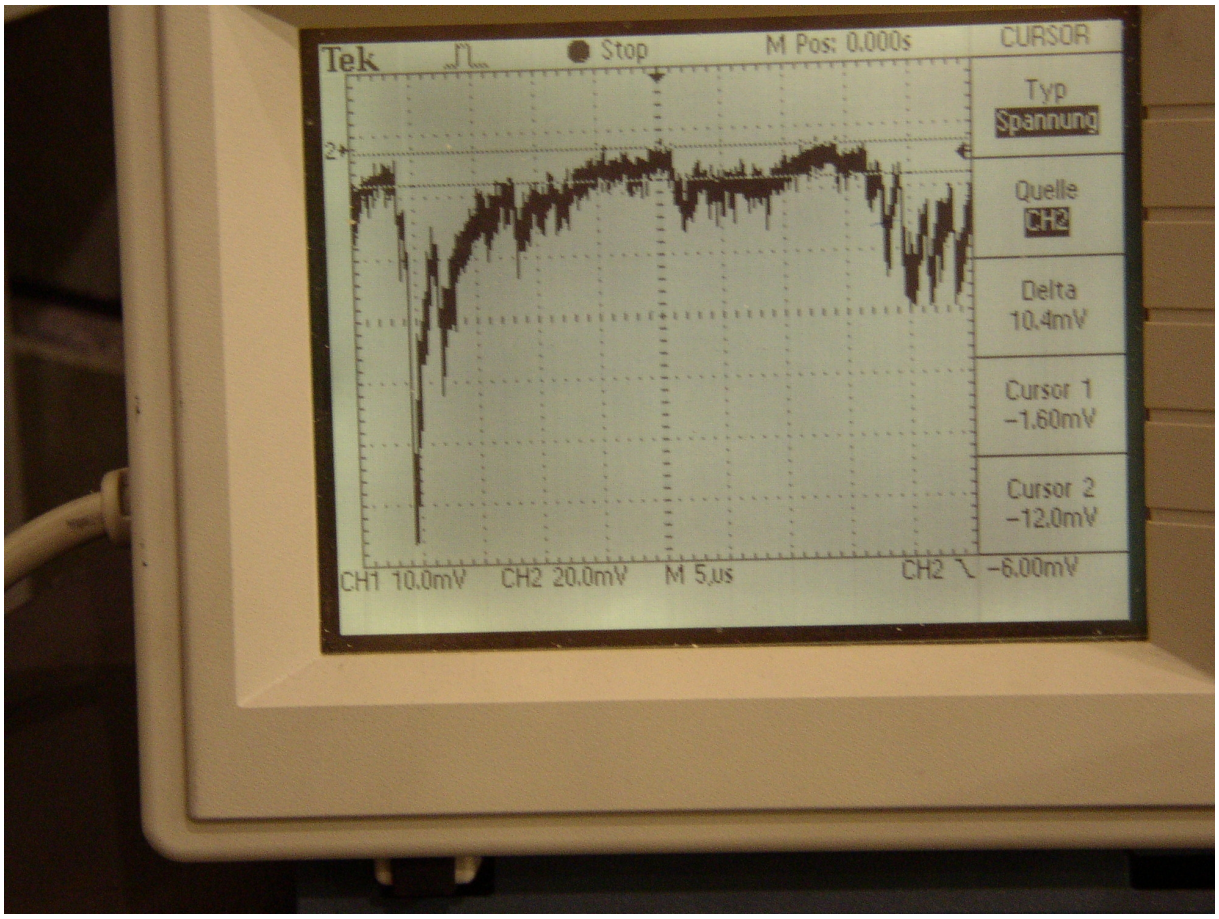


Abbildung 3.21 Signal am Ausgang eines CsI-Kristalle mit Photomultiplierauslese bei ersten Testmessungen am GDH-Strahlplatz mit sehr hohem elektromagnetischen Untergrund, aufgrund fehlender Bleiabschirmungen und fehlerhafter Strahlführung.

4 Grundlagen Analog-Digital-Konverter

Dieses Kapitel beschreibt die theoretischen Grundlagen für den Einsatz eines Flash-ADC-Systems am CB-Transregio-Experiment. Hierbei werden Digitalisierungsverfahren für ADC- und Flash-ADC-Systeme erläutert. Als Beispiel für ein ADC- und ein Flash-ADC-System dienen hierbei die am Experiment verwendeten Fastbus-ADCs, sowie ein Flash-ADC-System, welches im Aufbau des CB-Transregio-Experiments eingesetzt wird.

4.1 Allgemeines

In heutigen Kern- und Teilchenphysikexperimenten an Beschleunigern werden zumeist hohe Zählraten in den Subdetektoren erreicht. Die daraus resultierenden Anforderungen an die Erfassung von analogen und digitalen Signalen und ihre Verarbeitung sind sehr hoch. Die moderne Technik hat in den letzten Jahrzehnten dafür gesorgt, dass Experimente mit immer höheren Datenraten realisiert werden können. Das CB-Transregio-Experiment wird in den kommenden Jahren Baryonenspektroskopie mit einem polarisierten Photonenstrahl und einem polarisierten Target (Doppelpolarisationsexperiment) durchführen und dabei sehr große Strahlströme verwenden, wodurch hohe Ereignisraten und eine gute Statistik erzielt werden sollen. Aufgrund der Geometrie des Experiments mit einem festen Zielpunkt (Target) und dem einlaufenden Photonenstrahl ergibt sich aus der Kinematik der beteiligten physikalischen Prozesse eine starke Häufung der Reaktionsprodukte in Vorwärtsrichtung. Für die physikalische Auswertung ist die Erfassung dieser Reaktionsprodukte im Vorwärtsbereich sehr wichtig, um eine vollständige Kinematik der Reaktion über einen weiten kinematischen Bereich gewährleisten zu können. Aus diesem Grund sind für die Projektphasen in den nächsten 12 Jahren Erweiterungen im sogenannten Vorwärtsbereich des Detektors geplant und im Rahmen des Sonderforschungsbereichs Transregio 16 (SFB TR-16) von der Deutschen Forschungsgemeinschaft (DFG) in der ersten der drei möglichen Förderperioden (je 4 Jahre) genehmigt worden.

Die Erfassung von analogen Signalen in Kern- und Teilchenphysikexperimenten erfolgt häufig über integrierende Analog-Digital-Konverter. Dabei wird ein analoges Eingangssignal während eines bestimmten Zeitraumes (Gate) erfasst und in eine digitale Information umgewandelt, deren Wert proportional zum Integral des Signals über den Messzeitraum ist.

Abbildung 4.1 zeigt das Signal eines Kristalls des Crystal-Barrel-Kalorimeters, welches über einen Photomultiplier ausgelesen wurde. Der digitale Ausgangswert bzw. die Fläche unter der Signalkurve entspricht dabei im Allgemeinen der Energie eines detektierten Photons. Aufgrund der Integration über die Signalform gehen viele Informationen verloren, die im Nachhinein nicht wieder rekonstruiert werden können. Hierzu gehören insbesondere:

- die Anstiegszeit des Signals
- die zeitliche Auflösung des Signals (Signalform)
- die Position des Signals innerhalb des Erfassungszeitraums.

Warum sind diese Informationen interessant?

Aufgrund der hohen Datenraten im Vorwärtsbereich und mit Verwendung eines Butanol-Targets mit entsprechendem elektromagnetischem Untergrund kann es zu Doppelpulsen kommen. Hierbei ist die physikalische Ereignisrate so hoch, dass innerhalb der Erfassungszeit mehr als ein Teilchen den jeweiligen Detektor treffen und zwei Pulse erzeugen kann, die sich überlagern, wenn der Abstand zwischen den beiden Signalen genügend kurz ist. In Abbildung 4.1 ist eine solche Überlagerung von zwei Pulsen zu sehen.

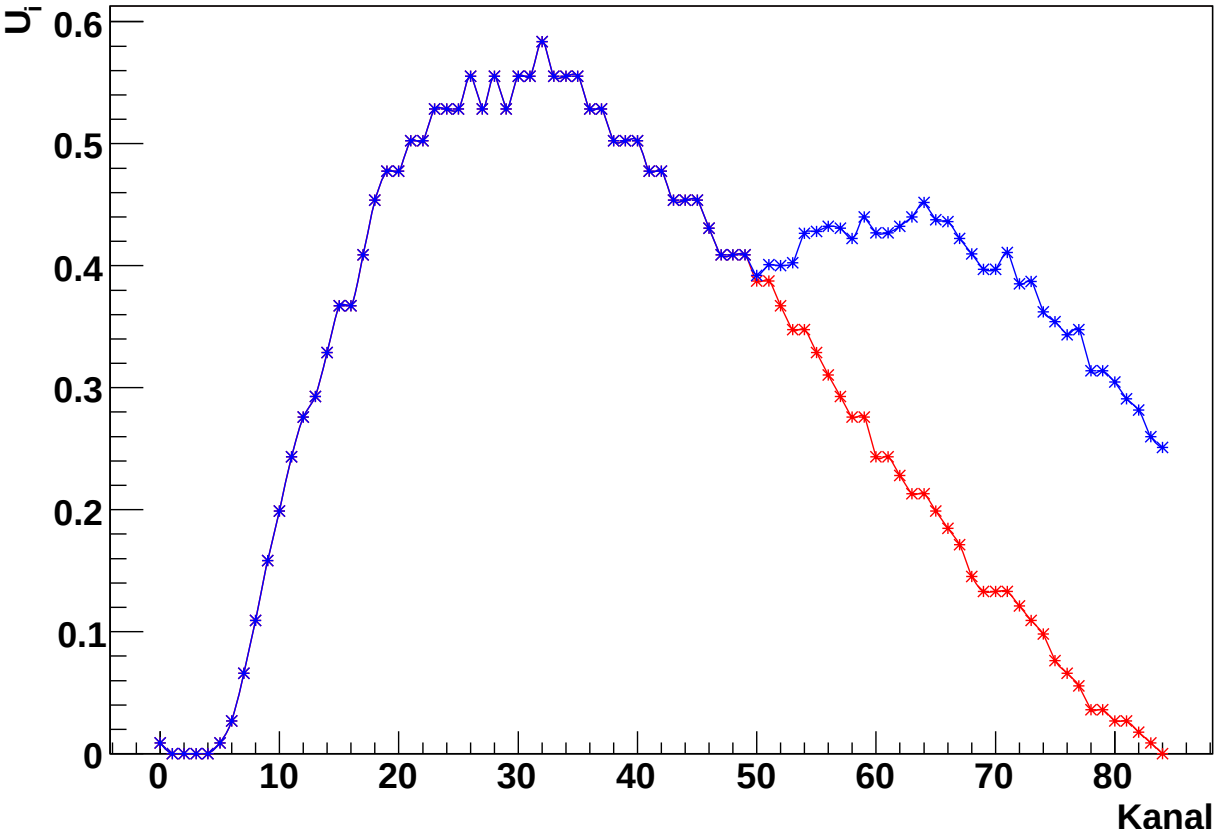


Abbildung 4.1 Simulierter Doppelpuls anhand der Überlagerung zweier Signale (blau). In Rot ist das erste Signal ohne Überlagerung zu sehen.

Wird ein integrierender ADC eingesetzt, so kann nicht zwischen den beiden Signalen unterschieden werden. Die beiden Pulse werden aufsummiert und die integrierte Fläche entspricht nicht mehr der Energiesumme eines Einzelsignals. Die Energiesumme wird somit durch das zusätzliche Signal verfälscht. Mit einem integrierenden ADC sind aus diesem Grund prinzipbedingt solche Doppelpulse nicht zu erkennen und die Energieinformation ist nicht korrekt ermittelbar. Mit Flash-ADCs lassen sich prinzipiell Doppelpulse erkennen und die Energie des Hauptsignals und des zweiten überlagerten Signals rekonstruieren [Fun02].

Neben den Doppelpulsen ist es aufgrund der Integration ebenfalls nicht möglich, zwischen geladenen und ungeladenen Teilchen zu unterscheiden. Diese erzeugen im Cäsiumjodid (CsI) verschiedene Signalformen, wobei die Unterschiede in der Anstiegszeit des Signals vom Sockel bis zum Maximum des Signals erkennbar sind. Geladene Teilchen erzeugen ein schneller an-

steigendes Signal und damit eine stärker ansteigende Signalflanke. Die Zeit zwischen Anstieg des Signals und dem Maximum ist somit deutlich kürzer.

Desweiteren kann nach einer Integration keine Aussage mehr über das zeitliche Verhalten eines Signals gemacht werden. Aufgrund der meist festgelegten Messzeit, die mit der typischen Länge des ausgelesenen Signals zusammenhängt, kann es passieren, dass ein gegen Ende der Messzeit eintreffendes Teilchen ein Signal erzeugt, welches nicht mit der physikalisch zu beobachtenden Reaktion korreliert ist. Das Signal trägt zum gebildeten Integral des Analog-Digital-Konverters bei.

Mit Kenntnis des vollen Signalverlaufes kann man über den Anstieg des Signals eine Zeitinformation erhalten, zu welchem relativen Zeitpunkt das Signal aufgetreten ist. Dazu kann der Anstieg extrapoliert werden, so dass man den Schnittpunkt des Signals mit dem Gleichspannungsoffset (Pedestal) bestimmen kann. Daraus erhält man eine Zeitinformation, die die absolute Lage dieses Signals innerhalb eines Ereignisses bestimmt.

Mit Flash-ADCs ist es möglich die gesamte Signalform zu digitalisieren. Die Digitalisierung erfasst dabei die Höhe und den zeitlichen Verlauf des Signals. Die Abtastfrequenz (Samplingrate) ist abhängig vom eingesetzen Flash-ADC-Typ und beträgt typischerweise bis zu 100 MHz. Bei dieser Frequenz wird das Eingangssignal alle 10 ns erfasst, digitalisiert und in einem Speicherbereich abgelegt. Über die aufgezeichneten Daten erhält man dann die Signalform. Als problematisch kann sich erweisen, dass aufgrund der hohen Abtastfrequenz eine sehr große Datenmenge zustande kommt. Durch die Abtastung des Signals entsteht pro Erfassungsintervall ein Datenwert, der entsprechend der Auflösung n (in Bit) des Flash-ADC somit auch n Bit für die Speicherung benötigt. Bei der Digitalisierung eines Signals von 4 μs Länge und 100 MHz Samplingrate entsteht eine Datenmenge von $N = 400 \cdot n$ Bit.

Bei einer Auflösung von 6 Bit ergibt dies mindestens eine Datenmenge von 2400 Bit bzw. 300 Bytes[22] pro erfasstem Signal und bei einer angenommenen mittleren Anzahl von 15 Signalen pro Ereignis eine Datenmenge von 4,5 kByte pro Ereignis. Bei einer angestrebten Ereignisrate von 1 kHz erreicht man somit einen kontinuierlichen Datenstrom des Flash-ADC-Systems für die Auslese der 90 Kristalle von insgesamt 4,5 MByte/s, was die bisher von jedem der Einzeldetektoren im Experiment erzielte Datenrate um ein Vielfaches übersteigt (zum Vergleich Crystal-Barrel-Detektor ca. 450 kbyte/s). Die Datenmenge des Flash-ADCs wird somit den Hauptanteil (mindestens 50 %) an der Gesamtdatenmenge des CB-TR-Experiments bilden. Für die Datenerfassung bedeutet dies auch, dass eine sehr große Datenmenge in kurzer Zeit erfasst, verarbeitet, übertragen und gespeichert werden muss. Eine rein „in Software“ arbeitende Lösung würde schnell an Grenzen stoßen. Hier bietet sich die Verwendung spezialisierter Hardware an, die die Verarbeitung der aufgenommenen Signale übernimmt.

[22] 8 Bit entsprechen einem Byte.

4.2 Digitalisierungsfehler von ADCs

Bei der Wandlung eines analogen Signals in eine digitale Information ist der digitale Wert immer fehlerbehaftet. Man unterscheidet bei einer Digitalisierung eines Signals zwischen mehreren Fehlerquellen, die bei der Digitalisierung eine Rolle spielen, unter anderem:

- Quantisierungsfehler,
- Linearitätsfehler,
- Nullpunktsfehler,
- integrale Nichtlinearität,
- differentielle Nichtlinearität.

Durch Umwandlung eines Signals in einen Digitalwert entsteht ein Quantisierungsfehler. Dieser Quantisierungsfehler ist durch die notwendige Rundung eines analogen Eingangswertes auf den nächst kleineren oder größeren Digitalisierungswert bedingt. Der maximale Abstand eines analogen Wertes zu einem Digitalwert beträgt hierbei 0,5 Bit. Der Quantisierungsfehler beträgt somit ebenfalls 0,5 Bit. Man nennt das letzte für einen Digitalwert signifikante Bit im Englischen least significant Bit (LSB). Bei einem 8-Bit ADC entspricht ein Fehler von 0,5 LSB einem Quantisierungsfehler von etwa 0,39 %.

Die meisten ADCs arbeiten linear. Damit ist gemeint, dass eine lineare Beziehung zwischen Ein- und Ausgangswerten besteht, unter Vernachlässigung der Quantisierung der erzeugten Digitalwerte. Jeder analoge Wert in einem Wertebereich zwischen

$$m \cdot k$$

und

$$m \cdot (k + 1)$$

mit der Konstanten m wird auf den Ausgangswert k abgebildet. Bei nichtlinearen ADCs ist der Wert abhängig vom Digitalisierungswert k. Dies findet häufig im Bereich der Sprachkommunikation Anwendung. Der in dieser Arbeit verwendete Flash-ADC bietet ebenfalls die Möglichkeit, nichtlinear betrieben zu werden, um einen größeren dynamischen Bereich abzudecken. Hierbei ergibt sich eine gewollte Abweichung der Übertragungsfunktion von einer Geraden. Dies wird für den Flash-ADC in Kapitel 4.3 beschrieben und auch später in allen Messungen genutzt. Technisch realisierte lineare Analog-Digital-Konverter besitzen aufgrund der Ungenauigkeiten der verwendeten Komponenten wie Widerstände und Kondensatoren allerdings immer auch einen Linearitätsfehler. Die differentielle Nichtlinearität bezeichnet dabei die Abweichung der einzelnen Übertragungsfunktionsstufen vom Sollwert. Unter integraler Nichtlinearität versteht man eine Abweichung der Linearität im gesamten Wandlungsbereich des ADC, also der Abweichung der gesamten Übertragungsfunktion von einer Geraden. Nullpunktsfehler ergeben sich

aufgrund einer verschobenen Kennlinie eines Konverters. Diese kann herstellungstechnisch gewünscht sein und beträgt dann zumeist 0,5 Bit.

4.3 Funktionsprinzip von Analog-Digital-Konvertern (ADCs)

Analog-Digital-Konverter finden sich heutzutage in vielen Geräten des täglichen Lebens. Überall dort, wo analoge Signale durch einen Mikrocontroller oder eine elektronische Steuerung verarbeitet werden müssen, sind Analog-Digital-Konverter (ADCs)[23] im Einsatz. Die Grundfunktion aller ADCs ist immer dieselbe. Ein analoges Signal wird dem Eingang eines ADCs zugeführt. Entsprechend der Auflösung des n-Bit ADC erzeugt dieser hieraus ein n-faches binäres Ausgangssignal. Im Weiteren soll hier kurz auf das Funktionsprinzip von einigen ausgewählten Analog-Digital-Konvertern eingegangen werden.

4.3.1 Integrierender Dual-Slope-ADC

Ein integrierender ADC, der nach dem Dual-Slope Verfahren arbeitet, setzt sich aus einem Integrator, einem Komparator und einer Logik für eine Zeitmessung zusammen. Die zu digitalisierende Spannung U_{in} wird für eine bekannte Zeit T_1 integriert. Zum Integrieren wird ein Operationsverstärker in Kombination mit einem Kondensator verwendet (Integrator). Der Kondensator lädt sich dabei in der Zeit T_1 auf eine Ladung

$$Q_{sig} = -T_1 I_{in} \tag{4.1}$$

auf. Nachfolgend wird der Kondensator mit dem bekannten Referenzstrom $\frac{U_{ref}}{R}$ entladen, bis die Ausgangsspannung den Wert 0 erreicht:

$$Q_{ref} = T_2 \frac{U_{ref}}{R} = -Q_{sig} = T_1 \frac{U_{in}}{R}. \tag{4.2}$$

Die zu messende Eingangsspannung ergibt sich dann zu:

$$U_{in} = U_{ref} \cdot \frac{T_2}{T_1}. \tag{4.3}$$

Die Eingangsspannung hängt somit nicht mehr von R oder C ab. Dies ist durch die Auf- und nachfolgende Entladung (dual slope) bedingt, da sich hier aufgrund der Ladungsmessung die Größen der Widerstände herauskürzen. Im einfacheren Single-Slope Verfahren wird nur ein Ladevorgang genutzt. Für die Wandlung des Signals wird eine gewisse Zeit benötigt, die abhängig von der Größe des Eingangssignals ist. ADCs mit Dual-Slope-Verfahren sind vergleichsweise langsam. Mit diesem Verfahren werden Konversionszeiten in der Größenordnung von einigen Millisekunden erreicht, welche nicht geeignet sind für ein physikalisches Experiment mit höchsten Raten, insbesondere dann, wenn viele Kanäle parallel verarbeitet werden müssen. Hierzu eignen sich andere Verfahren, wie das Flash-Verfahren.

[23] engl. analog-digital-converter (ADCs)

4.3.2 Flash-ADCs

Das Prinzip der Flash-Wandlung eines analogen Eingangssignals ist vergleichsweise einfach. Ein Eingangssignal U_{in} wird an 2^n Komparatoren eines n-Bit Flash-ADCs weitergeleitet. Jeder Komparator vergleicht das Eingangssignal mit einem über eine Widerstandskette angelegten Referenzsignal U_{ref}. Das Referenzsignal U_{ref} wird entsprechend der Anzahl der Komparatoren in 2^n äquidistante Spannungen mit den Amplituden $\frac{i}{2^n}U_{ref}$ mit $i = 1, 2, ..., 2^n$ zerlegt. Ein Prioritätsenkodierer oder eine Kombination aus XOR-Gattern[24] in Verbindung mit einem normalen Enkodierer erzeugt aus den einzelnen Vergleichswerten ein binäres Ausgangssignal mit einer linearen Kennlinie.

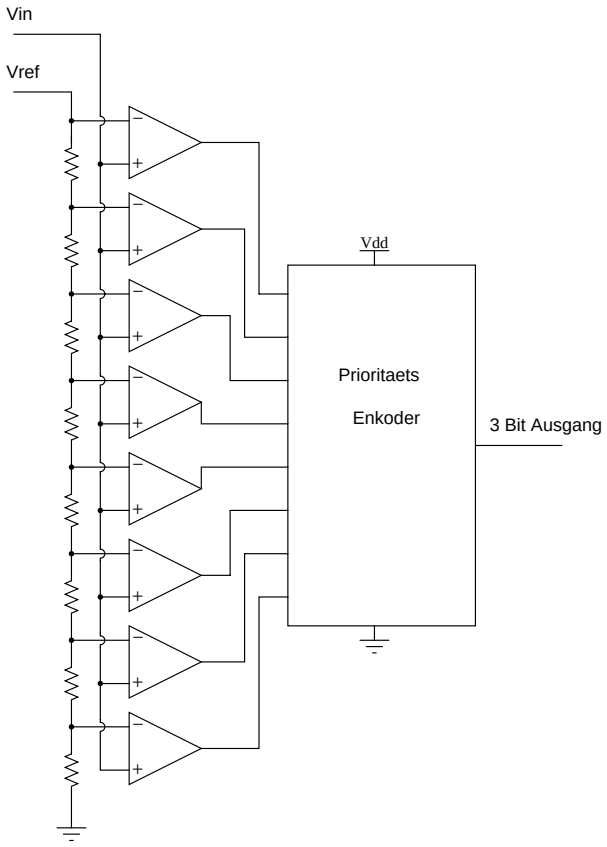


Abbildung 4.2 Prinzip eines Flash-ADCs mit 3-Bit Auflösung.

Einer der Vorteile eines solchen Analog-Digital-Konverters ist die sehr schnelle Umwandlungszeit. Die Wandlungszeit ist bei diesem Prinzip nur durch die Laufzeit der Signale durch die Komparatoren und die elektronischen Gatter begrenzt. Desweiteren kann durch die Anpassung der Widerstandswerte eine beliebige (nichtlineare) Kennlinie erzeugt werden. Hauptnachteil der Flash-Wandlung ist die hohe Anzahl der benötigten elektronischen Komponenten. Aufgrund der exponentiellen Abhängigkeit der Anzahl der Komparatoren von der benötigten Auflösung in Bit (n Bit erfordern 2^n Komparatoren) wird die Flash-Wandlung nur für Auflösungen bis zu maximal 12 Bit verwendet.

[24] XOR - Exklusives Oder

4.4 Beispiele für ADC-Systeme

Im Folgenden sollen kurz ein bereits im Crystal-Barrel-Experiment verwendetes ADC-System und das in dieser Arbeit verwendete System sowie deren interne Funktionsweise vorgestellt werden.

4.4.1 LeCroy 1885F Fastbus integrierender ADC

Seit einigen Jahren werden Fastbus-ADCs vom Typ 1885F der Firma LeCroy [LeC97] am Crystal-Barrel-Experiment zur Digitalisierung der Signale der Photodioden der CsI-Kristalle des CB-Detektors verwendet. Das System zeichnet sich durch eine sehr hohe Auflösung von 15 Bit und eine vergleichsweise kurze Konversionszeit von 265 μs aus. Auf jedem der Module sind 96 Kanäle verfügbar. Um eine solch hohe Dichte an ADC-Kanälen zu erreichen, wurde auf den Modulen nur ein einziger ADC-Baustein eingesetzt. Die Signale werden durch spezielle von LeCroy entwickelte Multiplexer-Bausteine MIQ 401 nacheinander auf den Eingang des ADC-Bausteins gegeben. Dabei wird die Ladung des Eingangssignals in den Multiplexern während eines anliegenden Signals (Gate) gespeichert und in drei Anteile aufgeteilt. Der ADC ist also ein integrierender ADC. Ein Anteil von 80 % und 10 % des Eingangssignals werden zur Digitalisierung verwendet (Dual-Range-Technik). Hierdurch wird ein höherer dynamischer Bereich erreicht. Die übrigen 10 % werden für interne Korrekturen verwendet (der Hersteller macht hierzu keine weiteren Angaben). Der digitalisierende ADC verfügt über eine Auflösung von 12 Bit. Aufgrund des Verhältnisses von 8:1 der anliegenden Signale im unteren (Low-Range) und oberen (High-Range) Bereich wird eine Gesamtauflösung von 15 Bit erzielt ($2^n = 8, n = 3$).

Der auf der Platine verbaute ADC vom Typ Datel ADC 521MC ist ein ADC mit 12-Bit-Ausgang [Dat88]. Intern arbeitet der ADC mit einem 7-Bit-Flash-ADC. Das Eingangssignal wird in einem Zwei-Schritt-Verfahren digitalisiert. Im ersten Schritt wird das Eingangssignal digitalisiert und als 7-Bit Wert an ein Register übergeben, das über einen 7-Bit Digital-Analog-Konverter das Signal in ein analoges Ausgangssignal wandelt, welches vom Analogeingang des Flash-ADC subtrahiert, verstärkt und dann erneut auf den Eingang des ADCs gegeben wird. Es erfolgt eine zweite Digitalisierung mit 7 Bit, deren Ergebnis in einem weiteren Register zwischengespeichert wird. Die beiden Register werden nachfolgend durch eine Korrekturlogik zu einem 12-Bit-Ausgangssignal kombiniert. Die maximale Gesamtverarbeitungszeit für ein Eingangssignal liegt bei 1050 ns. Die Genauigkeit beträgt typischerweise 3 LSB und schlechtestenfalls 8 LSB für die Digitalisierung eines 12-Bit-Datenwertes. Die Gesamtsensitivität des Fastbus-Systems wird vom Hersteller zu 50 fC/count $\pm$3 % im unteren Digitalisierungsbereich und zu 400 fC/count $\pm$5 % im oberen Digitalisierungsbereich angegeben.

4.4.2 Struck DL300 Flash-ADC

Ein ADC-System mit Flash-Verfahren ist das DL300-System der Firma Struck. Die ADC-Module DL305 und DL310 des Systems basieren auf dem Parallel- oder auch Flash-Verfahren. Hierzu wird auf den Karten der Flash-AD-Wandler SDA 5010 der Firma Siemens mit 6-Bit-Auflösung

verwendet. Das System kann ein Eingangssignal mit einer maximalen Erfassungsfrequenz von 100 MHz abtasten und die einzelnen Abtastungswerte in einem Speicher ablegen. Um einen größeren dynamischen Bereich abdecken zu können, wird der FADC mit einer nichtlinearen Kennlinie betrieben. Hierzu wird ein Teil des Eingangssignals als Treiber für die Referenzspannung des Bausteins benutzt. Die etwas vereinfachte Schaltung hierzu ist in Abbildung 4.3 gezeigt.

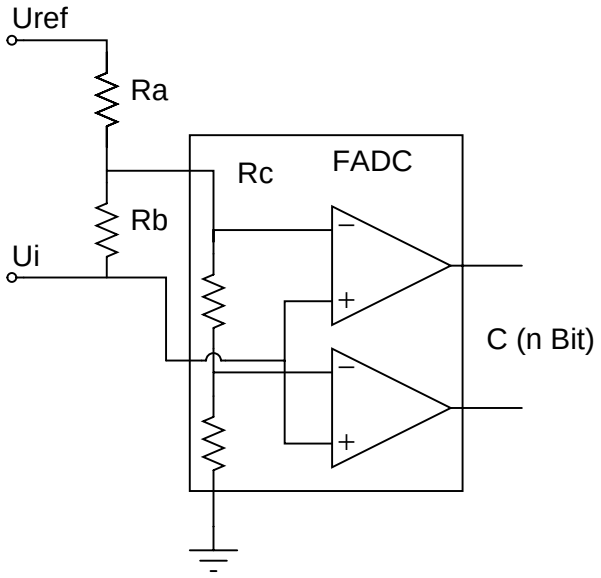


Abbildung 4.3 Schemazeichnung des Flash-ADC-Bausteins mit Kopplung der Referenzspannung an die Eingangsspannung.

R_c ist der Gesamtwert der internen Widerstandskette des SDA 5010. Die an R_c gegen Masse anliegende Referenzspannung U_{ref} wird über einen Spannungsteiler erzeugt. Am Spannungsteiler wird die Spannung U_{const} über den Widerstand R_a an U_{ref} und weiter über R_b an das Eingangssignal U_i angeschlossen. Mit dem Ansatz

$$U_0 = U_{ref}(U_i = 0V) = a\frac{R_b}{R_a}U_{const} \tag{4.4}$$

lässt sich eine Konstante a errechnen, die von R_a, R_b und R_c abhängt. Für a gilt dann[25]:

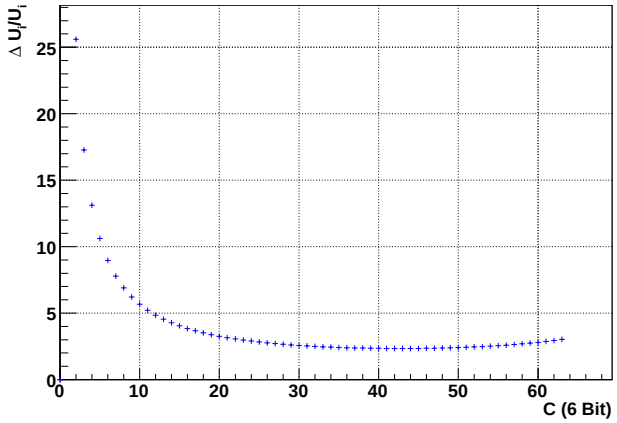
$$a = \frac{R_a \parallel R_c}{R_b + R_a \parallel R_c}. \tag{4.5}$$

Mit den Werten R_a = 100 Ω, R_b = 15,4 Ω und R_c = 100 Ω folgt a ≈ 0,75. Mittels der Konstanten a kann nun die nichtlineare Kennlinie des FADC zum Digitalisierungswert C durch

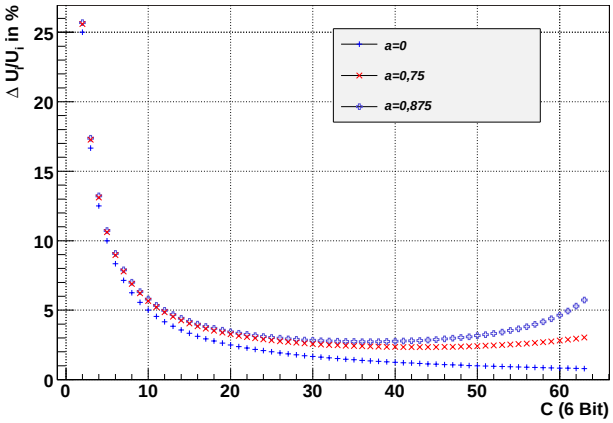
$$U_i = \frac{C}{C_{max} - a \cdot C} \cdot U_0 \tag{4.6}$$

ausgedrückt werden. Hierbei errechnet sich der maximale Digitalisierungswert C_{max} durch die Auflösung des Flash-ADC von 6 Bit zu $C_{max} = 2^6 = 64$. Die sich ergebende Kennlinie ist in Abbildung 4.4 zu sehen.

[25] $R_a \parallel R_b$ soll hierbei der resultierende Wert aus der Parallelschaltung von R_a und R_b sein



Nichtlineare Kennlinie des FADC für das Widerstandsverhältnis a=0,75



Digitalisierungsfehler des DL300-Flash-ADC für verschiedene Widerstandsverhältnisse a in %

Abbildung 4.4 Nichtlinearität des DL300-Flash-ADC.

Die effektive Auflösung erhöht sich für a=0,75 auf 8 Bit im unteren Amplitudenbereich. Das Widerstandverhältnis von a=0,75 ist durch den Hersteller vorgegeben, kann aber bei Bedarf modifiziert werden. Der Fehler hierzu beträgt

$$\frac{\Delta U_i}{U_i} = \frac{1}{2}\frac{1}{C}\frac{1}{1 - a\frac{C}{C_{max}}} \tag{4.7}$$

und bestimmt sich aus der Hälfte der Differenz zweier benachbarter Digitalisierungsstufen C+1 und C im Verhältnis zur Gesamthöhe des Signals (0,5 LSB), also dem Quantisierungsfehler. Aufgrund fehlender Herstellerangaben zum Digitalisierungsfehler des Gesamtsystems musste der Fehler über den Quantisierungsfehler abgeschätzt werden.

Bei maximaler Eingangsamplitude beträgt der Fehler etwa 3 % für a=0,75. Ist eine andere oder eine lineare Kennlinie erwünscht, so kann dies durch Anpassung der Widerstandswerte von R_a und R_b erreicht werden. Abbildung 4.4 (rechts) zeigt den Digitalisierungsfehler für verschiedene Werte von a des Flash-ADC-Moduls.

4.5 Vergleich des Digitalisierungsfehlers eines integrierenden ADC und des DL300-Flash-ADC

Für den Vergleich zwischen dem Fehler eines integrierenden Analog-Digital-Konverters und dem zu ermittelndem Integral des Kurvenverlaufes eines Flash-ADCs muss man zunächst eine Annahme über die Form des zu digitalisierenden Signals machen, da der Fehler abhängig von der Pulshöhe an jedem Digitalisierungszeitpunkt und somit abhängig von der Pulsform ist. Das Ausgangssignal eines Shapers lässt sich mathematisch durch eine Anregungsfunktion und eine Transferfunktion beschreiben [Cha95]. Hier soll die vereinfachte Transferfunktion mit exponentieller Anregung angewandt werden ([Cha95] Seite 7), die für eine Abschätzung des Digitalisierungsfehlers vollkommen ausreichend ist:

$$f(x) = Q(\frac{x^3}{6} \cdot e^{-x}) + A \tag{4.8}$$

Der Kurvenverlauf ist in Abbildung 4.5 zu sehen.

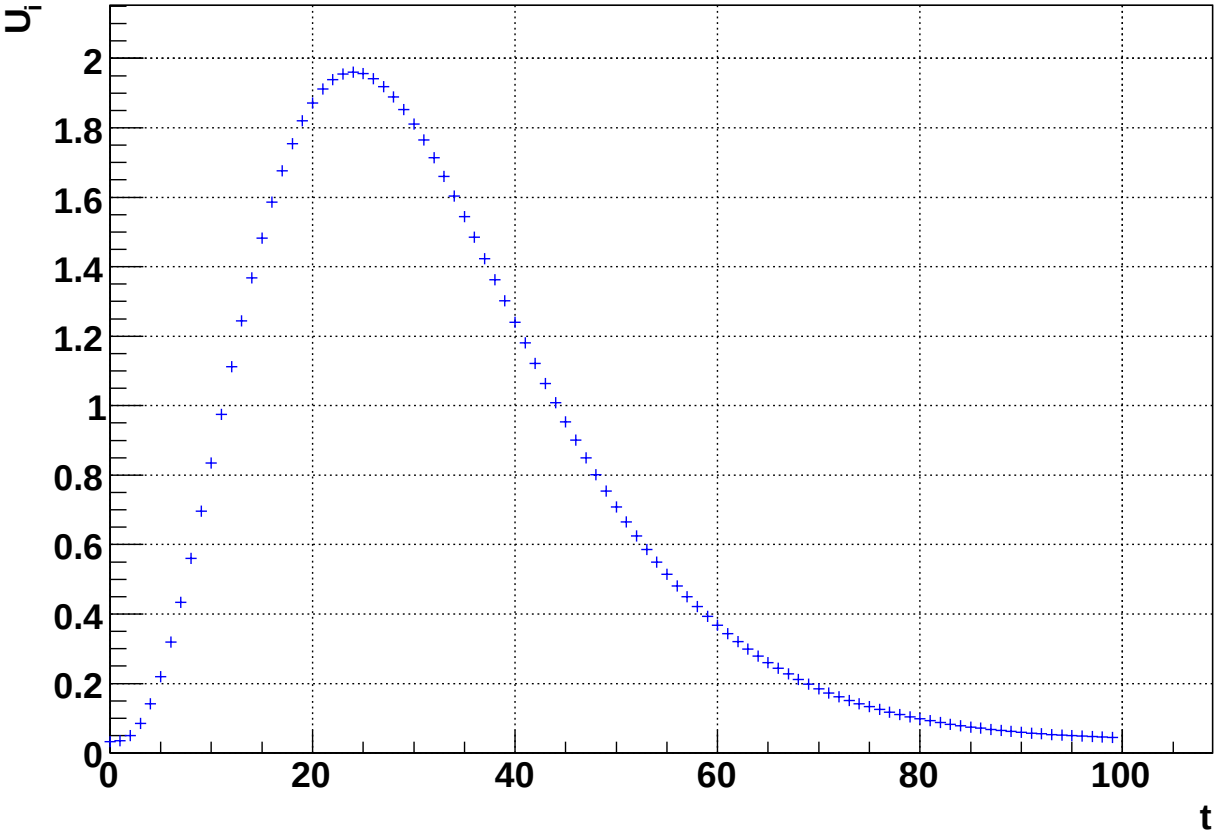


Abbildung 4.5 Vereinfachte Transferfunktion mit exponentieller Anregung zur Simulation eines typischen Signalverlaufes.

Der Offset-Parameter A wurde so gewählt, dass sich umgerechnet ein Digitalisierungswert C von 4 ergibt, der in Kapitel 7 auch für Messungen mit dem DL300-System als Pedestal-Wert verwendet werden wird. Der Wert Q wird so variiert, dass sich unterschiedlich hohe Signale ergeben, die über den gesamten Wertebereich des FADC variieren.

Jeder Funktionswert f(x) wird zunächst in einen Digitalisierungswert C umgerechnet, der auf eine natürliche Zahl gerundet werden muss. Danach wird aus dem Digitalisierungswert C wieder eine Spannung errechnet, sowie der Fehler zu jedem Digitalisierungswert. Die Linearisierung erfolgt für drei mögliche Werte des Widerstandsverhältnisses a. Als Fehler wurde erneut der Quantisierungsfehler 0,5 LSB angenommen. Alle ermittelten Digitalisierungswerte werden aufsummiert, wenn der Digitalisierungswert größer als der simulierte Pedestal-Wert von 4 ist und als FADC-Summe auf der Zeitachse gegen den Fehler in Prozent auf der y-Achse aufgetragen. Das Ergebnis ist in Abbildung 4.6 zu sehen.

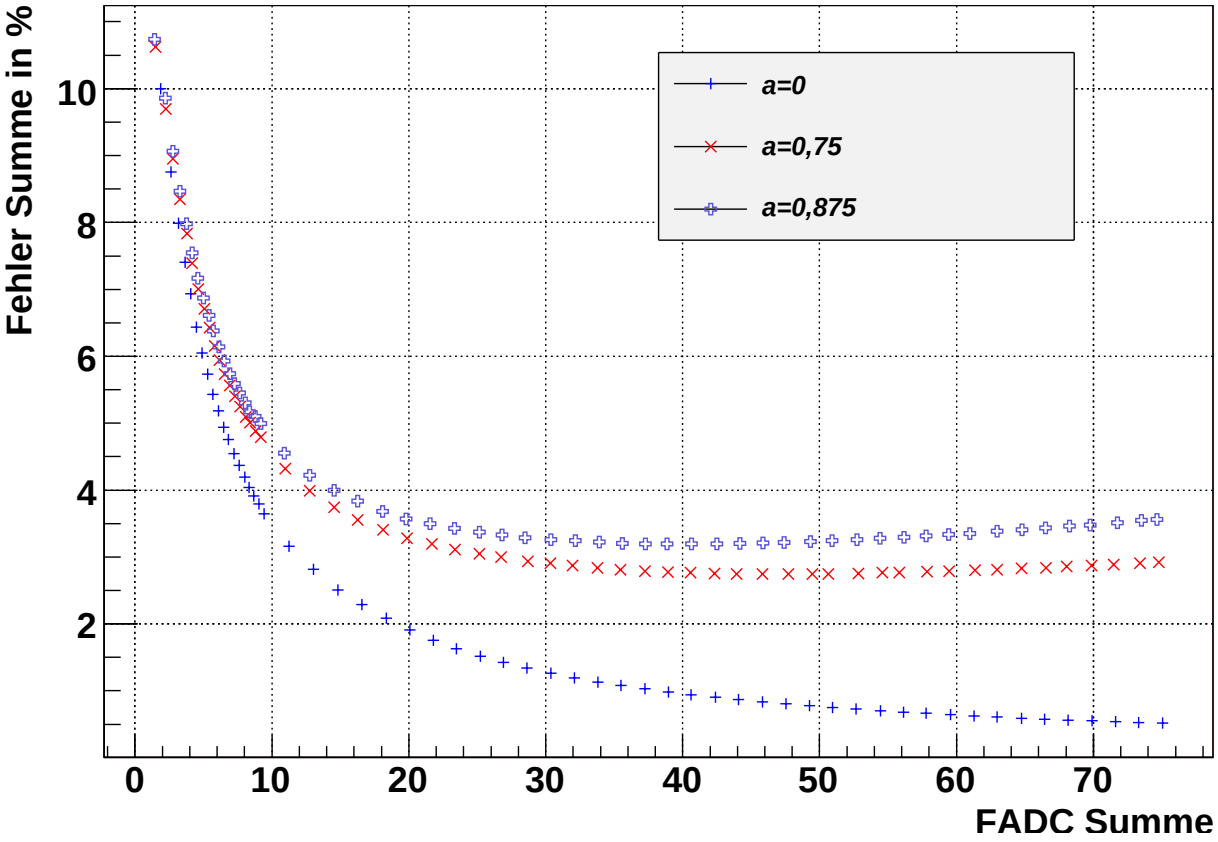


Abbildung 4.6 Simulierter Fehler des Integrals über einen Pulsverlauf bei einem DL300 Flash-ADC-System.

Für kleine Werte der FADC-Summe ergeben sich relativ hohe Fehler. Ab einer FADC-Summe von 10 beträgt der relative Fehler für alle Widerstandsverhältnisse weniger als 5 %. Für den linearen Betrieb mit a=0 sinkt der prozentuale Anteil des Fehlers bei steigender FADC-Summe immer weiter ab. Für einen nichtlinearen Betrieb steigt der relative Fehler ab einem Summenwert von etwa 45 wieder leicht an, was durch die nichtlineare Kennlinie und den größeren Abstand der aufeinander folgenden Digitalisierungswerte C bedingt ist.

Vergleicht man die erhaltenen Werte des abgeschätzten Fehlers für den Flash-ADC mit dem vom Hersteller des integrierenden ADC angegebenen Fehler von ± 3 %, so ergibt sich für FADC-Summen zwischen 20 und 70 ein Fehler in der selben Größenordnung. Hierbei sollte allerdings berücksichtigt werden, dass der Fehler für den FADC aufgrund fehlender Herstellerangaben geschätzt werden musste. Der Digitalisierungsfehler wird aufgrund von Ungenauigkeiten der verwendeten elektronischen Komponenten vermutlich etwas höher als der verwendete Quantisierungsfehler liegen.

Die Ergebnisse der Simulation ermöglichen die Bestimmung sinnvoller Messbereiche für die Verwendung des Flash-ADC-Systems zur Auslese der Kristalle am neuen Vorwärtsdetektor. Hierbei muss das Signal, ebenso wie beim Fastbus-ADC, in zwei Komponenten aufgeteilt werden, um den am Experiment auftretenden Energiebereich zwischen 20 MeV und 2 GeV abzudecken. Hierzu kann das Signal abgeschwächt auf einen zweiten FADC-Kanal gegeben werden.

Aufgrund des geringeren Digitalisierungsbereiches von effektiv 8 Bit wird ein etwas größeres Abschwächungsverhältnis von voraussichtlich 10:1 benötigt.

Im Weiteren wird die in dieser Arbeit durchgeführte technische Realisierung beschrieben.

5 Grundlagen technische Realisierung

Dieses Kapitel beschreibt die technischen Grundlagen für den Einsatz eines Flash-ADC-Systems (FADC) am CB-Transregio-Experiment. Hierzu werden die bereits vorhandenen Prozessoren (CPUs) des Experiments beschrieben, sowie eine Interface-Karte mit einem feldprogrammierbaren Logikbaustein, die in dieser Arbeit zur Implementierung einer schnellen Flash-ADC-Auslese verwendet wird.

5.1 DAQ CPUs - VMIC

Für die Datenerfassung werden VME-CPUs der Firma VMIC verwendet. Diese CPUs basieren auf Prozessoren der Firma Intel und werden im Experiment seit einigen Jahren erfolgreich für die Datenerfassung eingesetzt. Als Betriebssystem kommt hier Linux zum Einsatz[26]. Die älteren CPU-Karten verfügen über zwei 100-MBit-Ethernetschnittstellen, die neueren CPUs verfügen über eine 100-MBit- und eine 1-GBit-Ethernetschnittstelle. Von den beiden Schnittstellen wird im Experiment eine zur Steuerung und Kontrolle verwendet, die zweite Schnittstelle wird dediziert zur Übertragung der Daten genutzt, so dass hier eine hohe Übertragungsgeschwindigkeit von bis zu 100 MBit/s oder sogar 1 GBit/s möglich ist. Abbildung 5.2 enthält ein Schemabild der am Experiment verwendeten VME-CPUs.

Bild eines Tundra Universe II Chips für den Zugriff auf den VME-Bus

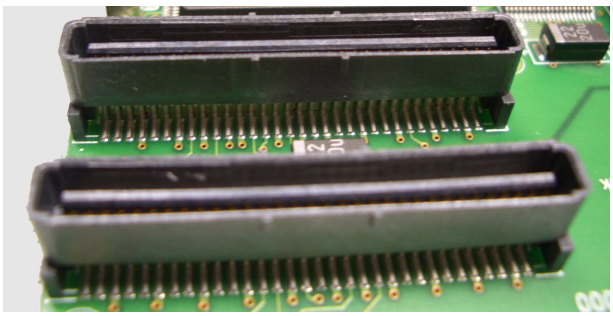


Bild eines 32-Bit PMC-Steckverbinders

Abbildung 5.1 VME-Chip und PMC-Schnittstelle einer VMIC VME-CPU.

Als Bridge-Chip zwischen PCI[27] und VME-Bus[28] kommt auf den Boards der Tundra Universe II Chip zum Einsatz (siehe Abbildung 5.1 links). Der am Experiment verwendete Linux-Treiber[29] für Intel-CPUs wurde am HISKP entwickelt und unterstützt DMA[30]- und Blocktransfers. Der

[26] Derzeit Debian Sarge mit Kernel 2.4.27.
[27] PCI - Abkürzung für den Busstandard Peripheral Component Interconnect
[28] VME - Abkürzung für den Busstandard VERSAmodule Eurocard Bus, entwickelt für Motorola 68000er CPUs
[29] http://universe2.sourceforge.net
[30] DMA (Direct Memory Access)

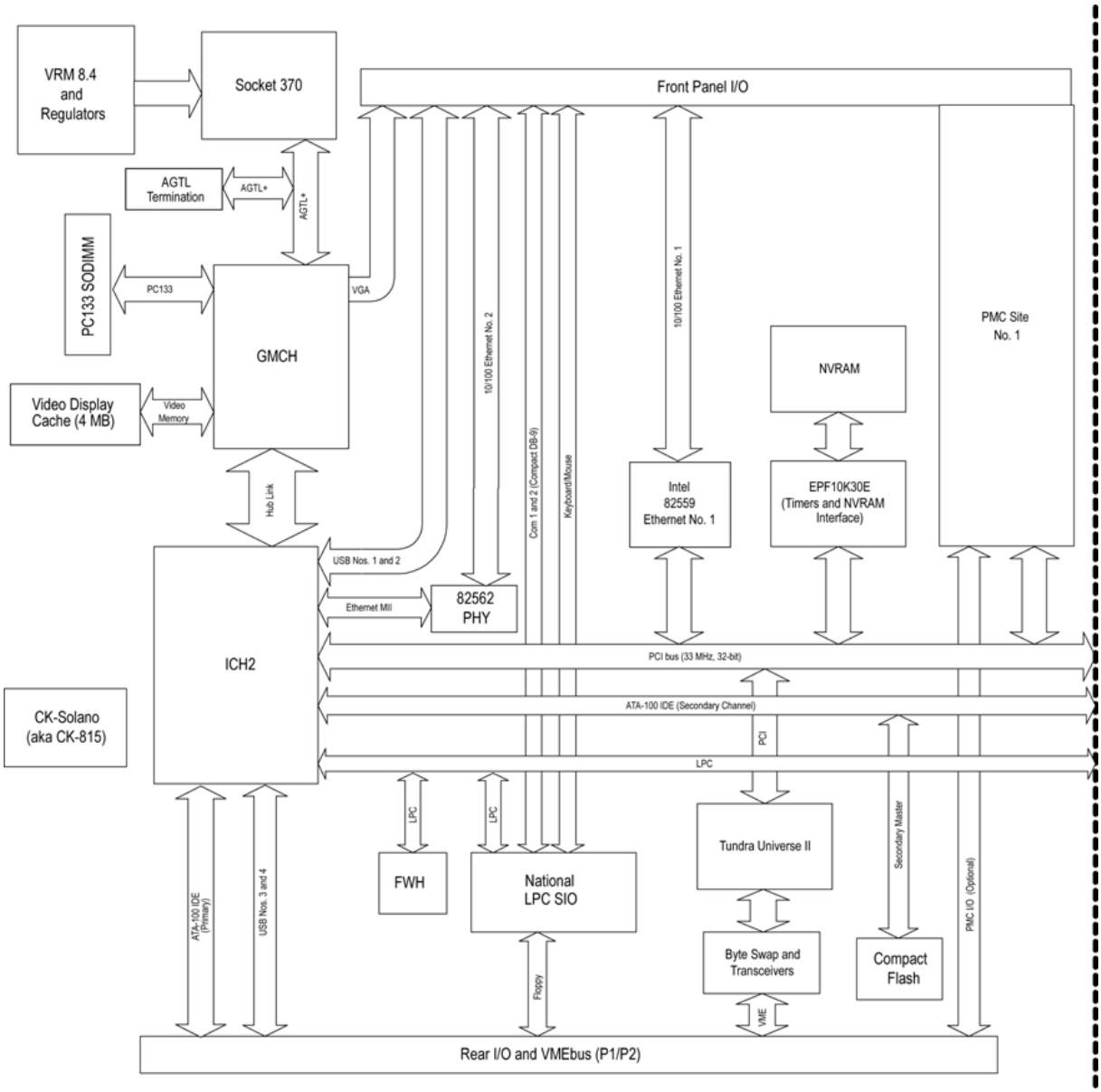


Abbildung 5.2 Schemazeichnung einer VMIVME-7750 CPU [GE 05].

Zugriff auf den Industriestandard VME wird benötigt, da der überwiegende Teil der Elektronik des Experiments als VME-Module realisiert ist.

Neben der Anbindung an den VME-Bus steht auf den CPUs ein PMC-Interface[31] zur Verfügung (siehe Abbildung 5.1 rechts). Dieser Interfacestandard entspricht einem PCI-Interface, wie es auch in einem handelsüblichen PC vorzufinden ist. Lediglich das Format der Steckkarte und der Anschluss sind kompakter. Eine VME-CPU mit aufgesteckter PMC-Karte benötigt nur einen Steckplatz in einem VME-Crate.

Der PCI-Bus ermöglicht einen direkten Datentransfer zwischen der angeschlossenen Peripherie und dem Hauptspeicher der VME-CPU mit einer sehr hohen Datenrate. Aufgrund der großen

[31] PMC - PCI Mezzanine Card, PCI-Steckkarte mit einem kompakten Steckerformat für Industriecomputer

zu erwartenden Datenmenge bei der Auslese mittels Flash-ADCs soll das Interface die Daten über den PCI-Bus übertragen.

Die derzeit eingesetzten CPUs verfügen über einen 32-Bit-PMC-Steckplatz mit 33,33 MHz Busfrequenz. Die maximale theoretische Übertragungsrate über den PCI-Bus beträgt hierbei etwa 133 MByte/s. Da aufgrund der Bus-Struktur mehrere Geräte angeschlossen sein können und der PCI-Bus ein geteilter Bus ist, kann bei Zugriffen auf andere Geräte am Bus die Übertragungsrate auch niedriger ausfallen. Hinzu kommen noch Verwaltungsaufgaben, bevor ein DMA-Transfer[32] initiiert werden kann, so dass die maximale Datentransferrate von 133 MByte/s faktisch nie erreicht wird. VME-CPUs der Firma VMIC mit einem 64-Bit-PCI-Bus mit 66 MHz sind seit kurzem ebenfalls verfügbar. Die maximale theoretische Datenübertragungsrate beträgt hier bis zu 533 MByte/s.

5.2 Struck DL300 Flash-ADC System

Das DL300-System der Firma Struck wurde am CERN am CB-LEAR-Experiment bereits für die Auslese der Jet-Drift-Chamber (JDC) verwendet [Mer88]. Das Flash-ADC-System ist ein modulares System bestehend aus einem Überrahmen (Crate) und verschiedenen Systemmodulen [Str87a]. Das System wurde in den 1980er Jahren am Physikalischen Institut der Universität Heidelberg entwickelt und von der Firma Struck in Serie produziert und vertrieben. Einsatz findet es in kernphysikalischen Forschungszentren, wie dem DESY und dem CERN bzw. in industriellen Forschungseinrichtungen. Der Haupteinsatzzweck waren Drahtkammern in kernphysikalischen Experimenten, bei denen das System zur Datenauslese verwendet wurde. Hieraus begründet sich auch die Tatsache, dass die Kanäle die Bezeichnungen Wire 0 left, Wire 0 right usw. tragen (siehe Anhang 6.2.1), da in den Experimenten die Drähte der Kammern jeweils am linken und am rechten Ende des Drahtes ausgelesen wurden.

Da mehr als 1200 Flash-ADC-Kanäle noch vom CB-LEAR-Experiment vorhanden sind und die Anschaffung neuerer Flash-ADC-Systeme sehr kostenintensiv ist (ca. 1000 Euro pro Flash-ADC-Kanal für ein SIS 3300 Flash-ADC), war es sinnvoll, Überlegungen zur möglichen weiteren Verwendung des DL300-Systems anzustellen. Ziel war es Kosten zu sparen und die noch vorhandene Elektronik für die Auslese des neuen Vorwärtsdetektors zu nutzen. Das System sollte so flexibel und performant gestaltet werden, dass es auch bei neuen Subdetektoren in den kommenden Förderperioden des Sonderforschungsbereiches Transregio 16 zum Einsatz kommen kann.

Hierbei musste jedoch das Interface zur DAQ-CPU neu entwickelt werden, damit eine wesentlich höhere Auslesegeschwindigkeit erzielt wird, um nicht die Datennahmerate des Experiments zu beschränken. Das am CERN eingesetzte DL300-System erreichte in ersten Tests eine Auslesegeschwindigkeit von ca. 30 Hz [Mer88]. Am CERN war während des Betriebs eine Auslese der Jet-Drift-Chamber von ca. 50 Hz möglich, wobei hier allerdings wesentlich weniger Daten

[32] DMA (Direct Memory Access) bezeichnet einen direkten Zugriff auf den Speicher eines Computers durch die angeschlossenen Peripheriegeräte ohne Umwege über die CPU.

übertragen werden mussten [Kal06]. Für das CB-Transregio-Experiment ist eine Auslesera-te von mindestens 1000 Ereignissen pro Sekunde (1000 Hz) mit einer wesentlich größeren Datenmenge je Ereignis angestrebt. Die Entwicklung eines mindestens 20 mal schnelleren CPU-Interface ist ein Hauptbestandteil dieser Arbeit.

Das System besteht aus Modulen in einem Überrahmen (Crate). Die Module werden in Unterkapitel 5.2.1 vorgestellt. Im Crate lassen sich zwei verschiedene FADC-Module einset-zen, die sich in der Speichertiefe unterscheiden, welche die maximale Erfassungszeit bei einer gewählten Erfassungsgeschwindigkeit (Samplingrate) bestimmt.

Die zentralen Leistungsdaten des DL300-Systems sind wie folgt:

- Erfassungsfrequenz (Samplingrate) von 25,50 und 100 MHz oder benutzerdefinierbar (bis max. 100 MHz),
- Auflösung 6 Bit nichtlinear (effektiv 8 Bit),
- Hardware-Hitscanner zur Suche nach „Treffern“ in den Datenspeichern.

Für den ersten Einsatz am neuen Vorwärtsdetektor zur Auslese der Crystal-Barrel-Kristalle ist die Auflösung von effektiv 8 Bit nicht ausreichend, um den gesamten Energiebereich von 20 MeV bis hin zu 2 GeV abzudecken. Aus diesem Grund wird das Signal geteilt und abgeschwächt auf einen zweiten Kanal gegeben. Hierdurch vergrößert sich der dynamische Bereich für die Messung, da zwei verschiedene Messbereiche zur Verfügung stehen.

5.2.1 Systembeschreibung

Insgesamt gibt es 7 Systemmodule [Str87a]. Zwei davon sind ADC-Module mit unterschiedli-chen Speichertiefen. Daneben sind noch Interface-, Test- und Prozessormodule verfügbar. Das DL300 besteht aus den folgenden Modulkomponenten:

- DL305 - 4-Kanal ADC-Modul mit 6 Bit Auflösung nichtlinear (effektiv 8 Bit), 256 Speicher-stellen und max. 100 MHz Samplingrate [Str87b]
- DL310 - wie DL305 nur mit 1024 Speicherstellen [Str87c]
- DL301 - Camac Interfacemodul
- DL302 - Zentrales Kontrollmodul mit Scanner- und Hitdetektoreinheit [Str87d]
- DL307 - Prozessormodul zum Anschluss an eine Motorola VME-CPU (CPU07) [Str87e]
- DL303 - Test- und Kalibrationsmodul
- DL309 - Busmonitor-, Speicher- und Monitormodul mit 12-Bit-DAC

Für die Module gibt es einen Überrahmen, der in 19-Zoll Racks nach DIN 41494 eingebaut werden kann. Das Crate inklusive Netzteil benötigt 6 Höheneinheiten (Doppel-Europakarten). Ein Crate kann insgesamt 28 Module mit einer Platinengröße von 223 mm x 160 mm aufnehmen. Die Modulsteckplätze sind von links nach rechts von 0 bis 27 durchnummeriert. Die Rückwandplatine (Backplane) des Cratesystems besteht aus einem analogen und einem digitalen Bus. Die Steckverbinder der digitalen Backplane sind hierbei kompatibel zu den Standard VME-Steckertypen. Der Transfer der digitalen Signale erfolgt über einen ECL[33] kompatiblen Bus mit einer 18-Bit-Adressleitung und 16-Bit-Datenleitungen.

Zum Anschluss der Signalkabel gibt es auf der Rückseite zwölf 40-polige Pfostenverbinder nach DIN 41651. Aufgrund der primären Verwendung des Systems für Driftkammern sind die Steckverbinder (willkürlich) aufgeteilt in Eingänge für die linken und rechten Signaldrahtenden.

Die Spannungsversorgung ist als primär getaktetes Schaltznetzteil aufgebaut. Der Stromverbrauch liegt bei über 50 A bei -5.2 V.

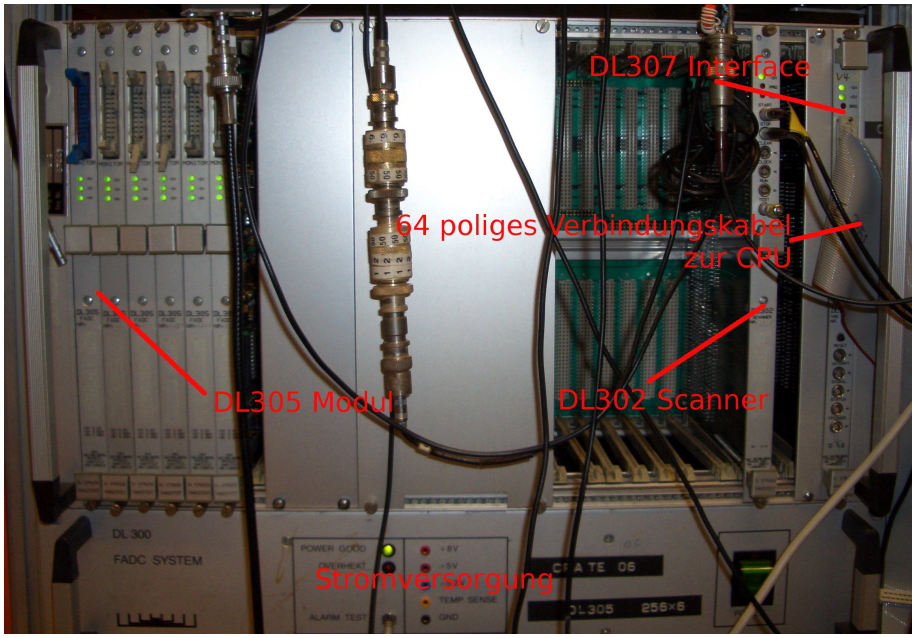


Abbildung 5.3 Foto eines Struck DL300 Flash-ADC-Systems.

Im System sind typischerweise 24 ADC-Module vom Typ DL305 oder DL310 im Einsatz, welche jeweils nur die Steckplätze von 0-23 verwenden dürfen. Sind weniger Module im Einsatz, so müssen die Steckplätze von Position 0 beginnend zuerst genutzt werden. Die Positionen 24-27 sind reserviert für Module zur Steuerung des Systems. Eine Standardbestückung sieht wie folgt aus:

- Position 0 bis 23 - ADC Module DL305 oder DL310
- Position 24 - Testermodul DL303

[33] Emitter Coupled Logic: Emittergekoppelte Logik

- Position 25 - Interfacemodul DL301 (Camac) oder DL307 (VME)
- Position 26 - Scanner- und Hitdetektormodul DL302
- Position 27 - Monitormodul DL309

Die Positionen der Test-, Interface- und Monitormodule sind allerdings nicht an die Positionen 24-27 gebunden. Diese Modultypen können jeden Platz im Cratesystem einnehmen. Für die Verbindung zu einer VME-CPU steht eine spezielle Interfacekarte DL307V zur Verfügung, die über ein 64-poliges Kabel mit der Interfacekarte DL307 im DL300-Crate verbunden wird. Die DL307V-Interfacekarte ist hierbei als Aufsteckkarte für eine CPU07 der Firma Microsys konzipiert und lässt sich nur mit dieser CPU verwenden. Die CPUs der Firma Microsys basieren auf Motorola-68000er-Prozessoren und genügen bei weitem nicht mehr den Anforderungen für eine Auslese mit hoher Datenrate. Die Aufsteckkarte DL307V ist in TTL-Logik ausgelegt. Ansatz für ein modernes und schnelles Interface an diesen Spezial-Bus sind kommerzielle hardwareprogrammierbare Adapterkarten auf der Basis rekonfigurierbarer Logik.

5.3 Rekonfigurierbare Logik (PALs, FPGAs und ASICs)

Standard integrierte Schaltkreise (z.B. TTL-Logik-ICs) haben den Nachteil, dass diese in ihrer Funktion festgelegt sind und nicht verändert werden können. In den letzten Jahrzehnten gab es daher mehrere Entwicklungen, die programmierbare Logikschaltkreise hervorbrachten. Die ersten vollständig konfigurierbaren Bausteine sind seit etwa 1978 erhältlich. Diese PAL (Programmable Array Logic) genannten ICs können von extern programmiert werden. Sie enthalten eine programmierbare logische Matrix mit UND- und ODER-Verknüpfungen, einen Eingabeblock und einen Ausgabeblock, sowie eine programmierbare Rückkopplung. Mit den UND- und ODER-Verknüpfungen lässt sich nahezu jede beliebige logische Verknüpfung zwischen Eingangs- und Ausgangssignalen erreichen. Ab einer Komplexität von mehr als 25000 Gattern spricht man von CPLDs (Complex Programmable Logic Devices). CPLDs haben den Vorteil, dass sich Signallaufzeiten aufgrund der fest definierten Signalflüsse exakt berechnen lassen. In den letzten Jahren sind neben den CPLDs feldprogrammierbare Gate-Arrays (FPGAs) immer beliebter geworden. FPGAs bestehen aus einer Matrixstruktur mit frei konfigurierbaren Logikeinheiten, Ein- und Ausgabebereichen (IOBs) und einer Verbindungsebene sowie SRAM-Speicherblöcken. Die Verbindungsebenen können die Logikeinheiten und die Ein- und Ausgänge miteinander verbinden. Es können nahezu beliebige Verbindungen zwischen den Logikblöcken geschaltet werden. Man spricht deswegen von einem „Verteilungsmeer“[34]. Im Gegensatz zu PLDs kann man bei FPGAs komplexe Verbindungsmuster schalten. Man unterscheidet zwischen SRAM-basierten Bausteinen, die beliebig häufig neuprogrammierbar sind und programmierbaren FPGAs in Anti-Fusetechnik. Letztere sind nur ein einziges Mal programmierbar. Die Komplexität und die maximalen Taktfrequenzen von FPGAs sind in den letzten Jahren stetig gewachsen. Aktuell sind Bausteine mit über 10 Millionen Gattern erhältlich. Neben der freiprogrammierbaren Logik enthalten FPGAs mittlerweile auch komplette Prozessoren, so dass ein Chip die Möglichkeit bietet,

[34] engl. sea of routing

Software auf der integrierten CPU auszuführen sowie eigene Logik auf dem FPGA zu implementieren. Der Vorteil von FPGAs gegenüber Standardprozessoren ist, dass auf den FPGA-Bausteinen massiv-parallele Berechnungen ausgeführt werden können. Prozessoren arbeiten die ausgeführten Programme zumeist sequentiell ab, jede Instruktion benötigt hierbei mehrere Taktzyklen. Es gibt bei neuen Prozessoren spezielle Instruktionen, wo ein Befehl auf eine große Menge an Daten angewandt werden kann (SIMD - Single Instruction Multiple Data). Allerdings ist bei FPGAs eine viel stärkere Parallelisierung möglich.

Abbildung 5.4 Bild eines Xilinx Virtex-II FPGA.

Neben FPGAs gibt es einen weiteren konkurrierenden Ansatz, um eine benutzerdefinierte Logik auf einem Chip unterzubringen: die sogenannten applikationsspezifischen integrierten Schaltkreise (ASICs)[35]. Hierbei steht am Ende des Entwicklungsprozesses ein fertiger Chip, der nicht mehr modifiziert werden kann. Die Vorteile von ASICs gegenüber FPGAs sind durch höhere Taktfrequenzen und eine größere Logikdichte gegeben. Hieraus ergibt sich auch ein niedrigerer Strombedarf. Allerdings sind prinzipbedingt keine Änderungen am Design mehr möglich. Sinnvoll ist der Einsatz von ASICs, wenn entsprechend hohe Stückzahlen notwendig werden oder der Stromverbrauch relevant ist (z.B. in Mobiltelefonen). Für kleinere Stückzahlen (kleiner 10000-100000 Stück) rechnet sich der Einsatz von ASICs im Allgemeinen nicht.

Einsatz finden FPGAs vor allem in Anwendungen, wo massiv-parallele Berechnungen notwendig sind. Paradebeispiele hierfür sind der Einsatz bei der Analyse von Signalen von Radioteleskopen mit Hilfe von schnellen Fouriertransformationen[36] oder in Schachcomputern. In der Physik werden FPGAs für Datenerfassungssysteme verwendet. Beispiele hierfür sind das ALICE-Experiment, welches in den nächsten Jahren in Genf am Large-Hadron-Collider (LHC) in Betrieb gehen wird und das COMPASS-Experiment, welches derzeit Messungen zur Spinstruktur des Protons am CERN vornimmt. Am COMPASS-Experiment werden die an der Universität Freiburg entwickelten Catch-Module verwendet, welche als Hauptkomponenten FPGAs der Firma Xilinx nutzen.

[35] engl. ASIC - Application Specific Integrated Circuits
[36] Fast-Fourier-Transformations - FFT

Für den Einsatz am CB-ELSA-Experiment ist wegen der geringen Stückzahl eine Verwendung von FPGAs naheliegend. CPLDs sind aufgrund der begrenzten Designmöglichkeiten keine Alternative, ebenso wie die Verwendung von ASICs, die sich erst für Massenproduktion eignen.

5.4 Hardwarebeschreibungssprache VHDL

VHDL oder auch VHSIC HDL steht für Very High Speed Integrated Circuit Hardware Description Language[37]. Die Sprache wurde in den frühen 1980er Jahren vom US-amerikanischen Verteidigungsministerium entwickelt und in einer ersten Fassung im Jahre 1985 kommerziell veröffentlicht. Seit 1993 ist die Sprache als IEEE-Standard akzeptiert und danach in mehreren Revisionen minimal modifiziert worden. Die Sprache dient der Beschreibung und Simulation von komplexen Schaltkreisen. Sie soll einen Austausch von Hardwarebeschreibungen und Dokumentation von Schaltkreisen in einem einheitlichen Format zwischen verschiedenen Institutionen ermöglichen. Desweiteren kann mit VHDL sehr effizient und modular ein Design eines elektronischen Schaltkreises beschrieben werden. Es ist sogar möglich komplexe Prozessoren mit mehreren Millionen Transistoren effizient in VHDL zu entwickeln.

Neben der reinen Beschreibung kann ein entwickelter Schaltkreis vor dem Einsatz auf dem Chip auf einem Rechner getestet werden. Hierzu können vom Entwickler Simulationen geschrieben werden, die alle möglichen Zustände des Schaltkreises testen. Dabei ist es wichtig, dass die Schaltkreise in kleine Teilmodule zerlegt werden, um somit die gewünschte Funktionalität separat testen zu können.

Neben VHDL existiert noch die Hardwarebeschreibungssprache Verilog. Diese wurde 1985 ursprünglich als reine Simulationssprache entworfen, bietet aber mittlerweile auch die Möglichkeit Schaltkreise zu beschreiben. Kommerzielle Entwicklungsumgebungen wie die in Kapitel 5.6 beschriebene ISE-Software können zumeist mit beiden Sprachvarianten umgehen. Für die Beschreibung von Intellectual-Property (IP) Modulen hat sich in den letzten Jahren allerdings VHDL durchgesetzt. IP-Module sind fertig entwickelte Schaltkreise, die zumeist käuflich zu erwerben sind. Die genauen Funktionen der Module sind dann spezifziert und können in eigene Designs eingebunden werden. Das erlaubt die Integration von komplexen Bussen, Protokollen und arithmetischen Operationen, ohne dass hierfür eigene Entwicklungsarbeit betrieben werden muss. Analog zur quelloffenen Software im GNU/Linux-Umfeld existieren mittlerweile Projekte, die auch komplexe Busse und Protokolle frei im Quellcode zur Verfügung stellen. Für die Nutzung in eigenen Projekten fallen somit keine Kosten an. Eines der bekanntesten Projekte ist das Opencores Projekt[38].

Ein Beispiel eines VHDL-Codes ist anhand eines einfachen D-Flip-Flops in Quellcode 5.1 gezeigt.

[37] engl. Very High Speed Integrated Circuit Hardware Description Language - Gerätebeschreibungssprache für integrierte Hochgeschwindigkeitsschaltkreise

[38] http://www.opencores.org/

```
1       entity dff is
2               port(
3                       d,clk: in BIT;
4                       q: out BIT);
5       end def;
6       architecture behav of dff is
7       begin
8       process(clk)
9               if (clk='1' and clk'event) then
10                      q <= d;
11              end if;
12              end process;
13      end behav;
```

Quellcode 5.1 Beispiel eines D-Flip-Flops in VHDL.

Zu jeder VHDL Designbeschreibung gehört zunächst eine Entity (Einheit/Gebilde) und eine Architecture (Architektur). Die Entity muss einen eindeutigen Namen tragen, der nach dem Wort Entity angegeben wird. Im Entity-Teil werden dann die Ein- und Ausgabesignale des Designs beschrieben. Im vorliegenden Fall sind dies die typischen Eingangssignale für ein D-Flip-Flop **d** und der Taktgeber **clk** (Clock). Als Ausgang ist das Signal **q** definiert. In der Entity-Definition wird jeweils ein Name für das Signal, sowie eine Richtung (Ein- oder Ausgang oder beides) und ein Typ benötigt. Der darauffolgende Architecture-Teil beschreibt dann die Funktion der Komponente (entity), auf die sich die Architektur explizit beziehen muss (architecture ... of ... is). Jede Entity benötigt mindestens einen Architecture-Teil. Es können auch mehrere Architekturen angegeben werden, um darüber eine Simulation zu schreiben oder auch um alternative Schaltkreise zu testen. Im Quelltext können danach dann lokale Signale und Konstanten definiert werden, die für die Interaktion des Schaltkreises mit der Außenwelt keine direkte Rolle spielen. Die definierten Signale sind dann nur lokal in der Architektur in Verwendung. Darauf folgt ein **begin**, welches am Ende des oben gezeigten Quelltextes durch **end behav** abgeschlossen wird. Dazwischen können beliebig viele Zuweisungen stehen. Mit einer Prozess-Anweisung lassen sich sequentiell arbeitende Schaltungen entwickeln. Ein Prozess wird in VHDL als eine eigenständige Einheit behandelt, die parallel zu allen anderen „ausgeführt“ wird. Ein Prozess kann über sensitive Eingänge verfügen (sensitivity lists). Diese werden in der Deklaration eines Prozesses angegeben. Der Prozess wird ausgeführt, wenn sich eines der Signale in der Sensitivitätsliste verändert. In Quellcode 5.2 ist ein Beispiel für einen Prozess mit einer Sensitivitätsliste angegeben.

Der Prozess reagiert auf die Signale **rst** und **clk**. Das heißt erst bei einer Änderung eines der beiden Signale, werden die zwischen Zeile 2 und Zeile 12 stehenden Bedingungen und Anweisungen ausgeführt. Ist das Signal **rst** aktiv (high), so wird das Register **q** auf 0 gesetzt. Andernfalls werden die zwischen 5 und 11 stehenden Anweisungen ausgeführt, wenn das Clock-Signal auf high gesetzt ist und sich das Signal geändert hat (**clk'event**). Hierdurch wird der „Programmteil“ somit nur auf der ansteigenden Flanke des Clock-Signals ausgeführt. Der oben gezeigte Prozess implementiert ein getaktetes Schieberegister. Auch wenn die VHDL-

```
1       process(rst, clk)
2       begin
3       if rst = '1' then
4               q <= "00000000";
5       elsif clk = '1' and clk'event then
6               if load = '1' then
7                       q <= Data_in;
8               else
9                       q <= q(1 to 7) & q(0);
10              end if;
11      end if;
12      end process;
```

Quellcode 5.2 Beispiel eines Prozesses mit Sensitivity-List.

Anweisungen bekannten Programmiersprachen ähneln, so ist die Entwicklung eines Designs nie ganz losgelöst von der darunterliegenden Schaltkreislogik zu sehen. Es müssen bestimmte Regeln und Voraussetzungen der zu beschreibenden Hardware berücksichtigt werden, damit ein Design mit einem VHDL-Compiler auch korrekt übersetzt und implementiert werden kann. Ein Verständnis dieser Grundlagen ist Voraussetzung, um ein korrektes VHDL-Design implementieren zu können. Ein weiterer großer Unterschied zu Programmiersprachen wie C oder C++ ist, dass die Entwicklungszeit pro Quelldateizeile bei VHDL wesentlich höher ist. Grund hierfür ist, dass durch VHDL ein elektronischer Schaltkreis modelliert wird. Die erstellte Logik kann über einen Testbench simuliert und intensiv getestet werden. Insbesondere durch die gleichzeitige „Ausführung“ des Programmcodes kann es zu einem unterschiedliches Zeitverhalten in den einzelnen Schaltkreisen kommen. Daher ist eine Synchronisierung zwischen den VHDL-Modulen notwendig, was bei einem linear ablaufenden C- oder C++-Code ohne Parallelverarbeitung nicht der Fall ist. Kommerzielle Entwicklungsumgebungen unterstützen das Arbeiten mit VHDL-Code durch Bereitstellung von Editoren, Funktionen zur Syntaxprüfung und logischen Prüfungen, Hardware-Compilern und die Programmdateierstellung für spezielle FPGA-Bausteine.

5.5 PMC-FPGA-Karte

Für die Anbindung des DL300-Systems an den PCI-Bus der VMIC-CPU unter Verwendung eines programmierbaren FPGAs gibt es Karten verschiedener Hersteller, die in Frage kommen. Eine typische Karte ist in Abbildung 5.5 gezeigt.

Bei der Auswahl des Herstellers war wichtig, dass Linux-Treiber für den Linux-Kernel 2.4 und 2.6 verfügbar sind und diese Treiber vom Hersteller im Quellcode mitgeliefert werden. Hierdurch sind individuelle Anpassungen und auch das Übersetzen für eine andere Linux-Distribution möglich. Dies war insbesondere wichtig, da im Experiment Debian Linux eingesetzt werden sollte, welches häufig nicht von einem Hardwarehersteller direkt mit Binärtreibern unterstützt wird. Zum Zeitpunkt der Entscheidung für eine PMC-FPGA Karte war die Firma Alpha-Data aus England der einzige Hersteller, der diese Kriterien mit einem Produkt voll erfüllte. Die

Abbildung 5.5 FPGA PMC-Karte der Firma Alpha-Data [Alp04a].

Firma produziert PMC-Karten in verschiedenen Varianten, welche neben einem FPGA, einem PCI-Controller und einem frei belegbaren Anschluss zusätzlich noch Speicher (SSRAM[39]) zur Verfügung stellt. Dieser Speicher kann vom FPGA und über den PCI-Bus via DMA angesteuert und ausgelesen werden. Schematisch sieht die Karte wie in Abbildung 5.6 gezeigt aus.

Die FPGA-Karte ist mit verschiedenen programmierbaren Bausteinen der Firma Xilinx lieferbar. Für diese Arbeit wurde das kleinste Modell ADM-XRC/400-4/4 ausgewählt, welches mit einem Xilinx Virtex V400 FPGA bestückt ist. Auf der Karte sind vier unabhängige RAM-Bänke mit ZBT-SSRAM-Speicher zu je 256k x 36 vorhanden (insgesamt 4 MByte SSRAM-Speicher). Die Karte kostet etwa 3000 Euro. Als Interface zum PCI-Bus dient auf der Karte der PCI-Bus-Chip PLX 9080. Der FPGA-Chip kann entweder über den PCI-Bus, über eine JTAG-Schnittstelle oder über einen auf der Karte vorhandenen Flash-Speicher programmiert werden. Sofern die Konfiguration über den Flash-Speicher erfolgt, wird der FPGA direkt beim Einschalten der CPU

[39] SSRAM - engl. Synchronous Static Random Access Memory - synchroner statischer Speicher mit wahlfreiem Zugriff (Anordnung des Speichers in einer Matrixform)

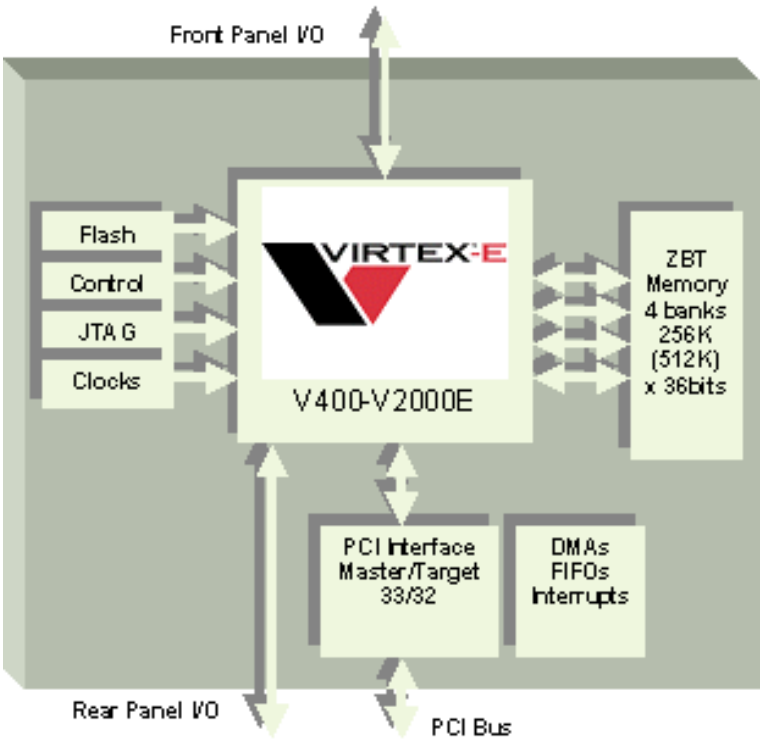


Abbildung 5.6 Schematische Darstellung der FPGA PMC-Karte der Firma Alpha-Data [Alp04b].

mit der im Flashspeicher abgelegten Datei (Bitstream-Datei) initialisiert. Neben dem Treiber im Quellcode wird vom Hersteller auch ein Software-Development-Kit (SDK)[40] mitgeliefert, welches die einfache Ansteuerung der Karte ermöglicht. Dies wird in Kapitel Anhang 6.1.2 im Detail beschrieben.

5.6 Xilinx Entwicklungsumgebung

Um den FPGA-Chip Virtex V400 der Firma Xilinx programmieren zu können wird die Entwicklungssoftware ISE von Xilinx benötigt. Kernbestandteil der Software ist der VHDL-Editor und Compiler, der aus einem VHDL-Code ein für den eingestellten Chiptyp kompatibles Bitstreamfile erzeugt. Die Software übernimmt dabei die sehr komplexe Aufgabe, aus dem VHDL-Code des Nutzers und einer User-Constraints-Datei, die die Funktion der Ein- und Ausgabe-Anschlüsse und zeitliche Vorgaben für Signalverläufe festlegt, eine fertige Bitstreamdatei zu erzeugen. Zunächst werden von der Software Syntaxüberprüfungen des eingegebenen Codes durchgeführt (Synthetisierung) und nachfolgend wird versucht den übersetzten Code in mehreren Schritten auf dem FPGA zu implementieren (Implementierung). Bei der Implementierung wird im Schritt Translate aus den im Synthetisierungsprozess erstellten Netzlisten eine Native Generic Database erstellt. In der Implementierung werden im Weiteren die Schritte Kartieren (Map) sowie Platzieren und Pfadplanung (Place & Route) durchgeführt. Hier sind manuelle Eingriffe möglich, um die Platzierung der Logikkomponenten auf dem FPGA abweichend von den Vorgaben

[40] engl. Software-Development-Kit (SDK) - Software-Entwicklungspaket

durch die Software durchzuführen. Nach der Implementierung kann dann die Bitstreamdatei erzeugt werden, welche über eine JTAG Schnittstelle an den FPGA übertragen werden kann. Die notwendige Software Impact kann diesen Schritt direkt ausführen. Da sich der FPGA auf der Alpha-Data-Karte über den PCI-Bus programmieren lässt, wird hiervon kein Gebrauch gemacht. Die Bitstreamdatei wird auf einem Dateisystem des lokalen Eventbuilders abgelegt und kann durch eigene Software an den Chip übertragen werden.

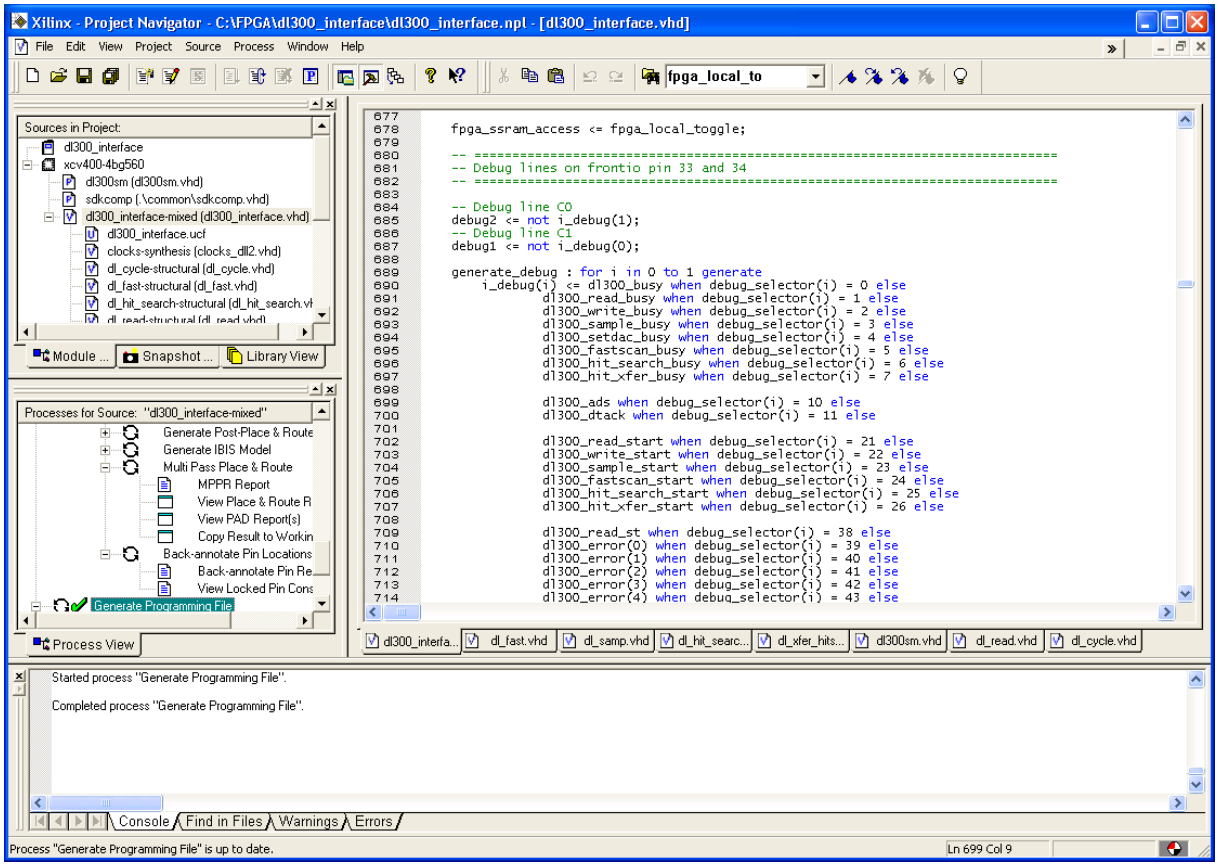


Abbildung 5.7 Xilinx ISE Software.

Die in dieser Arbeit erstellten VHDL-Dateien und deren Funktionsweise werden in Kapitel 6 präsentiert. In Kapitel 7 sollen die wesentlichen Ergebnisse bei ersten durchgeführten Messungen mit dem für diese Arbeit implementierten Interface und eine Analyse zur Auslesegeschwindigkeit vorgestellt werden.

6 Implementierung des DL300-CPU-Interfaces

Im Folgenden soll dokumentiert werden, wie das DL300-System der Firma Struck über den FPGA-Chip auf der Alpha-Data-Karte angesteuert wird. Hierzu wird auf die technischen Spezifikationen der verwendeten FPGA-PMC-Karte und des Flash-ADC-Systems im Detail eingegangen. Diese sind von den jeweiligen Herstellern vorgegeben. Es ist kein kommerziell erhältliches Interface für die schnelle Anbindung des DL300-Systems verfügbar, so dass dies für diese Arbeit zur Messung am geplanten Doppelpolarisationsexperiment an ELSA mit dem Crystal-Barrel-Detektor entwickelt werden musste. Im Weiteren werden die für das Interface im Rahmen diese Arbeit entwickelten Funktionen für die Anbindung der FPGA-Karte an den PCI-Bus, die Ansteuerung des SSRAM, die Fehlerdiagnosemöglichkeiten, das Datenformat und die Implementierung der Statemaschinen für die Kommunikation mit dem DL300-System im Detail beschrieben.

6.1 Alpha-Data-Karte

Die FPGA-Karte von Alpha-Data wurde bereits grob in Kapitel 5.5 beschrieben. Im Weiteren soll hier nun dargestellt werden, wie die Karte in dieser Arbeit verwendet wird.

6.1.1 Externer Anschluss

Über einen 34-poligen Frontstecker ermöglicht die FPGA-Karte den Anschluss von beliebigen Signalleitungen. Für die Anbindung an das DL300-System sind hierzu 32 Leitungen in Verwendung. Zwei zusätzliche Leitungen ermöglichen das Herausführen von internen FPGA-Signalen an den Frontstecker, um so über einen Logic-Analyzer oder ein Oszilloskop das zeitliche Verhalten von Signalen extern beobachten zu können. Die Zuordnung von externen Signalen zu FPGA-Pins erfolgt über das User-Constraints-File. Da die FPGAs auf den ADM-XRC-Karten fest verlötet sind, ist die Zuordnung von FPGA-Pins zu den Pins am Stecker an der Vorderseite durch den Hersteller der Karte vorgegeben. Die Zuordnung zu den internen VHDL-Signalnamen erfolgt in der UCF-Datei (User-Constraints-File). Folgende Signale des DL300-Systems müssen zugeordnet werden:

- 16 Datenleitungen D0-D15
- 11 Adressleitungen A0-A10
- Lese-/Schreibleitung R/W
- Adress-Strobe-Leitung ADS
- Reset-Leitung RST
- Data-Acknowledge-Leitung DTACK

Die genauen Zuordnungen sind im Anhang in Tabelle A.1 zu finden.

6.1.2 Application-Programming-Interface (API)

Das Application-Programming-Interface[41], welches zur FPGA-Karte mitgeliefert wurde, erlaubt die einfache Ansteuerung und Programmierung der Karte. Der Nutzer des Software-Development-Kit (SDK) kann sich somit vollständig auf das Programmieren der eigenen Anwendung konzentrieren. Das SDK bietet Funktionen für die Initialisierung der Karte, das Konfigurieren des FPGA mit einer externen Bitstreamdatei, das Setzen der Clock-Generatoren, Funktionen für Datentransfer und Interrupt-Handling und Funktionen zur Fehlerdiagnose.

Im Anhang A.2 sind die in Kapitel 6.11 verwendeten Funktionen und Variablen des SDK beschrieben. Für die Programmierung kam die Version 2.3.0 des SDK zum Einsatz. Für weitere Funktionen sei auf die Dokumentation des SDK von der Firma Alpha-Data verwiesen [Alp03].

6.2 DL300-System

Das verwendete DL300-System der Firma Struck soll im Folgenden beschrieben werden. Hierbei werden die FADC-Module DL305 und DL310 sowie die VME-Interfacekarte DL307V und DL307 vorgestellt. Das DL302-Scanner-Modul wird über die DL307-Interfacekarte angesprochen und steuert die Datenerfassung und die anschließende Suche nach Ansprechern im Speicherbereich der FADC-Module.

6.2.1 ADC-Module DL305 und DL310

Die ADC-Module DL305 und DL310 sind nahezu identisch aufgebaut, unterscheiden sich jedoch in der Speichergröße je Kanal. Jedes Modul verfügt über 4 unabhängige FADC-Kanäle mit je 6-Bit Auflösung (nichtlinear) und einem RAM-Speicher, welcher für jeden Kanal beim DL305 256 Worte zu je 6 Bit und beim DL310-Modul 1024 Worte ebenfalls zu je 6 Bit umfasst. Jedem FADC-Chip ist ein Verstärker vorgeschaltet. Der Speicher von zwei der vier Kanälen ist mit den Datenleitungen D0-D5 der digitalen Backplane verbunden. Der zweite Speicher der anderen beiden Kanäle ist mit den Datenleitungen D8-D13 verbunden. Bei einer Adressierung über den digitalen Bus werden somit zwei FADC-Kanäle parallel angesprochen, so dass diese auch parallel ausgelesen werden könnnen. Die Adressierung der Speicherstellen beim DL310 geschieht wie in Tabelle A.2 im Anhang angegeben. Bei den DL305-Modulen erfolgt dies analog, nur aufgrund der geringeren Speichergröße mit kleineren Adressabständen zwischen den Modulen (siehe Tabelle A.3 im Anhang).

6.2.2 DL307-Interface-Modul

Das DL307-Interfacemodul ermöglicht die Adressierung, Steuerung und Auslese des DL300-

[41] engl. Application-Programming-Interface (API) - Schnittstelle zur Anwendungsprogrammierung

Systems über eine VME-CPU. Hierbei wird das Modul durch ein 64-poliges Verbindungskabel (SV-Bus) mit einer Personality-Card DL307V verbunden, die ihrerseits für einen CPU07-Prozessor auf Basis einer Motorola-68000-CPU konzipiert ist. Da diese CPU nicht mehr verwendet wird und der Anschluss nun über die Alpha-Data-Karte erfolgen soll, wurden die Ausgänge der Personality-Card mit dem Frontanschluss der FPGA-Karte verbunden. Die Belegung des Frontanschlusses ist Tabelle A.1 im Anhang zu entnehmen. Die Anschlüsse auf der DL307V-Karte sind in der Dokumentation [Str87e] des DL300-Systems zu finden. Die Interfacekarte verfügt über einen Adressraum von 8 KByte. Um den gesamten Adressraum des DL300-Systems ansprechen zu können, sind spezielle Zugriffe notwendig. Hierzu sind auf der Karte zwei AC-Adress-Register[42] implementiert (siehe Tabelle 6.1), über die die Adressierung im DL300 festgelegt werden kann. Neben den AC-Registern gibt es weitere Register, mit denen Datenzugriffe durchgeführt und Statusmeldungen abgerufen werden können. Die Register in Tabelle 6.1 sind auf dem Interface verfügbar, hierbei bezeichnet der Typ den möglichen Datenzugriff (R für read - Lesezugriff, W für write - Schreibzugriff und R/W für read/write - Lese- und Schreibzugriff)

Der gestreckte Zyklus ist für das Schreiben der DAC (Digital-to-Analog-Converter) notwendig, da diese über ein spezielles Verfahren bitweise langsam beschrieben werden müssen (siehe Kapitel 6.7.5). Das Statusregister, welches über die Adresse 0x406 gelesen und geschrieben werden kann, enthält die im Anhang in Tabelle A.5 aufgeführten Informationen.

6.2.3 DL302-Scanner-Modul

Das Dl302-Scanner-Modul ist das zentrale Steuer- und Kontrollmodul des gesamten Systems. Der Scanner steuert die Datennahme und verfügt über einen eingebauten Hit-Scanner, der im Adressraum des DL300-Systems selbstständig nach Einträgen suchen kann, die einen vorgegebenen Schwellenwert überschreiten.

Man unterscheidet die zwei folgenden Modi:

- Sampling-Modus – Der Sampling-Modus selektiert alle FADC-Module gleichzeitig und sorgt dafür, das die Module den Digitalisierungsprozess ausführen und die digitalisierten Werte der Eingangssignale speichern.

- Fastscan-Modus – Der Fastscan-Modus selektiert die Module nacheinander entsprechend der (Speicher-) Position. Die ausgegebenen Daten werden dann von der Hitdetektor-Logik genutzt.

Für eine Datenerfassung von Messwerten wird zunächst der Scanner in den Sampling-Modus geschaltet. Für das CB-Transregio-Experiment, im Einsatz an den Kristallen in Vorwärtsrichtung, soll das System im Common-Stop-Modus betrieben werden. Dabei wird zunächst der

[42] engl. Adress Counter(AC) - Adresszähler

Name	Adresse	Typ	Bedeutung
IFDATA	0x400	R/W	Datenzugriff auf Adresse A0-A17, die über das AC-Register festgelegt wird
IFAUTO	0x401	R/W	Datenzugriff auf Adresse A0-A17 mit automatischer Adresserhöhung, dabei wird die Adresse über das AC-Register festgelegt und das Adressregister nach einem Schreibzugriff um 1 erhöht. Bei einem Lesezugriff wird die Adresse vor dem Lesezugriff erhöht
IFSCYCL	0x402	R/W	Datenzugriff mit gestrecktem Zyklus (gestreckt auf 14 μs), A0-A17 wird ebenfalls über AC-Register bestimmt. Der Zyklus ist für das Setzen der Baseline-DACs notwendig
IFACL	0x404	R/W	Zugriff auf die unteren 10-Bit des AC-Registers (Address Counter) über D0-D9
IFACH	0x405	R/W	Zugriff auf das AC-Register Bit 10-17 über D0-D7
IFCSR	0x406	R	Zugriff auf das Status-Register D0, D5, D6, D11, D12, D13, D14 und D15 (siehe Tabelle A.5)
IFCSR	0x406	W	Zugriff auf das Status-Register D0 und D1 (siehe Tabelle A.5)
IFSIGN	0x407	W	Setzen und Löschen des NIM-Signals über D2 am Frontpanel
IFWINDOW	0x000-0x3FF	R/W	Freier Speicherzugriff auf DL300-Adressen. Die Adresse wird dabei durch ADR=ACH+IFWINDOW bestimmt

Tabelle 6.1 DL307-Register.

Sampling-Modus gestartet und die FADC-Module führen dauerhaft eine Digitalisierung des Eingangssignals aus. Hierbei wird der auf dem Modul verfügbare Speicher aufsteigend beschrieben. Die für den Modus definierte Erfassungsfrequenz (Sampling-Frequenz) legt fest, wie das Eingangssignal zeitlich verarbeitet wird. Bei einer Frequenz von 100 MHz wird das Signal alle 10 ns in der Amplitude erfasst und digitalisiert. Der zeitliche Abstand zwischen den Speicherstellen beträgt somit ebenfalls 10 ns. Wird das Ende des physikalischen Speichers erreicht, so wird mit der Aufzeichnung am Beginn des Speicherblocks des Moduls wieder begonnen, wobei hierbei die Werte überschrieben werden, die zeitlich um mehr als 10 ns multipliziert mit der Speichertiefe zurückliegen.

Bei einer Datenerfassung im Common-Stop-Modus wird der Scanner über ein Eingangssignal an der Frontseite des Moduls gestoppt, wenn ein gewünschtes Ereignis im Detektor festgestellt wurde. Die Stop-Position wird im Stop-Pointer gespeichert. Diese Position ist für die zeitliche Auswertung des digitalisierten Signals notwendig. Ist die Digitalisierung durch das externe Signal gestoppt worden, kann mit dem Hitscanner der Adressraum des DL300-Systems nach Einträgen durchsucht werden, die eine definierte Schwelle überschritten haben. Die Schwelle lässt sich hierbei über das Hit-Threshold-Register des DL302-Moduls festlegen. Dies ermög-

licht eine Reduktion der auszulesenden Daten, da Kanäle ohne relevante Information nicht berücksichtigt werden müssen.

Bei Datenerfassung im Common-Start-Modus wird der Scanner durch ein externes Signal gestartet. Die Flash-ADC-Module erfassen dann die Eingangssignale. Der Erfassungsvorgang wird gestoppt, sobald die zuvor eingestellte Stop-Adresse nach einer festgelegten Zeit, abhängig von Kanalzahl und Samplingrate, erreicht wurde.

Für die Steuerung des Moduls sind Register auf dem Scanner-Modul vorhanden, die über die interne Adresse (IA) des Scanners angesprochen werden können. Die Adresse der Register errechnet sich wie folgt:

$$\text{Slot} \cdot 2^9 + \text{IA} \tag{6.1}$$

Die internen Adressen 4-15 haben dabei die in Tabelle A.6 aufgeführten Funktionen. Das Funktionsregister 8 bestimmt insbesondere, mit welcher Frequenz das Modul betrieben wird. Desweiteren kann über dieses Register festgelegt werden, in welchem der beiden verfügbaren Modi (Sampling oder Fastscan) das Modul arbeiten soll und ob es im Common-Stop- oder Common-Start-Modus betrieben wird. Die Belegung des Funktionsregisters ist in Tabelle A.7 im Anhang beschrieben.

6.3 Anforderungen und Aufgaben für die Implementierung

Nachdem die Rahmenbedingungen und die verwendeten Hardware- und Softwarekomponenten für die Implementierung vorgestellt worden sind, sollen die Anforderungen an die Implementierung des DL300-CPU-Interfaces definiert werden. Dementsprechend sollen die Aufgaben für die Implementierung auf dem feldprogrammierbaren Logikbaustein und einer Software zur Steuerung des Systems durch den zukünftigen Nutzer festgelegt werden.

Für die Steuerung und Auslese des DL300-Systems müssen zunächst die in Kapitel 6.2.2 beschriebenen Register der DL307-VME-Interfacekarte gelesen und beschrieben werden können. Nur über die Interfacekarte lässt sich ein Zugriff auf das DL300-Crate und die System-Module wie das DL302-Modul realisieren. Hierzu wird eine Lese- und Schreiboperation zum Zugriff auf die Register der DL307-Interfacekarte benötigt. Die Verarbeitung der Signale von und zum DL300-System sollte dabei vollständig auf dem FPGA erfolgen, um eine schnelle Verarbeitung zu gewährleisten. Zum Setzen der Digital-Analog-Konverter ist hierbei ein spezieller Zugriff notwendig, der ebenfalls berücksichtigt und korrekt implementiert werden muss. Für die Initiierung einer Datenerfassung und Suche nach Ansprechern müssen mehrere Lese- und Schreiboperationen auf dem DL302-Scanner-Modul nacheinander durchgeführt werden. Die notwendigen Parameter, die an das DL300-Crate übergeben werden, sollten vom Benutzer auf geeignete Art und Weise bei der Initialisierung des Interfaces vorgegeben und auch nachträglich geändert werden können. Die Steuerung und die Speicherung der Positionen der gefundenen Ansprecher sollte auch hier auf dem FPGA erfolgen, um eine hohe Verarbeitungsgeschwindigkeit zu erreichen.

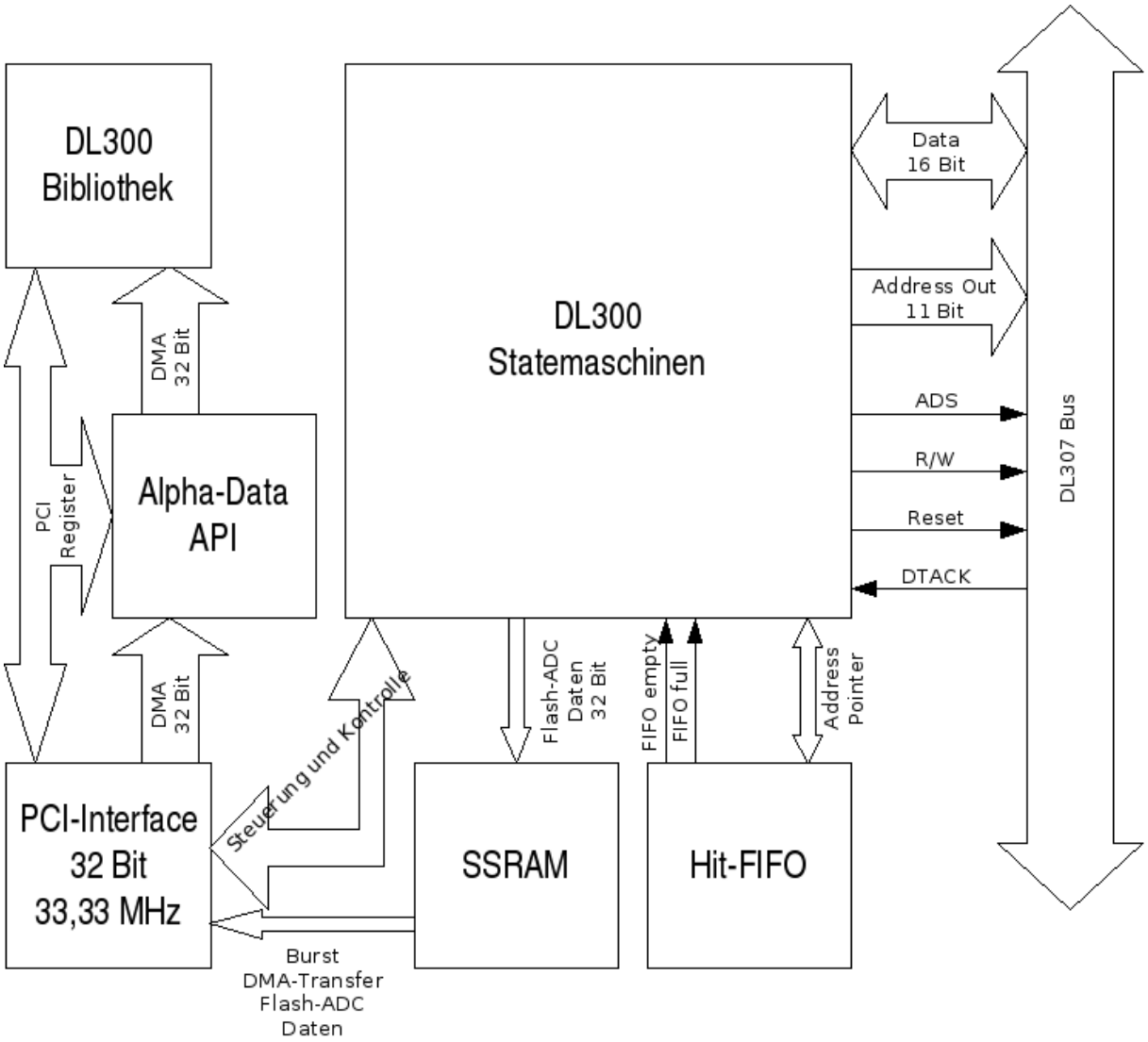


Abbildung 6.1 Schemazeichnung des DL300-CPU-Interfaces.

Die Auslese der gefundenen Ansprecher und der damit verknüpften Daten muss bei Verwendung der VMIC-CPUs und der Alpha-Data-Karte über die PCI-Schnittstelle erfolgen. Für einen schnellen Transfer großer Datenmengen steht auf der Karte ein PCI-Controller mit DMA-Kanälen[43] zur Verfügung, der für den Transfer der gesamten Daten eines Ereignisses genutzt werden kann. Idealerweise sollten die Daten im SSRAM-Speicher der Karte zwischengespeichert werden, um nur einen DMA-Transfer durchführen zu müssen. Für die Analyse der Daten parallel zur Auslese sollte es möglich sein, die Daten an VHDL-Module zu übergeben, die eine Auswertung durchführen. Da das DL300-System mit dem CPU-Interface in Zukunft an verschiedenen Detektorkomponenten zum Einsatz kommen wird, ist es wichtig, dass später neue Module leicht hinzugefügt werden können. Für den Benutzer sollte eine Schnittstelle zur Verfügung stehen, die die komplette Steuerung aller Aspekte des implementierten Interfaces ermöglicht, sowie eine schnelle und unkomplizierte Fehlerlokalisierung und Performanzanalyse bietet. Um hier kompatibel zu bereits vorhandener Software zu sein, sollte die Implementierung

[43] DMA - engl. Direct Memory Access - Direkter Speicherzugriff

der Schnittstelle in Form einer C++-Klasse als externe Bibliothek erfolgen, so wie dies auch für die am Experiment eingesetzte VME-, CAMAC- und Fastbus-Hardware erfolgt ist. In dem nachfolgenden Kapiteln wird die Umsetzung dieser definierten Aufgaben beschrieben. Kapitel 6.4 gibt einen Überblick über die für diese Arbeit erstellen VHDL-Dateien, deren Funktion dann im Weiteren im Detail erläutert wird. Die Software-Bibliothek zur Steuerung wird am Ende dieses Kapitels beschrieben. In Abbildung 6.1 ist die Struktur mit den wichtigsten Steuerungs- und Datenleitungen zu sehen.

6.4 VHDL-Dateien

Zunächst soll ein Gesamtüberblick über die erstellten VHDL-Dateien gegeben werden. In den folgenden Unterkapiteln werden die Dateien und deren Inhalt näher erläutert. Der Hauptanteil des erstellten VHDL-Codes befindet sich in der Datei dl300_interface.vhd. Hierin sind alle am Chip herausgeführten Signale definiert, die über das dazugehörige Constraints-File (dl300_interface.ucf) den Bausteinanschlüssen zugeordnet werden. Neben den in Kapitel 6.1.1 erwähnten Anschlüssen am vorderen Stecker, die zum DL300-System führen, sind in der Entity-Deklaration die Signale für den PCI-Bus, die Taktgeber und das ZBT-SSRAM[44] definiert. Diese werden in den nachfolgenden Kapiteln beschrieben. Die Hauptquelldatei bindet zwei Definitionen für Pakete ein. Hierbei handelt es sich zum einen um Komponenten aus dem SDK der Alpha-Data-Karte (sdkcomp.vhd), zum anderen um das Paket (Package) dl300sm.vhd, das alle für das DL300-System entwickelten Statemaschinen-Definitionen (der Begriff der Statemaschinen wird in Kapitel 6.7 erläutert) enthält. Das Paket sdkcomp enthält die Statemaschinen für den Zugriff über den PCI-Bus. Dazu gehören eine Demand-Mode-DMA-Statemaschine (component plxddsm), eine Direct-Master-Statemaschine (component plxdmsm) und eine Direct-Slave-Statemaschine (component plxdssm). Desweiteren stellt es ein synchrones 32-Bit-FIFO[45] mit 512 Worten Speicher zur Verfügung, welches Block-Rams auf dem Virtex-FPGA verwendet. Von den definierten Komponenten werden in dieser Arbeit nur die Direct-Slave-Statemaschine und das FIFO verwendet.

Die Statemaschinen für das DL300-System werden in Kapitel 6.7 beschrieben. In Tabelle A.9 im Anhang werden die Quellcodedateien angegeben, die in der Datei dl300sm.vhd als Paket definiert sind.

Jede Statemaschine wurde hierbei in ein separates File ausgelagert. Dies ermöglicht die spätere Verwendung der Statemaschinen in anderen Projekten, falls die DL300-Steuerung beispielsweise über ein eigenständige Kontrolkarte erfolgen soll, welches die Daten von mehreren DL300-Systemen auf einer Alpha-Data-Karte zusammenführt.

Neben den Package-Definitionen kommen noch weitere Quellcode-Dateien zum Einsatz. Diese sind im Anhang in Tabelle A.10 aufgeführt.

[44] ZBT-SSRAM - engl. Zero-Bus Turnaround Synchronous Static Random Access Memory - synchroner statischer Speicher mit wahlfreiem Zugriff (Anordnung des Speichers in einer Matrixform) ohne Latenzverlust auf dem Bus zwischen Lese- und Schreiboperationen (ZBT), zur Nutzung der maximalen Speicher-Bandbreite.

[45] first in - first out Prinzip

Die angebenen Statemaschinen ermöglichen die Auslese und Steuerung des DL300-Systems. Für die Analyse der gefundenen Treffer (Hits) ist ein modulares System vorgesehen, das verschiedene Funktionen auf die auszulesenden Daten anwenden kann. Die Daten werden dabei an eine externe Statemaschine weitergeleitet, und mit jedem neu eingetroffenen Datenwort wird ein Taktzyklus für die externe Statemaschine generiert. Tabelle A.11 im Anhang enthält einen Überblick über die implementierten Funktionen. Die Analysefunktionen sind dabei in der Datei dl_analyze zu einem Paket (Package) zusammengefügt.

6.5 PCI-Anbindung

Die Anbindung an den PCI-Bus der VME-CPU ist die Schnittstelle, über die alle Kommandos für das DL300-System übertragen werden. Mit Registern auf dem PCI-Bus werden dabei alle Parameter für das Flash-ADC-System festgelegt, und es können Statusinformationen des DL300-Systems sowie der Interfacekarte abgerufen werden. Für die Übertragung der Daten aus dem RAM-Speicher der FPGA-Karte an den Speicher auf dem VME-Board wird ein direkter Speichertransfer (DMA) verwendet.

6.5.1 Direct-Slave-PCI-Statemaschine und DMA-Memory-Transfer

Der DMA-Transfer (engl. Direct Memory Access) ermöglicht das direkte Beschreiben des Arbeitsspeichers der VME-CPU, ohne dass hierbei jedes übertragene Datenwort von der CPU bearbeitet werden muss. Das Peripheriegerät (in diesem Fall das Interfaceboard) kommuniziert direkt mit dem Arbeitsspeicher. Die CPU hat hierbei nur Verwaltungsaufgaben zu übernehmen und wird somit stark entlastet. Der hierfür notwendige DMA-Controller ist auf dem PCI-Bus Master-Interface-Chip der Firma PLX vom Typ PCI-9080 implementiert [PLX00]. Dieser bietet zwei unabhängige DMA-Kanäle, die für einen Transfer vom lokalen an den PCI-Bus oder umgekehrt genutzt werden können. In diesem Chip sind weiterhin 8 FIFO-Puffer[46] vorhanden, die die ankommenden Daten zwischenspeichern können. Der DMA-Transfer wird über die Direct-Slave-Statemaschine gesteuert. Alternativ hierzu könnte eine Demand-Mode-Statemaschine eingesetzt werden. Dieses hätte den Vorteil, dass bei sehr stark variierenden Datenmengen pro DMA-Zyklus die Entscheidung über die zu transferierende Datenmenge dem FPGA obliegt. Da dies aber nicht anzunehmen ist[47], wurde hier zunächst eine Direct-Slave-Maschine verwendet, die es erlaubt, einen DMA-Transfer, sowie auch die Register für die Steuerung der Interfacekarte zu implementieren. Hiermit steht mehr Chipfläche auf dem FPGA zur Verfügung, so dass spätere Erweiterungen zur Datenanalyse auf dem FPGA noch Platz finden. Der DMA-Transfer ist bis auf die verwendete Statemaschine aus dem DL300-API vollständig im Quellcode der Hauptdatei dl300_interface.vhd implementiert. Die Statemaschine für den PCI-Transfer mit der Bezeichnung plxdssm ist wie in Quellcode A.1 im Anhang gezeigt eingebunden.

Neben den Signalen Zurücksetzen **rst** und dem Taktgeber **clk** sind auf der rechten Seite der

[46] first in - first out Prinzip

[47] die Datenmenge pro Event wird sich nicht um mehrere Größenordnungen je Event ändern

Anschluss-Zuweisung Signalleitungen definiert, die den PCI-Signalen entsprechen[48], sowie Signale, die nur innerhalb des FPGA Verwendung finden[49]. Desweiteren gibt es solche, die anzeigen, ob das dazugehörige Signal ausgegeben werden soll (Output-Enable). Diese sind durch die Endung _oe gekennzeichnet. Die Endungen _o bzw. _i geben an, ob es sich um ein eingehendes (Input = _i) oder ein ausgehendes (Output = _o) Signal handelt. Die Leitung für Bereit **ds_ready** und Stop **ds_stop** müssen im FPGA-Code generiert und der Statemaschine zur Verfügung gestellt werden, wie in Quellcode 6.1 auszugsweise gezeigt wird. Diese generiert dann die notwendigen Signale auf dem PCI-Bus. **ds_ready** signalisiert, dass ein zu lesendes oder zu schreibendes Datenwort ansteht. Hierbei muss zwischen einem Lese- oder Schreibzugriff auf die PCI-Register (siehe Kapitel 6.5.2) und einem Zugriff auf das SSRAM unterschieden werden. Dies erfolgt anhand des Bits 21 der Adresse auf dem PCI-Bus (siehe Zeilen 8, 13 und 14 in Quellcode 6.1). Die **ds_ready** Leitung setzt sich hierbei aus einem logischen Oder von 4 anderen Leitungen zusammen (Zeilen 15-16). Diese vier werden bei einem Lese- oder Schreibzugriff auf das SSRAM oder die PCI-Register in den 4 möglichen Kombinationen generiert. Das Lesesignal wird dabei über ein Register implementiert (Zeilen 7-10), da das Signal entweder anhand der Adressdekodierungs-Leitung **ds_decode** in Zeile 7, die eine Adressdekodierung anzeigt, erzeugt werden kann oder über das Early-Valid **zbt_evalid** Signal, welches das Anliegen eines Datenworts am Ausgang des Speichers im Nachfolgenden Taktzyklus anzeigt (Zeile 10). Die Signale für das Schreiben des RAM-Speichers werden direkt über die Schreib-Leitung **ds_write** erzeugt (Zeile 14).

Das Stop-Signal **ds_stop** dient dazu, einen sogenannten Burst-Zugriff zu beenden (Zeilen 18-23). Bei einem Burst werden mehrere Datenworte aufeinanderfolgender Speicherzellen mit nur einer Adressierungsphase nacheinander übertragen. Eine Burst-Übertragung muss abgebrochen werden, wenn ein Zugriff auf ein PCI-Register (siehe Kapitel 6.5.2) erfolgt, da hier nur maximal ein 32-Bit-Datenwort übertragen wird. Außerdem muss ein Burst abgebrochen werden, wenn der RAM-Speicher angesprochen und die Grenze einer Speicherbank erreicht wird, da hier ohne neue Adressierung der Speicherbank kein weiterer sinnvoller Datentransfer durchgeführt werden kann. Die Statemaschine plxdssm des ADM-XRC API generiert hierzu das Signal local burst termination **lbterm_o**. Um nun auch das kontinuierliche Auslesen des SSRAM-Speichers zu ermöglichen, muss bei einem Lese- oder Schreibzugriff auf den Speicher noch die SSRAM-Adresse erhöht werden. Dies erfolgt, wenn der PCI-Bus Kontrolle über die RAM-Zugriffe hat[50] oder wenn eine Lese- oder Schreiboperation auf dem RAM-Speicher erfolgt. Die entsprechende Erhöhung der Adresse erfolgt dann im Prozess **gen_page_addr**.

6.5.2 PCI-Register

Für die Steuerung des DL300-Systems sind auf dem FPGA Register definiert worden. Die notwendigen Signalleitungen des lokalen Busses des PCI-9080 Chips der Firma PLX sind in

[48] **qlads**, **lblast_i**, **lwrite_i**, **lready_o**, **lbterm_o**
[49] **ds_xfer**, **ds_decode**, **ds_write**, **ds_ready** und **ds_stop**
[50] **fpga_ssram_access** ist logisch 0.

```
1       gen_rd_ready: process(rst, clk)
2       begin
3               if rst = '1' then
4                       regs_rd_ready <= '0';
5                       sram_rd_ready <= '0';
6               elsif clk'event and clk = '1' then
7                       regs_rd_ready <= ds_decode and not ds_write
8                               and not la_q(21);
9                       sram_rd_ready <= read_enable and
10                      OR_reduce(zbt_evalid and banksel);
11              end if;
12      end process;
13      regs_wr_ready <= ds_decode and ds_write and not la_q(21);
14      sram_wr_ready <= ds_decode and ds_write and la_q(21);
15      ds_ready <= sram_rd_ready or sram_wr_ready or regs_rd_ready
16              or regs_wr_ready;
17
18      ds_stop <= '1' when ds_decode = '1' and la_q(21) = '0' else
19              '1' when ds_decode = '1' and la_q(21) = '1' and
20              la_q(bterm_order + 1 downto 2) = bterm_bound_vec else
21              '1' when ds_xfer = '1' and la_q(21) = '1' and
22              la_q(bterm_order + 1 downto 2) = (bterm_bound_vec - 1) else
23      '0';
24
25      inc_page_addr <= (zbt_read or zbt_write) when
26              fpga_ssram_access = '0';
```

Quellcode 6.1 Generierung der Signale für die PCI-Statemaschine und das SSRAM (Auszug).

Kapitel 6.5.1 beschrieben, da diese auch für die DMA-Übertragung verwendet werden. Diese Beschreibung soll hier nicht wiederholt werden. Die Definition der Register erfolgt in drei Prozessen in der Hauptdatei dl300_interface.vhd. Hierbei dienen zwei Prozesse der Reaktion auf einen Register-Lesezugriff, während ein Prozess auf Schreibzugriffe reagiert. In den Tabellen im Anhang und im Folgenden werden die Zugriffsrichtungen lesen und schreiben durch die üblichen Buchstaben R[51] und W[52] abgekürzt. In Tabelle A.12 im Anhang sind die erstellten Prozesse zum Lesen und Schreiben der PCI-Register aufgeführt.

Die Prozesse **gen_oe_regs** und **gen_oe_regs_debug** erfüllen dieselbe Aufgabe. Lediglich zu Entwicklungszwecken sind die Register zur Fehlersuche in den Prozess **gen_oe_regs_debug** ausgegliedert worden. In den Prozessen (siehe Quellcode 6.2) werden die Ausgangsaktivierungssignale (engl. Output-Enable oder kurz OE) auf den Wert 0 (Zeile 4) initialisiert, sowie gesetzt, wenn ein Lesezugriff auf das Register stattfindet (Zeile 7). Die Adressierung des Registers wird über die Signalleitungen **la_q** festgestellt. Hierzu wird die anliegende Adresse mit der jeweiligen Registeradresse verglichen. Das Output-Enable-Signal aller Register wird dann

[51] engl. read - lesen
[52] engl. write - schreiben

zurückgesetzt, wenn ein Transfer durchgeführt wird (Zeile 14).

```
1       gen_oe_regs : process(rst,clk)
2       begin
3               if rst = '1' then
4                       oe_firmware_rev <= '0';
5               elsif clk'event and clk = '1' then
6               if ds_decode = '1' then
7                       if la_q(22 downto 21) = "00" and ds_write = '0' and
8                               la_q(13 downto 2) = X"000" then
9                               oe_firmware_rev <= '1';
10              end if;
12              else
13              if ds_xfer = '1' and (lblast_i= '1' or lbterm_o = '1') then
14                      oe_firmware_rev <= '0';
15              end if;
16              end if;
17      end process;
```

Quellcode 6.2 Aktivieren einer Output-Enable-Leitung am Beispiel des Firmwareregisters.

Die Ausgabe eines Registers erfolgt, wenn die zugehörige Output-Enable-Leitung (oe_signalname) gesetzt ist. Das auszugebende Signal wird in diesem Fall an **id** (Internal Databus) übergeben, ansonsten geht die Leitung in einen hochohmigen Zustand, einem sogenannten Tristate[53]. Dadurch können mehrere VHDL-Signale einzeln auf den Bus durchgeschaltet werden, wobei die derzeitige Signalquelle vom internen Datenbus isoliert wird. Der Datenbus kann somit von mehreren unterschiedlichen Signalquellen getrieben werden.

```
1       id <= X"0000000" & dl300\_samplemode when oe\_samplemode = '1'
2               else (others=>'Z');
```

Quellcode 6.3 Ausgabe eines Signals an den internen Datenbus.

Da die auszugebenden Daten unter Umständen nicht die volle PCI-Busbreite von 32-Bit nutzen, wird an den id-Bus immer ein voller 32-Bit-Datenwert übergeben. Die nicht genutzten Bits werden dabei mit dem binären Wert 0 gefüllt. In Quellcode 6.3 wird der 4-Bit breite Wert dl300_samplemode mit 28 führenden binären Nullen (entsprechend dem hexadezimalen Wert 0x0000000) gefüllt.

Der interne Datenbus **id** ist lokal im FPGA definiert und wird nicht nach außen gegeben. Die Ausgabe der Signale am internen Datenbus erfolgt synchron mit der PCI-Taktfrequenz. Hierbei wird das Signal der lokalen Datenleitung **ld_o** übergeben, welche zur Ausgabe nach extern an das Signal **ld** übergeben wird, sofern die hierfür aus der PCI-Bus Statemaschine kommende **oe_ld** Leitung gesetzt ist. Das Signal lokale Daten **ld** ist dann aus dem FPGA auf den PLX PCI-Chip hinausgeführt. Die Implementierung ist in Quellcode 6.4 gezeigt.

[53] In VHDL ist dies über den Standardtypen std_logic möglich, der hierbei den Wert Z erhält.

```
1       ld <= ld_o when oe_ld = '1' else (others => 'Z');
2
3       gen_ld_o: process(rst, clk)
4       begin
5               if rst = '1' then
6                       ld_o <= (others => '0');
7               elsif clk'event and clk = '1' then
8                       ld_o <= id;
9               end if;
10      end process;
```

Quellcode 6.4 Synchrone Ausgabe des internen Datenbus an den PCI-Bus.

Die Implementierung eines Schreibzugriffes auf den PCI-Bus im Prozess **registers** erfolgt analog. Hier werden die Signale, die die Inhalte der Register enthalten, anstelle der Output-Enable-Leitungen auf den am PCI-Bus anliegenden Wert gesetzt. Aufgrund einer Besonderheit des PCI-Busses gibt es eine Kommando/Byte-Enable-Leitung. Diese signalisiert bei einem Datenzugriff, welche der 4 Bytes der 32-Bit-Datenleitung Daten enthalten. Diese ist auf dem lokalen Bus als local byte enable **lbe** vorhanden und kann über die interne Variable **lbe_i**[54] geprüft werden. Ein Schreibzugriff ist nur auf eine Teilmenge der PCI-Register möglich. Die Initialisierung der Register erfolgt mit sinnvollen Standardwerten, die, so wie die gesamte Belegung, den Tabellen A.13, A.14 und A.15 im Anhang zu entnehmen sind.

Die Steuerung des Sampling- und des Fastscanvorgangs des DL300-Systems erfolgt über die beiden Register 0x26 und 0x27. Über einen Lesezugriff auf Register 0x26 kann ein Samplingvorgang initiiert werden, über einen Lesevorgang auf Register 0x27 kann ein Fastscan mit anschließender Hitsuche und Hittransfer gestartet werden. Alle übrigen Register dienen der Festlegung der Parameter für die Auslese des DL300-Systems, sowie für Fehlersuche und Diagnosezwecke (siehe Kapitel 6.9).

6.6 SSRAM-Anbindung

Das ZBT-SSRAM[55] auf der FPGA-Karte ist in Bänken angeordnet, die je nach Typ der verwendeten Alpha-Data-Karte unterschiedliche Größen haben können. Das Design ist so implementiert, dass alle derzeit von Alpha-Data verfügbaren FPGA-Karten mit dem vorliegenden Design mit wenigen Änderungen betrieben werden können. Für Anpassungen des Interfaces auf andere Karten sei auf die Dokumentation des Alpha-Data-SDK und auf das darin enthaltene Beispiel des ZBT-SSRAM verwiesen [Alp03]. Bei der vorliegenden ADM-XRC Karte sind vier Bänke 36-Bit SSRAM zu je 256k Größe (Cypress CY7C1355B 256kx36 RAMs) vorhanden. Die 4 Paritätsbits werden im Design nicht verwendet. Im Weiteren wird nur die RAM-Bank 0 verwendet,

[54] lbe_i ist die invertierte lbe_l Leitung

[55] ZBT-SSRAM - engl. Zero-bus turnaround Synchronous Static Random Access Memory - synchroner statischer Speicher mit wahlfreiem Zugriff (Anordnung des Speichers in einer Matrixform) ohne Latenzverlust auf dem Bus zwischen Lese- und Schreiboperationen (ZBT), zur Nutzung der maximalen Speicher-Bandbreite.

da der Speicher eines einzelnen RAM-Chip (1MByte) mehr als ausreichend für die vorliegende Anwendung ist. Der Zugriff auf die weiteren Bänke via PCI-Bus ist allerdings implementiert, so dass diese prinzipiell zur Verfügung stehen[56]. Die Implementierung des Zugriffes auf den Speicher erfolgt in Anlehnung an das Beispieldesign ZBT, des SDK der Alpha-Data Karte, sowie der Applikation memtest.c. Die Speicherbänke auf der FPGA-Karte können in ein 2 MByte großes Adressfenster auf dem lokalen Adressbus eingeblendet werden. Dies ist schematisch in Abbildung 6.2 gezeigt.

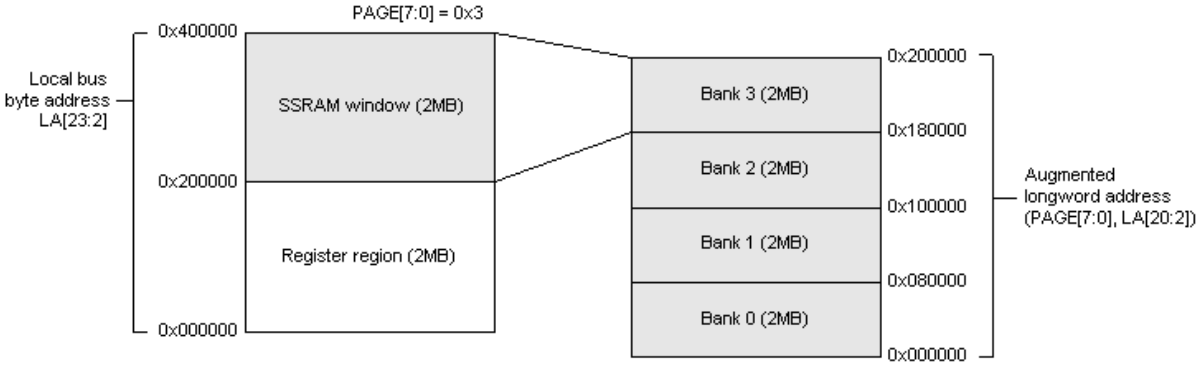


Abbildung 6.2 Einblendung eines 2 MByte großen Adressfensters auf dem lokalen Adressbus.

Ein Zugriff auf das SSRAM ist so implementiert, dass der RAM-Inhalt über einen DMA-Transfer auf die CPU übertragen werden kann. Außerdem ist über einen Multiplexer ein Zugriff des FPGAs auf den SSRAM-Speicher möglich. Dies wird von den Statemaschinen **dl_xfer_hits** und **dl_fast** genutzt, um Daten im RAM abzulegen. Die direkte Steuerung und Kontrolle der SSRAM-Bausteine wird durch die Module des SDK[57] übernommen. Hierbei werden für die vier Bänke Instanzen der Module **zbt_ports** erzeugt. Für alle auf der ADM-XRC FPGA-Karte nicht vorhandenen Bänke werden Dummy-Module **zbt_dummy** erzeugt. Die Anzahl der verwendeten Bänke ist ein Parameter, der je nach verwendeter Karte gesetzt werden kann. Allerdings wurden im Interface Teile der nicht verwendeten Signale auskommentiert, um den Übersetzungsvorgang zu verkürzen. Desweiteren wird ein **zbt_dpins** Modul instanziert, welches eine Strom von 16 mA an die Speicher-Bausteine liefert, so wie dies durch die Referenzimplementierung von Xilinx für ZBT-Speicher vorgegeben ist. Die SSRAM-Adresse **ssram_addr**, die Datenleitung local data **ld** und die Byte-Enable Leitung **be** wurden durch die Signalnamen **ssram_mux_addr**, **ld_mux** und **lbe_mux ersetzt**. Hierdurch können Signale von mehreren Quellen abhängig von der Signalleitung **fpga_ssram_access** dem Speicher zugeführt werden. Dadurch kann zum einen der lokale Bus als auch die internen Datenleitungen des FPGA **ld_fpga** und **lbe_fpga** Zugriff auf das SSRAM erhalten.

Das Signal **fpga_ssram_access** bzw. **fpga_local_toggle** regelt den Zugriff durch den FPGA auf das SSRAM. Im Prozess **gen_local_toggle** wird das Signal **fpga_local_toggle** auf 1 ge-

[56] Für den Zugriff vom FPGA muss gegebenfalls noch die Adressierung der verschiedenen RAM-Bausteine korrekt im VHDL-Code (über die verwendete fpga_ssram_addr) implementiert werden.

[57] SDK - Software-Development-Kit

setzt, wenn das Signal **dl300_hit_search_busy** oder das Signal **dl300_hit_xfer_busy** auf logisch 1 gesetzt sind und signalisieren, dass eine der beiden Statemaschinen, die für die Auslese des DL300-Systems zuständig sind, aktiv ist. Das Signal sorgt zum einen dafür, dass ohne einen PCI-Zugriff ein Beschreiben des Speicherinhaltes möglich ist, zum anderen werden die entsprechenden Multiplexer anhand dieses Signals umgesetzt.

Für die Adressierung der Speicherbänke und Speicherzellen sind die Prozesse **gen_page_addr**, **gen_banksel_addr** und **gen_banksel** zuständig. Der Prozess **gen_page_addr** setzt bei einem Beginn eines Schreib- oder Lesezugriffes über den PCI-Bus auf das SSRAM die Variable **page_addr** auf den adressierten Wert und erhöht diese sukzessiv bei jedem weiteren Lese- oder Schreibzugriff. Dadurch ist ein Burst-Transfer realisiert (siehe Kapitel 6.5.1). Der Prozess **gen_banksel_addr** generiert, basierend auf dem Wert in der Variablen **size_reg**, die Variable **banksel_addr**, die jeweils die höchsten beiden gültigen Bits der adressierten Speicherstelle enthält. Die Informationen zu dem auf der Karte verbauten Speicher können aus einem ROM auf der Karte von der Software ausgelesen werden und müssen über die Software in ein PCI-Register geschrieben werden, damit dem FPGA diese Information auch zur Verfügung steht. Implementiert sind entsprechend den möglichen Speicheranordnungen auf Alpha-Data Karten 128 k, 256 k, 512 k und 1 M RAM-Bausteine. Die Variable **banksel_addr** enthält jeweils die drei Bits, die außerhalb des Adressraumes für einen RAM-Baustein liegen[58]. Ein 128 k Baustein (mit 128 k Adressen) lässt sich mit 17 Bits vollständig adressieren (Bits 0-16). Die Variable **banksel_addr** erhält in diesem Fall den Wert der Adressierungsbits 17 bis 19. Dies gilt dann analog für die anderen Speichergrößen. Die erzeugte Variable **banksel_addr** wird dann im Prozess **gen_banksel** ausgewertet und es wird in dem Vektor **banksel** das Bit der angesprochenen Speicherbank auf logisch 1, sowie alle anderen auf logisch 0 gesetzt.

Als Adresse für das SSRAM wird dem **zbt_port** Modul die Adresse entsprechend der Adressierungsgröße der verbauten SSRAMs übergeben. Die ausgegebenen Daten der Speichermodule werden an das Signal **zbt_q(i)** übergeben, welches an den internen Datenbus ausgegeben wird, sofern das jeweilige Modul adressiert und gelesen wurde. Die eingehende Datenleitung **d** und die Byte-Enable Leitung **be** werden mit den „gemultiplexten“ Signalen **ld_mux** und **lbe_mux** verbunden. Schaut man sich die Verknüpfungen an, so ist in dieser Implementation ein Lese- und Schreibzugriff über den PCI-Bus möglich. Eine Lesezugriff auf das SSRAM durch den FPGA ist hingegen nicht möglich. Alle während der Auslese benötigten Daten werden auf dem FPGA zwischengespeichert. Dieser verfügt über kleinere Block-RAMs, die sich schneller als das SSRAM auslesen lassen. Ein direkter Lesezugriff des FPGA auf das SSRAM wird somit nicht benötigt.

[58] Die verwendete Adresse setzt sich aus der Adresse auf dem PCI-Bus (la_i), sowie einem Seitenregister zusammen, welches via Software gesetzt werden muss. Dieses selektiert dann die entsprechende Speicherseite und blendet sie in ein 2 MByte grosses Fenster ein. Die Details sind dem Quellcode zu entnehmen.

6.7 Statemaschinen

Für die Steuerung des DL300-Systems werden Statemaschinen verwendet. Statemaschinen bestehen aus Zuständen (States) und Übergängen (Transitions). An einen Übergang können Bedingungen geknüpft werden, so dass der Wechsel zwischen den Zuständen erst erfolgt, wenn die Bedingung erfüllt ist. Die Übergänge zwischen den Zuständen werden durch einen externen Frequenzgeber getaktet. Sie sind also synchron zur angelegten Taktfrequenz. Somit können Ablaufdiagramme erzeugt werden, was für die sequentielle Bearbeitungen von Befehlen und Anweisungen sinnvoll ist. Der Vorteil von Statemaschinen ist, dass diese visuell über graphische Eingabeprogramme erstellt werden können. Ein Ablauf einer Steuerung wird so für den Betrachter „sichtbar“. Die Xilinx ISE Software umfasst auch ein solches Statediagramm-Programm. Die StateCad-Software übersetzt den graphisch erstellten Ablauf in einen HDL-Code. Bei der Übersetzung werden Konsistenzprüfungen für die Übergangsbedingungen durchgeführt. Kommt es hierbei im Statediagramm zu Übergängen, die niemals erfüllt werden können (sog. Dead-Locks), so bricht der Übersetzungsvorgang ab. Hat ein Zustand mehrere Übergangsmöglichkeiten zu anderen Zuständen, so wird beim Übersetzungsvorgang geprüft, ob es zwischen den Übergangsbedingungen zu Konflikten bei den Bedingungen kommt. Wenn zum Beispiel ein Zustand über zwei gleiche Bedingungen einen anderen Zustand erreichen kann, so ist dies kein konsistentes Statediagramm. Die StateCad-Software verhindert, dass sich ungewollte Fehler in die Statediagramme einschleichen können. Die Statediagramme für die folgenden Statemaschinen wurden zunächst über das graphische StateCad-Programm von Xilinx eingegeben. Hierbei zeigte sich jedoch, dass dieses Verfahren bei komplexen Statediagrammen problematisch wird. Zum einen wurde die Komplexität der Diagramme so groß, dass diese sehr viel Platz einnahmen. Es war hier durch die Software nicht möglich, sich wiederholende Teile abzutrennen. Hierzu hätten separate Statemaschinen erstellt werden müssen, die dann mehrfach eingebunden worden wären. Zum anderen gab es, nachdem fast alle Statediagramme erstellt waren, Probleme mit dem StateCad-Compiler. Sobald ein Statediagramm eine gewisse Komplexität angenommen hatte, begann der Compiler die Übersetzung zwar, stoppte jedoch mitten im Übersetzungsvorgang und erzeugte keine Ausgabedatei mehr. Der Übersetzungsvorgang musste dann ohne Ergebnis abgebrochen werden. Dieses Phänomen war reproduzierbar, wenn eines der Statediagramme eine gewisse Komplexität erreicht hatte. Wurde ein Zustand wieder entfernt, so lief der Übersetzungsvorgang weiter durch. Als problematisch erwies sich weiterhin, dass die StateCad-Software von der Firma Xilinx aufgekauft worden war und seitdem nicht mehr weiterentwickelt wurde. Die Webseite[59] zeigte während der Entwicklung des Interfaces auf eine nicht existente Seite der Firma Xilinx. Die Software verwendet zudem noch alte Microsoft DOS-Dateinamen in der 8.3 Notation (8 Zeichen Dateiname, 3 Zeichen Endung). Zur Lösung dieses Problems gab es mehrere Alternativen:

- Supportanfrage bei Xilinx
- Trennung der Statemaschinen in mehrere Teilmaschinen

[59] http://www.statecad.com/

- Verwendung eines anderen Statediagramm-Editors
- Verzicht auf ein graphisches Eingabetool

Aufgrund der Tatsache, dass es sich um eine alte Software handelt, die auch nicht die gewünschten Möglichkeiten zur Trennung von Unterteilen von Statemaschinen ermöglicht, wurde recherchiert, ob es eine Alternative zum verwendeten StateCad-Programm gab. Ein entsprechendes Programmpaket von der Firma HDL Works erfüllte diese Bedingungen. Allerdings war diese Software sehr leistungsstark und hatte dementsprechend einen hohen Preis. Die Anschaffung wäre nur bei mehreren FPGA-Projekten in der Arbeitsgruppe sinnvoll gewesen. Eine Supportanfrage bei Xilinx hätte im Hinblick auf die veraltete Software vermutlich in kürzerer Zeit nicht zum gewünschten Ziel geführt. Die Trennung der Statemaschinen in mehrere Teilmaschinen, die sich gegenseitig aufrufen, ist nicht sehr elegant. Daraufhin wurde versucht, die Statemaschinen ohne ein graphisches Eingabetool zu erstellen. Hierbei muss jedoch sehr genau darauf geachtet werden, dass alle Werte korrekt initialisiert werden, und dass etwaige Übergangsbedingungen nicht zu Dead-Locks führen. Die Implementierung von Statemaschinen gestaltete sich einfach und insbesondere ließen sich Wiederholungen von Codesegmenten sehr elegant vermeiden. Der erstellte VHDL-Code war zudem kompakter. Auch war nach der Synthetisierung des Codes die genutzte Chipfläche geringer. Daraufhin wurden alle Statemaschinen auf eigenen VHDL-Code umgestellt. Bei der Implementierung wurde sehr genau darauf geachtet, Dead-Locks in den Statemaschinen zu vermeiden.

Für die Ansteuerung des DL300-Systems über den vorderseitigen Stecker wurde eine Busmaschine **dl_cycle** entwickelt, die einen Lese- oder Schreibzyklus auf dem DL300 initiieren kann. Darauf bauen die beiden Statemaschinen fürs Lesen **dl_read** und Schreiben **dl_write** auf, die einen vollständigen Lese- und Schreibzyklus ausführen können. Die übergeordneten Statemaschinen **dl_setdac**, **dl_samp**, **dl_fast**, **dl_hit_search** und **dl_xfer_hits** verwenden dann die Funktionalität der Statemaschinen **dl_read** und **dl_write**. Die Statemaschine **dl_fast** startet die Statemaschine **dl_hit_scan** und diese wiederum die Statemaschine **dl_xfer_hits** . Dadurch ergibt sich eine hierarchische Struktur, wie in Abbildung 6.3 zu sehen ist.

Eine mit der Frequenz **clk** getaktete Statemaschine wird im VHDL-Code über die in Quellcode 6.5 zu sehende Syntax definiert. Hierbei wird zunächst in Zeile 1 ein neuer Block definiert. Darauf folgend wird für jeden Zustand der Statemaschine im neu erstellten Typ ein Wert definiert (Zeile 2). Die Variable **c_state** kann dann jeden dieser definierten Werte annehmen (Zeile 3). Über den in Zeile 5 definierten Prozess **state_decoder** wird die Bindung an den externen Taktgeber **clk** erreicht, sowie die Möglichkeit des Zurücksetzens der Statemaschine über das asynchrone Signal **rst** geschaffen. Nach dem Einschalten und Laden des FPGAs wird die Maschine somit zunächst in den Ruhe-Zustand **idle** versetzt und wartet in diesem Zustand, bis das Start-Signal **start_me** den Wert 1 erhält. Daraufhin wird die Zustandsvariable **c_state** auf den Wert **st_start** gesetzt, und mit dem nächsten Taktsignal erreicht die Statemaschine dann den Zustand **st_start**. Hier wird das Signal **a** auf logisch 1 gesetzt und in den nächsten Zustand **st_stop** gewechselt, in welchem das Signal **b** auf logisch 1 gesetzt wird. Danach kehrt die Statemaschine zum Ausgangszustand **st_idle** zurück und löscht die beiden Signale.

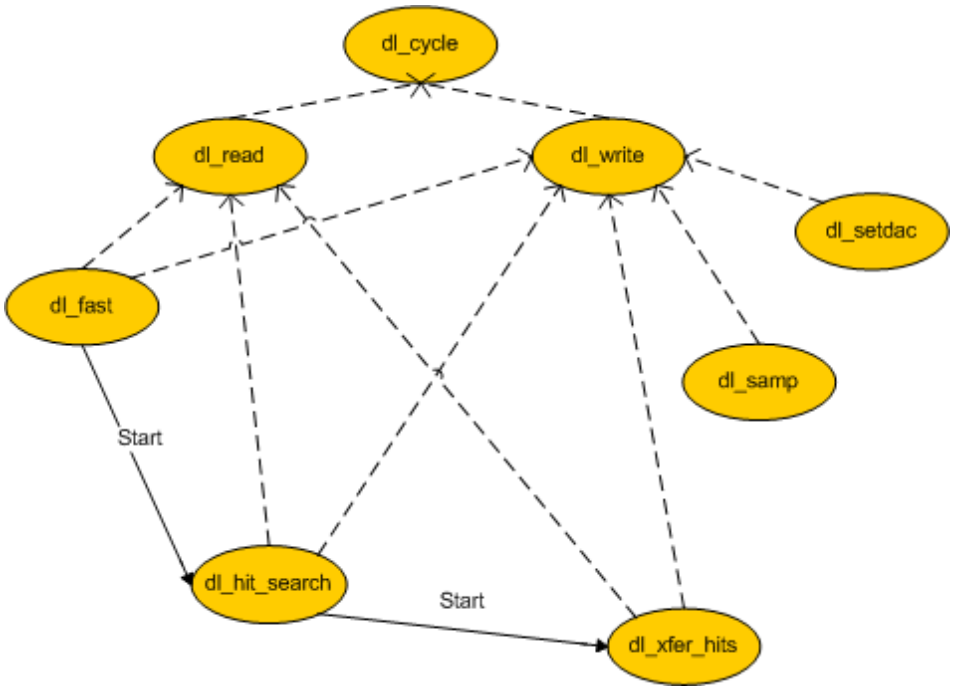


Abbildung 6.3 Diagramm der Statemaschinenstruktur.

Diese Grundstruktur wird im Weiteren in allen Statemaschinen verwendet. In Abbildung 6.4 ist dieselbe Statemaschine als Statediagramm gezeigt. Das Statediagramm wurde hierbei mit der Software Statecad der Firma Xilinx erzeugt. Auf die graphische Erstellung der Statemaschinen wurde aufgrund der bereits erwähnten Softwareprobleme verzichtet.

Die Statemaschinen für den Zugriff auf das DL300-System werden mit der doppelten PCI-Bus-Taktfrequenz betrieben. Hierdurch bleiben die Statemaschinen synchron zu den FPGA-Bereichen, in denen die PCI-Bus-Clock von 33,33 MHz verwendet wird. Durch die doppelte Frequenz von 66,66 MHz verkürzen sich jedoch die Zeiten, in denen übergeordnete Statemaschinen auf Signale der logisch untergeordneten Statemaschinen warten müssen. Das hierfür benötigte verdoppelte Taktsignal **clk2x** wird durch das Modul clocks generiert.

6.7.1 dl_cycle

Die im dl_cycle Modul implementierte Statemaschine übernimmt die Aufgabe, einen vollständigen Zyklus auf dem DL300-System durchzuführen. Ein vollständiger Zyklus umfasst hierbei das Setzen aller notwendigen Busleitungen, um einen Lese- oder Schreibbefehl an das DL300-System abzusetzen und zu überprüfen, ob dies innerhalb einer definierten Zeit erfolgreich möglich war. Der DL300-Bus verfügt dafür über die Eingangsleitungen ADS (address strobe), um von extern zu signalisieren, dass die Adressen für den nächsten Zyklus an den 11 Adressleitungen A0-A10 anliegen, sowie über den Ausgang DTACK (data acknowledge), der das Beenden eines Zyklus durch die DL300-Elektronik an den externen Kontroller meldet. Für die Unterscheidung zwischen einem Lese- und Schreibzugriff steht der Selektor R/W zur Verfügung, der dem DL300-System die Art des Zugriffs mitteilen kann. Hierbei gibt es die Möglichkeit eines Lese- (logisch 1) oder eines Schreibzugriffes (logisch 0). Das Signal ist logisch invertiert ($R/\bar{W}$). In-

```
1       statemachine : block
2       type states is (st_idle,st_start,st_stop);
3       signal c_state : states;
4       begin
5               state_decoder : process (clk,rst) is
6               begin
7                       if rst=1 then
8                               c_state <= st_idle;
9                       elsif (clk'event and clk='1') then
10                      case c_state is
11                              when st_idle =>
12                                      a <= '0'; b <= '0';
13                                      if start_me = '1' then
14                                              c_state <= st_start;
15                                      else
16                                              c_state <= st_idle;
17                                      end if;
18                              when st_start =>
19                                      a <= '1';
20                                      c_state <= st_stop;
21                              when st_stop =>
22                                      b <= '1';
24                                      c_state <= st_idle;
25                      end case;
26                      end if;
27              end process;
28      end block;
```

Quellcode 6.5 Beispiel einer Statemaschine mit Taktfrequenz clk.

tern wird das nichtinvertierte Signal verwendet (Schreibzugriff ist hierbei logisch 1), und erst bei der Ausgabe auf den DL300-Bus wird die Invertierung durchgeführt. Dies gilt ebenso für das externe ADS-Signal.

Für das zeitliche Verhalten ist es außerdem noch wichtig, dass zwischen einem normalen Zyklus und einem gestreckten unterschieden wird. Hierbei muss berücksichtigt werden, dass eine Zeitüberschreitung bei einem gestreckten Zyklus erst nach mehr als 14 μs auftritt. Der gestreckte Zyklus ist notwendig für das Setzen der DACs auf dem DL300-System. Die verwendeten Motorola-Chips benötigen hierbei ein spezielles Verfahren, um korrekt gesetzt zu werden. Dies wird noch in Kapitel 6.7.5 behandelt.

Die Unterscheidung erfolgt über das an **dl_cycle** übergebene 3-Bit Signal **dl300_cmd**. Ein Wert von 001 bedeutet hierbei, dass ein normaler Zugriff auf den Bus stattfinden soll. Jeder andere Wert bedeutet, dass ein verlängerter Zugriff durchgeführt wird. Die Portdeklaration für die Statemaschine dl_cycle sieht mit oben genannten Überlegungen wie in Quellcode 6.6 abgebildet aus.

Zunächst sind die Standardsignale für den Zeitgeber **clk** in Zeile 3 und die asynchrone Resetleitung **rst** in Zeile 4 in der Deklaration vorhanden. Über die zusätzliche Reset-Leitung

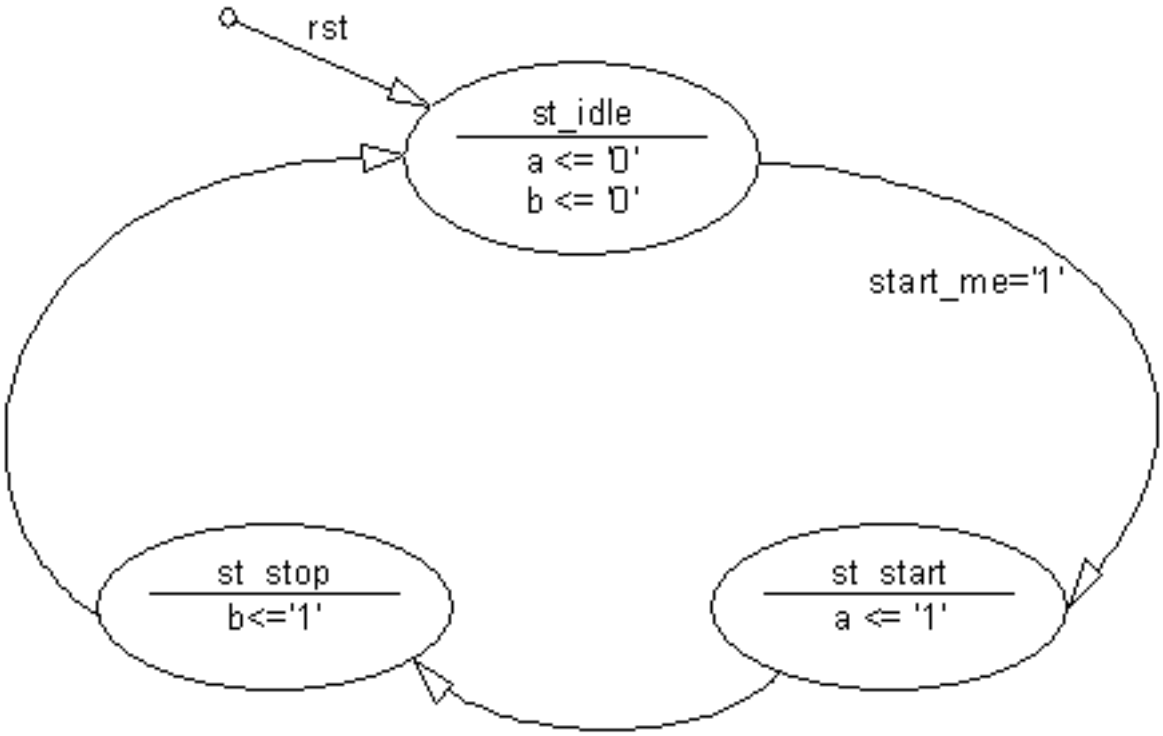


Abbildung 6.4 Statecad-Diagramm einer getakteten Statemaschine.

```
1               entity dl_cycle is
2               port (
3                       clk : in std_logic;
4                       rst : in std_logic;
5                       rst_force : in std_logic;
6
7                       dl300_cmd : in std_logic_vector (2 downto 0);
8                       dl300_dtack : in std_logic;
9                       dl300_rw : in std_logic;
10                      dl300_ads_out : out std_logic;
11                      dl300_busy : out std_logic;
12                      dl300_read_st : out std_logic;
13                      dl_error_out : out std_logic;
14                      dt_error_out : out std_logic;
15              );
16              end dl_cycle;
```

Quellcode 6.6 Entity Deklaration der dl_cycle Statemaschine.

rst_force kann über den PCI-Bus jede Statemaschine kontrolliert in den Ausgangs-Zustand gebracht werden. Hierbei wird dieselbe Initialisierung durchgeführt, die auch nach dem Laden des FPGAs vorgenommen wird. Dabei werden insbesondere die Fehlerleitungen **dl_error_out** und **dt_error_out** zurückgesetzt, ebenso wie alle intern verwendeten Variablen. Die Statemaschine wird auf den Ruhe-Zustand **st_idle** gesetzt. Neben der oben schon beschriebenen Kommando-Leitung **dl300_cmd**, die auch gleichzeitig das Startsignal für die Statemaschine

liefert, erhält die Maschine noch die Busleitungen DTACK und R/W als Eingangssignal, sowie das ADS-Signal **dl300_ads_out** als Ausgang. Die Statemaschine generiert die Signale Beschäftigt **dl300_busy** und Lesezugriff **dl300_read_st**. Das erste der beiden Signale wechselt von logisch 0 auf logisch 1, wenn die Statemaschine das Signal Start über die Bedingungen **dl300_cmd**>0 erhält. Das zweite Signal **dl300_read_st** wird gesetzt, wenn über den Bus von extern Daten gelesen werden können.

Als Hilfsmittel für die Statemaschine ist im Quellcode ein Zähler definiert. Dieser Zähler wird im Weiteren in nahezu jeder Statemaschine eingesetzt und soll hier kurz exemplarisch beschrieben werden. Der jeweilige Zähler kann über die Leitung **cnt_reset**=1 auf Null zurückgesetzt und gestoppt und über das Setzen von cnt_reset = 0 gestartet werden. Dies ist für die Bestimmung von Zeitüberschreitungen bei Fehlern im Transferablauf (falsche Adressierung, Fehlen von Hardware-Signalen) wichtig, da in diesem Falle ein Abbruch des Transfers herbeigeführt werden muss. Zu Beginn eines Zugriffes auf den DL300-Bus wird die Variable **cnt_reset** vom Wert 1 auf den Wert 0 gesetzt, wodurch der Zähler startet. Bei jedem Taktzyklus des Taktgebers **clk** wird der Zähler um eins erhöht. Der Zähler muss dabei entsprechend groß dimensioniert sein. Dieser einfache Zählerprozess ist wie in Quellcode A.2 im Anhang definiert. Wird ein maximaler Zählerstand erreicht, liegt eine Fehlerbedingung vor und das Statediagramm wird mit einer Fehlermeldung abnormal beendet.

Die Statemaschine befindet sich zunächst im Ruhe-Zustand **st_idle**, bis das Signal **dl300_cmd** gesetzt wird. Dies geschieht nur, sofern kein Fehler vorliegt, also wenn die Signale **dl_error** und **dt_error noch** auf 0 gesetzt sind. Die maximale Zykluszeit wird in der Variablen **max_cycle_time** festgelegt, die entsprechend dem Wert von **dl300_cmd** auf die Konstante **max_cycle_time_normal** oder **max_cycle_time_stretched** gesetzt wird. Die definierten Konstanten sind dabei ein Vielfaches der externen Taktfrequenz **clk** der Statemaschine. Bei der verwendeten Clock von 66,66 MHz entspricht dies einer Zeit von 15 ns. Die Statemaschine verlässt dann den State **st_idle** und wechselt zum Transfer-Zustand **st_transfer**. Hier wird das Beschäftigt-Signal **busy** gesetzt, welches an das nach extern weitergereichte Signal **dl300_busy** weitergegeben wird. Um einen Zeitüberlauf festzustellen, wird der Zähler gestartet und über Setzen des Adress-Strobe **dl300_ads** auf logisch 1 ein Zyklus auf dem DL300-System initiiert. Hierbei müssen dann bereits die Adressen auf den Adressleitungen A0-A11 anliegen, ebenso wie die Daten. Danach wechselt die Maschine in den Status **st_ack_cycle**, der darauf wartet, dass die **dl300_dtack** Leitung gesetzt wird. Sofern dies nicht innerhalb der definierten Zeitdauer passiert, wird ein Fehler ausgelöst, was intern im FPGA über Setzen des Fehler-Signals **dl_error** im Busfehler-Zustand **st_buserror** geschieht. Erreicht das Data-Acknowledge Signal den FPGA in der definierten Zeit, so wechselt die Statemaschine entweder in den Schreib-Zustand **st_write** oder in den Lese-Zustand **st_read**, abhängig vom Lese/Schreiben-Eingangssignal **dl300_rw** bzw. der damit verbundenen externen Busleitung. In den beiden Zuständen wird das Adressimpuls-Signal **dl300_ads** und der Zähler für die Zeitüberschreitung zurückgesetzt. Im Falle eines Lesezyklus wird noch das Signal Leseimpuls **dl300_read_st** gesetzt. Dieses signalisiert, dass die Datenleitungen am Bus einen gültigen Wert für den Lesezugriff haben. Im externen Code kann dann das Datenwort gelesen und gespeichert werden, wenn das Signal aktiv ist. Beide Zustände wechseln danach in den

abschließenden End-Zustand **st_end_cycle**, der zunächst den hierfür notwendigen Timeout-Zähler wieder startet und auf die Rücknahme des Daten-Akzeptiert-Signals **dl300_dtack** auf dem Bus wartet, was nach einem Löschen des Signals erfolgen sollte. Erfolgt dies nicht in der definierten Zeit, so wechselt die Statemaschine in den Fehler-Zustand **st_dtackerror**, ansonsten wird in den Ruhe-Zustand **st_idle** zurückgewechselt und der Zyklus ist mit Rücksetzen des Beschäftigt-Signals **busy** beendet.

6.7.2 dl_read

Die Statemaschine für den Lesezugriff **dl_read** führt unter Verwendung der zuvor erwähnten Zyklus-Statemaschine **dl_cycle** einen Lesezugriff auf dem DL300-Bus aus. Hierbei werden das Beschäftigt-Signal **busy** und das Leseimpuls-Signal **read_strobe** der Zyklus-Statemaschine verwendet, um festzustellen, wann ein Zugriff beendet worden ist und wann der entsprechende Lesebefehl auf das Datenwort ausgeführt werden soll. Die Deklaration der Statemaschine (siehe Quellcode 6.7) enthält zusätzlich ein Beschäftigt-Ausgangs-Signal **busy_out** als Ausgang, sowie eine Steuerungsleitung für das Kommando-Signal **dl300_cmd** und das Lese-Schreib-Signal **dl300_rw**, welches dann an die Zyklus-Statemaschine **dl_cycle** weitergegeben werden muss.

```
1       entity dl_read is
2               port (
3                       clk : in std_logic;
4                       rst : in std_logic;
5                       rst_force : in std_logic;
6
7                       dl_cycle_busy_in : std_logic;
8                       read_strobe_in : in std_logic;
9                       start_in : in std_logic;
10                      busy_out : out std_logic;
11                      dl300_cmd_out : out std_logic_vector (2 downto 0);
12                      dl300_rw_out : out std_logic;
13                      dl_read_err_out : out std_logic;
14                      dl300_error : in std_logic_vector(31 downto 0)
15              );
16      end dl_read;
```

Quellcode 6.7 Entity-Deklaration der dl_read Statemaschine.

Desweiteren sind Anschlüsse für Fehlerstatusmeldungen vorhanden (Zeile 13), sowie ein Eingangssignal für alle Fehlercodes innerhalb des DL300-Interfaces (Zeile 14), sowie Zeitgeber und Rücksetzleitungen analog zur Zyklus-Statemaschine **dl_cycle** (Zeilen 3-5). Für den Start eines Lesetransfers kann über das Start-Signal **start_in** (Zeile 9) die Statemaschine gestartet werden. Die Implementierung der verschiedenen Zustände ist in Quellcode 6.8 zu sehen. Zunächst wird nach Setzen des Start-Signals **start_in** in den Start-Lese-Zustand **st_start_read** gewechselt. Dort werden das Beschäftigt-Signal **busy**, das RW-Bit und das

Kommando für die Zyklus-Maschine **dl_cycle** gesetzt (Zeilen 11-13), um den Zyklus auf dem DL300-System zu starten. Nachfolgend wird in den Zustand zum Warten auf den Leseimpuls **st_wait_read_strobe** gewechselt (Zeile 15). Hier wird darauf gewartet, dass die Zyklus-Statemaschine **dl_cycle** das Beschäftigt-Signal **busy** setzt. Darauf wird das Kommando-Signal **dl300_cmd** zurückgenommen (Zeile 18), um nach Beendigung eines Zyklus nicht einen erneuten Zugriff zu starten. Intern wird nun die Variable für den Start der Zyklus-Statemaschine **dl_cycle_started** gesetzt (Zeile 19), um festzuhalten, ob ein Transfer erfolgreich initiiert werden konnte. Wenn dies der Fall ist und innerhalb der definierten Zeit die Zyklus-Statemaschine **dl_cycle** ein Leseimpuls-Signal generiert, so kann der Zyklus beendet werden. Die Maschine wartet dann im End-Zustand **st_wait_cycle_end** (Zeilen 31-33) auf die erfolgreiche Beendigung der Zyklus-Statemaschine **dl_cycle**, welches durch die Rücknahme des Beschäftigt-Signals **dl_cycle_busy** signalisiert wird. Wenn dies erfolgt und kein sonstiger Fehler aufgetreten ist (Bedingung **dl300_error**>0), dann wechselt die Maschine in den Ruhe-Zustand, das Beschäftigt-Signal **busy** wird zurückgenommen und der Lesezyklus ist erfolgreich beendet.

6.7.3 dl_write

Analog zur **dl_read** Maschine arbeitet die **dl_write** Statemaschine. Sie erhält zusätzlich zu den in **dl_read** verwendeten Ein- und Ausgängen noch einen binären Eingang, der eine Unterscheidung zwischen gestrecktem und normalem Zyklus ermöglicht (Selektor **stretched**). Über den Zustand dieser Signalleitung wird dann entschieden, wie hoch die Zeitüberschreitung sein darf, die für einen Zyklus akzeptabel ist und welchen Wert das Kommando-Signal **cmd** beim Starten des Zyklus **dl_cycle** erhält. Das Leseimpuls-Signal entfällt und wird auch nicht geprüft. Es wird lediglich geprüft, ob das Beschäftigt-Signal **busy** zurückgenommen wird und der Zyklus auf dem DL300-System korrekt abgeschlossen wurde. Die Steuerleitung **rw** erhält den Wert 1, um damit einen Schreibzyklus zu signalisieren.

6.7.4 Multiplexer

Der Begriff Multiplexer bezeichnet ein Selektionsschaltnetz. Dieses dient dazu, aus mehreren Eingängen ein Signal zu selektieren und auf einen Ausgang durchzuschalten. Die Auswahl des Eingangs erfolgt durch ein oder auch mehrere Steuersignale, die bestimmen, welches der Eingangssignale selektiert wird. Multiplexer werden in der Implementierung des DL300-Interfaces an vielen verschiedenen Stellen verwendet, insbesondere für die Steuerung der Ein- und Ausgangssignale der in Kapitel 6.7 beschriebenen Statemaschinen und der SSRAM-Bausteine.

Im Folgenden sollen kurz exemplarisch einige der Multiplexer für die Statemaschinen beschrieben werden, da diese wichtig für das Verständnis der Funktionsweise des Interfaces sind. In Kapitel 6.7.1 wurde bereits die Statemaschine **dl_cycle** beschrieben. Diese benötigt die Eingangssignale **cmd** und **rw**, welche mit den Signalen **dl300_cycle_cmd_in** und **dl300_cycle_rw_in** verbunden sind. Die Statemaschinen **dl_read** und **dl_write** nutzen beide die Funktionalität der Statemaschine **dl_cycle**. Die Ausgangssignale werden übergeben,

```
1       case c_state is
2               when st_idle =>
3                       busy <= '0';
4                       cnt_reset <= '1';
5                       cmd <= (others => '0');
6                       if (start_in = '1' and dl300_error=0) then
7                               c_state <= st_start_read;
8                       end if;
9                       dl_cycle_started <= '0';
10              when st_start_read =>
11                      busy <= '1';
12                      rw <= '0';
13                      cmd <= "001";
14                      cnt_reset <= '0';
15                      c_state <= st_wait_read_strobe;
16              when st_wait_read_strobe =>
17                      if (dl_cycle_busy_in = '1') then
18                              cmd <= "000";
19                              dl_cycle_started <= '1';
20                      end if;
21                      if (counter>max_cycle_time or dl300_error>0) then
22                              c_state <= st_error;
23                      end if;
24                      if (dl_cycle_started = '1' and read_strobe_in = '1') then
25                              c_state <= st_wait_cycle_end;
26                      end if;
27                      when st_wait_cycle_end =>
28                              if (counter>max_cycle_time or dl300_error>0) then
29                                      c_state <= st_error;
30                              end if;
31                              if (dl_cycle_busy_in = '0') then
32                                      c_state <= st_idle;
33                              end if;
34                      when st_error =>
35                              if (dl300_error>0) then
36                                      dl_read_err <= '0';
37                              else
38                                      dl_read_err <= '1';
39                              end if;
40                              c_state <= st_idle;
41      end case;
```

Quellcode 6.8 Zustände der dl_read Statemaschine.

wenn das Signal **busy** der übergeordneten Statemaschine gesetzt ist. Die Implementation des Multiplexers ist in Quellcode A.3 im Anhang zu sehen.

Für die Signale **dl300_read_start**, **dl300_write_start** und **dl300_write_data** ist analog ein Multiplexer implementiert. Da sich hier jedoch die Statemaschinen **dl_fast**, **dl_hit_search** und **dl_xfer_hits** geschachtelt aufrufen, sind die Bedingungen für die Übergabe der Ausgabewer-

te so implementiert, dass jeweils auch das Signal **busy** der übergeordneten Statemaschine verlangt wird. Die Implementierung hierzu ist Quellcode A.4 im Anhang zu entnehmen.

6.7.5 dl_setdac

Zum Setzen der Analog-Digital-Konverter des Typs Motorola MC144111 ist ein etwas komplizierterer Schreibzyklus notwendig, da dieser Baustein nur über ein serielles Dateninterface verfügt. Hierbei wird der schon beschriebene gestreckte Zyklus verwendet, der mit 14 μs wesentlich länger dauert als ein normaler Zyklus. Die DACs für die Kanäle 0-3 einer FADC-Karte werden dabei nacheinander gesetzt. Das Beschreiben eines einzigen FADC-Kanals ist nicht möglich, da alle Kanäle sukzessive abgearbeitet werden müssen. Die 6 Bit des jeweiligen Kanals werden bitweise in absteigender Reihenfolge geschrieben. Die Kanäle werden ebenfalls in der absteigenden Reihenfolge 3,2,1,0 gesetzt. Insgesamt sind somit 24 Schreibzyklen notwendig, um den DAC für eine FADC-Karte zu setzen. Hierfür wird ein gesonderter Zyklus in dieser Statemaschine programmiert, der den Schreibzugriff der Statemaschine **dl_write** verwendet (siehe Kapitel 6.7.3) und den bereits beschriebenen Selektor **stretched** nutzt. Zur Unterscheidung, ob ein Modul vom Typ DL305 oder DL310 adressiert wird, ist der Selektor moduletype vorhanden. Dieser ist logisch 1 für ein Modul des Typs DL310 (1024 Byte) und 0 für ein Modul des Typs DL305 (256 Byte) und muss von extern über ein PCI-Register gesetzt werden (siehe Kapitel 6.11).

Die Statemaschine erhält neben den bereits bekannten Ein- und Ausgängen zusätzliche Signalleitungen. Zum einen sind dies die Ausgänge für die Adresse auf dem DL300-System zum anderen die Datenleitungen. Als Eingangsparameter können die Nummer der FADC-Karte und die zu setzenden Werte für die 4 Kanäle der FADC-Karte übergeben werden (24 Bit). Die zu übergebenden 4 Kanäle sind dabei wie in Tabelle A.16 im Anhang zu sehen kodiert.

Für die Adressierung der DACs muss zunächst in den Registern 0x404 (ACL) und 0x405 (ACH) die korrekte Adresse für den DAC gesetzt werden. Dies geschieht in den beiden Zuständen **st_set_acl** und **st_set_ach**. Die Adresse errechnet sich, wie in Quellcode 6.9 zu sehen, aus der DAC-Nummer für ein Cratesystem bestückt mit DL305-Modulen (in VHDL Notation[60]), sowie für DL310-Module (siehe Quellcode 6.10).

```
1              ach = X"00" \& "001" \& dac\_number(5\ downto\ 1)
```

Quellcode 6.9 Berechnung des oberen Adress-Counters für das Setzen der DACs (DL305-Module).

[60] Das Symbol & beschreibt in VHDL die Verkettung von zwei Bitmustern: 010 & 011 ergibt die sechstellige Binärzahl 010011

```
1              ach = X"00" \& "1" \& dac\_number(5\ downto\ 1) \& "00"
```

Quellcode 6.10 Berechnung des oberen Adress-Counters für das Setzen der DACs (DL310-Module).

Für die Adresse im ACL-Window wird in beiden Fällen an der Bitposition 10 der Wert des Bit 0 von **dac_number** übergeben. Zum Schreiben im Stretched-Mode wird die Funktion 0x402 der DL307-Interfacekarte verwendet. Hierzu werden die 24 Schreibzugriffe durchgeführt, die oben bereits beschrieben wurden. Im VHDL-Code wird dies durch zwei Zähler realisiert, deren Werte **bitpos** mit 5 und **valpos** mit 3 initialisiert werden. Dies ist in Quellcode A.5 im Anhang zu sehen.

Mit diesen „Basis“-Statemaschinen ist nun die hardwaremäßige Steuerung des DL300-Systems implementiert. Im Folgenden werden die zusammengesetzten Schreib- und Leseoperationen beschrieben.

6.7.6 dl_samp

Um das DL300-System in den Datenerfassungs-Modus (Sampling) zu schalten sind mehrere Schritte notwendig. Zunächst muss das vom Scanner verwaltete Bus-Acknowledge-Signal (BAK) gesetzt werden. Solange der Sampling-Modus aktiv ist, bleibt das Signal gesetzt. Das BAK-Signal signalisiert eine Lese-/Schreiboperation des DL302-Moduls auf dem Bus. Danach muss der Adresszeiger auf das Scanner-Modul gesetzt werden, damit das Modul über die nachfolgenden Schreibzugriffe initialisiert und konfiguriert, sowie der Samplingvorgang gestartet werden kann. Dies geschieht über die Adresse 0x405 auf dem DL307-Interfacemodul. Um den Scanner anzusteuern wird an diese Adresse der Wert 0x00F0 geschrieben. Danach wird ein beliebiger Schreibzugriff auf die Adresse 0x0E (entspricht 14 dezimal) ausgeführt (siehe Tabelle A.6), welcher das Scanner-Modul zurücksetzt. Über einen Schreibzugriff auf das Register 0x0D wird der Scanner gestoppt, ein Zugriff auf das Register 0x0F löscht das Hit-Flag. Durch diese Initialisierungen befindet sich das Modul im Ausgangs-Zustand. Um in den Sampling-Modus zu gelangen, wird an die interne Adresse 0x08 des Scanners der Wert des Samplemode-Registers geschrieben, welches über den PCI-Bus von der Software gesetzt werden kann (siehe Kapitel 6.5.2). In Tabelle A.7 ist eine Übersicht über die möglichen Parameter für das Scanner-Modul gegeben. Um den Scanner in den Samplemodus zu schalten, dürfen die Bits 2 und 4 im Schreibzugriff auf das Register 0x08 nicht gesetzt (logisch 0) sein. Über Bit 3 kann ausgewählt werden, ob der Erfassungsvorgang im Common-Stop- oder im Common-Start-Modus betrieben werden soll. Über die Bits 0 und 1 kann die Auswahl der Frequenz erfolgen, mit der die Datenerfassung stattfinden soll. Hier ist es möglich, einen externen Frequenzgenerator zu verwenden, eine benutzerdefinierbare Frequenz auszuwählen, eine Frequenz von 50 MHz auszuwählen oder mit 100 MHz Datenrate zu erfassen. Nachdem der hierfür definierte Wert aus dem PCI-Register an die interne Adresse 0x08 des Scanners geschrieben wurde, wird die Stop-Adresse des Scanners über die interne Adresse 0x0A auf 0xFF gesetzt. Abschließend wird die Start-Adresse des Scanners initialisiert und auf 0 gesetzt (Interne-Adresse

0x09). Diese mehrfachen Schreibzugriffe könnten natürlich einzeln über die implementierten Hardwareregister durchgeführt werden, was allerdings erhebliche Ausführungszeiten und eine ständige Interaktion des Rechners zur Folge hätte. Daher wurde diese Schreibsequenz in einer Statemaschine implementiert, die nur einen Rechnerzugriff benötigt, um die gewünschte Aktion zu starten. Hier nun zusammengefasst die von der **dl_samp** Statemaschine ausgeführten Schreiboperationen:

- 0x0E an Adresse 0x406 des DL307-Interfacemoduls (BAK assert)
- 0xF0 an Adresse 0x405 des DL307-Interfacemoduls (ACH auf Scanner-Adresse)
- beliebiger Schreibzugriff auf 0x0E des DL302-Scanners (Clear-Scanner)
- beliebiger Schreibzugriff auf 0x0D des DL302-Scanners (Stop-Scanner)
- beliebiger Schreibzugriff auf 0x0F des DL302-Scanners (Clear Hit-Flag)
- Erfassungsmodus schreiben auf 0x08 des DL302-Scanners (Sample-Mode Parameter)
- 0xFF schreiben auf 0x0A des DL302-Scanners (Scanner-Stop-Adress)
- 0x00 schreiben auf 0x09 des DL302-Scanners (Set Scanner-Adress-Counter)

6.7.7 dl_fast

Für die Erfassung von Treffern (engl. hits) auf dem Speicherbereich des DL300-Systems ist der Fastscan-Modus zuständig. Um das Auffinden von Pulsinformationen in dem zusammenhängenden Speicherbereich der FADCs zu erleichtern, existiert auf dem Scanner-Modul eine Hardwareimplementierung, die bei einem autonomen konsekutiven Auslesen der Daten der FADC-Speicher auf dem internen ECL-Bus die Pulshöheninformation mit einem Schwellenwert vergleicht und die Auslese an der Stelle stoppt, an der die Schwellenbedingung erfüllt ist. Somit werden alle Speicherzellen übersprungen, die keine relevante Information enthalten. Die Statemaschine **dl_fast** implementiert alle Schritte die notwendig sind, um das DL300-System in den Fastscan-Modus zu versetzen und den Scan zu starten. Hierfür sind Parameter notwendig, die der Fastscan-Statemaschine extern über PCI-Register übergeben werden können. Dabei sind zur Zeit die folgenden Parameter implementiert:

- scan_stop_in
- scan_mode_in
- scan_thresh_in
- stop_pointer_out
- hit_search_out .

Im Ruhe-Zustand **st_idle** werden alle internen Variablen der Statemaschine initialisiert. Sobald das Startsignal **start_in** anliegt und kein Fehler vorliegt (dl300_error = 0), wechselt die Statemaschine in den Zustand Start **st_start**. Hier wird zunächst das interne Beschäftigt-Signal **busy** und das Signal **mybusy** gesetzt. Die Statemaschine wechselt daraufhin in den Zustand zum Setzen der Adresszähler **st_set_scan_ach**. In diesem Zustand wird dafür gesorgt, dass der Adresszeiger des DL300-Interface-Moduls auf den korrekten Wert für das Scanner-Modul gesetzt wird. Dazu wird in das Register 0x405 der Wert 0xF0 geschrieben. Der Schreibzugriff wird über ein spezielles Konstrukt durchgeführt, welches hier kurz beschrieben werden soll.

Um jeweils wiederkehrende Schreib- oder Lesezugriffe durchzuführen und nicht jedesmal den Schreibzugriff vollständig implementieren zu müssen, wurden eine Schreib- und eine Lesesubroutine aufgesetzt, in die aus der Statemaschine verzweigt werden kann. Als Parameter kann für die Rückkehr ein Zustand definiert werden, in welchen nach dem vollständig abgearbeiteten Lese- oder Schreibzugriff verzweigt werden soll. Dies wird über die Variable next_state erreicht, der vor dem jeweiligen Zugriff der neue Zustand übergeben wird. Für den Start eines Schreibzugriffes ist der Schreib-Zustand **st_pre_write** vorgesehen, für den Lesezugriff der Lese-Zustand **st_pre_read**. Über die Signale Start-Lesen **read_start** und Start-Schreiben **write_start** werden die Statemaschinen für den Schreib- und Lesezugriff **dl_read** und **dl_write** gestartet. Die Zustände **st_pre_write** und **st_pre_read** warten auf die vollständige Abarbeitung der beiden Statemaschinen, welche durch das Zurücksetzen der Beschäftigt-Signalleitungen **write_busy_in** und **read_busy_in** auf logisch 0 signalisiert wird.

```
1               when st_pre_write =>
2                       cnt_reset <= '1';
3                       write_start <= '1';
4                       write_started <= '0';
5                       c_state <= st_write;
6
7               when st_write =>
8                       cnt_reset <= '0';
9                       if (write_busy_in='1') then
10                      write_start <='0';
11                      write_started <= '1';
12              end if;
13              if (counter > max_cycle_time or dl300_error>0) then
14                      c_state <= st_error;
15              end if;
16              if (write_started='1' and write_busy_in='0') then
17                      c_state <= next_state;
18              end if;
```

Quellcode 6.11 Beispiel eines Schreibzugriffs unter Verwendung der dl_write Statemaschine.

Der Zeitüberschreitungszähler wird erst zurückgesetzt (Zeile 2) und danach gestartet (Zeile 8), so dass dieser beginnend mit 0 mit der Taktfrequenz der Statemaschine zählt. In Zeile 3

wird die Schreib-Statemaschine **dl_write** gestartet, da das Signal **write_start** mit dem Start-Eingangssignal der Statemaschine **dl_write** verknüpft ist, sofern die Maschine **dl_fast** aktiv ist (Multiplexer im Hauptquellcode, siehe Kapitel 6.7.4). Sobald das Signal **write_busy_in** von logisch 0 auf logisch 1 wechselt, wird die Variable **write_started** auf logisch 1 gesetzt und das Signal für das Starten der Statemaschine **dl_write** zurückgesetzt (Zeilen 9-12). Wenn das Beschäftigt-Signal **busy** nicht innerhalb der definierten Zeitdauer **max_cycle_time** zurückgenommen wird, wechselt die Statemaschine in den Fehler-Zustand **st_error** (Zeilen 13-15). Zeile 17 enthält die bereits erwähnte Anweisung, die nach Beendigung des Schreibzugriffes in den neuen Zustand wechselt, der zuvor in der Variable **next_state** gespeichert wurde. Der Lesezugriff ist analog implementiert.

Nach dem Setzen des ACH-Registers des Scanners wird zunächst die Stop-Position ausgelesen, an welcher der Samplevorgang beendet wurde. Die geschieht über einen Lesezugriff auf der internen Adresse 0x005 des DL302-Scanner-Moduls. Die Position (Stop-Pointer) wird daraufhin in der Variablen **stop_pointer** gespeichert. Danach wird der Scanner über einen Lesezugriff auf der internen Adresse 0x006 zurückgesetzt, und es werden die Parameter für den Fastscan an das Crate übergeben.

Zunächst wird der Scanmodus über die interne Adresse 0x008 gesetzt (siehe hierzu Tabelle A.7). Nachfolgend wird dann die Scanner-Start- und Stop-Position gesetzt. Die Start-Position ist hierbei fest auf die Adresse 0x0000 programmiert und die Stop-Position, bis zu der der Scanner maximal laufen soll, kann über ein PCI-Register festgelegt werden. Der Wert im PCI-Register wird über die interne Adresse 0x000A an das DL300-Interface übergeben.

Für die Erkennung eines Treffers kann für die beiden Kanäle links und rechts auf den FADC-Karten ein separater Wert als Schwelle eingestellt werden. Im Interface wurde dies über ein PCI-Register implementiert, welches über die interne Adresse 0x000B an den Scanner übergeben wird.

Vor dem Start des Scan-Vorgangs wird zunächst noch das Hit-Flag über die Funktion 0x000F zurückgesetzt. Das Hit-Flag ist ein internes Signal des DL302-Scanners, welches angibt, ob zum Zeitpunkt des Zugriffs eine Trefferbedingung vorliegt. Diese liegt vor, wenn der Speicherwert der aktuellen und der vorherigen Speicherzelle über dem eingestellten Schwellenwert liegt. Für die Initialisierung ist das Rücksetzen dieses Hit-Flags notwendig. Abschließend wir der Scanner an die neue Start-Position gesetzt. Die Fastscan-Statemaschine **dl_fast** wird allerdings an diesem Punkt noch nicht beendet, sondern die nächste Statemaschine **dl_hit_search** wird initiiert. Dazu dient das Signal **dl_hit_search_out**. Das Signal **mybusy** wird beim Starten der Statemaschine **dl_hit_search** auf den Wert 0 zurückgesetzt. Erst nach Beenden der geschachtelten Statemaschine – **hit_search_busy_in** wechselt von logisch 1 auf logisch 0 – wird auch die Statemaschine **dl_fast** beendet. Hierbei wird dann auch das Signal **busy** zurückgenommen.

6.7.8 dl_hit_search

Die Statemaschine **dl_hit_search** führt die eigentliche Suche nach Ansprechern auf Kanälen

des DL300-Systems aus. Nach einer erfolgreichen Initialisierung des Fastscans durch Setzen des von der Statemaschine **dl_fast** erhaltenen Start-Signals **start**, wechselt die Statemaschine vom Ruhe-Zustand **st_idle** in die Initialisierung **st_start**. Hierbei werden das Signal **busy** und das Signal **mybusy** auf logisch 1 gesetzt. Beim Starten der Statemaschine wird der verwendete FIFO-Puffer gelöscht, so dass dieser keine Einträge von vorhergegangenen Ereignissen enthält. Dieser Puffer dient der Zwischenspeicherung von gefundenen Ansprechern. Der Adresszeiger für das SSRAM wird auf einen Bereich über dem normalen Ereignisspeicher auf die Adresse 0x10000 gesetzt. An diese Speicherposition werden die gefundenen Ansprecher für Diagnosezwecke abgelegt. Diese können nach einem Ereignis ausgelesen werden. An die Adresse 0x10000 auf dem SSRAM wird in einem ersten Schreibvorgang der Stop-Pointer abgelegt, der zuvor durch die Statemaschine **dl_fast** ausgelesen und extern an das Signal **dl300_fastscan_stop_pointer** weitergegeben wurde. Dieser steht lokal in der Statemaschine unter dem Namen **stop_pointer** zur Verfügung. Vor dem 16-Bit Wert des Stop-Pointers wird zusätzlich eine Markierung eingefügt, die den Anfang des Speichers anzeigt. Hierfür wird der 16-Bit Wert 0xFAFA verwendet. Danach wird in einer Schleife das Statusregister des DL300-Systems über die Adresse 0x406 gelesen. Ist hier das oberste der 16 Bits gesetzt, so liegt ein Processor-Request PRQ vor und der Hitscanner auf dem DL300 hat einen Ansprecher gefunden. Die Statemaschine verzweigt dann in den Zustand **st_read_hitpos**, andernfalls wird weiterhin das Statusregister abgefragt. Sofern nicht innerhalb einer definierten Anzahl von Taktzyklen ein PRQ ausgelöst wird, verzweigt die Statemaschinen in den Fehler-Zustand **st_error**. Gleiches gilt für den Fall, dass die maximale Anzahl an Treffern gefunden wurde (Parameter **dl300_maxhits**). Liegt ein Treffer vor, so wird die Speicherposition im Crate über die Funktion 0x4 des Hit-Scanner-Moduls im Zustand **st_read_hitpos** gelesen und im Zustand **st_eval_hitpos** ausgewertet. Sofern die Speicherposition auf der Adresse 0x0000 am Anfang des Crates steht, wurde kein Ansprecher mehr gefunden. Die Sucher nach Treffern wird in diesem Fall abgebrochen und sofern mehr als 0 Treffer gefunden wurden, wir ein Transfer initiiert, der in der Statemaschine **dl_xfer_hits** durchgeführt wird. Falls jedoch kein einziger Treffer gefunden wurde, ist dies nicht notwendig, und die Statemaschine verzweigt in den Zustand **st_no_xfer**. Hier wird in diesem Fall eine Markierung geschrieben, die sich aus dem 16-Bit Wert 0xFBFB sowie dem 16-Bit Wert des Stop-Pointers zusammensetzt. Da zwischenzeitlich keine Erhöhung des SSRAM-Adresszählers stattgefunden hat, wird dieser Wert an die Speicherstelle 0x10000 geschrieben. Ist die Speicher-Position größer als der Wert 0x0000, so wurde ein Ansprecher an der zurückgelieferten Speicherstelle gefunden. Über die Differenz aus der Stop-Position und der Position des gefundenen Ansprechers lässt sich die Zeit relativ zum Stop-Zeitpunkt des Samplingvorgangs ermitteln. Über die PCI-Register 0x11 und 0x12 (siehe Kapitel 6.5.2) kann vom Benutzer ein Zeitfenster definiert werden. Ist ein solches definiert, so werden die Positionen der zurückgelieferten Ansprecher daraufhin geprüft, dass sie im vorgegebenen Zeitintervall liegen. Falls dies der Fall ist, wird das Ereignis akzeptiert und es findet eine Verzweigung in den Zustand **st_accept_event** statt. Ist kein Zeitfenster definiert, wird jedes Ereignis unabhängig von der zeitlichen Position akzeptiert. Dabei wird der Adresszeiger für den SSRAM-Speicher um eins erhöht und die gefundene Hit-Position bzw. die Position, ab der ausgelesen werden soll, wird im SSRAM sowie im FIFO abgespeichert. Aufgrund der Arbeitsweise des DL300-Systems sind bei der Einordnung der Ereignisse allerdings einige

Dinge zu berücksichtigen. Der Speicherbereich für einen Kanal wird beim Samplingvorgang beschrieben. Sobald das Ende des Speichers eines Kanals erreicht ist, wird die Speicherung der FADC-Werte an der Position 0 des Speichers des Kanals fortgeführt und die zuvor aufgenommenen Datenwerte werden überschrieben. Da es für eine Bestimmung des Pedestalwertes wünschenswert ist, eine Anzahl von Speicherstellen vor dem gefundenen Ansprecher zu speichern, kann vom Benutzer eine Anzahl von sogenannten Presamples definiert werden. Die Auslese der Daten soll dann entsprechend der Größe des Presample-Wertes vor der gefundenen Hitposition geschehen. Aufgrund der Struktur des Speichers kann es passieren, dass durch Abzug der Presamples die Speichergrenze 0 eines FADC-Kanals unterschritten wird. In diesem Fall muss die Speicherposition an die korrekte Position im Endbereich des Moduls gesetzt werden. Im FIFO-Puffer wird diese korrigierte Position abgelegt (16-Bit Wert). Im SSRAM wird hingegen die ursprüngliche Position ohne Abzug der Presamples abgelegt. Zusätzlich zu diesem 16-Bit Wert werden an den Bits 31-16 zwei 8-Bit Werte abgelegt. Die Bits 31-24 enthalten dabei einen Zähler, der die Anzahl der gefundenen Ansprecher im aktuellen Fastscan enthält. Die Bits 23-16 enthalten einen der in Tabelle A.17 im Anhang angegebenen Werte mit der aufgeführten Bedeutung.

Nach dem Speichern der Position oder bei einem nicht akzeptierten Ereignis wird der Scanner auf eine neue Position gesetzt und der Fastscan wird fortgeführt. Hierbei errechnet sich die neue Position aus der zuvor gefundenen Position zzgl. der Anzahl der für ein Ereignis auszulesenden Speicherstellen, die durch den Benutzer über ein PCI-Register vorgegeben werden können. Findet durch die Addition der auszulesenden Speicherstellen ein Wechsel in den Speicherbereich des folgenden Moduls statt, so wird der neue Startpunkt auf den Anfang des nächsten Moduls gesetzt, damit dieses vollständig durch den Hitscanner erfasst wird. Die neue Startposition wird an Adresse 0x9 des Hitscanners geschrieben und über die Funktion 0x7 wird das Löschen des Hit-Flags und ein Neustart des Scanners bewirkt.

Nachdem der Hitscanner bis zur vorher festgelegten End-Position das Crate-System gescannt hat, generiert dieser einen Prozessor-Request (PRQ) und liefert als Position die Speicherstelle 0x0000 zurück. Sofern Ansprecher gefunden wurden, wird in den Transfer-Zustand **st_xfer_start** verzweigt. Hier wird die Statemaschine **dl_xfer_hits** gestartet und das Beschäftigt-Signal **busy** zurückgenommen. Sobald diese vollständig abgearbeitet ist (xfer_busy = 0), wird die SSRAM-Adresse um eins erhöht, abschließend der Wert 0xFFDFFFDD auf den RAM-Speicher geschrieben, der Scanner über die Funktion 0xD initialisiert und die Statemaschine kehrt in den Ruhe-Zustand **st_idle** zurückversetzt, wobei das lokale Beschäftigt-Signal **mybusy** zurückgenommen wird.

6.7.9 dl_xfer_hits

Zum Transfer der kompletten Daten eines gefundenen Treffers erhält die Statemaschine aus dem zuvor durch die **dl_hit_search** Statemaschine beschriebenen FIFO-Speicher[61] die Speicherpositionen der Ansprecher. Die Statemaschine liest die Anzahl der durch den Benutzer

[61] engl. FIFO - first in - first out

vorgegebenen Samples aus dem Crate-System und speichert diese im RAM-Speicher ab. Zusätzlich werden die Daten während der Auslese an Analyseroutinen weitergegeben. Hierbei können mehrere Analysemethoden die ausgelesenen Daten auswerten und deren Ergebnisse dem DL300-Interface zur Verfügung stellen.

Nach dem Start der Statemaschine über das Start-Signal **start** werden das Beschäftigt-Signal **busy** und das interne Beschäftigt-Signal **mybusy** gesetzt und ein Reset-Signal für die externen Analyseroutinen generiert. Danach ist die Statemaschine solange in Funktion, wie noch Werte im FIFO-Puffer vorliegen. Ist der FIFO-Puffer noch gefüllt, so wird im folgenden Schritt der nächste Wert aus dem Puffer gelesen. Der gelesene Wert (also die Speicherposition des gefundenen Ansprechers) wird dann in das SSRAM geschrieben. Dabei wird zur Markierung des Wertes der 16-Bit Wert 0xACAC vorangestellt. Bereits während des Schreibvorgangs auf das SSRAM wird der Adresszeiger des DL300-Systems auf die im FIFO gespeicherte Adresse gesetzt. Für den ersten Lesezugriff ist die Verwendung der Lesefunktion 0x400 notwendig. Diese liest den Wert an der aktuell eingestellten Speicher-Position. Für das kontinuierliche Lesen kann die Funktion 0x401 des Interfaces verwendet werden. Diese erhöht zunächst die Adresse im ACL-Register um den Wert 1 und liest danach den Wert an der neuen Speicherposition aus. Nach dem ersten Lesezugriff werden die gelesenen 6-Bit Werte der parallel gelesenen Kanäle im SSRAM-Speicher abgelegt. Hierbei setzt sich der zu schreibende Wert aus einer 4-Bit langen Markierung 0xF, der aktuellen Speicherposition im DL300(16-Bit) und den beiden Werten den gelesenen Kanäle (je 6 Bit) zusammen. Das Format der Daten ist in Tabelle A.18 im Anhang zu finden.

Alle nachfolgenden Werte, die zum selben Ansprecher auf dem DL300-System gehören, werden als 24-Bit Werte zu je 4x6-Bit abgespeichert. Dabei wird hier der 4-Bit Wert 0xD vorangestellt, sowie die aktuelle Anzahl der gelesenen Einträge reduziert auf 4 Bits (**num_reads** bzw. **n**). Das Format der Daten ist ebenfalls im Anhang in Tabelle A.19 aufgeführt.

Parallel zur Auslese der Daten werden diese an das Analyse-Pluginsystem weitergegeben (siehe Kapitel 6.8). Dabei werden die Daten, wie vom DL300 gelesen, als 12-Bit Wert mit je zwei 6-Bit Werten je Kanal übergeben. Desweiteren erhält das Analysesystem die Position relativ zum Startpunkt des Signals, sowie ein Taktsignal, welches mit jedem neu gelesenen Wert auf logisch 1 gesetzt und im nächsten Taktzyklus zurückgesetzt wird. Die Statemaschine berücksichtigt die Speichergrenzen der Kanäle und sorgt dafür, dass beim Erreichen der oberen Speichergrenze eines Moduls der Adresszähler (ACL) auf den Anfang des Speicherbereiches gesetzt und zunächst über die Funktion 0x400 und dann erneut über die Funktion 0x401 ausgelesen wird. Diese Vorgehensweise sorgt dafür, dass die Pulsdaten in der richtigen zeitlichen Reihenfolge in den Daten abgelegt werden. Die Speicherung der Daten im SSRAM erfolgt parallel zur Auslese des nächsten Datenwortes, um hier keine Zeit zu verlieren. Die Zykluszeiten auf dem DL300-System sind genügend groß, um die Daten während eines einzigen Lesezugriffes im RAM zu speichern. Die für das SSRAM benötigte Zeit beträgt hierbei 4 Taktzyklen von 33,33 MHz, somit also 120 ns. Die Zykluszeit eines Lesezugriffes auf dem DL300 benötigt wesentlich mehr Zeit, wie später in Kapitel 7.3 gezeigt wird.

6.8 Analysefunktionen

Die Analysefunktionen bekommen parallel zur Auslese die gelesenen Daten zur Verfügung gestellt. Die Daten können hierbei an nahezu beliebig viele[62] sogenannte Analyse-Plugins weitergegeben werden. Die Verarbeitung der Daten erfolgt im Gegensatz zu softwarebasierten Ansätzen vollständig parallel. Zur Demonstration wurden zunächst zwei Plugins entwickelt, die eine Integration über die gefundenen Pulse durchführen. Die Grenzen für die Summenbildung können jedem Plugin separat über PCI-Register (siehe Kapitel 6.5.2) übergeben werden. In der aktuellen Implementation sind 4 Hardwaresummen implementiert, die für den linken und rechten Kanal eine Summation anhand der übergebenen Daten und der Position durchführt. Die Summen stehen direkt nach Abschluss der Auslese eines Pulses zur Verfügung. Hiermit erhält man eine Energieinformation, da die Fläche unter einem Signal proportional zur im Kristall deponierten Energie ist. Zur Auswahl stehen die Plugins **dl_analyze_sum** und **dl_analyze_sum_lin**. Das Plugin **dl_analyze_sum** summiert die Digitalisierungswerte in den vorgegebenen Grenzen auf. Hierbei bleibt die Nichtlinearität des FADC allerdings unberücksichtigt. Das Plugin **dl_analyze_sum_lin** linearisiert hingegen die gelesenen Datenwerte anhand einer Linearisierungstabelle. Die Funktion

$$C_{lin} = \frac{C}{64 - a \cdot C} \cdot 0,5 \cdot k \cdot N \tag{6.2}$$

wird zum Erzeugen der Tabelle verwendet. Die errechneten linearisierten Werte C_{lin} werden gerundet. Mittels der Parameter k und N lassen sich kleinere oder größere Datenwerte erzeugen. Hierbei wurde k zu 1/0,361446 (Kehrwert des sich ergebenden Datenwertes nach der Linearisierung zum Digitalisierungswert 30) gewählt, sowie N zu 120. Mit diesen willkürlich gewählten Werten ergibt sich ein sinnvoller Datenbereich, bei dem die Summenwerte den 16-Bit-Wertebereich bei einer Anzahl von 100 Datenwerten pro Summe nicht überschreiten. Soll eine geringere Anzahl an Datenwerten ausgelesen werden, so kann die Genauigkeit der Hardwaresummen durch Anpassung der beiden Parameter erhöht werden. Hierzu müsste dann die im FPGA hinterlegte Tabelle angepasst werden und der FPGA mit der neuen Tabelle programmiert werden.

6.9 Fehlerdiagnose

Zur Fehlerdiagnose und zur Leistungsmessung sind auf dem Interface Schnittstellen implementiert worden, die vor allem bei der Entwicklung und der dabei notwendigen Fehlersuche benötigt wurden. Diese wurden intensiv bei der Entwicklung genutzt, da das Zusammenspiel der verschiedenen Komponenten wie PCI-Bus, SSRAM und das externe Flash-ADC-System genau aufeinander abgestimmt werden musste. VHDL bietet zwar die Möglichkeit, die entsprechenden Funktionen vorab über Simulationen zu testen, jedoch standen für die FPGA-Karte mit PCI-Bus und SSRAM, sowie für das DL300-Interface keine vom Hersteller spezifizierten Simulationen zur Verfügung[63]. Für die Analyse von Signalen auf dem FPGA-Chip ist eine Software erhältlich, die

[62] Nur begrenzt durch die maximale Chipfläche.

ein fertiges Design auf einem FPGA analysieren kann. Die Software ChipScope stand allerdings ebenfalls nicht zur Verfügung. Die Fehlersuche wurde jedoch durch Verwendung eines Logik-Analysierers erleichtert, mit dem die Signalleitungen zum DL300-System analysiert werden konnten. Dies ermöglichte eine sehr gute Analyse der programmierten Abläufe. Neben den 32 Datenleitungen, die zur Interfacekarte DL307 laufen, sind noch zwei weitere Ausgänge am 34-poligen Stecker der FPGA-Karte verfügbar. Diese wurden ebenfalls mit dem Logik-Analysierer verbunden, so dass beliebige VHDL-Signale aus dem FPGA auf diese zwei Kanäle gegeben und durch den Logik-Analysierer oder ein Oszilloskop betrachtet werden konnten. Die beiden Leitungen sind als **debug1** und **debug2** im VHDL-Code gekennzeichnet (siehe Tabelle A.1). Da es teilweise notwendig wurde, die zu überwachenden Signale während der Laufzeit der Datenerfassung zu wechseln, wurden die Signale so implementiert, dass über ein PCI-Register die Beschaltung der beiden Signalleitungen geändert werden können. Hierzu wird in die Register 0x102 und 0x103 (siehe Tabellen A.13, A.14 und A.15) ein Datenwert geschrieben, der einem zuvor im FPGA-Code definierten Signal entspricht. In Tabelle A.20 und Tabelle A.21 im Anhang sind die entsprechenden Werte und deren zugeordneten VHDL-Signale aufgeführt.

Mit dem eingesetzten Logik-Analysierer ist es möglich, diese Fehlerdiagnose-Signale neben dem Bus des DL300-System aufzuzeichnen. Hierüber ist eine schnelle Fehlerdiagnose durchführbar. Neben den herausgeführten Signalen, die primär für die Entwicklung des Interfaces sinnvoll sind, wurden für die Verwendung im Experiment in den Statemaschinen Fehlersignale eingeführt, so dass eine Lokalisierung einer Störung aufgrund eines zurückgelieferten Statuswortes möglich ist. Das Register 0x30 auf dem PCI-Bus dient der Ausgabe dieses 32-Bit Fehlerregisters. Im VHDL-Code ist dem Fehlerregister das Signal **dl300_error** zugeordnet. Die Belegung des Registers kann Tabelle A.22 im Anhang entnommen werden. Die Statemaschinen können einen oder auch mehrere Fehlertypen erzeugen, die je einem Bit des Fehlerregisters zugeordnet sind.

6.10 Zeitmessung

Zur Messung der benötigten Zykluszeiten der programmierten Statemaschinen wurden Zähler implementiert, die eine Bestimmung der Anzahl der von den Statemaschinen zur Ausführung der jeweiligen Operationen benötigten Taktzyklen ermöglichen. Diese Zähler sind als 64-Bit-Register implementiert. Die Zähler werden um eins erhöht, wenn die zugehörige Statemaschine aktiv ist. Hierzu werden die Beschäftigt-Signale **busy** bzw. **mybusy** ausgewertet, die die Statemaschinen erzeugen. Zusätzlich ist ein weiterer „freier“ Zähler implementiert, der während der Laufzeit permanent erhöht wird. Die Zähler werden bei der Initialisierung des FPGA zurückgesetzt und können zustätzlich manuell über einen Lesezugriff auf das PCI-Register 0x60 auf den Wert 0 zurückgesetzt werden. Der 64-Bit-Wert lässt sich über zwei je 32-Bit PCI-Register auslesen. Die zugehörigen Register können Tabelle A.14 im Anhang entnommen werden.

[63] Nach Abschluss der Entwicklung des Interfaces wurde eine entsprechende Simulation für die FPGA-Karte vom Hersteller Alpha-Data veröffentlicht.

6.11 Software DL300-System - libdl300

Für die Ansteuerung des programmierten Interfaces wird eine Software benötigt. Diese ist analog zu bereits bestehender VME- und CAMAC-Hardware[64] als Bibliothek ausgelegt (libdl300), die zum Beispiel bei Nutzung eines Flash-ADC-Systems in einem lokalen Eventbuilder eingebunden werden kann. Die Bibliothek verwendet für DMA-Übertragungen und das Laden der FPGA-Bitstreamdatei das vom Hersteller der FPGA-Karte mitgelieferte Software-Paket[65]. Die entwickelte Software ist als C++-Klasse implementiert.

6.11.1 Konstruktor und Destruktor

Für die Initialisierung des DL300-Systems ist der Konstruktor der C++-Klasse zuständig. Als Parameter werden diesem der Modultyp (DL305 oder DL310), die Anzahl der Module im Cratesystem, sowie die Position des Hardwarehit-Scanners übergeben, wobei der letzte Parameter optional ist und auf 24 gesetzt wird, sofern keine Position übergeben wird. Der Konstruktor versucht zunächst die FPGA-Karte über den PCI-Bus anzusprechen. Hierfür muss der Treiber der FPGA-Karte installiert und geladen sein. Sofern die Karte angesprochen werden kann, wird ermittelt, um welchen Typ von Karte es sich handelt. Hierbei werden alle derzeit vom Hersteller produzierten Kartentypen erkannt und unterstützt. Je nach Kartentyp gibt es mehrere mögliche Speicherkonfigurationen (andere Größen und Bankanordnungen). Die Parameter werden ermittelt und in Variablen gespeichert bzw. über PCI-Register an die Karte weitergegeben. Nachfolgend werden die Taktfrequenzen der beiden Oszillatoren gesetzt. Hierbei wird eine Frequenz von 33,33 MHz für die PCI-Busclock gesetzt, die der Standardbusfrequenz entspricht. Zusätzlich wird der zweite Frequenzgenerator mit 100 MHz getaktet. Dieser wird nicht benötigt und somit derzeit nicht für das FPGA-Design verwendet. Für den DMA-Zugriff werden zwei Speicherbereiche allokiert. Danach wird versucht, die in der Bibliothek fest vorgegebene Bitstreamdatei zu laden und den FPGA mit dieser Datei zu konfigurieren. Es werden DMA-Deskriptoren für die beiden reservierten Speicherbereiche angelegt und ein Lese- und Schreibtest auf dem SSRAM Speicher mit verschiedenen Testbitmustern durchgeführt, um zu gewährleisten, dass der Speicher auf der FPGA-Karte korrekt arbeitet und durch den FPGA richtig angesprochen wird. Nach Abschluss des Speichertests wird der Speicher auf der Karte auf den Wert 0 zurückgesetzt. Über das PCI-Register 0x0 wird das Erstellungsdatum der Bitstreamdatei und die Versionsnummer aus dem FPGA ausgelesen und überprüft, ob es sich um eine Entwicklungsdatei handelt. Dies ist der Fall, wenn die Versionsnummer den Wert 255 (0xFF Hex) trägt. Eine Entwicklungsdatei wird nur akzeptiert, wenn der Schalter DL300_DEBUG in der Bibliothek definiert ist. Dies soll gewährleisten, dass keine Entwicklungsversionen im laufenden Betrieb benutzt werden. Danach werden entsprechend der dem Konstruktor übergebenen Parameter die PCI-Register gefüllt, die dem FPGA angeben, welche und wieviele Module im Cratesystem vorhanden sind. Zusätzlich werden einige Parameter des Interfaces auf sinnvolle Werte durch die Bibliothek initialisiert. Vor Verlassen des Konstruktors wird über einen

[64] VME und CAMAC sind zwei in der Experimentalphysik häufig verwendete Bussysteme

[65] engl. Application-Programming-Interface - kurz API

Lesezugriff auf das Statusregister der DL307-Interfacekarte versucht, auf das Crate-System zuzugreifen. Tritt während der Initialisierung an irgendeiner Stelle ein Fehler auf, so wird eine C++-Exception ausgelöst, die von dem Anwenderprogramm abgefangen werden kann. Hierbei werden ein Fehlercode, eine Fehlerbeschreibung und der Dateiname, sowie die Zeilennummer, in der ein Fehler aufgetreten ist zurückgeliefert.

6.11.2 Setzen und Lesen von Parametern oder Werten

Vom Konstruktor werden bereits einige der Funktionen zum Setzen von Parametern auf dem DL300-System genutzt. Hierbei werden die Parameter geprüft, ob diese innerhalb eines sinnvollen Bereiches liegen und, sofern dies gegeben ist, in das zugehörige PCI-Register geschrieben. Neben den Funktionen zum Setzen von Parametern für die Auslese gibt es weitere, die Werte aus dem Interface abfragen können. Die Funktionen sind in Tabelle A.23 und Tabelle A.24 im Anhang zusammengefasst.

Zur Ansteuerung des DL300-Systems sind in der Programmbibliothek verschiedene Routinen implementiert, die Lese- und Schreibzugriffe auf das Cratesystem ermöglichen. Hierbei sind für den Endnutzer insbesondere die Funktionen zum Starten der verschiedenen Datenerfassungsschritte auf dem Crate wichtig. Desweiteren sind Lese- und Schreiboperationen zum Setzen und Zurücksetzen von Signalen und der verschiedenen Speicher definiert. Die Funktionen hierzu sind in Tabelle A.25 im Anhang aufgeführt.

Zusätzlich beinhaltet die Klasse weitere Lese- und Schreiboperation, sowie Hilfefunktionen, die zur klasseninternen Verwendung dienen (private Deklaration) oder bei der Entwicklung des Interfaces benötigt wurden. Letztere wurden bewusst nicht entfernt, da diese für spätere Entwicklungsschritte sinnvoll sein könnten.

Mit dieser Implementation können nun erste Testmessungen und Performanzanalysen im Labor durchgeführt werden. Das nachfolgende Kapitel enthält erste Ergebnisse, die mit dem neu erstellten Interface im Rahmen dieser Arbeit gewonnen wurden.

7 Ergebnisse

In diesem Kapitel werden Ergebnisse aus durchgeführten Messungen mit dem Flash-ADC-System beschrieben. Hierbei sollen vor allem Vergleichsmessungen mit einem integrierenden ADC, sowie Analysen zur Auslesegeschwindigkeit des implementierten Interfaces und ein Vergleich der Energieauflösung des Flash-ADC und des integrierenden ADC vorgestellt werden. Für die Messungen stand nur ein Messplatz im Labor zur Verfügung, da der Beschleuniger und das Experiment während der Anfertigung dieser Disseration vollständig umgebaut wurden. Somit konnten keine Daten unter echten Betriebsbedinungen mit einem Photonenstrahl am Strahlplatz des Crystal-Barrel-Experiments genommen werden.

7.1 Vergleich der Energieauflösung

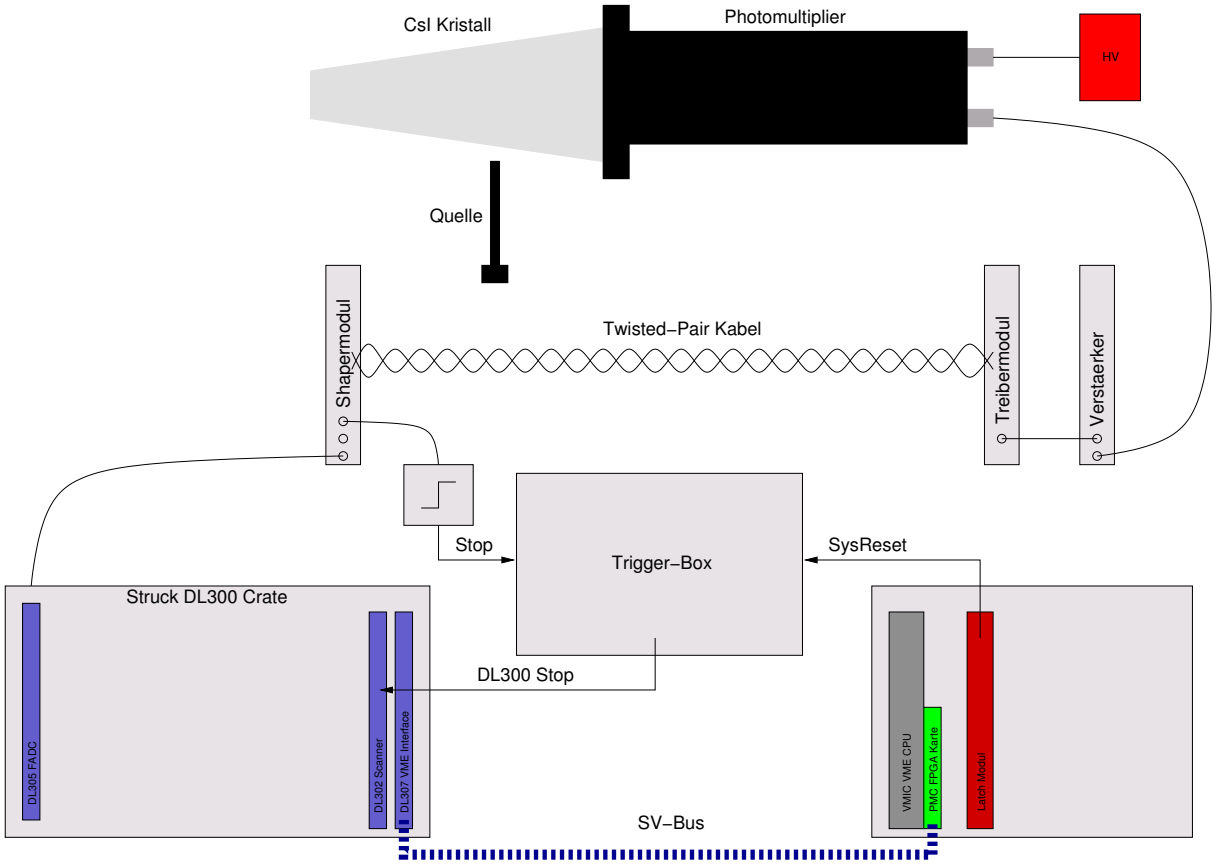


Abbildung 7.1 Versuchsaufbau zu Testmessungen mit einer radioaktiven Quelle.

Erste Messungen im Labor wurden in einem Versuchsaufbau mit einer radioaktiven Quelle durchgeführt (siehe Abbildung 7.1, hierbei wurde der β^+ Strahler ^{22}Na verwendet. Die emittierten Positronen annihilieren mit Elektronen bevorzugt unter Aussendung von zwei Gammaquanten mit je 511 keV Energie. Der Atomkern des nach dem Zerfall entstehenden ^{22}Ne ist in einem angeregten 2^+-Zustand und geht innerhalb von wenigen Pikosekunden in den Grundzustand

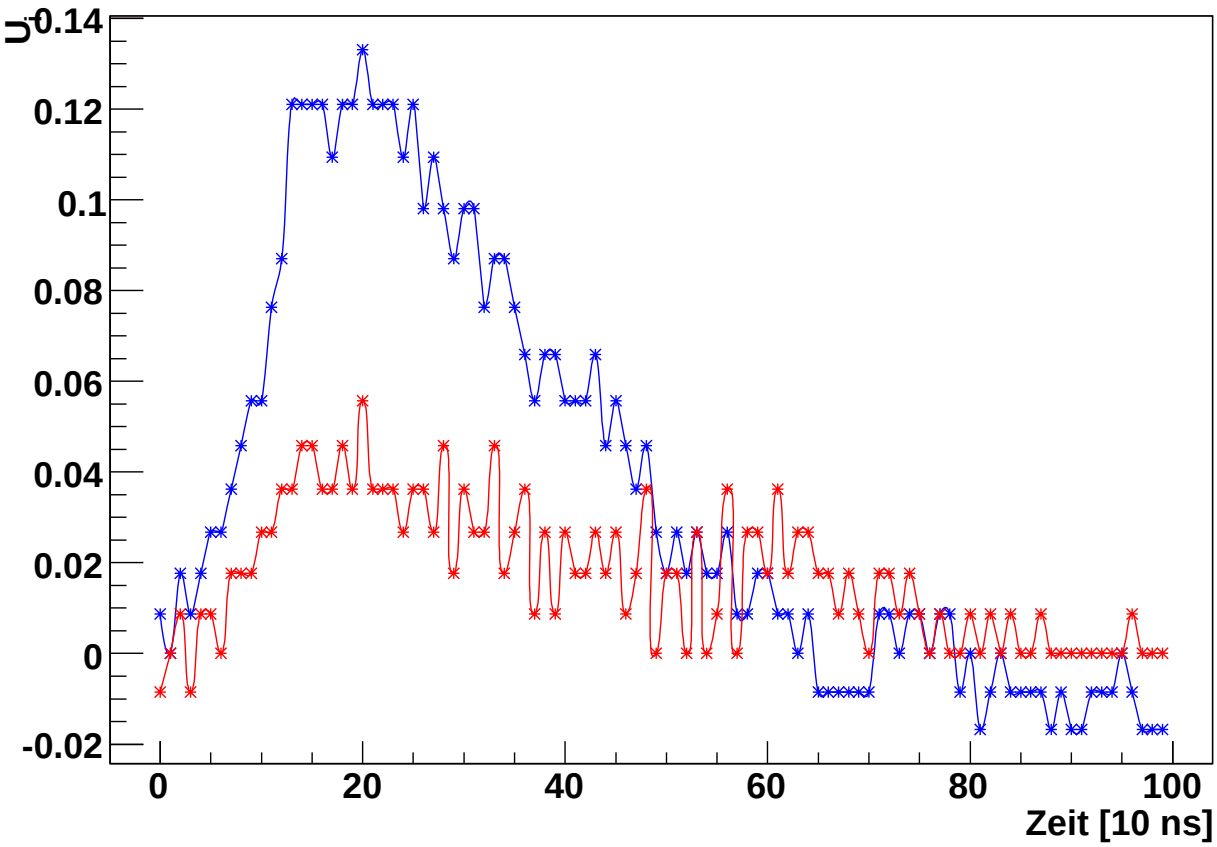


Abbildung 7.2 Signal eines aufgezeichneten Pulses im Flash-ADC bei Messung mit einer ^{22}Na-Quelle. Zu sehen ist das Signal (blau) und das abgeschwächte Signal (rot).

über. Hierbei wird ein weiteres γ-Quant emittiert, welches entsprechend der Energiedifferenz zwischen angeregtem Niveau und dem Grundzustand eine Energie von 1275 keV besitzt.

Die emittierten Gammaquanten mit Energien von 511 keV und 1275 keV werden mit einem Kristall aus dem Vorwärtswinkelbereich des Crystal-Barrel detektiert und über einen Photomultiplier in ein elektrisches Signal gewandelt. Das Signal wird über den speziell entwickelten Shaper [Hof04] geleitet. Der Energiebereich der Quelle (500keV) und der im Experiment zu erwartende Bereich (2 MeV - 3 GeV) sind zwar drastisch unterschiedlich, kann aber bei gleicher Photomultiplierverstärkung durch Verwendung eines 10-fach-Verstärkers angepasst werden, so dass die Signale im Bereich der niedrigsten im Experiment zu erwartenden Energien (einige MeV) liegen. Das verstärkte Signal wird auf den Eingang eines FADC-Moduls gegeben. Dabei wird es in zwei Anteile aufgeteilt. Der eine Anteil wird unverändert auf den rechten Kanal des FADC-Moduls gegeben, der zweite Anteil wird über eine Widerstandsschaltung abgeschwächt auf den linken Kanal gegeben. Das Flash-ADC-System wird im Common-Stop-Modus betrieben. Das Stop-Signal (Trigger) wird hierbei durch einen Diskriminator erzeugt, auf dem eine Schwelle für das eingehende Signal gesetzt werden kann. Das Stop-Signal für den FADC wird über eine feste Zeit verzögert, so dass der gesamte Puls im FADC erfasst wird. Eine Triggerlogik sorgt dafür, dass nach einem eingehenden Triggersignal kein zweites Stop-Signal generiert werden kann. Der Trigger wird erst wieder geöffnet, wenn ein von der CPU erzeugtes System-Reset-Signal generiert wurde. Die DACs des FADC wurden mittels der Funktion PedAdjust auf

den Basiswert 4 im Rahmen der Fehler justiert. Gemessen wurde mit insgesamt 100 Messpunkten bei einer Erfassungsfrequenz (Samplingfrequenz) von 100 MHz. Aufgezeichnet wurde also über einen Zeitraum von 1 μs mit einem Abstand von 10 ns zwischen den Messwerten. Das geeignet abgeschwächte Signal wurde parallel auf einen integrierenden 11-Bit-ADC vom Typ LeCroy Fera 4300 gegeben, um einen direkten Vergleich zwischen dem Flash-ADC und dem integrierenden ADC zu erhalten.

Abbildung 7.2 zeigt das Signal eines Pulses, welches bei der Messung mit einer ^{22}Na-Quelle aufgenommen wurde. Hierbei wurden die FADC-Werte auf der Y-Achse mittels der Korrekturfunktion 7.1 vom Digitalisierungswert C in linearisierte Spannungswerte U_{korr} gewandelt und um den eingestellten Pedestalwert 4 korrigiert.

$$U_{korr} = \frac{C}{64 - a \cdot C} \cdot \frac{1}{2} - \frac{4}{64 - a \cdot 4} \cdot \frac{1}{2} \tag{7.1}$$

Das Signal zeigt, dass die Aufnahme der Pulsform mit dem FADC möglich ist und dass das programmierte Interface prinzipiell funktioniert. Im Weiteren folgen Vergleiche mit einem integrierenden ADC und weitere Analysen.

7.1.1 Spektrum mit radioaktiver Quelle

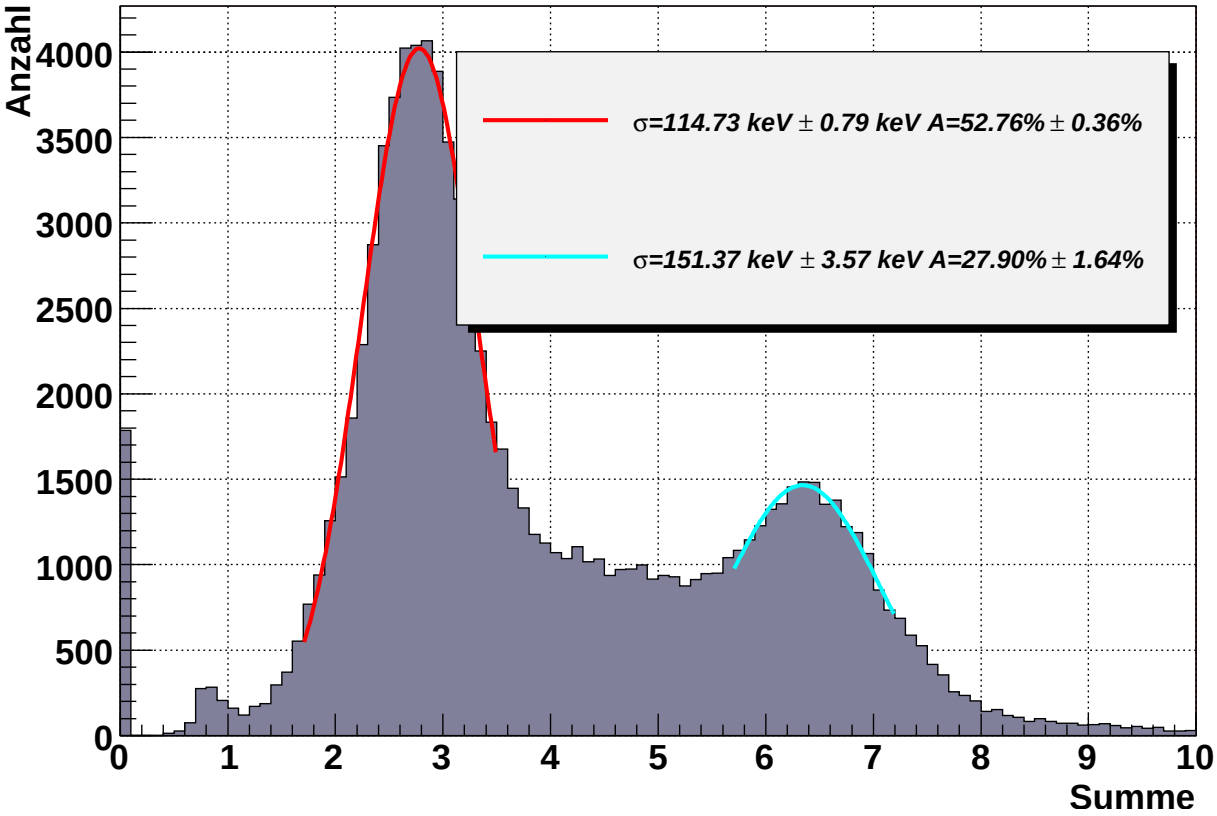


Abbildung 7.3 Spektrum der Na-22 Quelle, aufgenommen mit dem Flash-ADC, Integration in Software mittels linearisierter und um Pedestal korrigierter Messwerte.

Abbildung 7.3 zeigt das Spektrum, welches sich bei der Messung mit dem Flash-ADC-System nach 100000 detektierten Ereignissen ergibt. Im Histogramm aufgetragen ist die nachträglich

über die Software ermittelte Summe des Pulses. Hierbei wurden von den 100 aufgezeichneten Werten pro Puls nur diejenigen für die Summe genutzt, die über dem eingestellten Pedestalwert 4 liegen.

Die Werte wurden mit der Funktion 7.1 linearisiert. Im Histogramm zu sehen sind die Signale der bekannten 511 keV und der 1275 keV Linie der ^{22}Na-Quelle. Für beide Signale wurde eine Gauß-Funktion angepasst und der Mittelpunkt der beiden Linien in Einheiten der linearisierten Summe gebildet. Durch die bekannte Energie der beiden Linien und durch die Positionen der Gauß-Fits konnte dann die Breite (σ) der Linien in Einheiten von keV umgerechnet werden. Die Halbwertsbreite (HWB) entspricht dem 2,35-fachen von Sigma. Das Auflösungsvermögen ist wie in Formel 7.2 zu sehen definiert.

$$A = \frac{\Delta E}{E} = \frac{HWB}{Peakposition} \tag{7.2}$$

Für die 511 keV Linie ergibt sich mit dem Flash-ADC ein Auflösungsvermögen von A=52,76 % $\pm$ 0,36 % und für die 1275 keV Line ein Auflösungsvermögen von A=27,90 % $\pm$ 1,64 %. Im Vergleich hierzu ist in Abbildung 7.4 dasselbe Spektrum, aufgenommen mit dem integrierenden ADC, gezeigt.

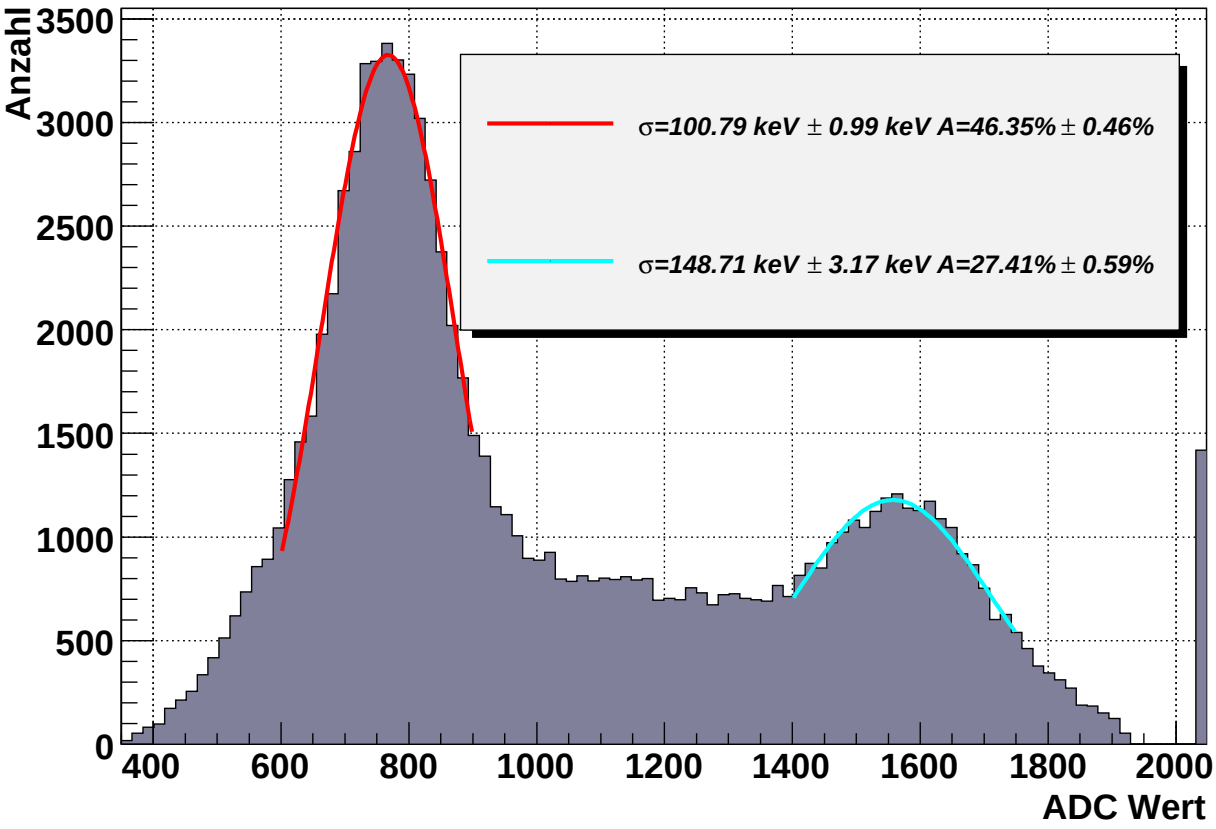


Abbildung 7.4 Spektrum der Na-22 Quelle, aufgenommen mit dem integrierenden ADC.

Das Auflösungsvermögen wird hier erneut bestimmt. Es ergibt sich zu A=46,35 % $\pm$ 0,46 % für die 511 keV Linie und A=27,41 % $\pm$ 0,59 % für die 1275 keV Linie. Das Auflösungsvermögen der beiden ADCs ist somit nahezu gleich. Lediglich im Bereich kleinerer Digitalisierungswerte

(511 keV Linie) ist das Auflösungsvermögen des FADC schlechter. Dies ist durch den Digitalisierungsfehler des FADC (siehe Abbildung 4.4) bei kleinen Messwerten bedingt. Möchte man diesen Bereich meiden, so kann beim FADC der Pedestalwert auf einen höheren Wert als 4 gesetzt werden. Dadurch wird der Digitalisierungsfehler, aufgrund der Verschiebung des Signals hin zu höheren Werten, bei kleineren Messwerten geringer. Hierbei verringert sich allerdings der Wertebereich, der für die Messung von Signalen zur Verfügung steht. Man muss im jeweiligen Anwendungsfall abwägen, ob ein geringerer Fehler bei kleinen Digitalisierungswerte oder ein größerer Messbereich gewünscht ist.

7.1.2 Vergleichsmessungen ADC-FADC

Vergleicht man nun die Messwerte des ADC mit denen des FADC, so sollte sich ein linearer Zusammenhang zwischen dem vom integrierenden ADC gelieferten Messwert und der vom FADC gebildeten und danach linearisierten Summe ergeben. In Abbildung 7.5 ist ein zweidimensionales Histogramm gezeigt, bei dem die Werte der beiden ADCs gegeneinander aufgetragen wurden. Deutlich zu sehen ist der lineare Verlauf. Im Bereich kleinerer ADC- und FADC-Werte verbreitert sich die Verteilung. Dies ist wiederum bedingt durch den Fehler bei kleinen Digitalisierungswerten beim FADC.

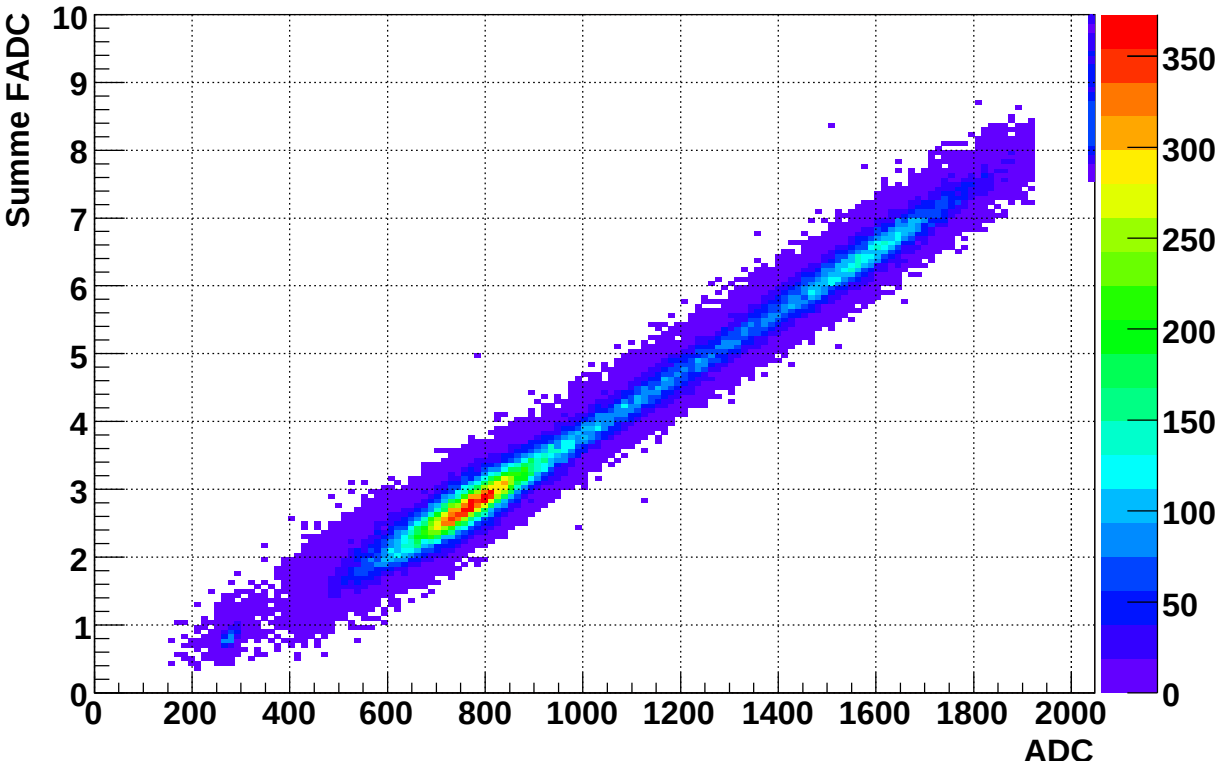


Abbildung 7.5 Vergleich linearisierte Summe FADC mit Datenwert des integrierenden ADC FERA 4300.

Für die Verwendung im Vorwärtsdetektor kann später eine Kalibrierung mittels der Fastbus-ADCs (15 Bit) stattfinden, die das Signal der Kristalle parallel digitalisieren werden.

7.1.3 Vergleich linker und rechter FADC-Kanal

Im Aufbau für den neuen Vorwärtsdetektor und für die Messung der Signale der CB-Kristalle, die über einen Photomultiplier ausgelesen werden, sollen die Signale aufgeteilt werden, wobei der zweite Anteil um einen konstanten Wert abgeschwächt wird. Dieser wird wie bereits beschrieben auf den linken FADC-Kanal gegeben, das nicht abgeschwächte Signal auf den rechten Kanal. Somit kann ein größerer Messbereich erreicht werden, als mit einem einzigen 6-Bit-FADC. Der Messbereich unterteilt sich dann in einen unteren Bereich (Low-Range) und einen oberen nicht so empfindlichen Bereich (High-Range). Für die spätere Messung am Experiment ist es wichtig, dass eine Kalibrierung der beiden Kanäle durchgeführt wird. Hierbei sollte sich im Messbereich, bei dem der rechte Kanal über den Signalverlauf nicht den maximalen Digitalisierungswert (63) erreicht, ein linearer Zusammenhang zwischen den beiden Kanälen zeigen. Überschreitet der FADC im unteren Messbereich die maximal messbare Spannung und somit den Digitalisierungswert 63, so erhält man einen Digitalisierungswert 0 (engl. Overflow). Für die Vergleichsmessung wurde ein anderer Photomultiplier verwendet, der mit einer höheren Spannung betrieben wird. Dadurch wird unter Verwendung der ^{22}Na-Quelle ein größerer Wertebereich abgedeckt, so dass der rechte Kanal den Überlauf erreicht.

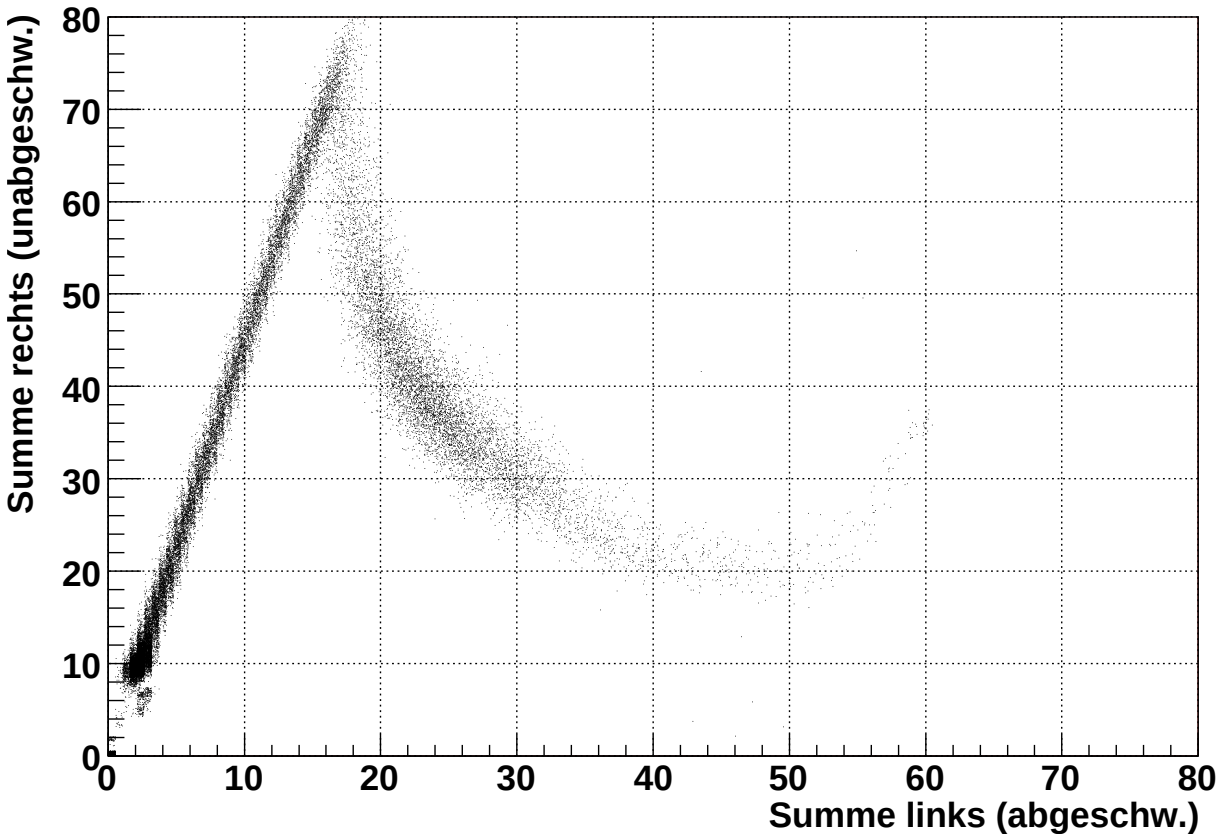


Abbildung 7.6 Vergleich linker (abgeschwächter) und rechter (unabgeschwächter) FADC-Kanal.

In Abbildung 7.6 ist in einem Histogramm der linke gegen den rechten FADC-Kanal aufgetragen. Bis zu einem Summenwert von 70-80 im rechten Kanal ist ein linearer Verlauf zu sehen. Für größere Summenwerte erreicht der rechte Kanal im Verlauf des Signals das Maximum. Hierbei

ergibt sich ein Digitalisierungswert von 0 (Überlauf). Je größer das Signal wird, desto größer wird der Signalbereich, der sich im Überlauf befindet. Die Summe im rechten Kanal wird aufgrund der vermehrt auftretenden Digitalisierungswerte 0 effektiv kleiner. Die Summe im linken Kanal steigt hingegen weiter an. Hierdurch erklärt sich der Bereich, der rechts neben dem linearen Bereich zu sehen ist. Wird das Signal noch größer, so befinden sich nur noch die ansteigende und die abfallende Flanke des Pulses außerhalb des Überlaufbereiches. Der Anteil im Überlauf nimmt bei zunehmender Signalhöhe im Verhältnis dadurch weniger stark zu. Deswegen ist im Verlauf des Graphen ein Bogen zu sehen, bei dem bei steigenden Werten der linken Summe die rechte Summe weniger stark abfällt und später sogar wieder ansteigt. Dieses Verhalten der FADCs kann im Experiment durch eine geeignete Kalibrationsfunktion berücksichtigt werden. Das Verhältnis zwischen unabgeschwächtem und abgeschwächtem Kanal kann über einen Fit ermittelt werden und beträgt mit dem für die Testmessungen verwendeten Abschwächer etwa 4,5:1.

7.2 Messungen mit kosmischer Strahlung

Neben der in Kapitel 7.1 beschriebenen Messung mit einer radioaktiven Quelle, wurden auch Messungen mit Sekundärteilchen der kosmischer Strahlung durchgeführt. Kosmische Strahlung besteht aus hochenergetischen Protonen und leichten Kernen, die in der Atmosphäre mit dem vorhandenen Stickstoff und Sauerstoff in Wechselwirkung treten und als Sekundärteilchen Myonen (sogenannte harte Komponente) und Elektronen und Positronen (weiche Komponente) erzeugen. Die mittlere Energie der Myonen beträgt auf Meeresniveau etwa 4 GeV. Der Fluss der Myonen beträgt 130 $\frac{1}{m^2 s}$. Die auf den Kristall auftreffenden Myonen deponieren nur einen Teil ihrer Energie im Kristall und können nicht gestoppt werden, da die deponierte Energiemenge pro cm CsI(Tl) 5,6 MeV beträgt.

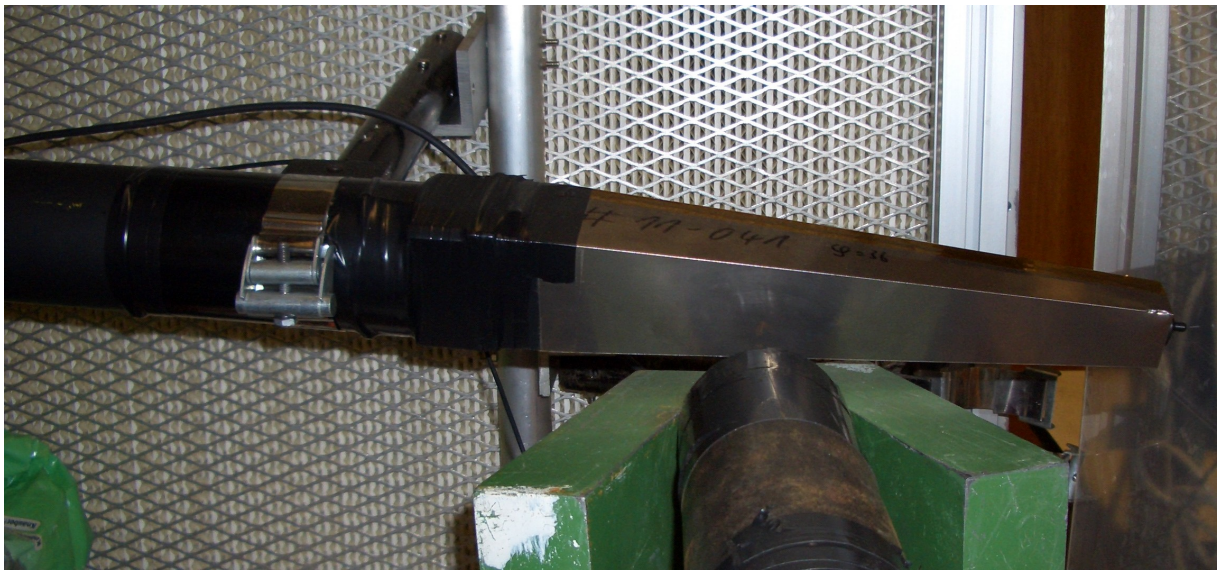


Abbildung 7.7 Kristall in Titanhülle mit Photomultiplierauslese. Unter dem Kristall befindet sich ein Plastikszintillator, der ebenfalls über einen Photomultiplier ausgelesen wird.

Der Versuchsaufbau ist in Abbildung 7.8 gezeigt und gleicht dem Aufbau zur Messung mit der

^{22}Na-Quelle. Aufgrund der höheren Energie der kosmischen Teilchen konnte auf den zuvor eingesetzten 10-fach-Verstärker verzichtet werden. Zusätzlich wurde ein schneller Plastikszintillator verwendet, der über einen Diskriminator das Start-Signal für einen 8-Kanal Zeit-Digital-Konverter (TDC[66]) vom Typ LeCroy 2228 liefert und parallel dazu das Stop-Signal für die Auslese des DL300-Systems generiert. Der Plastikszintillator ist unter dem Kristall montiert (wie in Abbildung 7.7 zu sehen), so dass alle durch den Plastikszintillator hindurchtretenden kosmischen Teilchen auch den Kristall durchqueren sollten. Das Stop-Signal für den TDC liefert der CsI-Kristall. Der Ausgang des Photomultipliers ist über einen weiteren Diskriminator mit dem Stop-Eingang des TDCs verbunden. Hierdurch wird es möglich, die Zeitbestimmung relativ zum schnellen Signal des Plastikszintillators für den Anstieg des CsI-Signals durchzuführen. Die Zeitbestimmung kann mit dem beschriebenen Aufbau durch den TDC erfolgen. Zusätzlich soll gezeigt werden, dass es möglich ist, eine Zeitinformation aus dem Signalverlauf zu erhalten, welche mit dem FADC aufgenommen wurde.

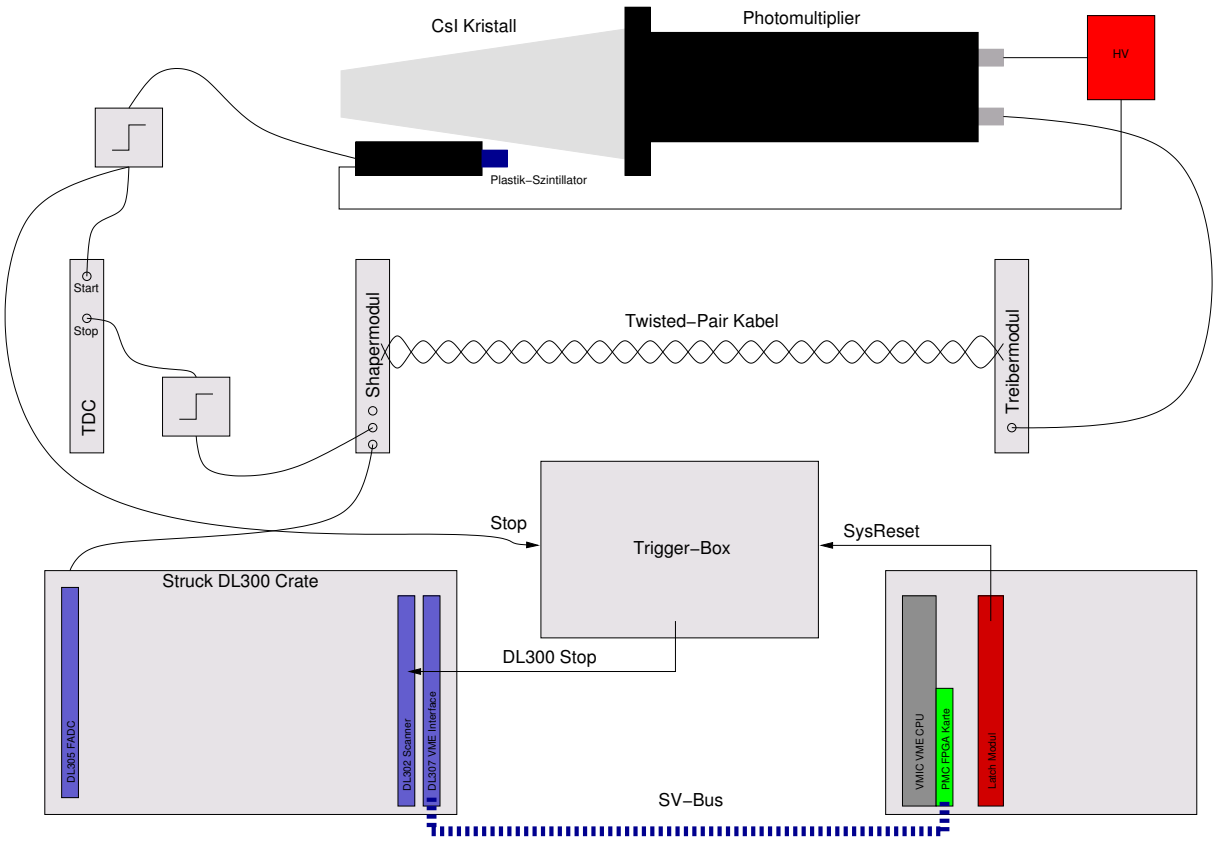


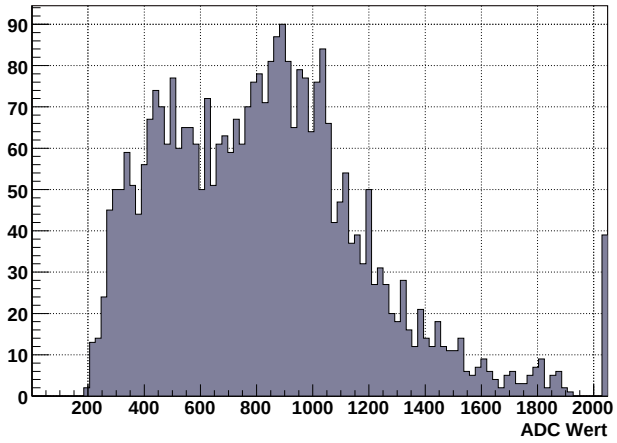
Abbildung 7.8 Versuchsaufbau zu Testmessungen mit kosmischen Teilchen.

7.2.1 Cosmic-Spektrum

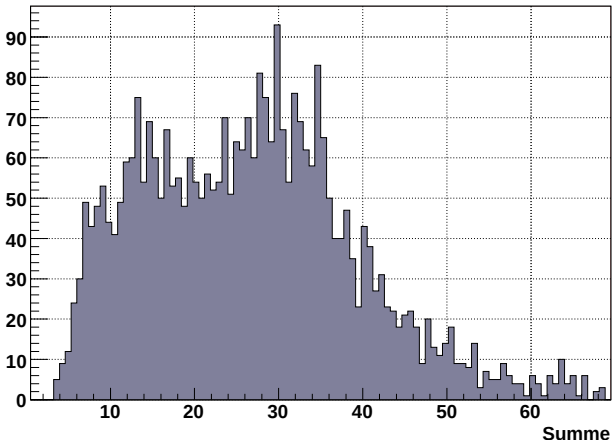
Das sich ergebende Spektrum bei Messung mit dem ADC und FADC ist in den Histogrammen in Abbildung 7.9 wiedergegeben. Hierbei wurden zwei Selektions-Schnitte angewandt, die die Datenmenge einschränken. Es wurde verlangt, dass die gemessenen Werte im integrierenden

[66] engl. TDC - time to digital converter - Zeit-Digital-Konverter

ADC größer als 200 sein sollen. Dieser Schnitt dient dazu, die relevanten Daten des Spektrums besser darzustellen (Unterdrückung von Nulleinträgen). Desweiteren wurde verlangt, dass das Maximum der Einzelwerte beim Abtasten des Signals im rechten, nicht abgeschwächten FADC-Kanal, kleiner als der Digitalisierungswert 62 sein soll. Hierdurch wird verhindert, dass bei Signalen die höher als der maximale Wertebereich des 6-Bit FADCs sind, verfälschte Summen in das Spektrum eingehen. Die Spektren bleiben somit vergleichbar. Für eine spätere Verwendung am Experiment ist hier natürlich zwingend eine Energieeichung zwischen den beiden FADC-Kanälen (abgeschwächt und nicht-abgeschwächt) mit der eingesetzten Abschwächerschaltung erforderlich. Das Spektrum, welches sich mit dem DL300-System ergibt, ist in Abbildung 7.9 zu sehen. Die beiden Spektren sind nahezu identisch.



Gemessen mit einem integrierenden ADC - Digitalisierungswerte größer 200 dargestellt.



Gemessen mit dem Flash-ADC-System - Summenwerte größer 2 dargestellt.

Abbildung 7.9 Cosmic-Spektrum

Dies bestätigt, dass beide Messmethoden, was das Energieintegral anbelangt, im zu erwarteten Energiebereich äquivalent sind.

7.2.2 Analyse der Pulsform und Vergleich mit TDC-Daten

Interessant ist es, die vom konventionellen TDC gemessenen Zeiten mit denen zu vergleichen, die sich mit der aufgezeichneten Signalform ermitteln lassen. Mit einer guten Zeitauflösung lässt sich im Experiment bestimmen, ob ein Photon zum aktuellen Ereignis dazugehört oder zeitlich versetzt auftritt und nicht berücksichtigt werden sollte. Zunächst wurde das TDC-Signal analysiert. Das sich ergebende TDC-Spektrum ist in Abbildung 7.10 zu sehen. Der TDC arbeitet mit einer Zeitauflösung von 150 ps je Kanal. Die gezeigten Werte ergeben sich aus den Digitalisierungswerten multipliziert mit einem Faktor 0,15 ns. An das Signal wurde eine Gauß-Funktion angepasst (engl. Fit). Der Gauß-Fit liefert eine Standardabweichung von σ=5,39 ns $\pm$ 0,18 ns. Diese relativ große Breite des prompten Zeitpeaks ist im Wesentlichen auf die langsame intrinsische Anstiegszeit der Lichtemission der CsI-Kristalle (ca. 100 ns) und auf die Verwendung eines Leading-Edge-Diskriminators zurückzuführen.

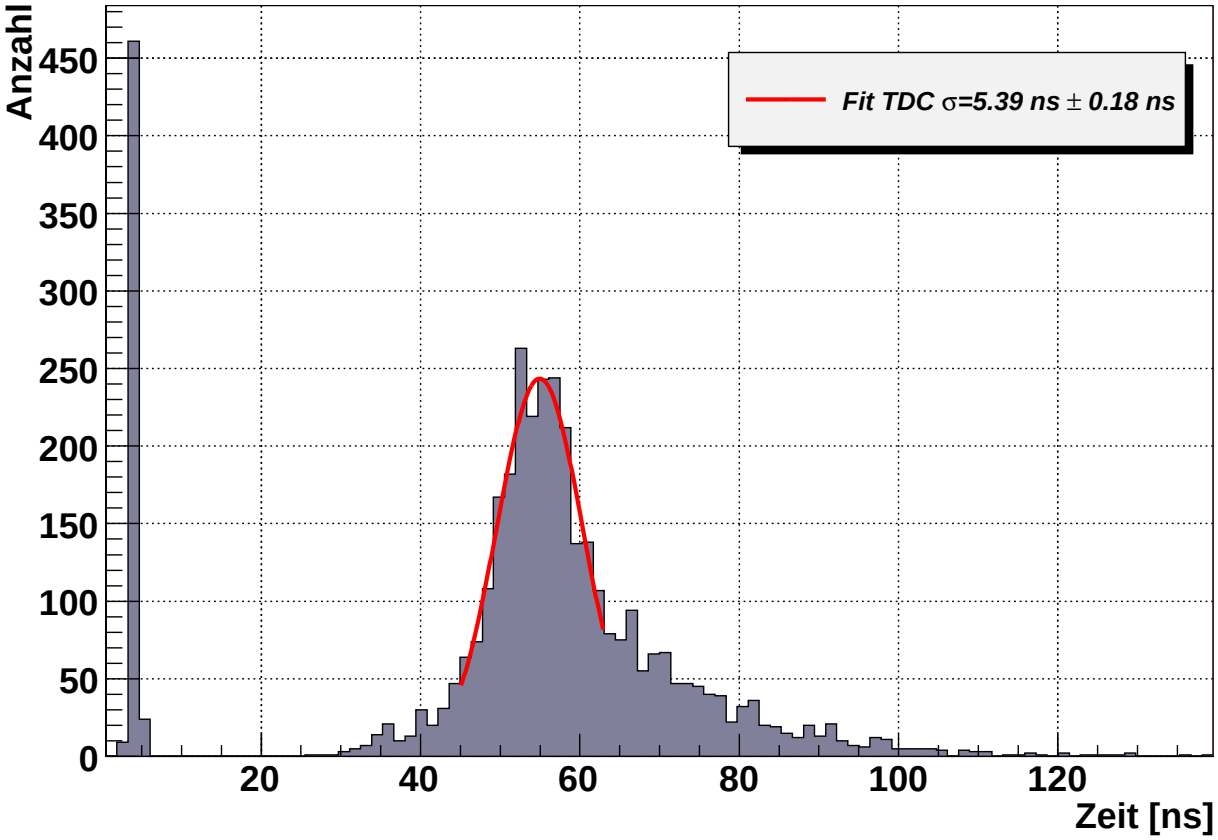


Abbildung 7.10 TDC-Spektrum der Cosmic-Daten.

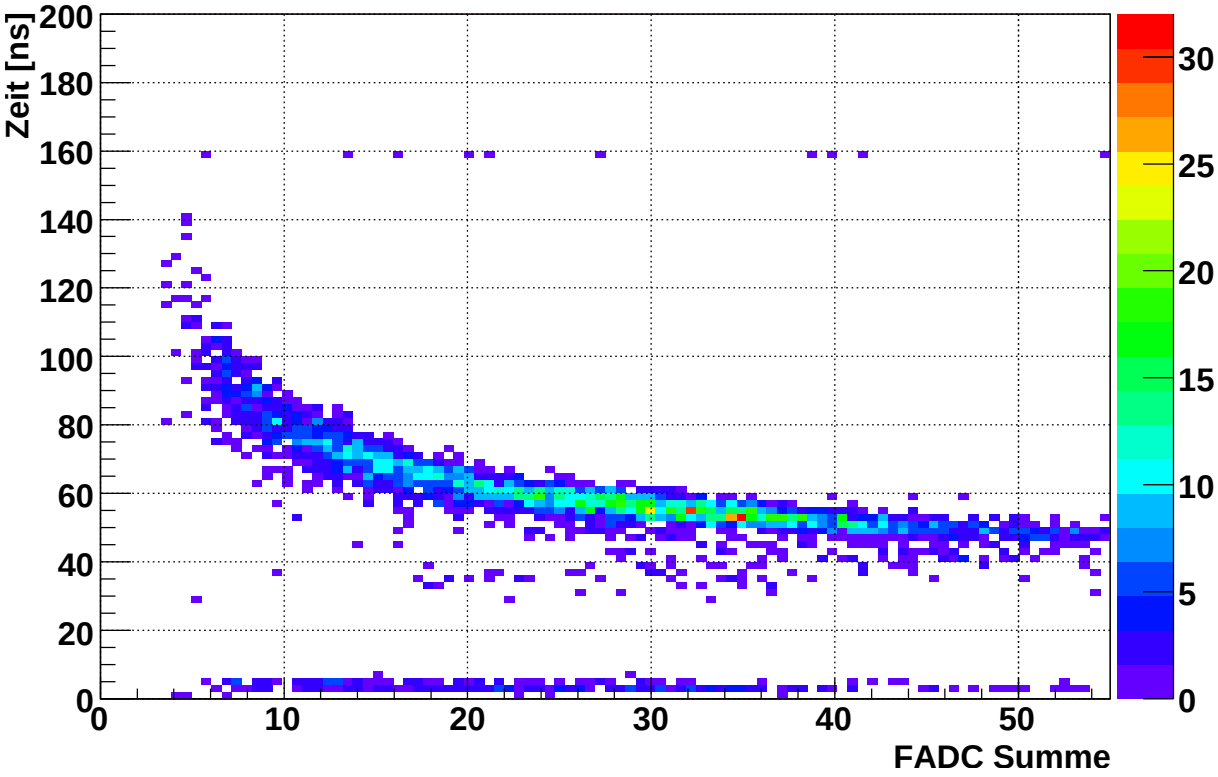


Abbildung 7.11 TDC-Spektrum der Cosmic-Daten, aufgetragen gegen das Integral des Signals.

Trägt man nun, wie in Abbildung 7.11 zu sehen, die TDC-Information gegen das Integral des Signals auf, so erkennt man eine deutliche Abhängigkeit der Zeitinformation von der ermittelten FADC-Summe, die der Energie des kosmischen Teilchens entspricht. Für kleinere Signale ist die gemessene Zeit im TDC größer. Dies ist durch die Erzeugung des Stop-Signals über den Leading-Edge-Diskriminator und die vergleichsweise langsame Anstiegszeit von Signalen im CsI-Kristall bedingt. Ein kleines Signal überschreitet die eingestellte Diskriminatorschwelle somit zu einem späteren Zeitpunkt.

Um aus der Signalform eine Zeitinformation zu erhalten, muss der Anstieg des Signals genau bekannt sein und der Schnittpunkt mit der Zeitachse bestimmt werden, da bei einer Abtastrate von 100 MHz bei direkter Verwendung der FADC-Daten nur eine Genauigkeit von $\pm$10 ns erreicht werden kann. Die aus dem Flash-ADC auszulesenden Daten werden anhand der gefundenen Hit-Position zeitlich aufsteigend geordnet. Hierbei werden eine gewisse vom Nutzer zu definierende Anzahl an Speicherstellen vor der gefundenen Hit-Position ausgelesen, um auch den Anstieg des Signals vollständig zu erfassen. Die Hit-Position und die Stop-Position werden bei der Auslese mitgespeichert. Mit diesen Daten lässt sich dann eine relative Position des gefundenen Ansprechers zur Stop-Position bzw. zum Stop-Zeitpunkt errechnen. Diese relative Hit-Position errechnet sich aus dem Abstand zur Stop-Position, wobei die Speichergrenzen der DL305- oder DL310-Module berücksichtigt werden müssen. Die Position der gefundenen Ansprecher relativ zum Stop-Zeitpunkt sind in Abbildung 7.12 gezeigt. Die Breite der Verteilung beträgt σ= 8,73 ns $\pm$ 0,10 ns.

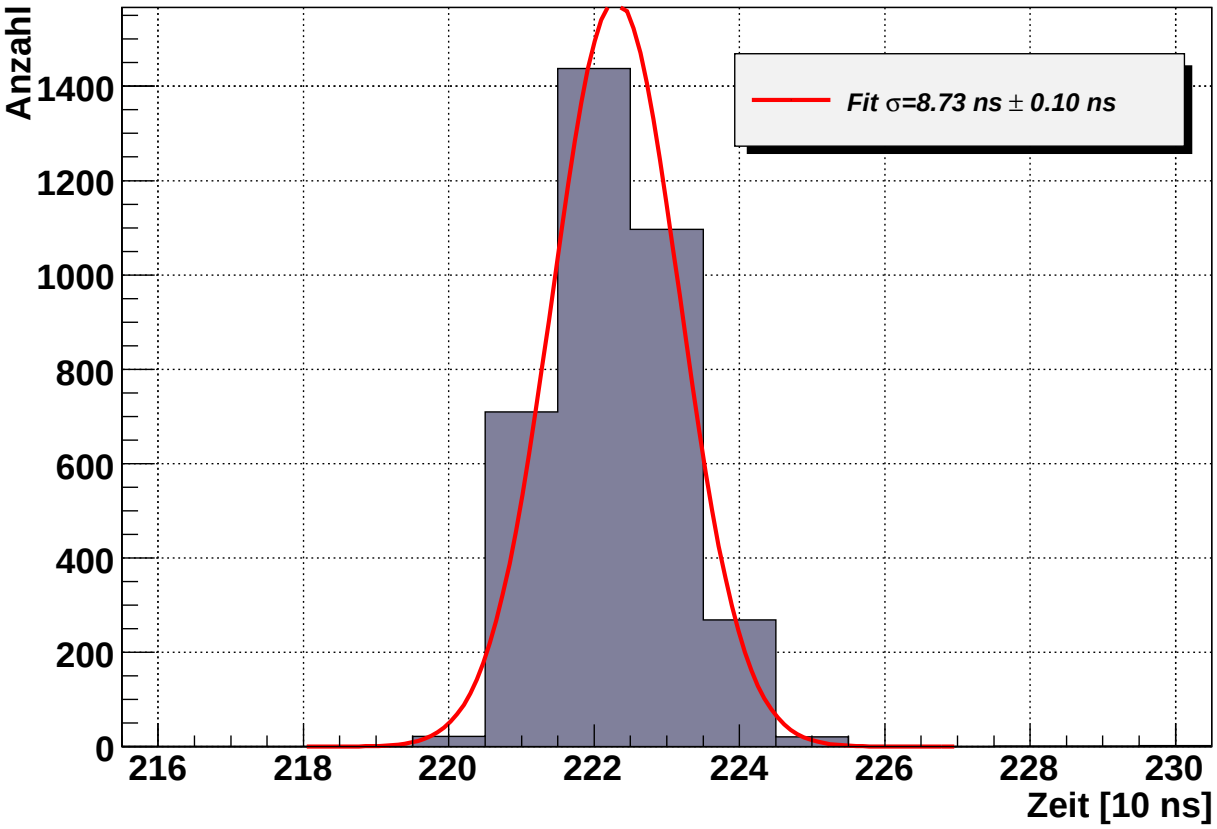
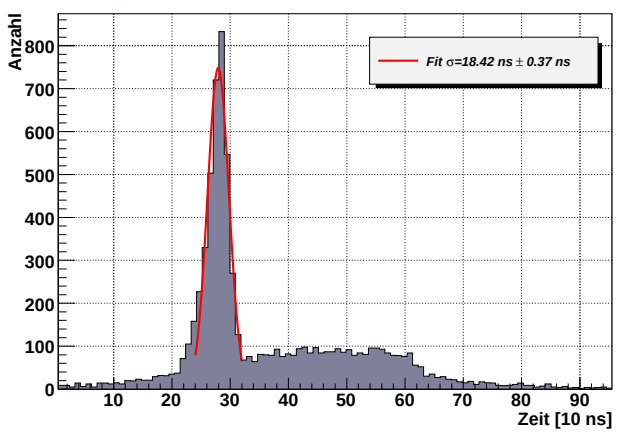


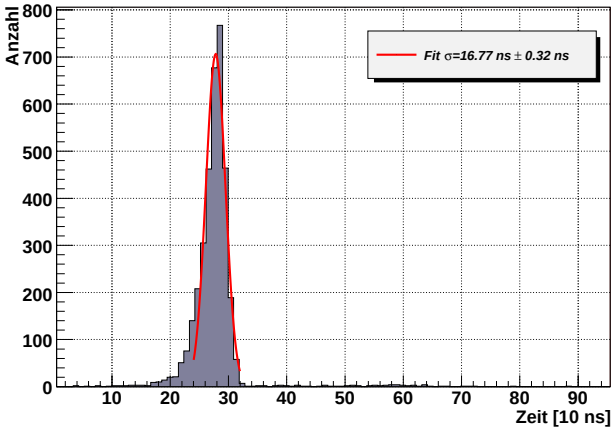
Abbildung 7.12 Relative Hitposition zum Stop-Signal.

Für die Bestimmung des Schnittpunktes mit der Zeitachse lassen sich verschiedene Verfahren anwenden, die unterschiedlich gute Genauigkeiten liefern. Die Daten lieferte eine Messung mit kosmischer Strahlung. Die verwendeten Daten wurden zuvor linearisiert und um den eingestellten Pedestalwert korrigiert. Dem ersten Wert jedes ausgelesenen Pulses wird die X-Position 0 zugeordnet. Um hieraus eine Zeit relativ zum Stop-Signal zu erhalten, wurde die relative Hitposition hinzuaddiert. Da die relativen Hitpositionen im verwendeten Aufbau bei Werten oberhalb von 200 liegen, wurde der willkürliche Wert 200 abgezogen, um kleinere Datenwerte zu erhalten.

Es wurden Zeitanalysen mit zwei verschiedenen Methoden durchgeführt. Im Folgenden sollen diese zwei Verfahren vorgestellt werden, welche sich im Hinblick auf die Güte der erzielten Ergebnisse, sowie auf den Rechenaufwand des analysierenden PCs unterscheiden. Für das erste Verfahren wurde die Tatsache genutzt, dass der Anstieg eines Signals im Bereich vor dem Maximum nahezu linear verläuft. An den Anstieg des Signals wurde dann eine Gerade angepasst. Darauf folgend wurde der Schnittpunkt dieser Geraden mit der Zeitachse bestimmt.



Signal ohne Selektions-Schnitt.



Signal mit Selektions-Schnitt (Summe FADC > 2).

Abbildung 7.13 Zeitinformation anhand eines Verfahrens mit Anpassung einer Geraden zur Bestimmung des Schnittpunktes mit der Zeitachse eines Signals unter Verwendung des Cosmic-Datensatzes.

Bei der Anpassung werden Hilfswerte genutzt, die aus dem Signalverlauf bestimmt werden. Zum einen wurde eine Start-Position ermittelt, die so festgelegt wurde, dass das Signal den Wert des eingestellten Pedestals um den Digitalisierungswert 2 überschreiten musste (Datenwert>PED+2). Desweiteren wurde das Maximum des Pulses über den Signalverlauf ermittelt, wobei bei mehreren identischen Maxima-Werten der erste vorkommende Wert für die Bestimmung der Maximum-Position genutzt wurde. Von der Position des Maximums wurde der Zahlenwert 10 abgezogen, der sich bei den Stichproben als sinnvoll ergab, so dass der nichtlineare Bereich vor Erreichen des Maximalwertes nicht für die Anpassung der Geraden verwendet wird maxpos-10). Der Fit wurde mittels ROOT [ROO] und der darin verwendeten Fitroutinen durchgeführt. Zum Schluss wurde noch der Schnittpunkt der Geraden mit der Zeitachse bestimmt. Der sich ergebende Datenwert wurde in eine Zeit relativ zum Stop-Signal umgerechnet und

in einem Histogramm aufgetragen. Das sich ergebende Signal ist in Abbildung 7.13 links zu sehen.

An den im Histogramm zu sehenden Prompt-Peak wurde eine Gauß-Funktion angepasst. Es ergibt sich eine Standardabweichung von σ=16,77 ns $\pm$ 0,32 ns. Im Vergleich zu den TDC-Daten ergibt sich eine schlechtere Zeitauflösung, die primär durch die Abtastfrequenz von 100 MHz bedingt ist. Der Bereich rechts und links neben dem Prompt-Peak enthält Einträge, die durch sehr kleine Pulse mit einer kleinen FADC-Summe erzeugt werden. Diese verschwinden, sofern eine minimale Schwelle für die Energiesumme verlangt wird (Abbildung 7.13 rechts). In Abbildung 7.14 ist die FADC-Summe gegen die ermittelte Zeitinformation aufgetragen.

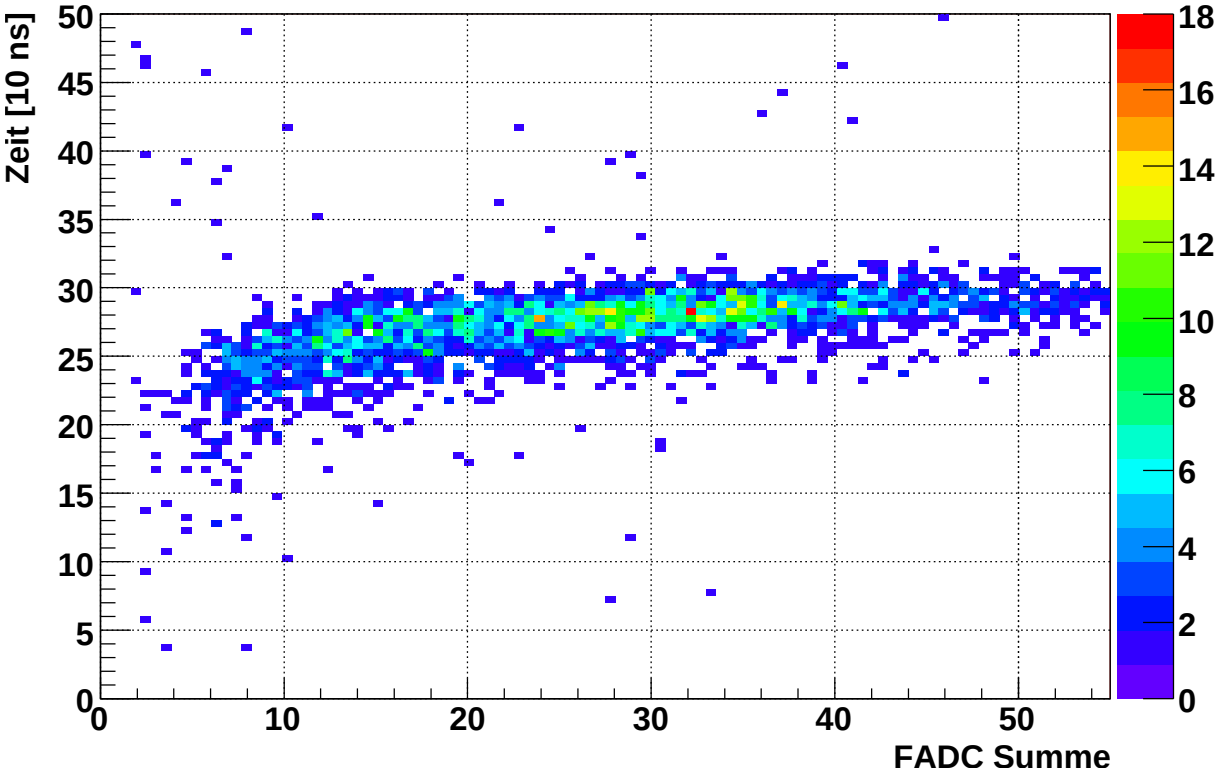
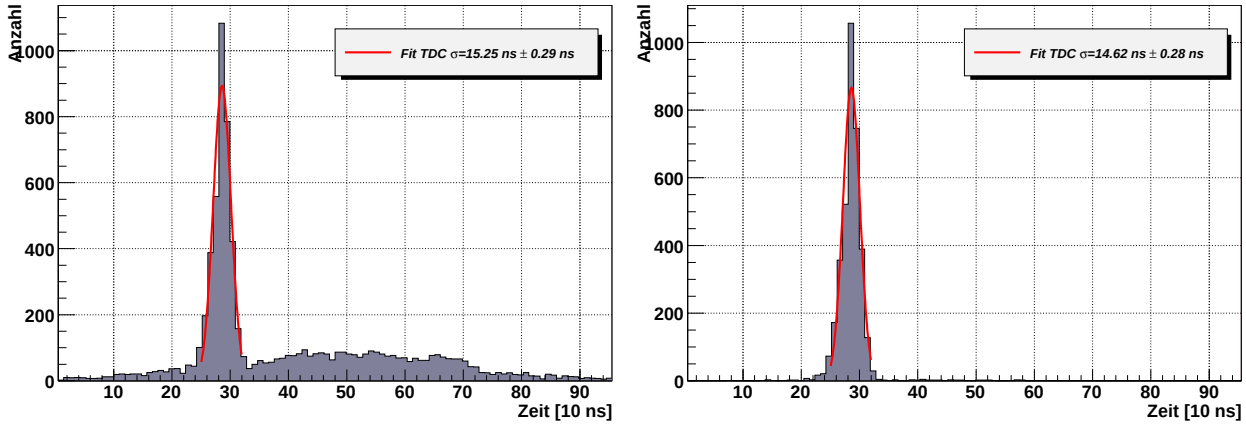


Abbildung 7.14 Zeitinformation anhand eines Verfahrens mit Anpassung einer Geraden zur Bestimmung des Schnittpunktes mit der Zeitachse eines Signals, aufgetragen gegen das Integral des Signals. Zur besseren Darstellung sind nur Werte aufgetragen, bei denen die FADC-Summe größer dem Wert 2 ist.

Die Daten zeigen hier, im Gegensatz zur Messung mit dem TDC, nahezu keine Abhängigkeit der Zeitinformation vom ermittelten Integral über das Signal. Da die Bestimmung des Startzeitpunktes nicht mittels des Leading-Edge-Diskriminators stattfindet, kann prinzipbedingt eine Verschiebung bei kleinen Signalen bzw. Signalsummen nicht auftreten. Für sehr kleine FADC-Summen zeigt sich, dass die Bestimmung des Schnittpunktes mit der Zeitachse geringfügig kleinere Zeiten liefert. Diese Abhängigkeit ist im gezeigten zweidimensionalen Histogramm zu sehen. Das Verfahren der linearen Anpassung wurde gewählt, da es sich (eventuell in vereinfachter Form) relativ leicht auf einem FPGA implementieren lässt. Die Analyse des Signals könnte während der Auslese durchgeführt werden, so dass die Zeitinformation im SSRAM des FPGAs zur Verfügung steht.

Alternativ wurde ein rechenzeitintensiveres Verfahren getestet, bei dem die gesamte Signalinformation verarbeitet wird. Hierbei wurde das Signal durch ein Polynom 5. Grades gefittet[67]. Die Nullstelle im Startbereich des Signals wurde über das Newtonverfahren ermittelt. Der Fit der Daten wurde im Bereich von der ermittelten Start-Position bis zur Position 95 durchgeführt. Das resultierende Histogramm nach Verrechnung der relativen Hitposition und Abzug des willkürlich gewählten Wertes 200 ist in Abbildung 7.15 links zu sehen. Rechts in der Abbildung ist erneut das Histogramm nach einem Selektions-Schnitt zu sehen, bei dem eine Summe im FADC größer 2 gefordert wurde. Es ergibt sich eine Standardabweichung von σ=14,62 ns $\pm$ 0,28 ns.



Signal ohne Selektions-Schnitt

Signal mit Selektions-Schnitt (Summe FADC > 2)

Abbildung 7.15 Zeitinformation mittels Fit eines Polynoms 5. Grades an die Signalform.

In Abbildung 7.16 ist wiederum die ermittelte Zeitinformation gegen das Integral des Signals aufgetragen. Beide Verfahren eignen sich prinzipiell für die Ermittlung einer Zeitinformation aus der Pulsform des Signals. Es werden Ergebnisse erzielt, die verglichen mit einer echten TDC-Information schlechter sind. Die Analysen zeigen, mit welcher Genauigkeit eine absolute Zeitinformation mit dem FADC-System zu ermitteln ist. Die Zeitauflösung ist begrenzt durch die maximale Samplingrate von 100 MHz, die einem Abstand von 10 ns je erfasstem Datenwert enspricht. Für eine Bestimmung von relativen Zeiten verschiedener Kristalle innerhalb eines Ereignisses hat dies jedoch keinen Einfluss. Die relative Genauigkeit kann deutlich besser sein, als die hier ermittelte für absolute Zeiten.

Für die Bestimmung einer absoluten Zeitinformation bereits während der Auslese, lässt sich ein vereinfachtes Verfahren anwenden, welches auf dem FPGA implementiert werden kann. Dies hat den Vorteil, das innerhalb der Auslesezeit eines Ereignisses diese Information bereits bestimmt und in einer Triggerentscheidung genutzt werden könnte.

[67] Getestet wurden auch Polynome höherer oder niedriger Ordnung. Für die vorliegenden Daten ergab sich mit einem Polynom 5. Grades die geringste Standardabweichung des Prompt-Peaks.

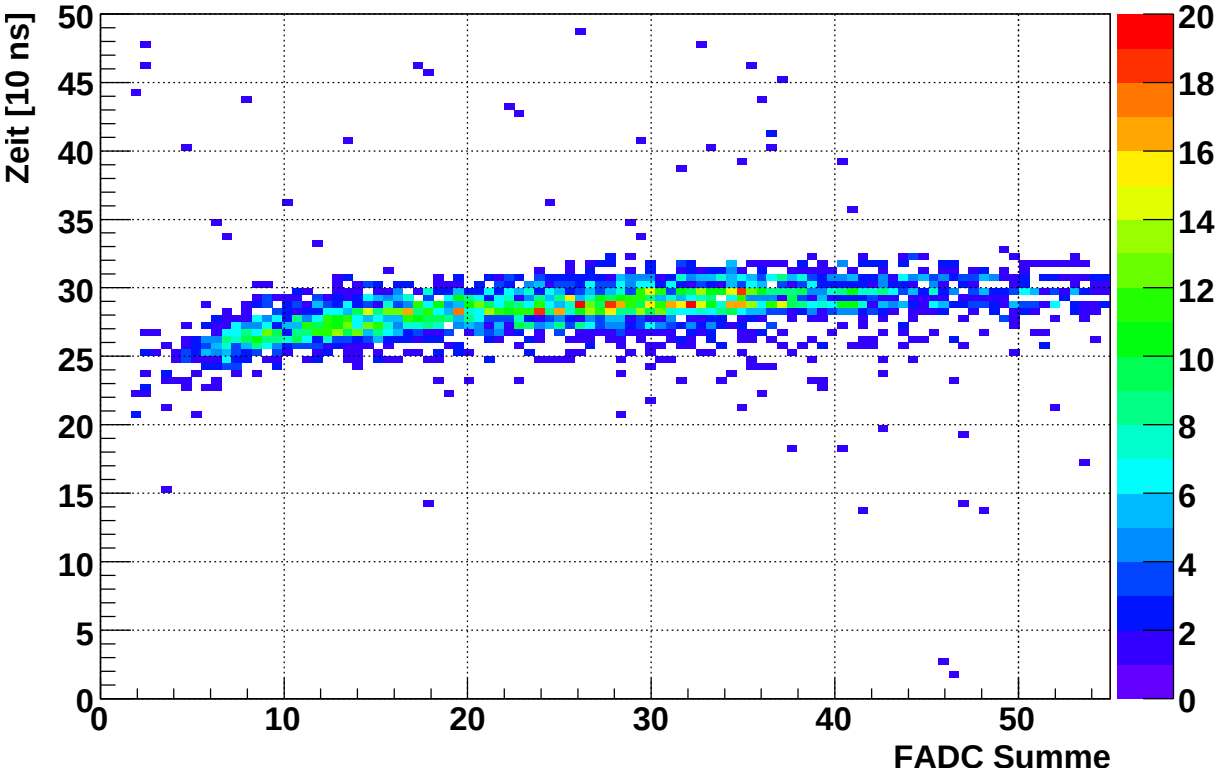


Abbildung 7.16 Zeitinformation mittels Fit eines Polynoms 5. Grades an die gesamte Signalform gegen das Integral des Signals. Zur besseren Darstellung sind nur Werte aufgetragen, bei denen die FADC-Summe größer dem Wert 2 ist.

7.3 Auslesegeschwindigkeit des FPGA-Interfaces

Die Auslesegeschwindigkeit des programmierten FPGA-Interfaces ist relevant für die später zu erwartende Datenrate bei der Verwendung des DL300-Systems im Betrieb am Experiment. In den folgenden Unterkapiteln sollen deshalb Ergebnisse vorgestellt werden, die eine Abschätzung der maximalen Datenrate im Experiment ermöglichen. Desweiteren werden die Abhängigkeiten von den, die Auslesezeit beeinflussenden Parametern, analysiert und erläutert. Das Zeitverhalten des Interfaces wurde dabei mittels eines Logik-Analysierers (Logic-Analyzer) vom Typ dli PL-1000E ermittelt, der am Buskabel zwischen der Personality-Card DL307V und dem DL307-Interfacemodul angeschlossen wurde, um den zeitlichen Verlauf der Datenleitungen zu erfassen. Hierbei wurden die beiden zur Fehlerdiagnose verwendeten Leitungen, die in Kapitel 6.9 beschrieben wurden verwendet, um die Zeitdauern zu ermitteln, die von den eingesetzten Statemaschinen benötigt werden. Dazu wurde das Signal **busy** der Statemaschinen über die Fehlerdiagnoseleitungen ausgegeben, so dass anhand der Signale die benötigte Zeit ermittelt werden konnte. Der Aufbau ist in Abbildung 7.17 zu sehen. Der Logik-Analysierer arbeitet mit einer maximalen Erfassungsfrequenz von 100 MHz, so dass alle 10 ns ein Datenwert digitalisiert wird.

7.3.1 Analyse einer Leseoperation

Zunächst soll das Zeitverhalten einer Leseoperation auf dem DL300-System anhand eines

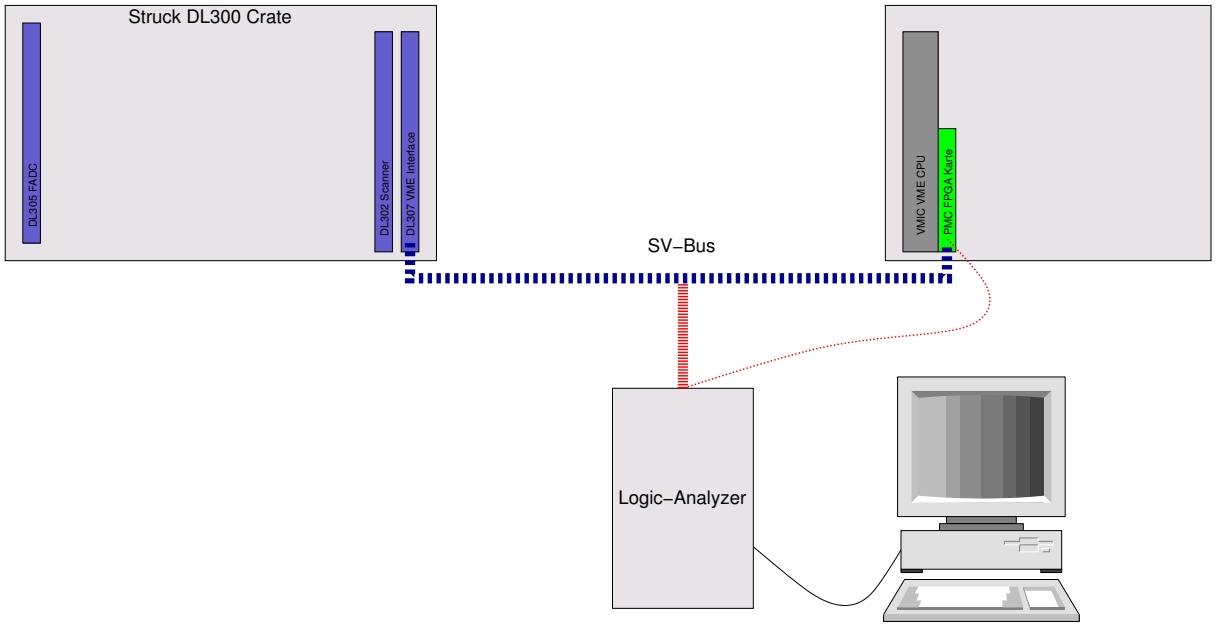


Abbildung 7.17 Versuchsaufbau zur Messung der Auslesegeschwindigkeit.

Lesezugriffs auf das Statusregister 0x406 der DL307-Interfacekarte diskutiert werden. In Abbildung 7.18 ist der zeitliche Verlauf der Signale zu sehen. Die beiden Markierungen C und S zeigen den Bereich, in dem die Fehlerausgabeleitung (C0) den logischen Wert 1 annimmt. Die Fehlerausgabeleitung wurde so gesetzt, dass diese dem kombinierten Busy-Signal des Interfaces **dl_busy** entspricht. Der gesamte Lesezyklus benötigt etwa 360 ns.

Gezeigt sind desweiteren die Adressleitungen A0-A10 (auf Anschluss F5-F7 und D0-D7), die Datenleitungen D0-D15 (auf Anschluss B0-B7 und A0-A7), sowie die Signale RW, ADS und DTACK. Die drei Signale RW, ADS und DTACK sind hierbei auf dem Bus logisch invertiert. Die Adressleitungen und die RW-Leitung ändern sich während der gesamten gezeigten Zeitdauer nicht. Dies ist dadurch zu erklären, dass vor dem gezeigten Zeitintervall bereits ein Lesezugriff (R/W logisch 1) auf dieselbe Adresse stattgefunden hat. Nach einer Zeitdauer von ca. 30 ns wechselt das ADS-Signal von logisch 1 auf logisch 0. Dies entspricht zwei Taktzyklen des Zeitgebers clk2x, der mit der doppelten PCI-Bus Geschwindigkeit von 66,66 MHz arbeitet (15 ns pro Taktzyklus). Dies ist dadurch zu erklären, dass bei einem Lesezugriff zunächst das Startsignal für die Statemaschine dl_read generiert wird. Daraufhin wird das Busy-Signal der Statemaschine auf 1 gesetzt und damit auch das globale Busy-Signal, welches auf der Leitung C0 zu sehen ist. Die Statemaschine dl_read setzt nun das Signal **dl300_cmd**, welches von der Statemaschine **dl_cycle** im nächsten Taktzyklus als Start-Signal interpretiert wird. Im nachfolgenden zweiten Taktzyklus wechselt dann die Statemaschine **dl_cycle** vom Zustand **st_idle** in den Zustand **st_transfer**, wobei das ADS-Signal gesetzt wird. Die folgende Zeitdauer von etwa 170 ns zwischen dem Setzen des ADS-Signals durch die Statemaschine und dem Setzen der DTACK-Leitung durch die DL307-Interfacekarte wird alleine durch das DL300-System bestimmt. Danach werden 80 ns benötigt, bis die ADS-Leitung durch die Statemaschine zurückgenommen wird. Dies ist dadurch bedingt, dass die Signale der DTACK-Leitung vor dem Erreichen der FPGA-Karte zunächst über eine Logik auf der DL307V-Personality-Karte geführt

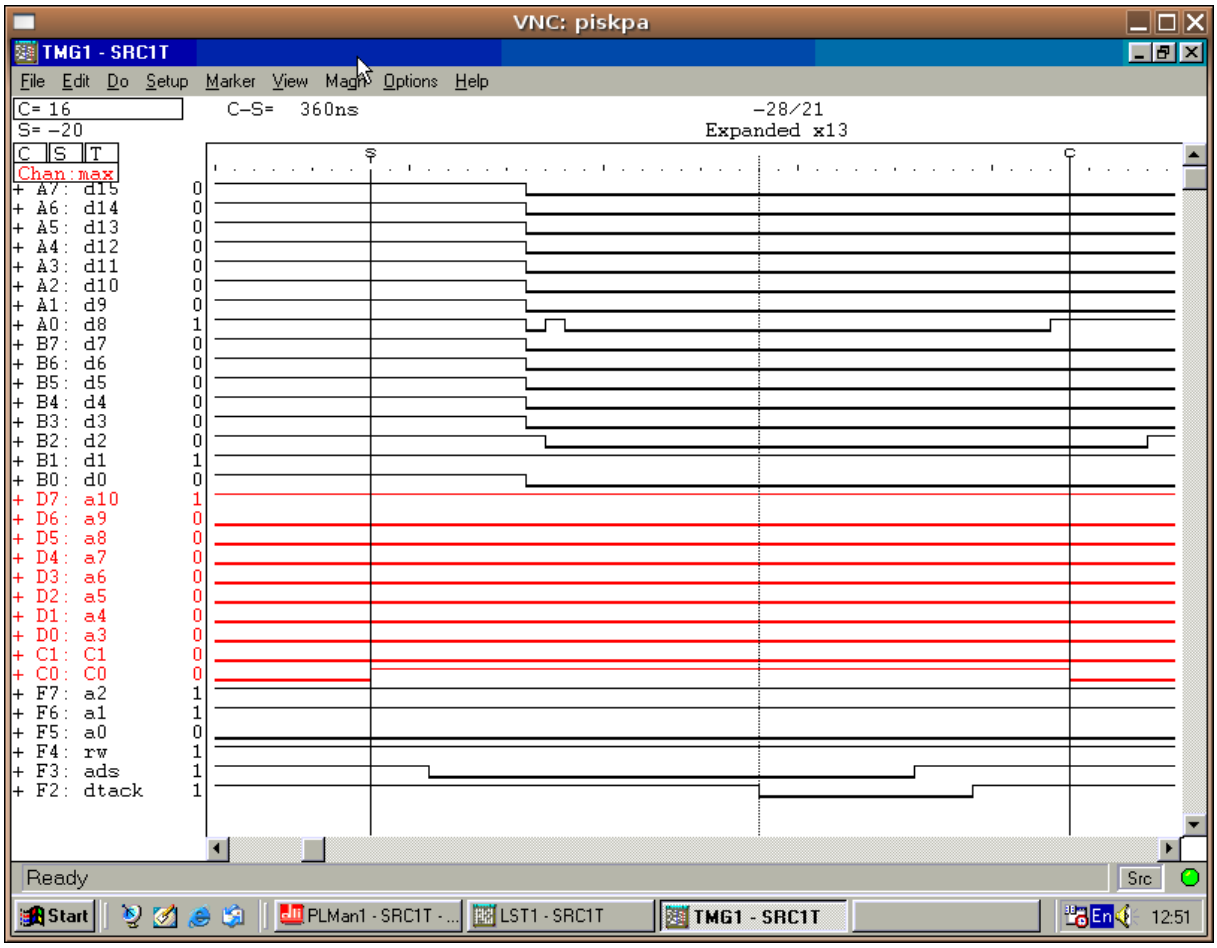


Abbildung 7.18 Ausgabe eines Logik-Analysierers, angeschlossen am Ausgang der FPGA-Karte. Gezeigt ist die zeitliche Abfolge der Signale beim Lesen des Statusregisters 0x406.

werden, die entsprechende Propagationszeiten besitzen (Oktaler Puffer und Leitungstreiber 74LS541). Desweiteren ist der ADS-Ausgang ebenfalls über eine Logik geführt, der Bausteine vom Typ 74LS32 enthält. Hinzu kommt noch die ebenfalls nicht zu vernachlässigende Laufzeit über das Buskabel zwischen der DL307V-Personality-Karte und der DL307-Interfacekarte. Nach der Rücknahme des ADS-Signals werden weitere 30 ns benötigt, bis das DL300-Crate die DTACK-Leitung zurücknimmt. Die Statemaschinen verlassen daraufhin nacheinander den Arbeitsmodus und wechseln in den Zustand **st_idle**. Durch die Schachtelung der Statemaschinen werden erneut 30 ns benötigt. Hinzu kommt noch die Propagationszeit des DTACK-Signals durch die Logik auf der DL307V-Karte. Gemessen wurden etwa 50 ns.

Da die Zeit für einen Zyklus wesentlich für die Gesamtperformanz des Interfaces ist, kann man Überlegungen anstellen, wie die Zeit für einen Zyklus verkürzt werden kann. Zum einen wäre es möglich, die Funktionalität der Statemaschine **dl_cycle** in die Statemaschinen **dl_read**, **dl_write** und **dl_setdac** zu übernehmen. Hierdurch könnten bei einem Zugriff auf das DL300-Crate mindestens 30 ns eingespart werden. Desweiteren könnte man durch einen Entwurf einer Personality-Card mit schnelleren Logik-Bausteinen kürzere Propagationszeiten erreichen. Grob geschätzt könnte man mit etwas Aufwand eine um etwa 10-15 % kürzere Zykluszeit erzielen. Darauf wurde jedoch zunächst verzichtet, da bei Tests mit leicht veränderter Logik auf der

DL307V-Karte kein stabiler Betrieb erreicht werden konnte. Es müsste hier ein neues Design der Karte erfolgen, welches ausgiebig getestet wird. Auf eine Integration der **dl_cycle** Statemaschine in die anderen drei Statemaschinen wurde bewusst verzichtet, um einen modularen Quellcode beizubehalten. Die erzielte Auslesegeschwindigkeit ist wie im Weiteren gezeigt wird zudem ausreichend, um die angestrebte Datenrate von über 1 kHz zu erreichen.

7.3.2 Initialisierung Sampling

Für die Datenaufnahme muss zunächst das DL300-System in den Samplingmodus geschaltet werden. Hierbei werden die dafür nötigen Parameter, wie Samplingfrequenz und der zu verwendende Modus (Common-Start oder Common-Stop) gesetzt. Desweiteren wird der Vorgang über Schreibkommandos auf die DL307-Interfacekarte gestartet. Die Realisierung des Vorgangs wird durch die Statemaschine dl_samp durchgeführt, die in Kapitel 6.7.6 beschrieben wird. Die Ausgabe des Logik-Analysieres ist in Abbildung 7.19 zu sehen.

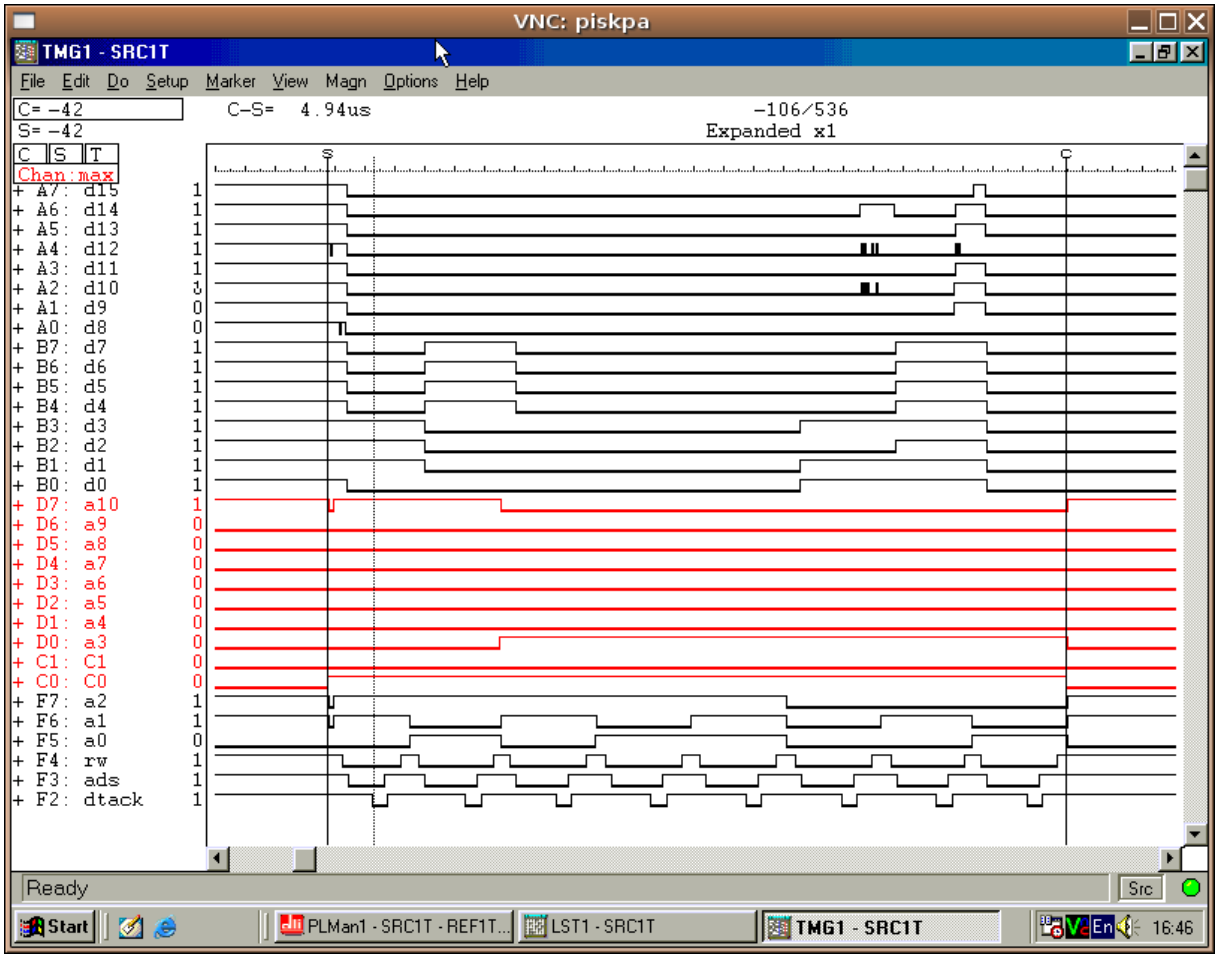


Abbildung 7.19 Ausgabe eines Logik-Analysierers, angeschlossen am Ausgang der FPGA-Karte. Markiert ist der Bereich der Initialisierung eines Samplingvorgangs.

Der Vorgang benötigt aufgrund der festen Befehlssequenz stets dieselbe Zeitdauer von etwa 5 μs. In der Ausgabe des Logic-Analyzers sind die separaten Kommandos anhand der Änderung von logisch 1 auf logisch 0 des DTACK-Signals (Data-Acknowledge) zu sehen. Im markierten

Zeitraum werden somit 8 Operationen durchgeführt, die an die DL307-Interfacekarte weitergegeben werden. Die Änderung des Signalpegels bei Lese- und Schreiboperationen ist dabei anhand der Lese- und Schreibleitung (F4: rw) zu sehen.

7.3.3 Initialisierung Fastscan

Die Initialisierung eines Fastscans wird über die Statemaschine **dl_fast** realisiert, die nach der erfolgreichen Initialisierung die Statemaschine **dl_hit_search** startet. Das Signal **mybusy** der Statemaschine kann über die Diagnoseleitung C0 ausgegeben werden. Ist diese Leitung auf logisch 1, so ist nur diese Statemaschine aktiv. Der Bereich, in dem die Signalleitung logisch 1 ist, markiert also die Zeitdauer, in der die Initialisierung des Fastscans erfolgt. Die an die DL307-Interfacekarte übergebenen Kommandos lassen sich mittels der aufgezeichneten Signale der Adress- und Signalleitungen verfolgen. Die übermittelten Lese- und Schreiboperationen wurden bereits in Kapitel 6.7.7 beschrieben. Diese Operationen sind wie bei der Initialisierung des Samplingvorgangs anhand des DTACK-Signals zu erkennen. In diesem Fall werden 9 Lese- und Schreiboperationen durchgeführt.

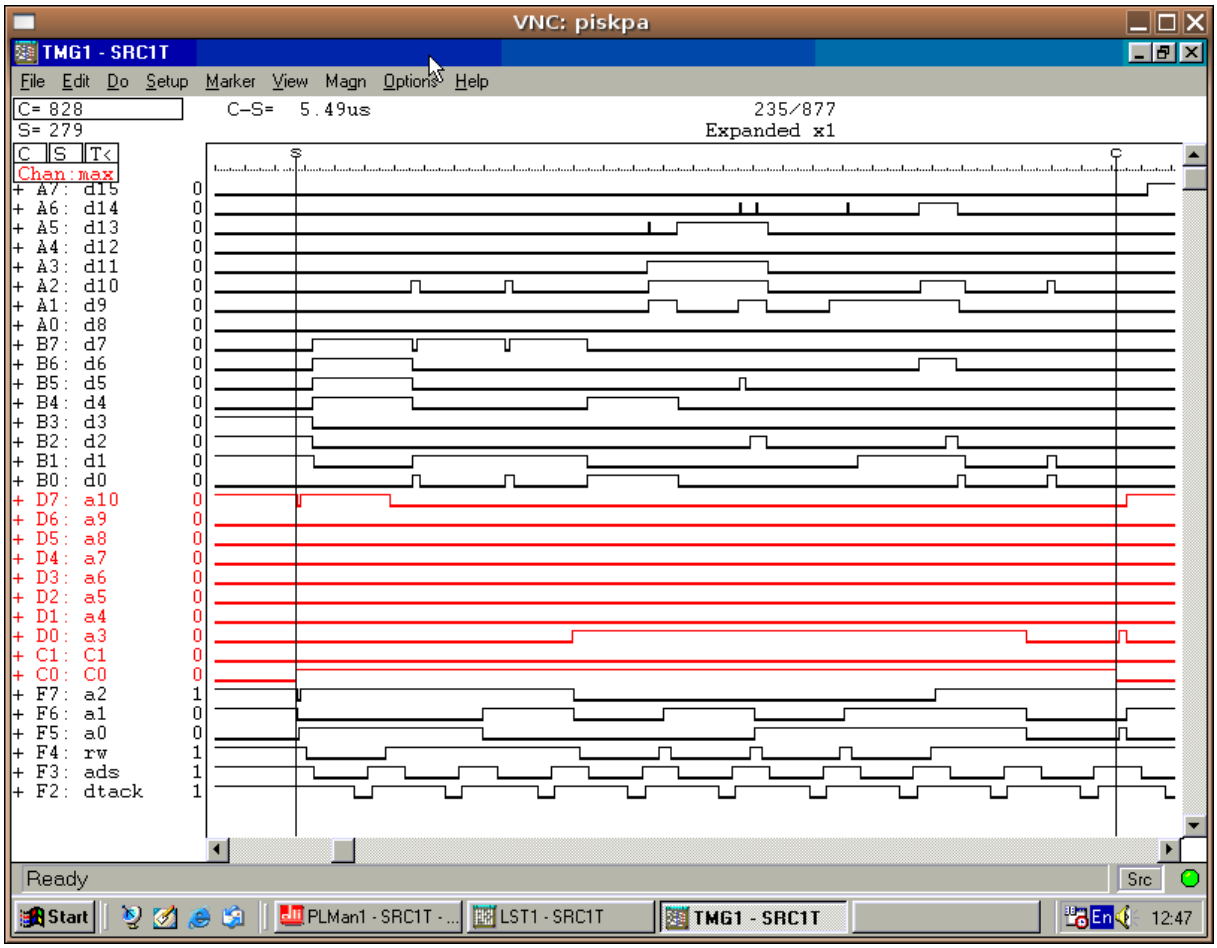


Abbildung 7.20 Ausgabe eines Logik-Analysierers, angeschlossen am Ausgang der FPGA-Karte. Markiert ist der Bereich der Initialisierung einer Fastscan-Operation.

Die Zeitdauer für die Initialisierung des Fastscan beträgt etwa 5,5 μs. Diese Zeitdauer ist fest, da

sich die Befehlssequenz nicht ändert. Lediglich die übermittelten Fastscan-Parameter können unterschiedliche Werte besitzen. Die Änderung der Parameter und somit der Datenleitungen hat jedoch keinen Einfluss auf die benötigte Zeit.

7.3.4 Suche nach Ansprechern - Hitscan

Für die Suche nach Ansprechern innerhalb des Speicherbereiches der verwendeten FADC-Karten ist das DL302-Scanner-Modul zuständig. Der zuvor initialisierte Fastscan wird durchgeführt, bis das Scanner-Modul einen Ansprecher im Speicherbereich der DL305- oder DL310-Module findet. Ein Hitscan sollte, damit dieser mit den eingestellten Schwellen einwandfrei funktioniert, mit einer Frequenz von 25 MHz durchgeführt werden. Ein Scan mit höheren Geschwindigkeiten – also höheren Scanfrequenzen – ist prinzipiell auch möglich. Entsprechende Tests wurden durchgeführt.

Die Performanz eines Hitscans wird durch mehrere Parameter beeinflusst. Hierzu gehören:

- Anzahl der Module im Crate-System,
- Speichergröße der eingesetzten Module (DL305 oder DL310),
- Scanfrequenz 25 MHz, 50 MHz oder 100 MHz,
- Anzahl der Ansprecher, die im Crate-System gefunden werden,
- Position der Ansprecher innerhalb der Speicherbereiche der FADC-Module.

Durch die Anzahl und Größe der eingesetzten Module wird die zu durchsuchende Speichergröße bestimmt. Jede Speicherstelle benötigt je nach eingestellter Scanfrequenz f eine Zeitdauer t von $t = \frac{1}{f}$, um auf einen Ansprecher überprüft zu werden. Bei einer Scanfrequenz von 25 MHz entspricht dies einer Zeitdauer von 40 ns pro Speicherstelle. Ein voll bestücktes Cratesystem mit DL305-Modulen hat eine Speichergröße von 12288 Speicherstellen, ein System mit DL310-Modulen eine Speichergröße von 49152 Speicherstellen.

In Abbildung 7.21 ist ein Hitscan zu sehen, durchgeführt mit einer Frequenz von 25 MHz, mit einem vollbestückten DL305-Crate mit 24 Modulen. Hierbei war nur der erste Kanal des FADC-Systems mit dem Ausgang des Signals eines Photomultipliers verbunden. Alle anderen Kanäle wurden nicht benutzt. Mit einer Schwelle von 10 für den Scan und einem zuvor eingestellten Pedestalwert 4 wurde dann die Busaktivität eines Hitscan-Vorgangs aufgenommen, der bei der Messung mit dem in Kapitel 7.2 beschriebenen Aufbau vorgenommen wurde. Als Anzahl der später auszulesenden Speicherstellen wurde die Anzahl 100 vorgegeben.

Der durchgeführte und aufgezeichnete Hitscan benötigt eine Zeitdauer von 451 μs. Der gefundene Ansprecher ist deutlich im Zeitdiagramm der aufgenommenen Signale am Anfang zwischen den beiden gesetzten Markierungen zu sehen. Um nun den Zusammenhang zwischen der Scanzeit und der Anzahl der Ansprecher im Cratesystem zu zeigen, wurde mit identischem

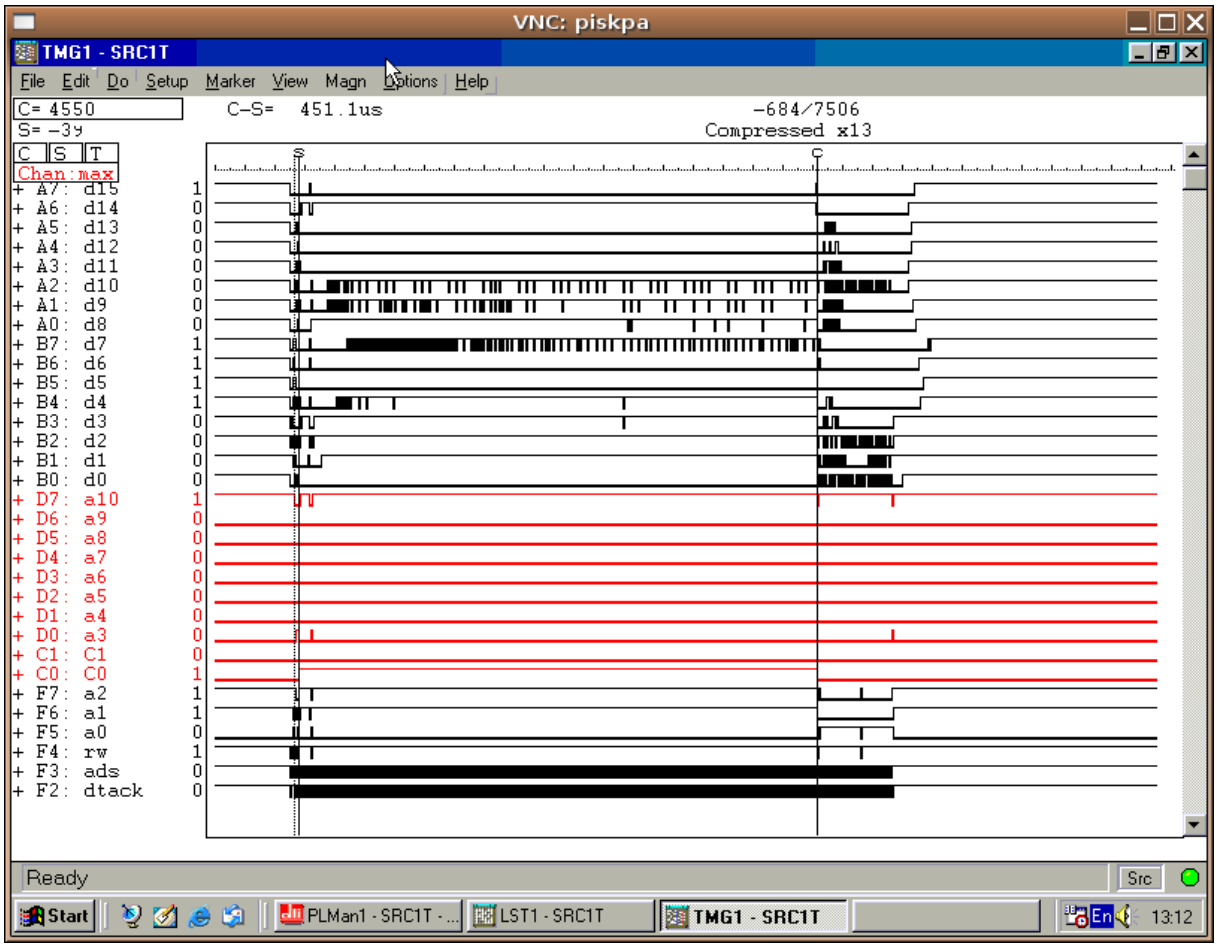


Abbildung 7.21 Ausgabe eines Logik-Analysierers, angeschlossen am Ausgang der FPGA-Karte. Markiert ist der Bereich der Suche nach Ansprechern in einem vollbestückten DL300-Crate mit einem einzigen Ansprecher.

Aufbau ein weiterer Hitscan durchgeführt. Hierbei wurde allerdings die Schwelle zur Erkennung von Ansprechern auf den Digitalisierungswert 2 gesenkt. Dadurch, dass die Schwelle unter dem eingestellten Pedestalwert 4 liegt, wird innerhalb von 2 Speicheradressen sofort ein Ansprecher gefunden, da die Bedingung für die Erkennung eines Ansprechers hier immer erfüllt ist. Um genau einen Ansprecher je Crate zu erhalten, wurde ein Zeitfenster für die Erkennung von Ansprechern definiert, so dass nur Ansprecher innerhalb eines definierten Zeit-Intervalles gespeichert werden. In Abbildung 7.22 ist der resultierende zeitliche Verlauf der Signalleitungen zu sehen.

Die gesamte Zeit für die Suche nach Ansprechern beträgt hier etwa 383 μs bei 48 gefundenen Ansprechern. Die Verkürzung der Dauer des Hitscans kommt dadurch zustande, dass nach einem gefundenen Ansprecher der Startpunkt für die Fortführung des Scans auf die aktuelle Speicherposition zuzüglich der im Interface eingestellten Anzahl an auszulesenden Speicherstellen pro Ansprecher (hier 100) gesetzt wird, die für jeden Puls ausgelesen werden. Sollte sich die neue Speicherposition nicht mehr innerhalb des aktuellen FADC-Moduls befinden, so wird die neue Position auf den Anfang des folgenden Moduls gesetzt. Hierdurch ergibt sich bei einer vorgegebenen Anzahl von 100 auszulesenden Datenpunkten eine theoretische Verkür-

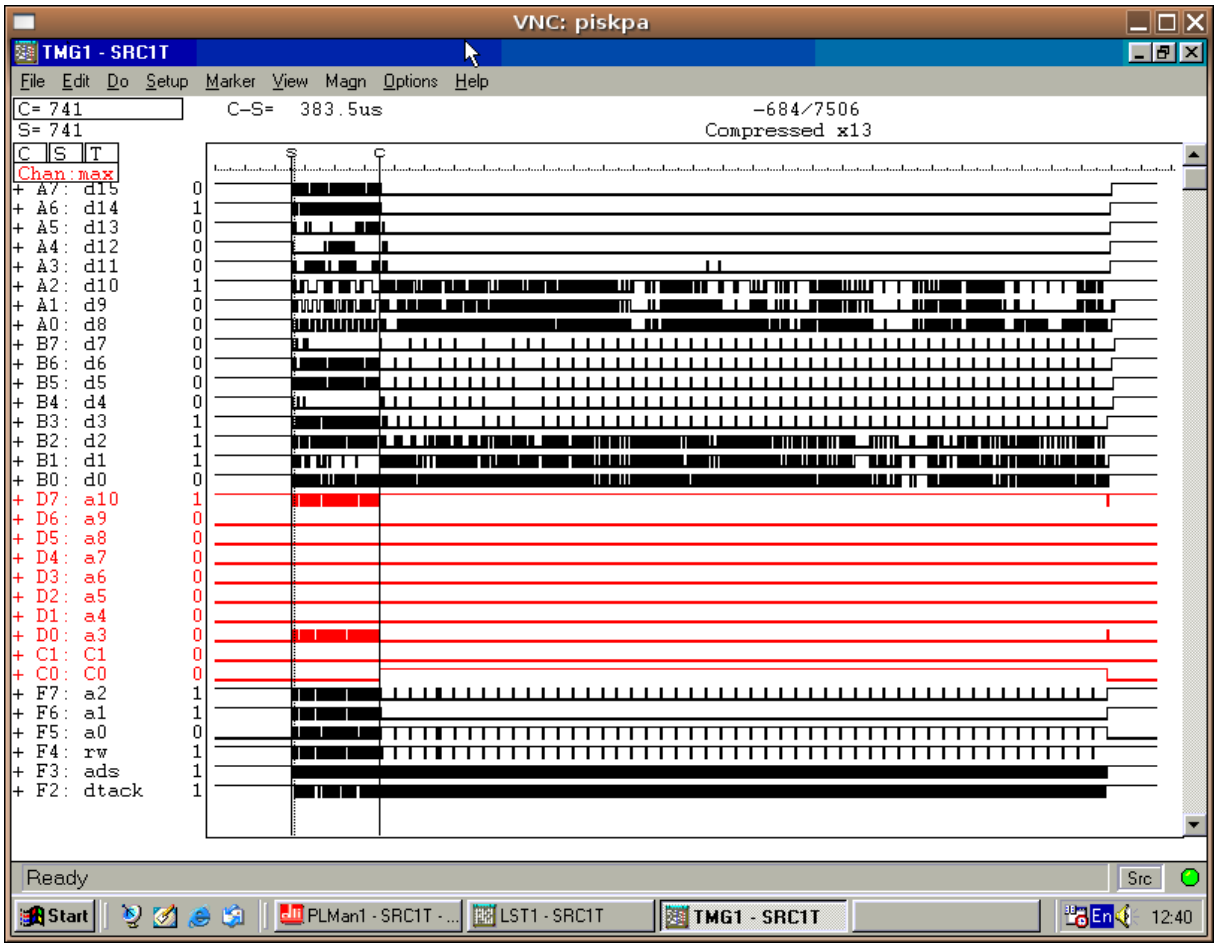


Abbildung 7.22 Ausgabe eines Logik-Analysierers, angeschlossen am Ausgang der FPGA-Karte. Markiert ist der Bereich der Suche nach Ansprechern in einem vollbestückten DL300-Crate. Die Schwelle zur Erkennung von Ansprechern wurde auf den Wert 2 gesenkt.

zung der Scanzeit um maximal 4 μs. Hinzu kommen pro gefundenem Ansprecher die Zeiten für die Befehle zur Neupositionierung des Scanner-Moduls, die Zeit für die Verarbeitung der ausgelesenen Position sowie die Befehle zum erneuten Start des Scanvorgangs. Desweiteren kann die Anzahl der übersprungenen Speicherstellen geringer als der voreingestellte Wert sein, wenn aufgrund des Stop-Signals und der aufgezeichneten Signalform der gefundene Ansprecher im Endbereich des Speichers eines FADC-Moduls liegt. Dadurch wird die Anzahl der übersprungenden Speicherstellen drastisch verringert, so dass die benötigte Zeitdauer wieder größer wird. Die gemessene Zeit von 383 μs entspricht aufgrund des Tests mit einer Hitschwelle unterhalb des Pedestalwertes etwa der minimal zu erwartenden Zeit für einen Fastscan mit den gegebenen Parametern. Die Zeit für eine Hitsuche mit im Mittel 7 Ansprechern pro DL300-Crate, wie im Aufbau am Vorwärtsdetektor zu erwarten, wird im Mittel bei etwa 430 μs liegen. Maximal sind etwa 460 μs zu erwarten. Bei Verwendung von DL310-Modulen sind aufgrund der vier mal höheren Speichergröße entsprechend vier mal längere Zeiten für die Suche nach Ansprechern zu erwarten.

7.3.5 Transfer der Daten vom DL300-Crate ins SSRAM

Nach einem erfolgreichen Hitscan werden die Daten der gefundenen Ansprecher vom DL300-Crate-System ins SSRAM der FPGA-Karte kopiert. Hierzu werden die im FIFO zwischengespeicherten gefundenen Positionen der Ansprecher verwendet, um je Position die vom Benutzer vorzugebende Anzahl Datenwerte an das SSRAM zu übertragen. Dabei wird zunächst die Adresse des Ansprechers an die DL300-Interfacekarte übergeben, um ab dieser Speicherposition einmalig über die Funktion 0x400, sowie nachfolgend dann über die Funktion 0x401 sukzessive den Speicherinhalt zu übertragen. Wird eine Modulgrenze erreicht, so muss der Adresszeiger auf der DL307-Karte auf den Beginn des aktuellen Moduls gesetzt werden, wo dann mit der Auslese fortgefahren werden kann. Die Ausgabe des Logic-Analyzers in Abbildung 7.23 zeigt dabei den Zeitraum der Übertragung eines Ansprechers, der mittels der Marker C und S markiert wurde. Innerhalb der Marker ist zu sehen, dass sich die Adressleitungen sowie die Lese- und Schreibleitung während der Auslese für einen Zugriff ändern. Dies zeigt, dass bei der dargestellten Auslese die Modulgrenze erreicht und der Adresszeiger auf den Beginn des Moduls zurückgesetzt wird.

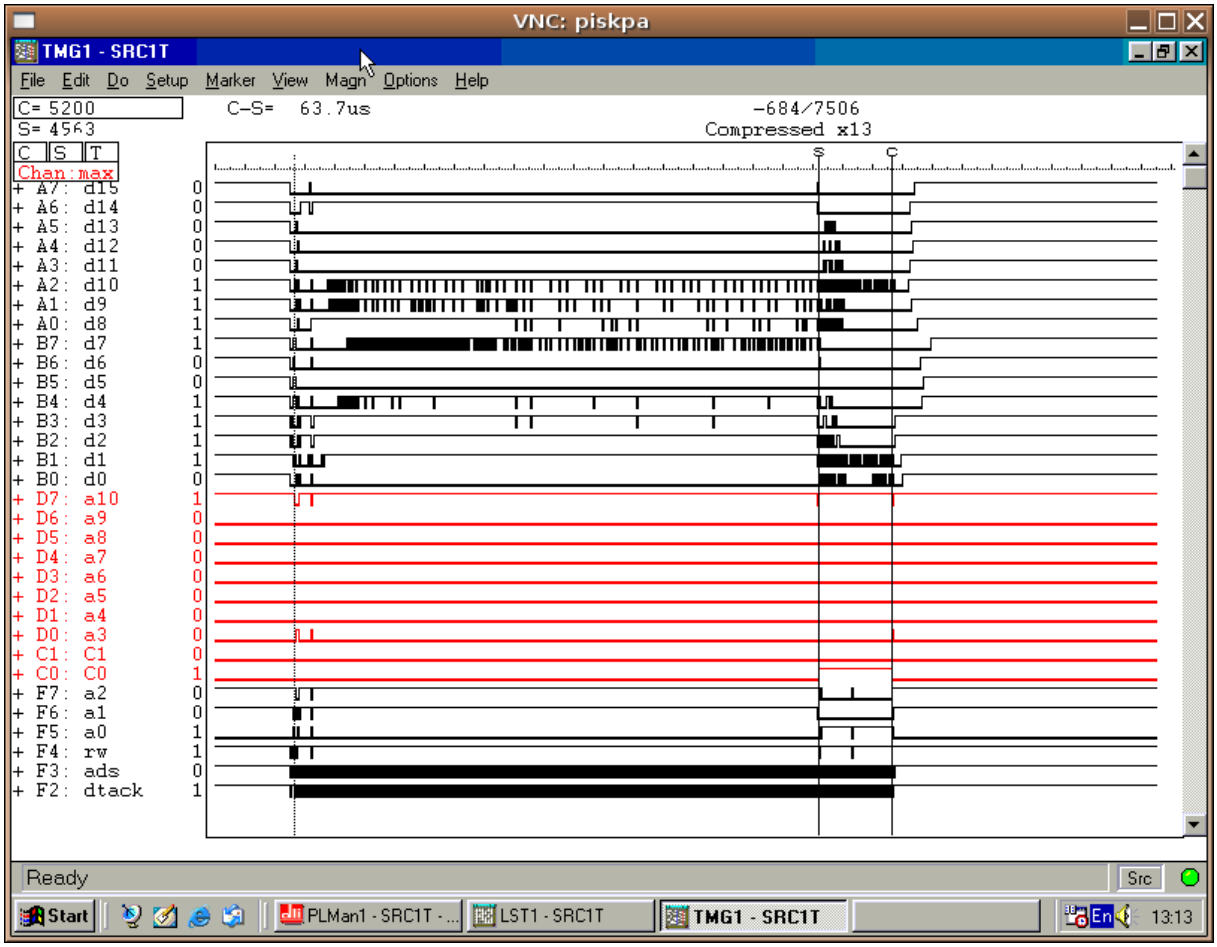


Abbildung 7.23 Ausgabe eines Logik-Analysierers, angeschlossen am Ausgang der FPGA-Karte. Markiert ist die Übertragung des gefundenen Ansprechers vom DL300-Crate ins SSRAM der FPGA-Karte.

Die Übertragung der 100 Datenworte benötigt etwa 64 μs. Die Auslesezeiten sind dabei im Wesentlichen nur von der vom Benutzer vorgegebenen Anzahl an zu lesenden Speicherstellen abhängig. Das Setzen des Adresszeigers auf dem DL307-Modul kann vernachlässigt werden, da die typische Anzahl der Lesezyklen zum Lesen der Datenworte wesentlich größer ist. Die Zeit hierfür beträgt im gezeigten Beispiel maximal etwa 4 % der Gesamtzykluszeit.

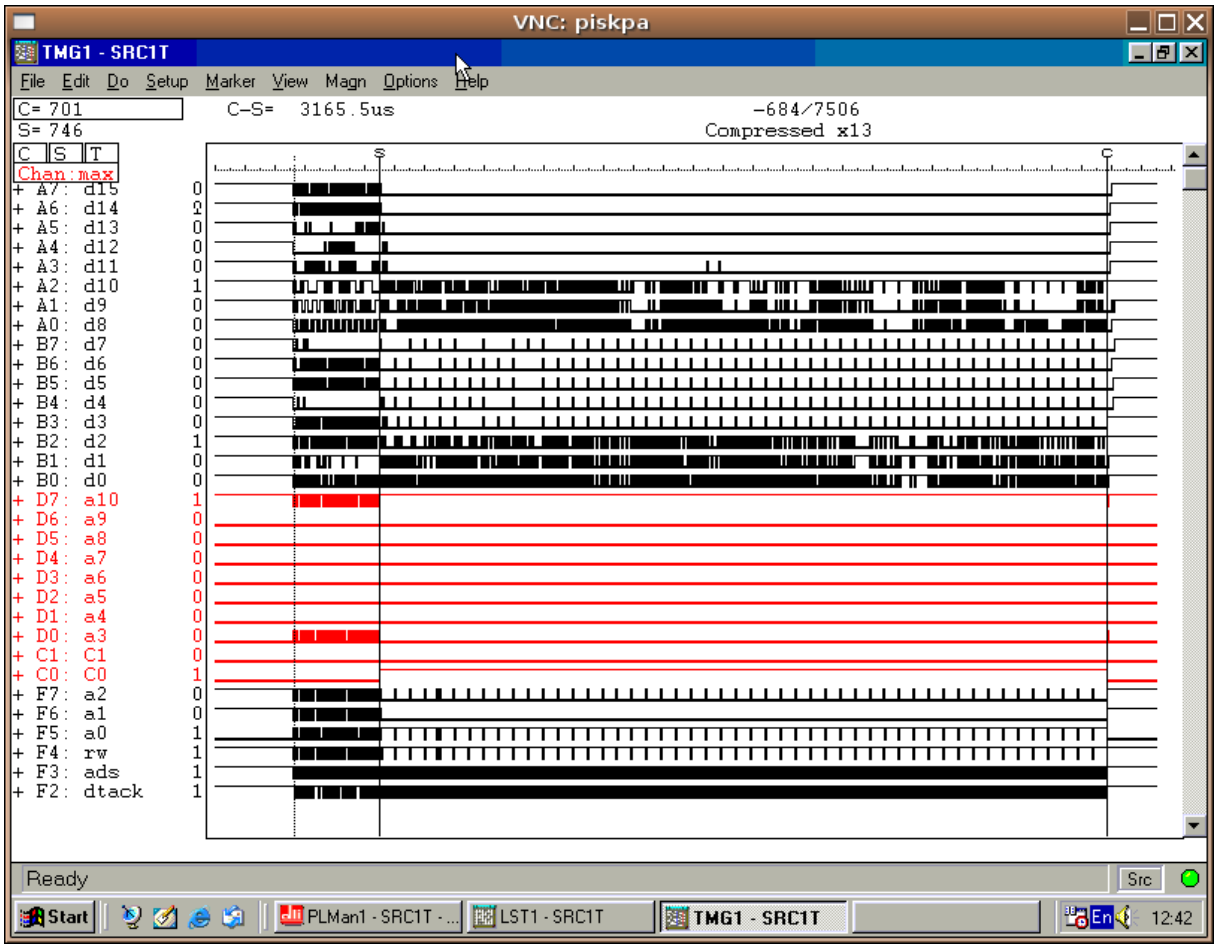


Abbildung 7.24 Ausgabe eines Logik-Analysierers, angeschlossen am Ausgang der FPGA-Karte. Markiert ist die Übertragung der gefundenen 48 Ansprecher vom DL300-Crate ins SS-RAM der FPGA-Karte.

In dem in Abbildung 7.24 abgebildeten Transfer der Ansprecher wurde der Schwellenwert verringert. Für eine mittlere Anzahl von 7 Ansprechern mit je 100 auszulesenden Datenworten pro Crate werden somit etwa 450 μs benötigt.

7.3.6 DMA-Transfer der Daten

Nach erfolgreicher Übertragung der Daten vom DL300-Crate ins SSRAM der FPGA-Karte können die Daten über einen DMA-Transfer an die DAQ-CPU übertragen werden. Der DMA-Modus ermöglicht dabei den direkten Transfer der Daten vom SSRAM-Speicher in den Speicher der angeschlossenen CPU, ohne dass hierbei während des Transfers Kontroll- oder Steuerungsaufgaben durch die CPU übernommen werden müssen. Der Transfer wird hierbei durch den

DMA-Controller gesteuert. Dieses Übertragungsverfahren hat gegenüber dem Transfer einzelner Speicherzellen den Vorteil, dass dieser wesentlich schneller vorgenommen wird. Dies soll hier anhand eines Beispiels für die beiden Transferverfahren gezeigt werden. Nachteilig bei einem DMA-Transfer ist die hohe Initialisierungszeit. Je nach Datenmenge muss man hier abwägen, ob ein einfacher Zugriff oder ein DMA-Transfer sinnvoll ist.

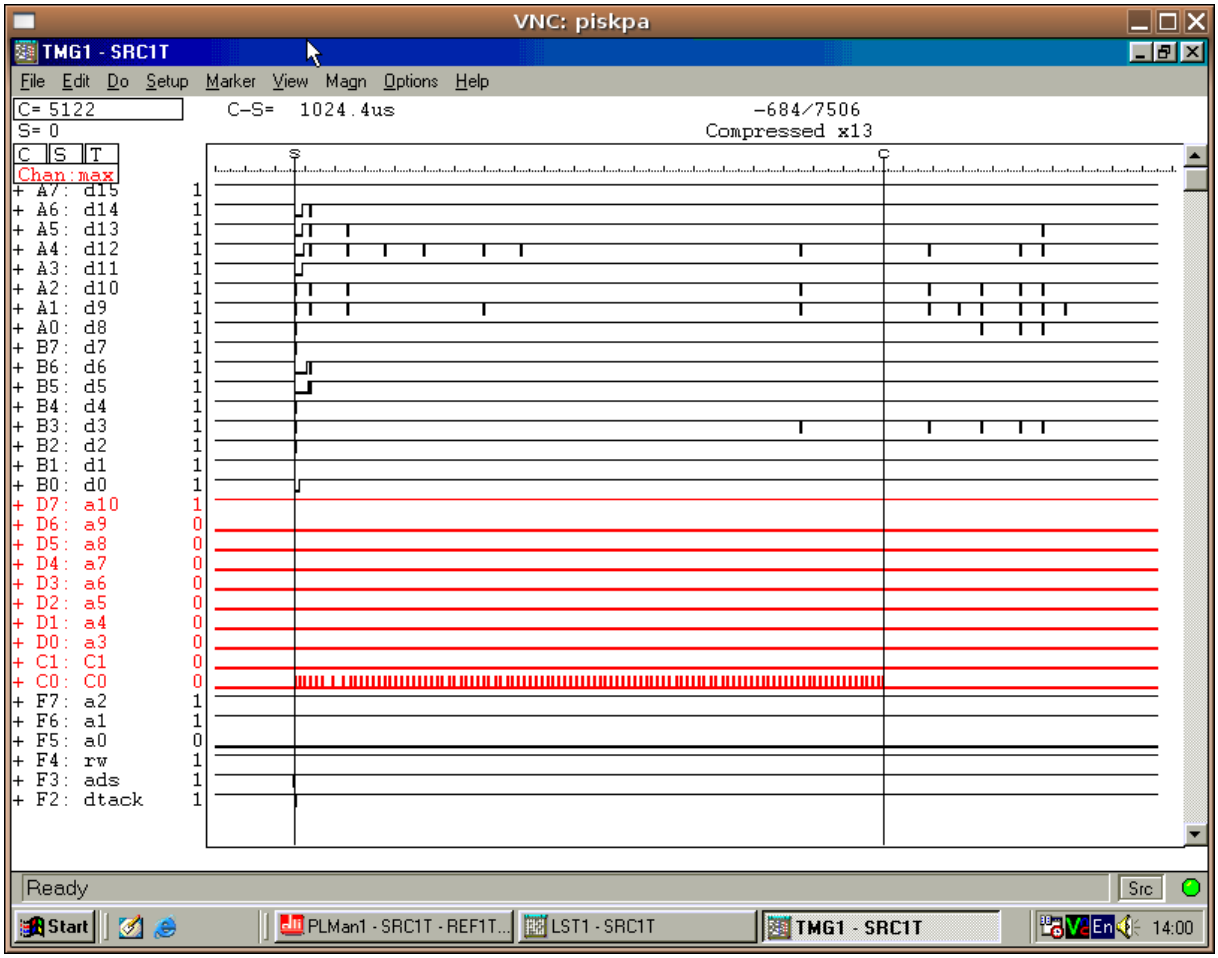


Abbildung 7.25 Ausgabe eines Logik-Analysierers, angeschlossen am Ausgang der FPGA-Karte. Markiert ist die Übertragung von 1000 32-Bit-Datenworten über einzelne Transfers von der FPGA-Karte ohne Zugriff auf das SSRAM.

In Abbildung 7.25 ist ein Transfer eines PCI-Registers zu sehen, welcher durch die Software 1000 mal wiederholt wird. Hierbei werden 1000 32-Bit-Datenworte vom FPGA an die CPU des angeschlossenen Rechners übertragen. Vor diesem Transfer wurde ein Zugriff auf das Statusregister durchgeführt, um einen Trigger für den Logic-Analyzer auszulösen. Markiert ist der Bereich, in dem die 1000 Datenworte übertragen werden. Dies ist anhand der Leitung C0 zu erkennen, die in der gezeigten Konfiguration das FPGA-Signal **ds_xfer** ausgibt. Die benötigte Zeit liegt bei etwa 1024 μs. Jeder Einzeltransfer nimmt also etwa 1 μs in Anspruch. Im Gegensatz hierzu ist in Abbildung 7.26 ein Transfer von 1000 Datenworten aus dem SSRAM der FPGA-Karte zu sehen. Vor dem Transfer wurde ein Lesezugriff auf das Statusregister durchgeführt. Auf der Leitung C0 ist das **ds_xfer** Signal zu sehen. Die Zeitdauer für den gesamten Datentransfer beträgt hier nur etwa 53 μs.

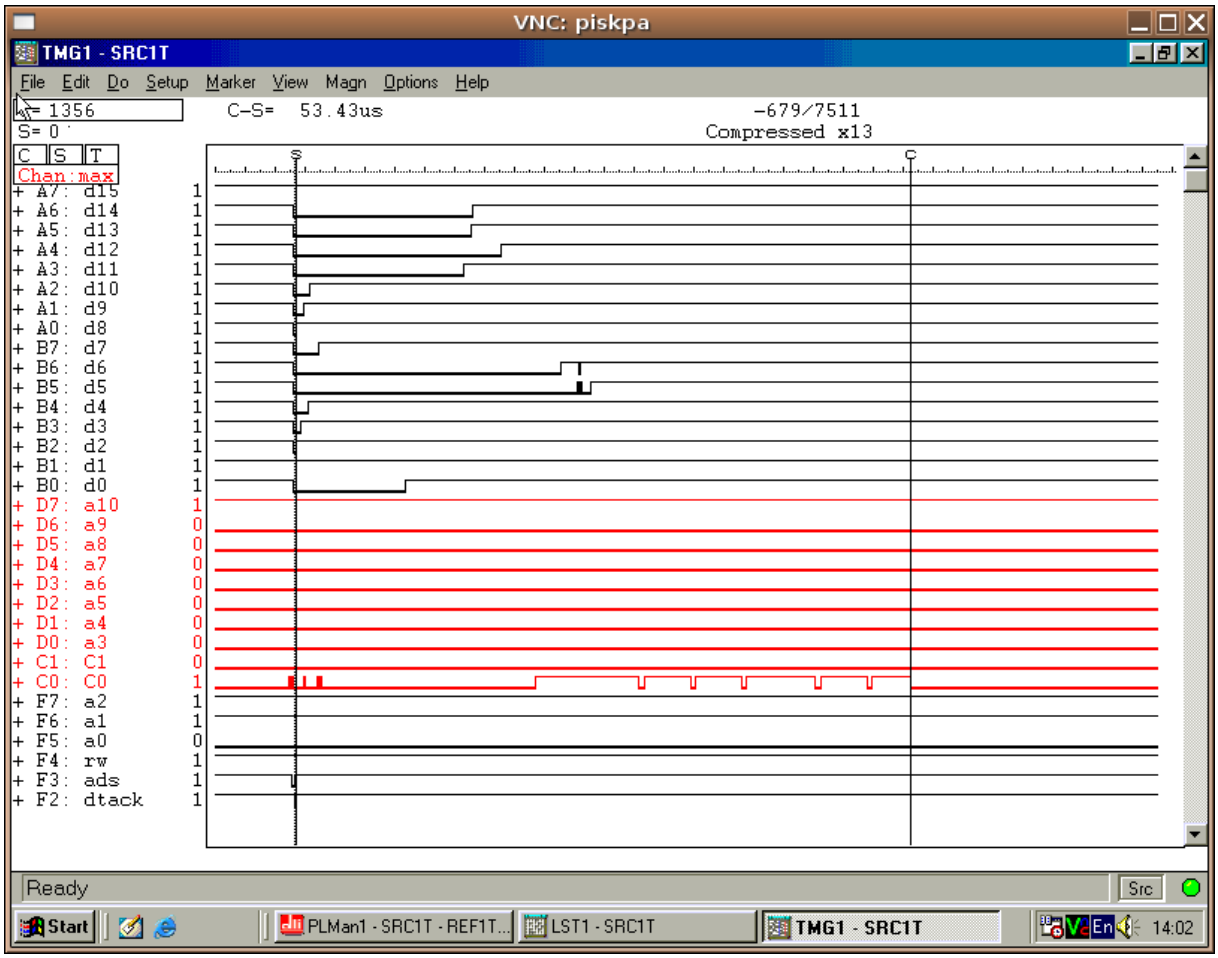


Abbildung 7.26 Ausgabe eines Logik-Analysierers, angeschlossen am Ausgang der FPGA-Karte. Markiert ist die Übertragung von 1000 32-Bit-Datenworten über einen DMA-Transfer vom SSRAM auf der FPGA-Karte zur CPU.

Die Zeitdauer für den DMA-Transfer setzt sich dabei aus zwei Anteilen zusammen. Der erste Anteil ist im vorderen Bereich zu sehen, in dem die C0-Leitung nur selten in den Zustand logisch 1 wechselt. Hier wird innerhalb von etwa 21μs der DMA-Transfer vorbereitet. Diese Zeit ist unabhängig von der Menge der zu übertragenen Daten, sofern der Transfer die Speichergrenzen des auszulesenden SSRAM-Bausteins nicht überschreitet, da in diesem Fall eine Aufteilung in zwei separate Übertragungen erforderlich ist. Danach folgt der Bereich des eigentlichen Datentransfers, welcher eine Zeit von etwa 32 μs benötigt. Dies entspricht einer Zeitdauer von 32 ns je übertragenem 32-Bit-Datenwort und ist begrenzt durch die verwendete PCI-Busfrequenz von 33,33 MHz. Die Übertragung der Datenworte selbst ist also mindestens um den Faktor 30 schneller, als bei der Verwendung eines Einzeltransfers. Nimmt man die benötigte Zeit zum Starten des DMAs von 21 μs hinzu, so ergibt sich ein zeitlicher Vorteil der DMA-Übertragung beim Transfer von mehr als 22 32-Bit-Datenworten. Im zunächst verwendeten Aufbau am Vorwärtsdetektor werden pro Ansprecher vorraussichtlich 100 Speicherpositionen zu je 2 mal 6 Bit übertragen, welche aufgrund der Zusammenfassung in 32-Bit-Worten und weiteren Summen-Informationen sowie Markierungen etwa 60 Datenworten zu je 32-Bit entsprichen. Die Datenmenge ist also ausreichend, um hier durch einen DMA Zeit zu sparen.

7.3.7 Zusammenfassung und prognostizierte Datenrate für den Vorwärtsdetektor

Die zuvor ermittelten Datenwerte ermöglichen eine Prognose für die am Vorwärtsdetektor zu erwartende Ausleserate, sowie eine Abschätzung der Datenrate für die Verwendung an weiteren Detektoren. Am Vorwärtsdetektor sollen die Signale von 90 Kristallen aufgeteilt auf 2 FADC-Crates mit je 48 Doppel-Kanälen werden. In den 90 Kristallen sind etwa 14 Ansprecher zu erwarten [Fun04], so dass im Mittel mit etwa 7 Ansprecher je FADC-Crate zu rechnen ist. Die in Tabelle 7.1 prognostizierten Werte basieren auf diesen Annahmen. Die Zeitwerte sind dabei aufgeteilt nach festen Zeiten, die unabhängig von der Anzahl der Ansprecher sind, sowie einer dynamischen Zeit, die von der Anzahl der gefundenen und auszulesenden Ansprechern abhängt.

Vorgang	feste Zeit	dynamische Zeit
Initialisierung Sampling 100 MHz	5 μs	-
Initialisierung Fastscan	5,5 μs	-
Hitsuche (Fastscan) mit 25 MHz oder 100 MHz	460 μs oder 115 μs	-
Hit-Transfer	-	450 μs
DMA-Transfer	21 μs	13,4 μs
Gesamt	491,5 μs oder 146,5 μs	463,4 μs

Tabelle 7.1 Prognose Auslesezeiten getrennt nach Abschnitten der Auslese.

Es ergibt sich ein Anteil von 491,5 μs mit einer Fastscan-Frequenz von 25 MHz oder alternativ von 146,6 μs bei einer Frequenz von 100 MHz. Der Zeitaufwand wird hauptsächlich durch den durchgeführten Fastscan verursacht. Hinzu kommt eine Zeitdauer von 463,4 μs die allerdings von der Anzahl der Signale im Vorwärtsdetektor und von der auszulesenden Anzahl an Speicherstellen abhängt. Insgesamt ergibt sich eine mittlere theoretische Auslesezeit von etwa 955 μs mit einem 25 MHz Fastscan oder 609,9 μs bei einem 100 MHz Fastscan. Da die Daten bereits vorformatiert an den Speicher der CPU übergeben werden, sollte sich die tatsächliche Auslesezeit in der Datenerfassung am Experiment von der theoretischen Auslesezeit kaum unterscheiden. Dies ist das Ergebnis früherer Analysen der Auslesezeiten im verwendeten DAQ-System [Sch04]. Die aufgeführten Zeiten entsprechen in der Summe der Totzeit der Datenerfassung. Man unterscheidet zwischen der Totzeit eines DAQ-Systems, während der die Auslese durchgeführt wird und der Lebendzeit, in der auf ein eingehendes Auslesesignal gewartet wird. Die Lebendzeit ist hierbei abhängig von der Rate des eingehenden Triggersignals.

Die Fastscanfrequenz sollte laut Spezifikation den Wert von 25 MHz nicht überschreiten. Tests zeigen jedoch, das ein Fastscan mit 100 MHz funktioniert, sofern die Schwelle zur Erkennung eines Ansprechers erhöht wird. Ob dies auch am Vorwärtsdetektor am Experiment einwandfrei funktioniert, muss ein entsprechender Test dort zeigen. Um die Auslesezeit weiter zu verringern, kann die Samplingfrequenz auf einen Wert von 50 MHz gesenkt werden. Dadurch verringert

sich die Anzahl der auszulesenden Speicherstellen und damit auch die für den Hit-Transfer benötigte Auslesezeit. Nachteilig wirkt sich diese Änderung auf das zeitliche Auflösungsvermögen des Signalverlaufes aus. Eine weitere Alternative für die Verringerung der Auslesezeit ist die Aufteilung der 90 Signale der Kristalle auf drei FADC-Crates. Da genügend FADC-Kanäle, sowie drei FPGA-Karten zur Verfügung stehen ist dies mit einer zusätzlichen CPU relativ einfach realisierbar.

Die zu erzielende Datenrate von 1 kHz für den Vorwärtsdetektor lässt sich mit dem implementierten Interface und der vorhandenen Elektronik realisieren. Die effektiv erreichte Datenrate hängt von den gewünschten Anforderungen an die Auslese in Bezug auf Schwellenwerte und zeitliche Auflösung ab. Soll die zeitliche Auflösung 10 ns betragen und der Schwellenwert vergleichsweise gering sein, so müssen zum Erreichen von 1 kHz Datenrate drei FADC-Crates mit eigener Auslese-CPU eingesetzt werden.

7.4 FPGA-Nutzung

Für weitere Entwicklungen des DL300-Interfaces ist interessant, wieviel von der auf dem Chip vorhandenen Logik noch verfügbar ist. In Abbildung 7.27 ist graphisch zu sehen, welcher Anteil der Logikeinheiten auf dem FPGA-Chip der Alpha-Data FPGA-Karte in Verwendung ist.

Der Xilinx Virtex FPGA vom Typ XCV400-BG560 verfügt über 704 Anschlussbeine (Pins), von denen 4 allerdings nicht verwendet werden. Von den restlichen Anschlussbeinen sind 132 für den Anschluß der Masse (GND) vorgesehen und 148 für Versorgungsspannungen. 404 Pins stehen dem Nutzer zur Verfügung, die aufgrund der durch den Hersteller festgelegten Verdrahtung bereits fest einem Bestimmungszweck zugewiesen sind. Die in Tabelle 7.2 aufgeführten IOBs entsprechen den 404 nutzbaren Pins des FPGA. Von diesen sind 342 in Benutzung.

Für spätere Erweiterungen sind die Ressourcen der Register (Slice Register), der assoziativen Felder (Look-Up-Table - LUT) und der Puffer (TBuf) von Bedeutung. Die Nutzung dieser Ressourcen liegt zwischen 20 % und knapp 40 %. Die Speicherbereiche (Block RAMs) auf dem FPGA sind primär durch den implementierten FIFO-Puffer in Benutzung. Von den 20 verfügbaren Block RAMs sind auf dem FPGA zwei hierfür genutzt worden. Die Ressourcenbelegung zeigt, dass noch genügend Platz auf dem FPGA für Erweiterungen zur Verfügung steht. Es ist im Mittel etwa ein Viertel der Ressourcen verwendet worden. Insbesondere im Hinblick auf die Analyse von Drahtkammersignalen in späteren Förderungsperioden von CB-TR ist dies von Bedeutung. Die Analysen der Signale können dann auf dem FPGA implementiert werden und die Auslese-CPU entlasten.

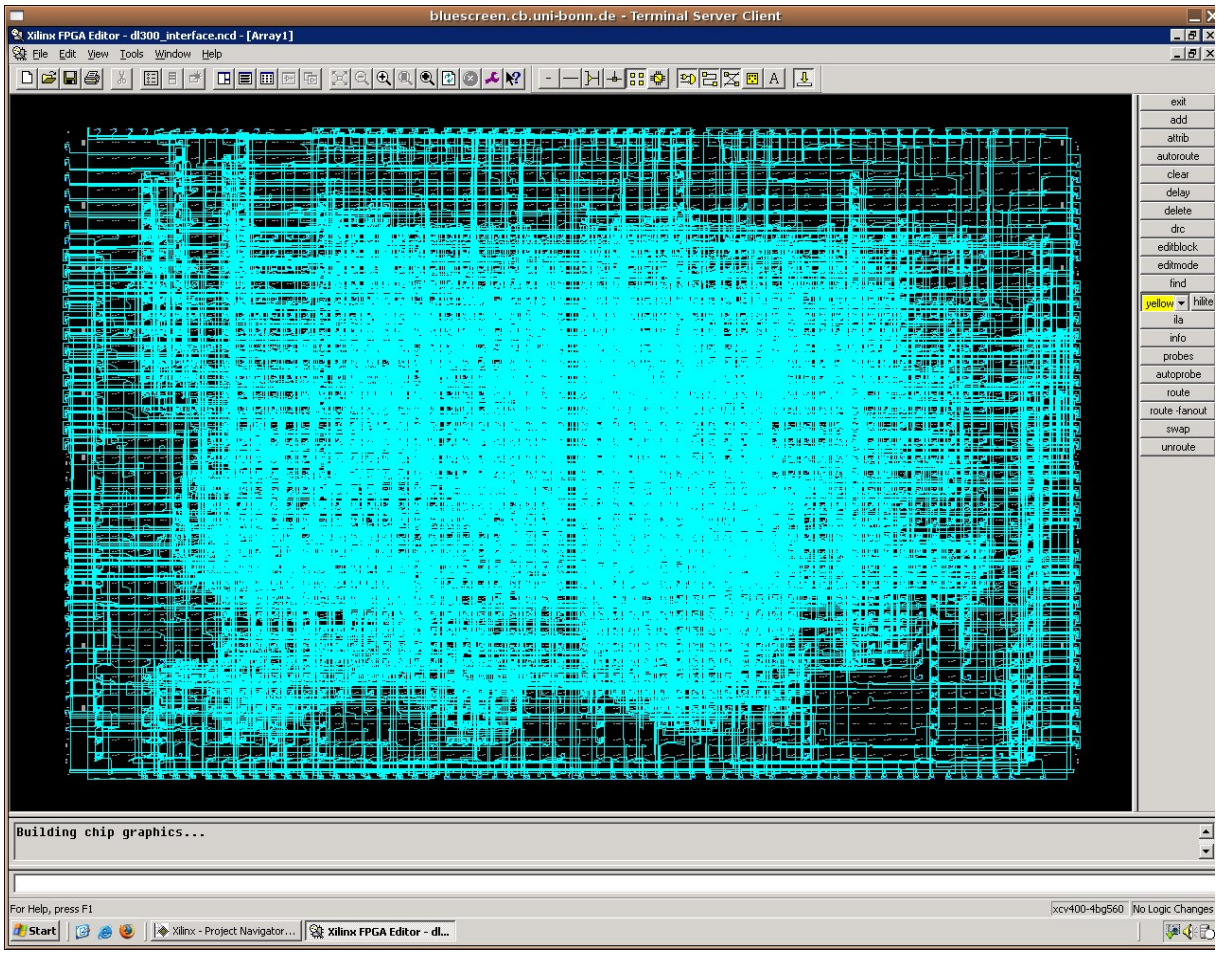


Abbildung 7.27 Graphische Darstellung der Nutzung des FPGAs im FPGA-Editor. Markiert sind die genutzten Logikeinheiten und Verbindungen auf der Chip-Fläche des verwendeten FPGAs.

Ressource	genutzte Einheiten	gesamte Einheiten Xilinx Virtex	Nutzung in %
Slice Register	1944	9600	20 %
4 input LUTs	3684	9600	38 %
IOBs	342	404	84 %
TBufs	1600	4960	32 %
Block RAMs	2	20	10 %

Tabelle 7.2 Ausnutzung der Logikeinheiten auf dem verwendeten Xilinx Virtex XCV400-BG560 FPGA.

8 Zusammenfassung und Ausblick

Im Rahmen dieser Dissertation wurde ein neues Prozessor-Interface für ein Flash-ADC-System vom Typ DL300 der Firma Struck auf Basis eines feldprogrammierbaren Logikbausteins entwickelt, welches zunächst am Crystal-Barrel-Experiment am neuen Vorwärtsdetektor zur Messung in einem Photoproduktionsexperiment mit Doppelpolarisation eingesetzt werden soll. Das Interface ermöglicht die Auslese des Flash-ADC-Systems unter Verwendung der am Experiment eingesetzten VME-CPUs. Die Steuerung des DL300-Crates erfolgt dabei vollständig autonom auf dem Logikbaustein. Die Daten werden vorformatiert auf dem Speicher der eingesetzten FPGA-Karte abgelegt und können über einen DMA-Transfer an die CPU übertragen werden. Die Auslesegeschwindigkeit der Flash-ADC-Kanäle ist durch das in dieser Arbeit erstellte Interface signifikant verbessert worden. Für den Einsatz am neuen Vorwärtsdetektor für das CB-TR-Experiment ist eine Datenrate von mehr als 1 kHz erzielbar. Für erste Analysen der aufgezeichneten Daten wurden zwei Analysefunktionen entwickelt, die parallel zur Auslese der Daten eine Integration „in Hardware“ über die detektierten Signale ermöglichen. Hierbei wird eine Summe gebildet, die dem Messergebnis eines integrierenden ADCs entspricht. Zusätzlich wird die gesamte Signalform ausgelesen und vollständig gespeichert. Dies ermöglicht die spätere Analyse der Experiment-Daten im Hinblick auf mögliche Doppelansprecher durch elektromagnetischen Untergrund. Hierdurch können nun verfälschte ADC-Summen erkannt werden, die für eine fehlerhafte Energiebestimmung verantwortlich sind, was ohne Verwendung von Flash-ADCs prinzipbedingt unmöglich ist. Die Energiesumme kann ebenfalls bestimmt werden. Hierbei ist der Fehler des Integrals allerdings abhängig von der Form des Pulses, so dass für kleine Photomultiplier-Signale die Auflösung des Flash-ADCs nicht so gut ist, wie die eines integrierenden 11-Bit ADCs. Für größere Signale ist die Auflösung allerdings vergleichbar mit dem integrierenden Fera-4300-ADC (A = 27 %). Desweiteren ist die Bestimmung einer Zeitinformation möglich, die absolut mit einer Standardabweichung von σ= 14 ns erfolgen kann. Die relative Zeit zwischen zwei Signalen in verschiedenen FADC-Kanälen lässt sich mit einer höheren Genauigkeit bestimmen.

Das programmierte Interface wurde modular entwickelt, so dass eine spätere Erweiterung der Funktionalität möglich ist. Auf dem feldprogrammierbaren Logikbaustein sind genügend Ressourcen verfügbar, um weitere Analysen parallel zur Auslese der Daten zu implementieren. Denkbar ist hier die automatische Erkennung von Doppelansprechern bereits während der Auslese, die Bestimmung einer relativen Zeitinformation der aufgezeichneten Pulse oder auch eine Erkennung von geladenen und ungeladenen Teilchen aufgrund der Pulsform. In späteren Förderungsperioden des Experiments kann das Interface für die Auslese von neuen Detektoren verwendet werden. Geplant sind hier Draht- und Driftkammern. Das DL300 Flash-ADC-System in Kombination mit dem FPGA-Interface ist für eine kostengünstige Realisierung einer Auslese der neuen Detektoren ideal geeignet, da auf bereits vorhandene FADC-Kanäle zurückgegriffen werden kann. Eine Analyse der Drahtkammer-Signale kann auf der FPGA-Hardware erfolgen und ermöglicht somit eine sehr schnelle Analyse der aufgezeichneten Signale. Diese Arbeit liefert hierfür die notwendigen Voraussetzungen.

A FPGA-Interface

A.1 Belegung des Frontio-Steckers

VHDL-Name	Bit	Signalrichtung	Pin	VHDL-Name	Bit	Signalrichtung	Pin
dl300_data_ext	0	ein und aus	W33	dl300_addr_ext	0	aus	AE33
dl300_data_ext	1	ein und aus	W31	dl300_addr_ext	1	aus	AC29
dl300_data_ext	2	ein und aus	W30	dl300_addr_ext	2	aus	AE32
dl300_data_ext	3	ein und aus	W29	dl300_addr_ext	3	aus	AD30
dl300_data_ext	4	ein und aus	Y30	dl300_addr_ext	4	aus	AF32
dl300_data_ext	5	ein und aus	AA33	dl300_addr_ext	5	aus	AD29
dl300_data_ext	6	ein und aus	Y29	dl300_addr_ext	6	aus	AF31
dl300_data_ext	7	ein und aus	AA32	dl300_addr_ext	7	aus	AE30
dl300_data_ext	8	ein und aus	AA31	dl300_addr_ext	8	aus	AG33
dl300_data_ext	9	ein und aus	AA29	dl300_addr_ext	9	aus	AH33
dl300_data_ext	10	ein und aus	AB31	dl300_addr_ext	10	aus	AF30
dl300_data_ext	11	ein und aus	AB30	dl300_rw_ext	-	aus	AH32
dl300_data_ext	12	ein und aus	AC33	dl300_ads_ext	-	aus	AJ32
dl300_data_ext	13	ein und aus	AC31	dl300_reset_ext	-	aus	AF29
dl300_data_ext	13	ein und aus	AC31	dl300_reset_ext	-	aus	AF29
dl300_data_ext	14	ein und aus	AB29	dl300_dtack_ext	-	ein	AH31
dl300_data_ext	15	ein und aus	AC30	debug1	-	aus	AJ31
				debug2	-	aus	AG29

Tabelle A.1 Belegung des Frontio-Steckers.

A.2 Funktionen des Alpha-Data SDK

- **ADMXRC2_OpenCardbyIndex()**

  Sucht im System und auf dem Bus eine ADM-XRC-Karte. Im Falle der eingesetzten PMC-Karte ist dies der PCI-Bus des Systems. Der erste übergebene Parameter ist der Kartenindex (Integer Wert). Der Kartenindex ist ein vom Bus abhängiger Parameter. Die Ordnung des Index ist nicht beeinflussbar. Für die Anwendung in Bezug auf das DL300-System hat dies jedoch keinen Einfluss, da hier auf den verwendeten VMIC-CPUs nur eine Karte eingesetzt werden kann. Der zweite Parameter ist ein Pointer vom Typ ADMXRC2_Handle. Diese Referenz wird bei fast allen weiteren Funktionsaufrufen als Referenz auf die geöffnete Karte verwendet. Die Variable wird bei allen folgenden Funktionen als erster Parameter mit übergeben. Dies wird im Weiteren nicht mehr explizit erwähnt. Als Rückgabewert der Funktion erhält man eine Variable vom Typ ADMXRC2_Status. Diese Variable kann den Wert ADMXRC2_SUCCESS annehmen, sofern eine Karte gefunden wurde sowie ADMXRC2_CARD_NOT_FOUND im Fehlerfall. Für fast alle weiteren Funktionen wird als Rückgabewert bei einem erfolgreichem Aufruf ADMXRC2_SUCESS zurückgegeben, und falls das Handle keine gültige Referenz darstellt, ein ADMXRC2_INVALID_HANDLE. Sofern kein expliziter Rückgabewert der Funktion erwähnt wird ist dies der Fall.

- **ADMXRC2_ConfigureFromFile()**

  Konfiguriert den FPGA auf der PMC-Karte mit der als zweiten Parameter übergebenen Bitstreamdatei, welche mit der Xilinx ISE Software erstellt wurde. Hierbei wird überprüft, ob sich die Datei öffnen lässt (ADMXRC_FILE_NOT_FOUND im Fehlerfall), ob die Datei eine gültige Bitstreamdatei ist (falls nicht: ADMXRC2_INVALID_FILE), ob die Datei für den auf der PMC-Karte verwendeten FPGA-Chip übersetzt wurde (ADMXRC2_FPGA_MISMATCH) und ob genügend Speicher vorhanden ist, um die Datei temporär in den Speicher zu laden (ADMXRC2_NO_MEMORY).

- **ADMXRC2_SetClockRate()**

  Setzt die jeweiligen Frequenzen der Taktgeneratoren auf der Karte. Der zweite Parameter gibt den Index des Frequenzgenerators an. Bei der verwendeten Karte sind zwei Frequenzgeneratoren vorhanden. Zum einen ein Frequenzgenerator für den lokalen Bus (Index 0) und zum anderen ein Frequenzgenerator (Index 1), der beliebig vom Nutzer verwendet werden kann. Der lokale Busfrequenzgenerator ist an die Geschwindigkeit des PCI-Bus gebunden. Hier sind Frequenzen von 40 kHz bis 40 MHz möglich. Die reguläre Geschwindigkeit für den PCI-Bus beträgt für die CPU und die Karte 33,33 MHz. Diese wird auch für die Implementierung so verwendet. Für den zweiten Frequenzgenerator sind Frequenzen zwischen 0 Hz und 100 MHz möglich. Als Übergabeparameter für die Funktion können einmal die gewünschte Frequenz sowie die tatsächlich gesetzte Frequenz an die Funktion als Parameter 3 und 4 übergeben werden.

- **ADMXRC2_CloseCard()**

  Die Funktion CloseCard() deinitialisiert die verwendete Karte.

- **ADMXRC2_Malloc()**

  Über diese Funktion wird Speicher ähnlich zur malloc() Systemfunktion reserviert. Im Gegensatz zur Systemfunktion kann der mit ADMXRC2_Malloc() reservierte Speicher auch zur Nutzung für DMA-Transfer verwendet werden.

- **ADMXRC2_Free()**

  Hierüber findet eine Freigabe des über malloc() reservierten Speichers analog zur Systemfunktion free() statt. Der mit malloc() für DMAs reservierte Speicher wird mit dieser Funktion wieder freigegeben.

- **ADMXRC2_GetCardInfo()**

  Das C-Struct ADMXRC2_CARD_INFO enthält Informationen über die verwendete Karte. Mit der Funktion GetCardInfo kann dieses Struct mit den für die geöffnete Karte gültigen Daten gefüllt werden. Das Struct enthält Informationen über die Seriennummer der Karte, über die Anzahl der Frequenzgeneratoren, DMA-Kanäle und RAM-Bänke, sowie über die Board- und Logik-Revisionsnummer.

- **ADMXRC2_GetSpaceInfo()**

  Liefert Informationen über den lokalen Adressraum. Wichtig ist hier insbesondere die virtuelle Basisadresse, über die die Register auf dem PCI-Bus angesprochen werden können.

- **ADMXRC2_GetBankInfo()**

  Die Funktion ermittelt die Werte der als zweitem Parameter übergebenen Memory-Bank und füllt das als Pointer übergebene Struct mit den Werten, die für die SSRAM-Bank gültig sind. Das Mitglied fitted des Structs gibt an, ob auf der Karte der Speicher überhaupt bestückt ist. Nur wenn dieser Integer-Wert größer als 0 ist, sind die Struct-Werte überhaupt gültig. Die anderen Werte enthalten Informationen über den Typ und die Größe der Memory-Bank.

- **ADMXRC2_GetStatusString()**

  Ein numerischer Fehlercode kann durch die Funktion GetStatusString in eine lesbare Fehlermeldung umgewandelt werden. Als Rückgabewert erhält man in diesem Fall einen String. Als einziger Parameter ist der Fehlercode zu übergeben.

- **ADMXRC2_SetupDMA()**

  SetupDMA ermöglicht die Zuweisung eines Speicherbereiches zu einem DMA-Kanal, der durch malloc() reserviert wurde. Der Speicherbereich wird als zweiter Parameter übergeben

und dessen Größe als dritter Parameter. Es können zusätzlich noch weitere Parameter übergeben werden. Dies ist im vorliegenden SDK allerdings noch nicht implementiert. Hier muss derzeit zwingend noch eine Null übergeben werden. Als Rückgabewert erhält man einen DMA-Deskriptor, der danach für die Durchführung eines DMA-Transfers verwendet werden kann.

- **ADMXRC2_BuildDMAModeWord()**

  Für die Ausführung eines DMA-Transfers (Funktion DoDMA) wird ein 32-Bit-Wort benötigt, welches die Parameter für den Transfer enthält. Hier hilft die Funktion BuildDMAModeWord, die als Parameter 1 den verwendeten Typen des ADM-Boards übergeben bekommen muss. Im vorliegenden Fall ist dies ADMXRC2_BOARD_ADMXRC. Desweiteren wird die Breite für den Bustransfer übergeben. Da es PCI/PMC-Karten auch für 64 Bit gibt, bzw. es möglich ist, einen Transfer mit einer kleineren als der maximalen Busbreite durchzuführen kann dies hier selektiert werden. Im Weiteren wird nur die für die verwendete Karte maximale Breite von 32 Bit verwendet, da dies die maximale Tranferrate ermöglicht. Als dritter Parameter kann noch die Anzahl der WaitStates spezifiziert werden, die laut Alpha-Data aus Kompatibilitätsgründen nur den Wert 0 erhalten sollten. Als vierter Wert werden noch Parameter für den durchzuführenden DMA Transfer übergeben.

- **ADMXRC2_DoDMA()**

  Die Funktion führt einen DMA-Transfer aus und schreibt in den vorher mit malloc() reservierten und mit SetupDMA zugewiesenen Speicherbereich. Der über SetupDMA zugewiesene DMA-Deskriptor wird als zweiter Parameter übergeben. Es können auch weiterhin noch ein Offset im lokalen Speicher, die Länge, die Richtung, der DMA-Kanal und ein Timeout-Parameter für den DMA-Transfer spezifiert werden.

- **ADMXRC2_UnsetupDMA()**

  Die Funktion entfernt einen DMA-Deskriptor, der zuvor mit SetupDMA initialisiert worden ist. Parameter ist hier der zu entfernende DMA-Deskriptor.

A.3 Speicheradressierung der DL300-Module

Slot	high Byte	low Byte	Adresse (Hex)
0	wire 0 right	wire 0 left	0x0000
0	wire 1 right	wire 1 left	0x0400
1	wire 2 right	wire 2 left	0x0800
1	wire 3 right	wire 3 left	0x0C00
...	...	...	...

Tabelle A.2 DL310-Speicheradressierung

Slot	high Byte	low Byte	Adresse (Hex)
0	wire 0 right	wire 0 left	0x0000
0	wire 1 right	wire 1 left	0x0100
1	wire 2 right	wire 2 left	0x0200
1	wire 3 right	wire 3 left	0x0300
...	...	...	...

Tabelle A.3 DL305-Speicheradressierung

A.4 Register der DL300-Module

Name	Adresse	Typ	Bedeutung
IFDATA	0x400	R/W	Datenzugriff auf Adresse A0-A17, die über das AC-Register festgelegt wird
IFAUTO	0x401	R/W	Datenzugriff auf Adresse A0-A17 mit automatischer Adresserhöhung, dabei wird die Adresse über das AC-Register festgelegt und das Adressregister nach einem Schreibzugriff um 1 erhöht. Bei einem Lesezugriff wird die Adresse vor dem Lesezugriff erhöht
IFSCYCL	0x402	R/W	Datenzugriff mit gestrecktem Zyklus (gestreckt auf 14 μs), A0-A17 wird ebenfalls über AC-Register bestimmt. Der Zyklus ist für das Setzen der Baseline-DACs notwendig
IFACL	0x404	R/W	Zugriff auf die unteren 10-Bit des AC-Registers (Address Counter) über D0-D9
IFACH	0x405	R/W	Zugriff auf das AC-Register Bit 10-17 über D0-D7
IFCSR	0x406	R	Zugriff auf das Status-Register D0, D5, D6, D11, D12, D13, D14 und D15 (siehe Tabelle A.5)
IFCSR	0x406	W	Zugriff auf das Status-Register D0 und D1 (siehe Tabelle A.5)
IFSIGN	0x407	W	Setzen und Löschen des NIM-Signals über D2 am Frontpanel
IFWINDOW	0x000-0x3FF	R/W	Freier Speicherzugriff auf DL300-Adressen. Die Adresse wird dabei durch ADR=ACH+IFWINDOW bestimmt

Tabelle A.4 DL307-Register.

[68] In einigen Versionen der vorliegenden Dokumentation des DL300-Systems ist dies fehlerhaft angegeben

Datenleitung	Typ	Bedeutung
D0	R	Temperaturüberschreitung
D5	R	Fehler bei +8 Volt Spannungsversorgung
D6	R	Fehler bei -5 Volt Spannungsversorgung
D11	R	Interrupt-Flip-Flop durch externes Signal am Eingang gesetzt
D12	R	Externes NIM-Status-Signal aktiv
D13	R	Bus-Request (BRQ) auf internem Datenbus gesetzt
D14	R	Bus-Acknowledge (BAK) auf internem Datenbus gesetzt
D15	R	Processor-Request (PRQ) auf internem Datenbus gesetzt
D0	W	setzt NIM-Signal auf externem Trigger Ausgang
D1	W	löscht das Interrupt-Flip-Flop

Tabelle A.5 DL307-Statusregister.

Interne Adresse (IA)	Beschreibung
4	Lesen des Address-Counters und Start des Scanners (Sampling- oder Fastscan-Modus)
5	Lesen des Address-Counters und Stop des Scanners (Sampling- oder Fastscan-Modus)
6	Lesen des Address-Counters und Zurücksetzen des Scanners (Initialisierung)
7	Lesen des Address-Counters, Löschen des Hit-Flags und fortführen des Fastscan
8	Laden des Funktionsregisters (siehe Tabelle A.7)
9	Setzen des Scanner-Address-Counters (Startadresse für Sample-Modus oder Fastscan-Modus)
10	Scannerüberlauf-Register, setzt die maximale Scanneradresse für einen Fastscan (Modulbegrenzung)
11	Setzen des Hit-Threshold-Registers, Wire left über Bit 0-5, Wire right über Bit 8-13[68]
12	Starten des Scanners
13	Stoppen des Scanners
14	Zurücksetzen des Scanners
15	Löschen des Hit-Flags

Tabelle A.6 DL302-Scanner - Interne Adressen.

Bit	Beschreibung
0,1	Frequenzwahl: 00 - externer Frequenzeingang, 01 - externe Taktfrequenz, 10 - 50MHz, 11 - 100MHz
2	Fastscan-Modus, erkenne Ende eines Treffers
3	Common-Start (0) oder Common-Stop-Modus (1)
4	Fastscan, erkenne Anfang eines Treffers
2,4	Bit 2+4 0: Sample-Modus

Tabelle A.7 DL302-Scanner Funktionsregister.

A.5 Quellcodedateien des Interfaces

Komponente	Beschreibung	Dateiname
plxddsm	Demand-Mode-DMA-Statemaschine	-
plxdmsm	Direct-Master-Statemaschine	-
plxdssm	Direct-Slave-Statemaschine	plxdssm.vhd
sfifo	Synchrones 32-Bit-FIFO	sfifo.vhd und sfiforam_v.vhd

Tabelle A.8 Package sdkcomp (Datei sdkcomp.vhd)

Komponente	Beschreibung	Dateiname
dl_cycle	Statemaschine DL300 Adressierungszyklus	dl_cycle.vhd
dl_read	Statemaschine DL300 Lesezugriff	dl_read.vhd
dl_write	Statemaschine DL300 Schreibzugriff	dl_write.vhd
dl_setdac	Statemaschine DL300 DAC-Ansteuerung (Pedestals)	dl_setdac.vhd
dl_samp	Statemaschine DL300 Samplingmodus	dl_samp.vhd
dl_fast	Statemaschine DL300 Initiierung Fastscanmodus	dl_fast.vhd
dl_hit_search	Statemaschine DL300 Fastscan und Suche nach Treffern	dl_hit_search.vhd
dl_xfer_hits	Statemaschine DL300 Transfer der Daten vom DL300 ins SSRAM	dl_xfer_hits.vhd

Tabelle A.9 Package dl300sm (Datei dl300sm.vhd)

Komponente	Beschreibung	Dateiname
dl300_interface	Hauptprojektdatei, welche alle anderen Dateien einbindet	dl300_interface.vhd
zbt_dpins	Generiert notwendige Ein- und Ausgabe-Puffer für die SSRAM Bauteile	zbt_dpins.vhd
zbt_port	Modul, das die Verbindung zu einem SSRAM-Bauteil definiert	zbt_port.vhd
clocks	Generiert den Frequenzgeber für das ZBT-SSRAM	clocks_dll2.vhd

Tabelle A.10 Quellcodedateien des DL300-Interface-Projekts.

Komponente	Beschreibung	Dateiname
dl_analyze_sum	Bildung einer Hardwaresumme (Integral)	dl_analyze_sum.vhd
dl_analyze_sum_lin	Bildung einer linearisierten Hardwaresumme (Integral)	dl_analyze_sum_lin.vhd

Tabelle A.11 Package dl_analyze.

A.6 Einbindung der Direct-Slave-Statemaschine

```
1               plxdssm0: plxdssm
2                       port map(
3                               clk         => clk,
4                               rst         => rst,
5                               sr          => logic0,
6                               qlads       => qlads,
7                               lblast      => lblast_i,
8                               lwrite      => lwrite_i,
9                               ld_oe       => oe_ld,
10                              lready      => lready_o,
11                              lready_oe   => lready_oe,
12                              lbterm      => lbterm_o,
13                              lbterm_oe   => lbterm_oe,
14                              transfer    => ds_xfer,
15                              decode      => ds_decode,
16                              write       => ds_write,
17                              ready       => ds_ready,
18                              stop        => ds_stop);
```

Quellcode A.1 Statemaschine für PCI-Transfers.

A.7 Prozesse zum Lesen und Schreiben der PCI-Register

Prozessname	Beschreibung	R/W
gen_oe_regs	Register 0x0-0x6A,0x200-0x213	R
gen_oe_regs_debug	Register 0x100-0x101 (Debug)	R
registers	Register 0x01,0x2,0x3,0x8,0x9,0xA,0xC-0x12,0x20,0x21,0x23,0x50,0x51,0x100,0x102,0x103,0x200-0x203	W

Tabelle A.12 Prozesse zum Lesen und Schreiben der PCI-Register.

A.8 PCI-Register des DL300-Interfaces

Adresse	Bit	R/W	VHDL Signalname	Beschreibung	Initwert
0x0	32	R	firmware_release	Firmwaredatum und Revisionsnummer	abhängig von Versionsnummer
0x1	1	R/W	pipeline_reg	SSRAM-Pipeline-Register	0
0x2	2	R/W	size_reg	SSRAM-Size-Register	0
0x3	8	R/W	page_reg	SSRAM-Page-Mode-Register	0
0x4	32	R	info_reg	SSRAM-Info-Register	0
0x5	32	R	status_reg	SSRAM-Status-Register	0
0x6	1	R	fpga_status_reg	FPGA-Status-Register	0
0x7	32	R	hit_fifo_status	Hit-FIFO-Status-Register	0
0x8	16	R/W	dl300_fastscanmode	Fastscan-Modus	0
0x9	16	R/W	dl300_scannerstoppos	Scanner-Stop-Position	0
0xA	16	R/W	dl300_scanthreshold	Trefferschwelle	0
0xB	8	R	dl300_hitcounter	Anzahl gefundener Ansprecher	0
0xC	1	R/W	dl300_moduletype	Modultyp	0
0xD	8	R/W	dl300_maxhits	Maximale Hitanzahl	0
0xE	8	R/W	dl300_readhits	Anzahl zu lesender Bytes	0
0xF	4	R/W	dl300_samplemode	Sampling-Modus	0
0x10	8	R/W	dl300_presamples	Anzahl Presamples vor Hitstart	0
0x11	10	R/W	dl300_hit_window_start	Hit-Zeitfenster Anfang	0
0x12	10	R/W	dl300_hit_window_stop	Hit-Zeitfenster Ende	0
0x13	-	R	-	Setze i_debug(1) auf logisch 1	0
0x14	-	R	-	Setze i_debug(1) auf logisch 0	0
0x20	11	R/W	dl300_debug_addr	Debug-Register DL300-Leseadresse	0
0x21	11	R/W	dl300_debug_addr	Debug-Register DL300-Leseadresse	0
0x22	16	R	dl300_read_data	Ausgabe Debug-Daten (gelesen)	0

Tabelle A.13 PCI-Register des DL300-Interfaces - Teil 1.

Adresse	Bit	R/W	VHDL Signalname	Beschreibung	Initwert
0x23	16	R/W	dl300_debug_write_data	Debug-Register DL300-Daten	0
0x24	-	R	dl300_debug_read_start	Debug Lesezugriff starten	0
0x25	-	R	dl300_debug_write_start	Debug Schreibzugriff starten	0
0x26	-	R	dl300_sample_start	Sampling starten	0
0x27	-	R	dl300_fastscan_start	Fastscan starten	0
0x28	-	R	- / (dl300_hit_xfer_start)	Reserviert / (Hittransfer starten)	0
0x30	32	R	dl300_error	Ausgabe Fehler-Register	0
0x31	16	R	dl300_fastscan_stop_pointer	Ausgabe Stop-Punkt Fastscan	0
0x40	19	R	fpga_ssram_addr	Ausgabe SSRAM-Adresse	0
0x41	1	R	dl300_rst_ram_addr	SSRAM-Adresse auf 0 zurücksetzen	0
0x42	1	R	dl300_statemachines_reset	Statemaschinen zurücksetzen (im Fehlerfall)	0
0x50	6	R/W	dl300_dac_number	DAC-Nummer die gesetzt werden soll	0
0x51	32	R/W	dl300_dac_value	DAC-Werte	0
0x60	32	R	-	Performance-Zähler zurücksetzen	0
0x61	32	R	dl300_free_counter	Freien Zähler auslesen (Bits 63 bis 32)	0
0x62	32	R	dl300_free_counter	Freien Zähler auslesen (Bits 31 bis 0)	0
0x63	32	R	dl300_sample_counter	Freien Zähler auslesen (Bits 63 bis 32)	0
0x64	32	R	dl300_sample_counter	Freien Zähler auslesen (Bits 31 bis 0)	0
0x65	32	R	dl300_fastscan_counter	Fastscan Zähler auslesen (Bits 63 bis 32)	0
0x66	32	R	dl300_fastscan_counter	Fastscan Zähler auslesen (Bits 31 bis 0)	0
0x67	32	R	dl300_hit_search_counter	Hitsuche Zähler auslesen (Bits 63 bis 32)	0
0x68	32	R	dl300_hit_search_counter	Hitsuche Zähler auslesen (Bits 31 bis 0)	0
0x69	32	R	dl300_hit_xfer_counter	Hit-Transfer Zähler auslesen (Bits 63 bis 32)	0
0x6A	32	R	dl300_hit_xfer_counter	Hit-Transfer Zähler auslesen (Bits 31 bis 0)	0

Tabelle A.14 PCI-Register des DL300-Interfaces - Teil 2.

Adresse	Bit	R/W	VHDL Signalname	Beschreibung	Initwert
0x100	32	R/W	dl300_dac_value	DAC-Werte	0
0x101	32	R	reg_read	Debug Leseregister ausgeben	0
0x102	32	R/W	debug_selector(0)	Setzen der Debug Ausgangsleitung 0	0
0x103	32	R/W	debug_selector(1)	Setzen der Debug Ausgangsleitung 1	0
0x200	32	R/W	dl300_hit_sum_start_stop(0)	Setzen des Integrationsbereiches für Hitsumme 0	0
...	...	...	...	...	...
0x203	32	R/W	dl300_hit_sum_start_stop(3)	Setzen des Integrationsbereiches für Hitsumme 3	0

Tabelle A.15 PCI-Register des DL300-Interfaces - Teil 3.

A.9 Beispiel Zeitüberschreitungszähler

```
1               gen_counter : process (rst,rst_force,clk,cnt_reset) is
2               begin
3                       if (rst='1' or rst_force='1' or cnt_reset='1') then
4                               counter <= (others => '0');
5                       elsif (clk'event and clk='1') then
6                               counter <= counter+1;
7                       end if;
8               end process;
```

Quellcode A.2 Beispiel eines Zählers zum Feststellen einer Zeitüberschreitung (Timeout).

A.10 Beispiele für Multiplexer

```
1       dl300_cycle_cmd_in <= dl300_cmd
2               when (dl300_read_busy = '0' and dl300_write_busy = '0') else
3               dl300_read_cmd_out
4               when (dl300_read_busy = '1' and dl300_write_busy = '0') else
5               dl300_write_cmd_out
6               when (dl300_read_busy = '0' and dl300_write_busy = '1') else
7               (others => '0');
8
9       dl300_cycle_rw_in <= dl300_rw
10              when (dl300_read_busy = '0' and dl300_write_busy = '0') else
11              dl300_read_rw_out
12              when (dl300_read_busy = '1' and dl300_write_busy = '0') else
13              dl300_write_rw_out
14              when (dl300_read_busy = '0' and dl300_write_busy = '1') else
15              '0';
```

Quellcode A.3 Multiplexer für dl_cycle Statemaschine, zur Übergabe des Signals cmd und rw.

```
1       dl300_read_start <= dl300_fastscan_read_start when
2               (dl300_fastscan_busy = '1') and (dl300_hit_search_busy = '0')
3               and (dl300_hit_xfer_busy = '0') else
4               dl300_hit_search_read_start when
5               (dl300_fastscan_busy = '1') and (dl300_hit_search_busy = '1')
6               and (dl300_hit_xfer_busy = '0') else
7               dl300_hit_xfer_read_start when
8               (dl300_fastscan_busy = '1') and (dl300_hit_search_busy = '1')
9               and (dl300_hit_xfer_busy = '1') else
10              dl300_debug_read_start when
11              (dl300_fastscan_busy = '0') and (dl300_hit_search_busy = '0')
12              and (dl300_hit_xfer_busy = '0') else
13              '0';
```

Quellcode A.4 Multiplexer für Signal start bei Lesezugriffen.

A.11 Format der Daten zum Setzen der DACs

23...18	17...12	11...6	5...0
Kanal 3	Kanal 2	Kanal 1	Kanal 0

Tabelle A.16 Format der Daten für das Setzen der Digital-Analog Konverters (DACs).

A.12 Setzen der Digital-Analog-Konverter

```
1       when st_write_dac_value =>
2               write_addr <= X"0402";
3               if (bitpos = 0) then
4                       next_state <= st_next_value;
5               else
6                       next_state <= st_next_bit;
7               end if;
8               if dac_value (6*valpos+bitpos)='1' then
9                       write_data <= "0000000000000001";
10              else
11                      write_data <= "0000000000000000";
12              end if;
13              c_state <= st_pre_write;
14
15      when st_next_bit =>
16              bitpos <= bitpos-1;
17              c_state <= st_write_dac_value;
18
19      when st_next_value =>
20              bitpos <= 5;
21              if (valpos>0) then
22                      valpos <= valpos-1;
23                      c_state <= st_write_dac_value;
24              else
25                      c_state <= st_idle;
26              end if;
```

Quellcode A.5 Setzen der DACs (dl_setdac).

A.13 Datenformat im SSRAM

Markierung (Bit 23-16)	Zeitfenster verwendet	Modulwechsel
11111110	nein	nein
11101110	nein	ja
01101110	ja	nein
01101110	ja	ja

Tabelle A.17 SSRAM-Hitmarkierung.

31...28	27...12	11...6	5...0
0xF	Position	rechts	links

Tabelle A.18 Format im SSRAM, erstes gelesenes Wertepaar.

31...28	27...24	23...18	17...12	11...6	5...0
0xD	n	Daten n rechts	Daten n links	Daten n-1 rechts	Daten n-1 links

Tabelle A.19 Format im SSRAM, weitere gelesene Werte (Datenwert n).

A.14 Registerbelegung DL300-Interfaces

Wert	VHDL-Signal	Wert	VHDL-Signal
0	dl300_busy	25	dl300_hit_search_start
1	dl300_read_busy	26	dl300_hit_xfer_start
2	dl300_write_busy	38	dl300_read_st
3	dl300_sample_busy	39	dl300_error(0)
4	dl300_setdac_busy	...	...
5	dl300_fastscan_busy	...	...
6	dl300_hit_search_busy	46	dl300_error(7)
7	dl300_xfer_busy	47	OR_reduce(dl300_error)
10	dl300_adc	50	fpga_local_toggle
11	dl300_dtack	51	fpga_ssram_rw
21	dl300_read_start	52	fpga_ssram_enable
22	dl300_write_start	60	dl300_stretched_mode
23	dl300_sample_start	70	clk
24	dl300_fastscan_start	71	clk2x

Tabelle A.20 Belegung Fehlerdiagnoseregister (Debugregister) Teil 1.

Wert	VHDL-Signal	Wert	VHDL-Signal
80	ds_xfer	91	zbt_write
81	ds_ready	100	Logischer Oder Busy Statemaschinen
82	ds_stop	101	dl300_sample_busy
83	ds_decode	102	dl300_fastscan_mybusy
84	ds_write	103	dl300_hit_search_mybusy
90	zbt_read	104	dl300_hit_xfer_mybusy

Tabelle A.21 Belegung Fehlerdiagnoseregister (Debugregister) Teil 2.

VHDL-Signal (Bit)	Statemaschine	Fehlersignal	Beschreibung
dl300_error(0)	dl_cycle	dl_error_out	Buserror - DTACK-Signal wird nach vorausgehendem ADS nicht gesetzt
dl300_error(1)	dl_cycle	dt_error_out	DTACK-Error - DTACK-Signal wird nicht zurückgenommen
dl300_error(2)	dl_read	dl_read_error_out	maximale Zykluszeit wird überschritten
dl300_error(3)	dl_write	dl_write_error_out	maximale Zykluszeit wird überschritten
dl300_error(4)	dl_samp	dl_sample_error_out	maximale Zykluszeit wird überschritten
dl300_error(5)	dl_setdac	dl_setdac_error_out	maximale Zykluszeit wird überschritten
dl300_error(6)	dl_fastscan	dl_fastscan_error_out	maximale Zykluszeit wird überschritten
dl300_error(7)	dl_hit_search	dl_hit_search_error_out	maximale Zykluszeit wird überschritten, zuviele Hits gefunden oder kein PRQ erhalten
dl300_error(8)	dl_xfer_hits	dl_xfer_hits_error_out	maximale Zykluszeit wird überschritten

Tabelle A.22 Belegung Fehlerregister (dl300_error).

A.15 API-Funktionen

Funktionsname	Beschreibung	Parameterbeschreibung
ReadHitCounter	Anzahl Ansprecher auslesen	Rückgabewert entspricht der Anzahl der Ansprecher auf dem gescannten Speicherbereich
SetHitWindowStartStop	Setzen des Hitzeitfensters mit Start und Stop	Start- und Stop-Position innerhalb der Speichergröße eines Moduls
GetHitWindowStart	Lesen des aktuellen Hitzeitfensters (Start)	Liefert Start-Position
GetHitWindowStop	Lesen des aktuellen Hitzeitfensters (Stop)	Liefert Stop-Position
SetScanThreshold	Setzen der Schwellen für einen Hitscan	Setzt den ScanThreshold entweder als kombinierten 16-Bit Wert oder mit zwei separaten Werten (high, low)
SetHitStumStartStop	Setzen des Hitsummenbereiches für die Hardwarehitsumme n	erster Parameter n Nummer der Hitsumme, zweiter und dritter Parameter Start- bzw. Stop-Position
GetHitSum	Lesen der Hardwarehitsumme n	erwartet einen Parameter n, liefert aktuelle Hardwarehitsumme n zurück
SetFastscanMode	Setzen des Fastscanmodus	Moduswort
SetScannerStop	Setzen des Fastscanmodus	Speicheradresse der Stop-Position. Muss kleiner oder gleich der Modulspeichergröße multipliziert mit der Modulanzahl sein
SetSampleMode	Setzen des Samplingmodus	Moduswort
SetSamplePresamples	Setzen der Anzahl Presamples	Anzahl der Speicherstellen, die als Presamples verwendet werden sollen

Tabelle A.23 API-Funktionen zum Setzen oder Lesen von Werten (Teil 1).

Funktionsname	Beschreibung	Parameterbeschreibung
SetModuleType	Setzen des Modultyps	DL305 entspricht dem Wert 0, DL310 entspricht dem Wert 1
SetNumModules	Setzen Anzahl der FADC-Module	Anzahl der Module, maximal 24 Module
SetReadHits	Zu lesende Speicherstellen	Anzahl der zu lesenden Speicherstellen je gefundenem Ansprecher
SetMaxHits	Maximale Hitanzahl	Anzahl der maximal erlaubten Ansprecher
GetStopPointer	Stop-Pointer lesen	liefert Stoppointer eines Fastscans zurück
GetFPGASSRAMAddr	SSRAM-Adresse lesen	SSRAM-Zeiger
GetErrorRegister	Fehlerregister lesen	liefert Fehlerdatenwort zurück
SetDebugSelector	Setzt Fehlerdiagnoseausgang	erster Parameter Nummer des Ausgangs, zweiter Parameter auszugebendes Signal
GetCounters	Zeitzähler lesen	Liefert C++-STL Vektor mit den 10 32-Bit-Werten der 5 64-Bit Zähler

Tabelle A.24 API-Funktionen zum Setzen oder Lesen von Werten (Teil 2).

Funktionsname	Beschreibung
StartSampling	Starten eines Erfassungsvorgangs
PerformFastScan	Starten eines Hitscanvorgangs mit anschließender Auslese
Reset	Ausführen eines Resets des DL300-Systems (Crate-Reset)
ResetRAMAddr	Zurücksetzen der SSRAM-Adresse auf Position 0
ResetStatemaschines	Statemaschinen in den Zustand st_idle versetzen und alle Parameter und Fehler zurücksetzen
ResetCounters	Setzt Zähler für Zeitmessung zurück
SetNIMOutput	Setzt NIM-Signal am Ausgang der DL307-Interfacekarte
GenerateTrigger	Erzeugt ein 100 ns Triggersignal am TRIGGER-Ausgang
ClearInterruptFF	Setzt das Interrupt Flip-Flop zurück
SSRAMRead	Auslesen des SSRAM-Speichers via DMA
DL307ReadStatusRegister	Lesen des Statusregisters (IFCSR) des DL307-Interface-Moduls
SetDAC	Setzen der Digital-Analog-Konverter zum Einstellen der Pedestals
PedAdjust	Automatisches Einstellen der Pedestals auf einen vorgegebenen Baselinewert
ReadSampleMemory	Lesen des Flash-ADC-Speichers über einfache Leseoperationen ohne DMA

Tabelle A.25 API-Funktionen zum Ausführen von Vorgängen oder Lese- und Schreibzugriffen.

B DAQ-Steuerungssoftware und Logsystem

Die Datenerfassung des Crystal-Barrel-Experiments besteht aus mehreren lokalen Eventbuildern. Hierbei verfügt jeder der Subdetektoren über mindestens einen eigenen lokalen Eventbuilder. Die zentrale Triggerkontrolle und die Synchronisierung erfolgt durch den globalen Eventbuilder. Die lokalen und der globale Eventbuilder schicken Ihre Daten an eine zentrale CPU, die die Daten aufbereitet und auf einem Speicher-System ablegt. Die Koordinierung und die Kontrolle der verschiedenen Prozesse auf den beteiligten CPUs erfolgt hierbei über eine Netzwerkschnittstelle. Die Funktionalität zum Steuern der lokalen Eventbuilder wird durch die Bibliothek daqctrl zur Verfügung gestellt. Der Experimentator benötigt eine Steuerungssoftware, mit der er über eine grafische Oberfläche die Steuerung und die Kontrolle des Experimentes vornehmen kann. Hierbei kann durch die Software eine Datenerfassung des Experiments (engl. Run) gestartet und gestoppt werden. Die DAQControl Software wurde unter dem Open-Source Desktop Gnome 1.4 im Rahmen dieser Arbeit entwickelt. Hierbei kamen die Technologien GConf, GTK, CommonC++, PostgreSQL und MySQL zum Einsatz.

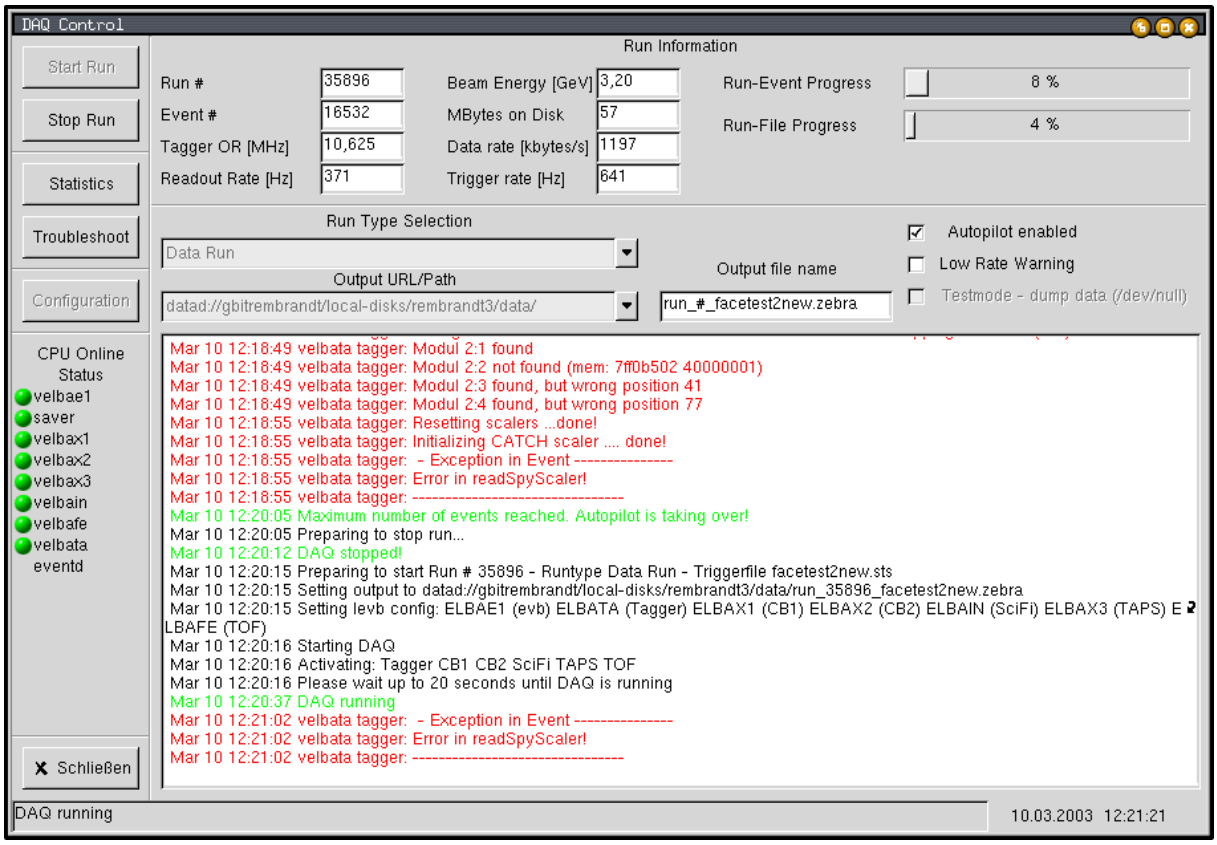


Abbildung B.1 DAQ-Steuerungssoftware DAQControl.

Die Elemente der Oberfläche wurden mit dem grafischen Tool Glade erstellt, welches Benutzeroberflächen für Gnome/GTK-Programme erstellen kann. Der Benutzer erhält die Möglichkeit, über Auswahlfelder und Knöpfe sowie Unterfenster Einstellungen zur Run-Nummer, zum verwendeten Triggerfile, zum Ausgabepfad und zum Dateinamen zu machen. Das Symbol # im

Dateinamen wird in der DAQControl durch die aktuelle Run-Nummer eingestellt, welche von der Software automatisch nach jedem Run hochgezählt wird, sofern es sich nicht um einen Testlauf handelt. Im Testmodus reagiert die Software genauso wie im normalen Produktionsbetrieb, es werden jedoch keine Daten in der Run-Datenbank abgelegt. Bei der Run-Datenbank handelt es sich um eine SQL-Datenbank, in der alle Informationen zum gestarteten bzw. abgeschlossenen Run abgelegt werden. Hierbei werden die Anzahl der Ereignisse im aktuellen Run, die Triggerbedigungen, sowie die Dateinamen und der Speicherort für spätere Abfragen festgehalten und mit Zeitmarken für den Start und den Stop des Runs versehen. In einem Unterfenster können desweiteren noch Angaben zum verwendeten Target und der Polarisation des Strahls eingegeben werden, welche ebenfalls in der Datenbank gespeichert werden.

Die Software ermöglicht es dem Nutzer, eine maximale Anzahl an Ereignissen vorzugeben, nach denen die DAQControl automatisch den Run stoppt und einen Neuen startet. Desweiteren ist es möglich eine maximale Dateigröße vorzugeben, nach deren Erreichen ein Run abgeschlossen und ein Neuer gestartet wird. In diesem automatischen Modus kann die Software die Datennahme vollständig autonom steuern, sofern es nicht zu einem Fehler im Datenerfassungssystem oder beim Beschleuniger kommt.

Aufgrund der Komplexität der verwendeten Software und insbesondere der vielfältigen und komplexen Hardware sind Fehlfunktionen oder Hardwaredefekte nie ganz auszuschließen. Aus diesem Grund werden im Zentralbereich des Hauptfensters neben Statusmeldungen der DAQControl (grün), Statusmeldungen der CPUs (schwarz) auch Fehlermeldungen der lokalen Eventbuilder, des globalen Eventbuilders oder des Eventsavers angezeigt (rot). Die hierfür notwendigen Daten erhält die DAQControl aus den lokalen Logdateien des Systems. Jeder lokale Eventbuilder, der globale Eventbuilder und der Eventsaver verfügen über eigene Logdaten. Diese Logdaten werden über Remote-Logging an den Syslog-Dienst übermittelt und zentral auf einem Syslogging-Server gesammelt, der gleichzeitig der zentrale Steuerrechner für die DAQ ist, auf welchem auch die DAQControl-Software betrieben wird. Dadurch sind alle Logdaten auf dem Rechner für die Anzeige in der DAQControl bzw. in Terminals verfügbar.

Für den Loggingdienst wurden die für die lokale Benutzung definierten Logging-Facilities local0 bis local7 verwendet. Diese wurden, wie in Tabelle B.1 zu sehen, den verschiedenen Schichten der Auslese zugeordnet.

Durch die hier definierten Ebenen ist es möglich, die Fehlerinformationen der Schichten getrennt in Dateien zu speichern und somit auch im Fehlerfall getrennt zu analyiseren. Neben den benutzten Schichten, die auf Syslog-Facilities abgebildet werden, können auch verschiedene Fehlerklassen unterschieden werden. Hierbei werden die Syslog-Fehlerklassen debug, info, notice, warn und error verwendet und ebenfalls in eigene Dateien unterteilt. Damit ergibt sich auf dem zentralen Syslogging-Server die Konfiguration, die in Quellcode B.1 zu sehen ist.

Die weiteren Facilities local1 bis local7 sind analog implementiert. Desweiteren sind für die Zusammenführung der Loginformationen Dateien definiert, welche die vollständigen Daten aller

Facility	DAQ Schicht	Beschreibung
local0	SYNC	Synchronisation (Syncmodul, Eventsaver)
local1	READOUT	Auslese der Subdetektoren
local2	TRANSPORT_SENDER	Datentransport (Senderseite)
local3	TRANSPORT_RECEIVER	Datentransport (Empfängerseite)
local4	TRANSPORT_ASSEMBLER	Zusammensetzung der Einzeldaten zu vollständigen Ereignissen
local5	ENCODING	Erzeugung der Datenstrukturen
local6	STORAGE	Speicherung der Daten
local7	CONTROL	Steuerung und Kontrolle des Experiments (DAQControl und Slowcontrol)

Tabelle B.1 Zuordnung Logging-Facilities zu DAQ-Schichten.

```
local0.debug                    -/var/log/DAQ/sync.debug
local0.info                     -/var/log/DAQ/sync.info
local0.notice                   -/var/log/DAQ/sync.notice
local0.warn                     -/var/log/DAQ/sync.warn
local0.err                      -/var/log/DAQ/sync.error
```

Quellcode B.1 /etc/syslog.conf des Remote Logging-Servers.

Facilities enthalten. Auf diese greift die DAQControl zu, um die Informationen für die Darstellung im zentralen Logfenster zu gewinnen. Für den Syslogdienst sind auf den lokalen Maschinen die Einträge in der syslog.conf Datei definiert, die in Quellcode B.2 zu sehen sind.

```
local0.*                        @cbcontrol1
local1.*                        @cbcontrol1
local2.*                        @cbcontrol1
local3.*                        @cbcontrol1
local4.*                        @cbcontrol1
local5.*                        @cbcontrol1
local6.*                        @cbcontrol1
local7.*                        @cbcontrol1
```

Quellcode B.2 /etc/syslog.conf der Client-PCs.

Durch die Verwendung von Logdateien ist es möglich, dass auch entfernte Benutzer die Ausgaben der Datenerfassung analysieren und beobachten können. Hierzu ist ein Login über einen Secure-Shell-Dienst vorgesehen, so dass eine Problemanalyse schnell von externen Nutzern erfolgen kann, ohne dass hierzu Informationen vom lokalen Terminal benötigt werden.

In Abbildung B.1 ist das Hauptfenster der Software zu sehen. Im weißen Bereich unten ist das

bereits erwähnte Fenster für Lognachrichten sichtbar, das den überwiegenden Teil des Platzes einnimmt. Im linken Bereich sind oben die Steuerungsknöpfe zu sehen, die es dem Schichtpersonal ermöglichen eine Datenerfassung zu starten bzw. zu stoppen. Alle CPUs sind darunter mit einem Statuslämpchen abgebildet. Grün bedeutet hierbei, dass die CPU betriebsbereit ist, die DAQ-Software auf dem Gerät in Funktion ist und über das lokale Steuerungsnetz zu erreichen ist[69]. Notwendig für den Start der DAQControl-Software ist eine Kontrollverbindung über TCP/IP zu den beiden Maschinen velbae1 (Globaler Eventbuilder) und saver (Eventsaver). Die Verbindungen werden während der Laufzeit der DAQControl überwacht und im Fehlerfall oder bei Verbindungsabbruch (z.B. Neustart einer Maschine) automatisch wieder neu initiiert. Die Anzeigen im oberen Bereich enthalten (von oben nach unten und links nach rechts) die Werte für die aktuelle Run-Nummer, die Eventnummer, die Tagger-OR-Rate (Indikator für den Photonenfluss), die aktuelle Ausleserate (in Hz), die aktuelle Elektronenstrahlenergie im Beschleuniger, die Menge der geschriebenen Daten in MByte, die Datenrate in kbyte/s und die aktuelle Triggerrate. Im Bereich rechts oben sind zwei Fortschrittsbalken zu sehen, die jeweils den Anteil der erreichten Ereignisse (Events) oder den Fortschritt beim Schreiben der Daten in Prozent der maximal vorgegebenen Dateilänge oder der maximalen Ereignisanzahl zeigen. Im Automatikmodus (Autopilot) wird der Run gestoppt, sobald eine der Anzeigen 100% erreicht. Daraufhin wird ein neuer Run gestartet. Der Autopilot lässt sich über ein Kontrollkästchen aktivieren, das sich unter den beiden Fortschrittsanzeigen befindet. Desweiteren lässt sich ein Testmodus aktivieren sowie eine Warnung bei einer zu geringen Tagging-Rate.

Die DAQControl lässt sich auch bei laufender Datenerfassung schließen, ohne dass hierbei die Datenerfassung unterbrochen werden muss. Desweiteren wurde die Software so konzipiert, dass mehrere Benutzer gleichzeitig eine DAQControl öffnen können. Jeder Benutzer verfügte bisher über eine eigene Konfiguration in der GConf-Datenbank des jeweiligen Gnome-Desktops. Damit alle Nutzer über dieselbe Konfiguration verfügen, wurde in der Umbauphase für das Transregio-Experiment die Datenbank von GConf auf eine zentrale SQL-Datenbank umgestellt, die ebenfalls in der Umbauphase auf ein neues hochverfügbares Serversystem migriert wurde. Die Run-Datenbank wurde hierbei von PostgreSQL auf MySQL umgestellt, so dass hiermit jetzt eine einheitliche Datenbank für die Runinformationen, die DAQControl und die Slowcontrol existiert. Die grafische Oberfläche wurde seit der Verwendung während der CB-ELSA Strahlzeiten seit dem Jahr 2002 nicht weiterentwickelt, da hier aufgrund der allgemeinen Entwicklung des Gnome-Desktops eine aufwendigere Migration auf die neuen GTK-Bibliotheken ansteht. Auf mittel- bis langfristige Sicht ist hier zu überlegen, ob die bestehende Applikation auf die neuen GTK-Bibliotheken portiert werden soll oder ob später ein webbasierter Ansatz verfolgt wird. Da die Daten der DAQControl in der MySQL-Datenbank vorhanden sind und eine C/C++-Bibliothek für die Ansteuerung der DAQ (libdaqctrl) vorhanden ist, wäre die Entwicklung einer webbasierten Oberfläche sehr schnell möglich und könnte parallel zur DAQControl betrieben bzw. weiterentwickelt werden. Dadurch lässt sich eine Steuerung auch ohne spezielle Software realisieren.

[69] TCP/IP-Verbindung aufgebaut

C Synchronisationsmodul

Um zu gewährleisten, dass die Daten der Subprozessoren synchron sind und die Ereignisse der lokalen Eventbuilder auf dem zentralen Eventsaver synchronisiert werden können, wurde im Rahmen dieser Arbeit ein Synchronisationsmodul entwickelt, welches die Synchronisation der Daten durch eine dedizierte Hardware unterstützen soll.

C.1 Hardware

Für das Modul wurde das bestehende Design eines 32-Bit-Zählermoduls verwendet und modifiziert. Das Zählermodul wurde zuvor im Saphir-Experiment zur Auslese der Proportionaldrahtkammern des Tagging-Systems verwendet [Wit95]. Das Modul verfügt über 4 NIM-Eingänge und einen 16 Bit breiten ECL-Bus. Über einen der NIM-Eingänge wird das Readout-Signal des Experiments zugeführt (das Computer-Trigger-Signal), um eine einstellbare Zeit verzögert und an die serielle Schnittstelle des lokalen Eventbuilders als Interrupt-Signal weiterzugeben (siehe Kapitel D). Ein weiterer Eingang dient zum Zurücksetzen des Moduls (SYSRESET). Beide Signale werden von der Trigger-Control-Unit (TCU) erzeugt. Auf dem dritten Eingang wird ein konstantes 20 MHz Signal zugeführt, welches an die 8 auf dem Modul vorhandenen Zähler weitergeleitet wird und als deren Taktgeber dient.

Die Aktivierung der Zähler erfolgt durch die auf den programmierbaren Bausteinen (PALs bzw. GALSs) gesetzten logischen Gleichungen. Da die Zähler zu unterschiedlichen Zeiten während der Datennahme aktiviert werden, können hierbei sehr genaue Zeitanalysen des Datenerfassungssystems durchgeführt werden. Tabelle C.1 enthält die Belegung des 16-Bit breiten ECL-Bus.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Frei	Frei	Frei	Frei	Frei	Frei	Addr Select	Cmd Strobe	Buf Bit 4	Buf Bit 3	Buf Bit 2	Buf Bit 1	Buf Bit 0	Int_2	Int_1	Int_0

Tabelle C.1 Belegung ECL-Synchronisationsbus.

Die unteren drei Bits können Informationen zum Triggertyp (Interruptqualifier) tragen. Dadurch kann im lokalen Eventbuilder eine spezielle Auslese aktiviert werden, wenn dies gewünscht ist.

Jeder lokale Eventbuildern verfügt über ein Modul. Im globalen Eventbuilder wird zusätzlich ein Latchmodul eingesetzt, das über ECL und NIM Aus- und Eingänge verfügt und die Datenleitungen OK und Busy der lokalen Eventbuilder auslesen, sowie eine einheitliche Puffernummer (Bit 3-7) an die Synchronisationsmodule verteilen kann. Die Aktivierung der LEVBs geschieht wie bereits beschrieben über das Setzen der ADDR_SELECT-Leitung. Das Latchmodul im globalen Eventbuilder verfügt insgesamt über 16 Ein- sowie 16 ECL-Ausgaberegister und ist über VME

steuerbar. Die Eingangsregister werden vollständig für die Verarbeitung der von den CPUs eingehenden BUSY- und OK-Signale verwendet. Da jede CPU beide Signale generiert, ergeben sich mit 8 CPUs insgesamt 16 Eingangssignale. Somit ist es möglich mit einem Latchmodul im globalen Eventbuilder insgesamt 7 LEVBs und das Synchronisationsmodul im globalen Eventbuilder zu versorgen.

15	14	13	12	11	10	9	8
BUSY CPU 8	BUSY CPU 7	BUSY CPU 6	BUSY CPU 5	BUSY CPU 4	BUSY CPU 3	BUSY CPU 2	BUSY CPU 1
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OK CPU 8	OK CPU 7	OK CPU 6	OK CPU 5	OK CPU 4	OK CPU 3	OK CPU 2	OK CPU 1

Tabelle C.2 Belegung ECL-Eingang Latchmodul.

Die 16 Ausgangsregister werden auf den ECL-Bus gegeben, an den auch die lokalen Eventbuilder angeschlossen sind. Die Belegung ist analog zu der des ECL-Busses an den lokalen Eventbuildern (siehe Tabelle C.1).

Das Synchronisationsmodul wurde im Rahmen dieser Arbeit umgebaut und außerdem wurde die entsprechende Bibliothek für die Ansteuerung des Moduls entwickelt. Im weiteren soll hier kurz die Software und deren Funktionsweise beschrieben werden.

C.2 Software

In der Bibliothek libsync sind die Funktionen für das Modul im globalen und im lokalen Eventbuilder zusammengefasst. Die Bibliothek besteht im Wesentlichen aus den beiden C++-Klassen:

- CSyncClientModule - Klasse für den lokalen Eventbuilder
- CSyncServerModule - Klasse für den globalen Eventbuilder

Die Ein- und Ausgänge der beiden Module lassen sich via VME setzen bzw. lesen. Da die Ausführung von Lese- bzw. Schreibkommandos und von Bitoperationen auf den einzelnen VME-Registern sehr zeitkritisch ist, wurde eine C++-Klasse entwickelt, die den Wert eines Registers nach einer Lese- oder Schreiboperation in einer lokalen Variablen speichert. Dadurch können Bitoperationen auf diesen Registern mit den Schattenkopien (engl. Shadow) arbeiten und müssen keine Leseoperationen auf dem VME-Bus ausführen. Für ein Register auf dem VME-Bus wird kein direkter Zugriff über die VME-Klasse durchgeführt, sondern es wird ein Shadow-Register angelegt.

Hierfür wurden die beiden Klassen

- CVMERegister - Klasse ohne Software Byteswapping (benötigt Hardwareswapping)
- CVMERegisterSwapped - Klasse mit Software Byteswapping

erstellt. Hierbei fand die Klasse CVMERegisterSwapped nur auf einem ersten VME-Testsystem der Firma Eltec Einsatz, da auf diesem System keine Hardwarekonvertierung zwischen Little- und Big-Endian[70] möglich ist und somit in Software bei jedem VME-Zugriff realisiert werden musste. Durch Modifikation des Quellcodes ist für Maschinen ohne Hardwarekonvertierung der Einsatz der Synchronisationsbibliothek ebenfalls möglich. Am Experiment wurden allerdings im Weiteren nur CPUs der Firma VMIC eingesetzt, die alle über eine Hardwarekonvertierung verfügen.

Die Shadowregister-Klasse enthält Funktionen für das Lesen und Zuweisen eines Wertes sowie für die Bitoperationen logisch ODER und logisch UND (engl. OR und AND).

Die Klasse CSyncClientModule enthält die folgenden relevanten Funktionen:

```
1  inline void setOK();
2  inline void clearOK();
3  inline void setBusy();
4  inline void clearBusy();
5  void waitActive();
6
7  inline int waitTrigger();
8  inline unsigned char readBufferNumber();
9
10 inline void clearCounters();
11 inline unsigned long readCounter(int n);
12 vector <unsigned long> readCounters();
13 inline unsigned long readEventCounter();
14 inline unsigned long readFreeClockCounter();
15 inline unsigned long readLifetimeCounter();
16 inline unsigned long readDeadTimeCounter1();
17 inline unsigned long readDeadTimeCounter2();
18
19 inline unsigned char activeDeadTimeCounter();
20 inline unsigned char readInterruptqualifier();
21
22 inline bool active();
23 void clearTrigger();
```

Quellcode C.1 CSyncClientModule-Klasse.

[70] Little- und Big-Endian bezeichnen zwei verschiedene Methoden, um einen Datenwert im Speicher einer CPU abzulegen. Sind die Ziffern mit fallender Wertigkeit von links nach rechts angeordnet, so nennt man dies Big-Endian, im umgekehrten Fall Little-Endian. Der VME-Bus arbeitet mit Big-Endian-Kodierung, Intel x86-Prozessoren mit Little-Endian-Kodierung.

Zwei ECL-Leitungen des Moduls sind für die Statusmeldungen an den globalen Eventbuilder vorgesehen. Dies sind die beiden Leitungen OK und BUSY. Sobald der lokale Eventbuilder das Startsignal für die Auslese erhalten hat, wird das BUSY-Signal (beschäftigt) gesetzt. Nach Abschluss der Auslese wird das OK-Signal gesetzt. Diese Signalisierung ermöglicht es dem globalen Eventbuilder festzustellen, wann die lokalen Eventbuilder die Auslese beginnen oder diese beendet haben. In der Datenerfassung werden hierzu die Funktionen setOK, clearOK, setBusy und clearBusy verwendet, die in der gemeinsamen Interfaceklasse zur DAQ (IControlClient) ebenfalls definiert sind. Ob ein lokaler Eventbuilder an der Auslese teilnehmen soll, kann über den globalen Eventbuilder festgelegt werden. Zur Aktivierung eines LEVB wird die Buffernummer gesetzt. Parallel dazu wird die ADDR_SELECT-Leitung auf Logisch 1 gesetzt, um zu signalisieren, dass eine Aktivierung eines Moduls stattfinden soll. Bei der Übermittlung der Buffernummer bleibt dieses Signal normalerweise auf Logisch 0. Ein lokaler Eventbuilder ruft nach der Initialisierung die Funktion waitActive (Zeile 5) auf. Diese enthält eine Warteschleife, die erst beendet wird, wenn die Active-Leitung des jeweiligen LEVB vom globalen Eventbuilder gesetzt wird. Sofern keine Datennahme durchgeführt wird oder der lokale Eventbuilder an der Datennahme nicht beteiligt ist, verharrt der LEVB im Ruhezustand in dieser Funktion.

Die Funktion waitTrigger (Zeile 7) wartet auf ein Triggerereignis, welches am seriellen Port generiert wird. Dies wird detailliert in Anhang D beschrieben.

Die Funktion readBufferNumber liest die aktuelle Buffernummer, die vom globalen Eventbuilder vorgegeben wird. Diese ist für die Synchronisation der Ereignisse zuständig und wird in jedem Event an den Eventsaver vom LEVB übermittelt. Die Buffernummer wird bei jedem Ereignis vom globalen Eventbuilder erhöht und durchläuft zyklisch die Werte von 0-31. Der Eventsaver kann anhand der Werte der Buffernummer überprüfen, ob einer der lokalen Eventbuilder eines der Ereignisse übersprungen oder verloren hat und somit die Synchronisierung der Daten durch einen Hardwarezähler verifizieren.

Die Funktion clearCounters löscht alle 8 Hardwarezähler auf dem Modul. Mit der Funktion readCounter kann ein Zähler gelesen werden. Als Parameter werden hierzu die Werte 0-7 akzeptiert. Die Funktion readCounters liest alle 8 Zähler nacheinander aus und übergibt die Werte in einem STL-Vector an das aufrufende Programm (Zeile 12).

Aktuell sind auf dem Hardwareboard 5 der 8 Hardwarezähler implementiert. Die jeweilige Funktion der Zähler ist der Tabelle C.3 zu entnehmen.

Zähler 4 dient der Normierung. Hierzu wird der Zähler permanent mit einer konstanten Frequenz von 20 MHz getaktet. Die übrigen Zeitzähler erhalten ebenfalls diese Taktfrequenz, zählen jedoch nur zu bestimmten Zeiten weiter. Die Lebendzeitzähler (Lifetime) und Totzeitzähler (Deadtime) werden jeweils dann weitergezählt, wenn sich die Datenerfassung im jeweiligen Zustand befindet. Dabei bezeichnet die Lebendzeit die Zeitdauer, während der die Datenerfassung auf ein neues eingehendes Ereignis wartet. Die beiden Totzeitzähler werden hingegen während der Auslese des jeweiligen Detektors weitergezählt. Der Eventcounter (Zähler 3) zählt

Zählernummer	Funktion	Bedeutung
3	readEventCounter	Anzahl der Ereignisse
4	readFreeClockCounter	20 MHz-Zähler
5	readLifeTimeCounter	Lifetime-Zähler
6	readDeadTimeCounter1	Totzeitzähler 1
7	readDeadTimeCounter2	Totzeitzähler 2

Tabelle C.3 Hardwarezähler auf dem Syncmodul.

die Anzahl der Ereignisse, die durch die Datenerfassung aufgezeichnet wurden und ist somit nicht mit dem Taktgenerator für die 20 MHz Taktfrequenz verbunden. Für genaue Zeitanalysen der Datenerfassung des Experiments auf Basis der oben genannten Zähler sei auf eine Dissertation [Sch04] verwiesen.

D Serielles Interrupt-Kernelmodul

Für jedes aufzunehmende Ereignis wird im Experiment ein Triggersignal erzeugt. Dieses Signal dient dazu, die Datenerfassung aller am Experiment aktiven Detektoren zu starten. Hierbei ist es wünschenswert, eine Unterbrechung (engl. Interrupt) bei den beteiligten Prozessoren zu erzeugen. Dies ermöglicht die zeitnahe Bearbeitung und Auslese der Ereignisse und belastet die Ressourcen des jeweiligen Rechners nur minimal, da die Prozessoreinheit nicht ständig den Triggerstatus überwachen muss.

Bevor die neue Datenerfassung im CB-ELSA-Experiment auf Basis der in Kapitel 2.11 beschriebenen VMIC-CPUs umgestellt wurde, fand die Erzeugung der Interrupts über die parallele Schnittstelle der Motorola VME-CPUs statt. Da dies für den Einsatz mit den neuen CPUs so nicht mehr möglich war, wurde nach entsprechenden Alternativen gesucht. Aufgrund der begrenzten Anschlussmöglichkeiten kamen hierbei die serielle und die USB-Schnittstelle in Frage. Prädestiniert für den Anschluss von Interruptleitungen wäre auch hier eine parallele Schnittstelle gewesen, die allerdings auf den VMIC VME-CPUs nicht zur Verfügung stand. Hierdurch wäre es möglich gewesen, mehrere Eingangsleitungen für verschiedene Interruptsignale zu verwenden sowie über Ausgangsleitungen weitere Signale erzeugen zu können.

Zunächst wurde versucht, über einen USB-nach-Parallel-Konverter mit einem entsprechenden USB-Treiber die gewünschte Funktionalität zu erreichen. Allerdings wurde dies schnell wieder verworfen, da durch die USB-Treiber ein zusätzlicher Overhead entstanden wäre, der die Interrupts nicht zeitnah genug hätte auslösen können.

Als elegante Methode bietet sich die Verwendung der Steuerungsleitungen für die Flusskontrolle der seriellen Schnittstelle an. Bei einer seriellen Schnittstelle nach RS-232 Standard kommt hier die Leitung CTS (clear-to-send) in Frage. Diese Eingangsleitung liegt auf Pin 8 einer 9-poligen RS-232 Schnittstelle an und wird im Normalfall bei einer Hardwareflusskontrolle z.B. zwischen einem Modem und einem PC verwendet. Die sogenannte RTS (ready-to-send) Leitung ist ein Ausgang, der als Eingang auf der CTS-Leitung der Gegenstelle anliegt. Die beiden Leitungen sind also zwischen zwei Geräten im Normalfall über kreuz angeschlossen.

Um nun sensitiv auf die Änderung des CTS-Signals zu sein, musste der serielle Treiber des verwendeten Betriebssystems modifiziert werden; nach Möglichkeit so, dass nur eine der beiden Schnittstellen durch den seriellen Interrupttreiber blockiert wird. Dadurch bleibt die zweite Schnittstelle der VMIC-CPU frei zur anderweitigen Nutzung. Hierzu wurde der serielle Treiber des Linux 2.4.18 Kernels (SuSE 8.0 Standardkernel) modifiziert und später an den Kernel 2.4.27 der Debian 3.1 Sarge Distribution angepasst.

Zunächst wurden alle Funktionen entfernt, die für den Treiber nicht relevant sind. Dies sind insbesondere die Funktionen für die Übertragung von Zeichen über die serielle Schnittstelle, wie in Quellcode D.1 aufgeführt.

```
inline int serial_paranoia_check()
void rs_stop()
void rs_start()
_INLINE_ void rs_sched_event()
_INLINE_ void receive_chars()
_INLINE_ void transmit_chars()
_INLINE_ void check_modem_status()
void rs_put_char()
void rs_flush_chars()
int rs_write()
int rs_write_room()
int rs_chars_in_buffer()
void rs_flush_buffer()
void rs_send_xchar()
void rs_throttle()
void rs_unthrottle()
int get_serial_info()
int set_serial_info()
int get_lsr_info()
int get_modem_info()
int set_modem_info()
int do_autoconfig()
void send_break()
void rs_break()
int get_multiport_struct()
int set_multiport_struct()
void rs_set_termios()
void rs_hangup()
inline void wait_for_xmitr()
void serial_console_write()
int serial_console_wait_key()
kdev_t serial_console_device()
int __init serial_console_setup()
```

Quellcode D.1 Aus dem seriellen Treiber entfernte Funktionen.

Für die Kommunikation mit dem Gerät wurde dafür die sogenannte Major-Nummer[71] 244 verwendet. Diese wird vom Standard-Linuxkernel nicht verwendet und ist für Treiber verwendbar, die nicht offiziell im Linux-Quellcode enthalten sind.

Die entsprechende Gerätedatei in /dev wird über das in Quellcode D.2 gezeigte Kommando angelegt.

[71] Majornumber: jeder block- oder zeichenorientierte Treiber im Kernel ist über eine Major- und Minor-Nummer definiert. Die vom Linux-Kernel definierten Majornummern sind der Datei Documentation/devices.txt in Linuxquellcode zu entnehmen. Das statische Major/Minor-System wird derzeit durch das Konzept udev abgelöst. Siehe http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf

```
mknod /dev/serial_intr char 244 0
```

Quellcode D.2 Makenode Kommando (mknod) für die Erzeugung der Gerätedatei.

erzeugt. Das Kommando erstellt dabei eine zeichenorientierte Datei (Parameter char) mit der Majornummer 244 und der Minornummer 0. Der Treiber wurde so modifiziert, dass er jeweils nur den ersten im System vorhandenen seriellen Port verwendet (#define NR_PORTS 1). Da der serielle Treiber alle weiteren Controller verwendet, muss der Interrupttreiber vor dem seriellen Treiber geladen werden.

Um dies sicherzustellen, sollten in der Datei /etc/modules.conf die in Quellcode D.3 aufgeführten Anweisungen für die Reihenfolge beim Laden der Module vorhanden sein.

```
1  post-install serial_intr modprobe serial
2  pre-install serial modprobe serial_intr
```

Quellcode D.3 Laden des seriellen Gerätetreibers vor dem seriellen Systemtreiber.

Diese Zeilen sorgen dafür, dass vor dem Laden des serial-Moduls versucht wird den seriellen Interrupttreiber (serial_intr) zu laden (Zeile 2) und dass nach dem Laden des serial_intr-Moduls immer auch der serielle Treiber geladen wird.

Unter Debian Linux, welches in den CB-Transregio Strahlzeiten zum Einsatz kommen soll, werden die Definitionen nicht direkt in die Datei /etc/modules.conf geschrieben, sondern in /etc/modutils/setserial abgelegt. Ein Aufruf von update-modules erzeugt dann die Datei /etc/modules.conf aus den Dateien in /etc/modutils.

Damit beim Zugriff auf das mit mknod erzeugte Device /dev/serial_intr der richtige Treiber geladen wird, muss desweiteren noch ein Alias-Name erzeugt werden. Dieser sorgt dafür, dass das Linuxsystem beim Zugriff auf die Gerätedatei mit der definierten Major- und Minor-Nummer die korrekte Kernelmoduldatei lädt. Dies wird über den in Quellcode D.4 gezeigten Alias-Eintrag erreicht. Unter Debian Linux wird dieser Alias-Eintrag im Verzeichnis /etc/modutils in der Datei aliases erzeugt.

```
alias char-major-244 serial_intr
```

Quellcode D.4 Alias-Eintrag für den seriellen Interrupttreiber.

Für die weitere Verwendung als serieller Interrupttreiber wurden in der Funktion rs_ioctl die Teile entfernt, die nicht für den Treiber relevant sind. In der Funktion sind die Ein- und Ausgabekontrollen (Input/Output Controls) definiert, die über die Systemfunktion ioctl auf den Gerätetreiber angewandt werden können. Hier wurden die drei neuen IOCTLs

- IOCTL_SET_RTS - setzt im Modemkontrollregister die RTS Leitung
- IOCTL_WAIT - wartet auf eine Änderung des CTS Signals
- IOCTL_FIRST_WAIT - gleiche Funktion wie IOCTL_WAIT, jedoch mit kurzem Timeout (Initialisierung)

definiert. Die Funktion IOCTL_SET_RTS setzt bzw. löscht das RTS-Flag im Modemkontrollregister, welches wiederum bestimmt, welcher Signalpegel auf der externen RTS (read-to-send) Leitung des seriellen Ports anliegt. Dies kann zu Diagnosezwecken verwendet werden, indem das RTS-Signal mit der CTS-Leitung verbunden wird (Selbsttrigger). Es kann hiermit die Zeitdauer bestimmt werden, die für Verarbeitung des Interrupts benötigt wird. Die Funktion IOCTL_WAIT definiert zunächst einen Timer von der Zeitdauer IRQ_TIMEOUT_VALUE. Durch diesen Timer wird die Wartedauer auf einen von einem Benutzerprogramm ausgeführten ioctl Befehl definiert, die das Programm maximal auf eine Änderung des CTS-Signals warten soll. Ändert sich das Signal nicht in der vorgegebenen Zeitdauer, so kehrt die IOCTL_WAIT-Funktion dennoch zurück. Die korrespondierende Funktion rs_interrupt_single_timeout setzt dann die Variable timeout_detected auf den Wert 1 und weckt die Warteschlange eventWait auf. In diesem Fall liefert die ioctl Funktion den Wert 0 zurück (kein Ereignis eingetreten). Sofern innerhalb der maximalen Wartezeit ein Ereignis am CTS-Eingang aufgetreten ist, wird die Funktion rs_interrupt_single aufgerufen. Diese ist als Interrupthandler definiert, wenn es sich um einen seriellen Port handelt, der über eine eigene Interruptsteuerungsleitung verfügt. Dies ist bei Intel-PC-Systemen soweit immer gegeben, wenn es sich um einen Standard RS-232-Port handelt, was bei der verwendeten VMIC-CPU der Fall ist. Aus diesem Grund wurde nur die Funktion rs_interrupt_single implementiert. Die beiden Funktionen rs_interrupt und rs_interrupt_multi wurden nicht um Interrupt-Funktionalität erweitert. Die Funktion rs_interrupt_single testet zunächst, ob der Interrupt auch wirklich zur Interrupt-Request-Nummer (IRQ-Nummer) passt, die dem seriellen Port zugeordnet ist. Ist dies der Fall, so wird das Modem-Status-Register (MSR) über die Funktion serial_in gelesen. Dies bewirkt, dass der neue Zustand des MSR gespeichert wird. Desweiteren wird ein Zähler für die Anzahl der ausgelösten Interrupts erhöht und die Warteschlange eventWait aufgeweckt. Dies hat zur Folge, dass der Wartevorgang im IOCTL beendet wird, welcher durch den Aufruf des System-Schedulers in der Funktion rs_ioctl im IOCTL_WAIT initiiert wird (siehe Zeile 7). Sofern ein Interrupt ausgelöst wurde liefert die Funktion IOCTL_WAIT die Anzahl der aufgetretenen Interrupts zurück (Zeile 26).

Ist ein Interrupt schon vor dem Aufruf des IOCTL_WAIT aufgetreten, so wird ein entsprechender Hinweis im Systemlog ausgegeben. Die IOCTL-Funktion ruft dann nicht den Scheduler auf, sondern verlässt den IOCTL Teil sofort. Es wird hier ebenfalls die Anzahl der bereits ausgelösten Interrupts zurückgegeben.

Die IOCTL Funktion IOCTL_FIRST_WAIT dient dazu bei der Initialisierung einmalig auf einen Interrupt zu warten. Hierbei wird der Zeitüberschreitungszähler auf einen sehr kleinen Wert gesetzt, wodurch erreicht wird, dass – sofern bisher kein Interrupt aufgetreten ist – nahezu

```
1  irq_timeout_timer.expires=jiffies+IRQ_TIMEOUT_VALUE*HZ;
2  irq_timeout_timer.function=rs_interrupt_single_timeout;
3  add_timer(&irq_timeout_timer);
4  add_wait_queue(&eventWait,&wait);
5  current->state=TASK_INTERRUPTIBLE;
6  if (interrupt_counts==0) {
7    schedule();
8    cntInterrupts = interrupt_counts;
9    interrupt_counts=0;
10 }
11 else {
12   printk("serial_intr: Interrupt before wait queue %i\n",interrupt_counts);
13   cntInterrupts = interrupt_counts;
14   interrupt_counts=0;
15   current->state=TASK_RUNNING;
16 }
17
18 del_timer(&irq_timeout_timer);
19 remove_wait_queue(&eventWait,&wait);
20
21 if (timeout_detected==1) {
22   timeout_detected=0;
23   return 0;
24 }
25 else
26   return cntInterrupts;
27 break;
```

Quellcode D.5 IOCTL_WAIT

sofort aus der Wartefunktion (dem Scheduler des Betriebssystems) zurückgekehrt wird. Ist ein Interrupt aufgetreten, so wird durch den Funktionsaufruf die Variable, die die Anzahl der aufgretenen Interrupts zählt, auf den Wert Null zurückgesetzt.

Für den Nutzer des seriellen Interrupttreibers ist eine Bibliothek entwickelt worden, die die Synchronisationsmodule des Experiments ansteuert. Die Bibliothek libsync enthält eine C++-Klasse, die im Konstruktor die Gerätedatei des seriellen Interrupttreibers öffnet und im Destruktor wieder schließt. Die Funktion CSyncClientModule::waitTrigger ruft hierbei lediglich die Systemfunktion ioctl auf und übergibt den Dateideskriptor des seriellen Interruptgerätes und die Konstante für den Aufruf der IOCTL-Funktion IOCTL_WAIT.

Der hier beschriebene Interrupttreiber wird für alle lokalen Eventbuilder des Experiments verwendet und hat seit der ersten Verwendung im Jahre 2001 mehrere hundert Millionen Ereignisse gehandhabt. Dabei haben sich keine Fehlfunktionen gezeigt, die auf den programmierten Treiber zurückzuführen waren.

E Literatur

Abe94 Abe, F. et al., (1994). Evidence for top quark production in $p\bar{p}$ collisions at $\sqrt{s} = 1.8TeV$. *Phys. Rev. Lett.*, 73:225-231.

Abe95 Abe, F. et al., (1995). Observation of Top Quark Production in pp Collisions with the Collider Detector at Fermilab. *Phys. Rev. Lett.*, 74:2626-2631.

Alp03 Alpha-Data, (2003). ADM-XRC SDK Documentation. Version 2.3 ftp://ftp.alphadata.co.uk/pub/admxrc/linux/.

Alp04a Alpha-Data, (2004a). ADM-XRC-Karte. Foto einer ADM-XRC-Karte, entnommen von http://www.alpha-data.com mit Genehmigung der Firma Alpha-Data.

Alp04b Alpha-Data, (2004b). Schemazeichnung ADM-XRC-Karte. Schemazeichnung einer ADM-XRC-Karte, entnommen von http://www.alpha-data.com mit Genehmigung der Firma Alpha-Data.

Ani05 Anisovich, A.V. and Klempt, E. and Sarantsev, A. and Thoma, U., (2005). Partial wave decomposition of pion and photoproduction amplitudes. *Eur. Phys. J.*, A24:111-128.

Ani05 Anisovich, A.V. and Sarantsev, A. and Bartholomy, O. and Klempt, E. and Nikonov, V.A. and Thoma, U., (2005). Photoproduction of Baryons Decaying into $N\pi$ and $N\eta$. *European Physical Journal A*, 25:427.

Ass03 Assafiri, Y. et al., (2003). Double π^0 photoproduction on the proton at GRAAL. *Phys. Rev. Lett.*, 90:222001.

Bar05 Bartholomy, O. and Crede, V. and van Pee, H., (2005). Neutral pion photoproduction off protons in the energy range 0.3 GeV < E(gamma) < 3 GeV. *Physical Review Letters*, 94:012003.

Bog01 Bogendörfer, R., (2001). Effizienzbestimmung für den Innendetektor des Crystal-Barrel-Experiments an ELSA. Diplomarbeit, Universität Erlangen-Nürnberg.

Cha95 Chase, R.L. et al., (1995). A fast monolithic shaper for the ATLAS E.M. Calorimeter. ATLAS Internal Note.

Chu95 Chung, S.U. and Brose, J. and Hackmann, R. and Klempt, E. and Spanier, S. and Strassburger, C., (1995). Partial Wave Analysis in the K-matrix formalism. *Ann. der Physik*, 4:404.

Cre05 Crede, V. and Bartholomy, O., (2005). Photoproduction of eta mesons off protons for photon energies from 0.75 GeV to 3 GeV. *Physical Review Letters*, 94:012004.

Cry05 Crystal Barrel Collaboration, (2005). Measurement of Double Polarisation Observables in 2 π^0-Photoproduction with the Crystal Barrel Detector at ELSA.

Dat88 Datel, (1988). ADC-520, ADC-521 12-Bit, Ultra-Fast, Low-Power A/D Converters. Product Data Sheet.

Dut95 Dutz, H. et al., (1995). An internal superconducting 'holding coil' for frozen spin targets.. *Nucl. Instr. and Meth.*, A356:111.

Fla76 Flatte, S., (1976). Coupled-channel analysis of the $\pi\eta$ $K\bar{K}$ threshold. *Phys. Lett. B*, 63:224.

For04 Fornet-Ponse, K., (2004). Entwurf eines Fokalebenendetektors für die Photonenmarkierungsanlage an ELSA. Diplomarbeit, Universität Bonn.

Fuc05 Fuchs, M., (2005). Photoproduktion neutraler Pionpaare mit dem Crystal-Barrel-Detektor an ELSA. Dissertation, Universität Bonn.

Fun02 Funke, C., (2002). Untersuchungen zur Energieauflösung von CsI-Kristallen mit Hochgeschwindigkeits ADCs. Diplomarbeit, Universität Bonn.

Fun04 Funke, C., (2004). Persönliche Mitteilung.

Fö00 Fösel, A., (2000). Entwicklung und Bau eines Innendetektors für das Crystal-Barrel-Experiment an ELSA in Bonn. Dissertation, Universität Erlangen.

GE 05 GE Fanuc Embedded Systems, (2005). Schemazeichnung VMIVME-7750 Main Board. Schmeazeichnung VMIVME-7750, entnommen von http://www.gefanuc.com/en/documents/vme7750_spec.pdf Seite 7 mit Genehmigung der Firma GE Fanuc.

Gla70 Glashow, S.L. and Iliopoulos, J. and Maiani, L., (1970). Weak interactions with lepton-hadron symmetry. *Phys. Rev. D*, 7:1285-1292.

Gó94 Gómez-Tejedor, J. A. and Oset, E., (1994). A Model for the $\gamma\mathrm{p} \rightarrow \mathrm{p}\,\pi^+\pi^-$ reaction. *Nucl. Phys.*, A571:667-693.

Gó96 Gómez-Tejedor, J. A. and Oset, E., (1996). Double pion photoproduction on the nucleon: Study of the isospin channels. *Nucl. Phys.*, A600:413-435.

Hel02 Helbing, K. and Anton, G. and Fausten, M. and Menze, D. and Michel, T. and Nagel, A. and Ryckbosch, D. and Speckner, T. and Van de Vyver, R. and Zeitler, G., (2002). The GDH-Detector. *Nuclear Instruments and Methods in Physics Research A*, 484:129-139.

Her77 Herb, S.W. et al., (1977). Observation of a Dimuon Resonance at 9.5 GeV in 400-GeV Proton-Nucleus Collisions. , 39:252-255.

Hof04 Hoffmeister, P., (2004). Konzipierung der Analogsignalverarbeitung für den Crystal Barrel Vorwärtsdetektor. Diplomarbeit, Universität Bonn.

Hus88 Husmann, D. and Schwille, W. J., (1988). ELSA - Der neue Elektronen Stretcher Ring in Bonn. *Phys. Bl.*, 44:40-44.

Hä97 Härter, F. et al., (1997). Two neutral pion photoproduction off the proton between threshold and 800 MeV. *Phys. Lett.*, B401:229-233.

Hö00 Höffgen, S., (2000). Einbindung eines großflächigen Flugzeitspektrometers als Vorwärtsdetektor für Experimente mit CB-ELSA. Diplomarbeit, Universität Bonn.

Isi24 Ising, G., (1924). Prinzip einer Methode zur Herstellung von Kanalstrahlen hoher Voltzahl. *Arkiv för matematik o. fysik*, 18, Nr.30:1-4.

Kal06 Kalinowsky, H., (2006). Persönliche Mitteilung.

Kno95 Knochlein, G. and Drechsel, D. and Tiator, L., (1995). Photo- and Electroproduction of Eta Mesons. *Z.Phys.A*, 352:327.

Kod01 Kodama, K. and others, (2001). Observation of tau-neutrino interactions. *Phys. Lett.*, B504:218-224.

Kon01 Konrad, M., (2001). Ortssensitiver Detektor für hochenergetische Photonen bei höchsten Raten. Diplomarbeit, Universität Bonn.

Kop02 Kopf, B. (2002). Untersuchung der photoinduzierten Reaktionen $\gamma\mathrm{p} \rightarrow \mathrm{p}\,\pi^0\pi^0$ und $\gamma\mathrm{p} \rightarrow \mathrm{p}\,\pi^0\eta$ an einem Flüssig-Wasserstoff-Target. Dissertation, Technische Universität Dresden.

Kot04 Kotulla, M. (2004). Recent results from $2\pi^0$ photoproduction off the proton. *AIP Conf. Proc.*, 717:842-847.

LeC97 LeCroy, (1997). Model 1882F and 1885F Fastbus Analog-to-Digital Converters. Operators Manual.

Lic69 Lichtenberg, D. B., (1969). Baryon supermultiplets of SU(6) x O(3) in a quark - diquark model. *Phys. Rev.*, 178:2197-2200.

Lö01 Löring, Ulrich and Kretzschmar, Klaus and Metsch, Bernard C. and Petry, Herbert R., (2001). Relativistic quark models of baryons with instantaneous forces. *Eur. Phys. J.*, A10:309-346.

Lü71 Lüke, D. and Söding, P., (1971). Multiple pion photoproduction in the s channel resonance region. *Springer Tracts in Modern Physics*, 59:39.

McM45 McMillan, E.M., (1945). The Synchrotron - A proposed High Energy Particle Accelerator. *Phys. Rev.*, 68:143-144.

Mer88 Merkel, M., (1988). Aufbau und Test eines Datenerfassungssystems für den Zentraldetektor des Crystal Barrel Spektrometers. Diplomarbeit, Universität Mainz.

Moe32 Moeller, C., (1932). Zur Theorie des Durchgangs schneller Elektronen durch Materie. *Annalen der Physik*, 14:531.

Mur95 Murphy, L. Y. and Laget, J.M., (1995). Reaction mechanisms in two-pion photoproduction on the proton. *Service de physique nucleaire*. DAPNIA-SPhN 95-42.

Nii76 Niinkoski, T.O. and Udo, F., (1976). Frozen-Spin Polarized Target. *Nucl. Instr. and Meth.*, 134:219-233.

Ols59 Olsen, H. and Maximon, L. C., (1959). Circular polarization transfer. *Phys. Rev.*, 114:887.

Par04 Particle Data Group [PDG], (2004). Review of particle physics. *Phys. Lett. B*, 592:1.

Per75 Perl, M.L. et al., (1975). Evidence for anomalous lepton production in $e^+ - e^-$ annihilation. *Phys. Rev. Lett.*, 35:1489-1492.

PLX00 PLX Technology, (2000). *PCI 9080 Data Book*. Version 1.06.

Rob05 Roberts, W. and Oed, T., (2005). Polarization Observables for Two-Pion Production off the Nucleon. *Physical Review C*, 71:055201.

ROO ROOT collaboration, ROOT: An Object Oriented Data Analysis Framework. http://root.cern.ch.

Sar05 Sarantsev, A.V. and Nikonov, V.A. and Anisovich, A.V. and Klempt, E. and Thoma, U., (2005). Decays of Baryon Resonances into $K\Lambda$, $K^+\Sigma^0$ and $K^0\Sigma^+$. *European Physical Journal A*, 25:441.

Sch04 Schmidt, C., (2004). Entwicklung eines Datenakquisitionssystems für das Crystal-Barrel-Experiment zur Messung photoinduzierter Reaktionen an ELSA. Dissertation, Universität Bonn.

Str87d Struck, Dr. B., (1985-1987d). DL302 Scanner. Technical Manual.

Str87b Struck, Dr. B., (1985-1987b). DL305 FADC Module. Technical Manual.

Str87e Struck, Dr. B., (1985-1987e). DL307/307V VME-Interface IF-Submodule. Technical Manual.

Str87c Struck, Dr. B., (1985-1987c). DL310 FADC Module. Technical Manual.

Str87a Struck, Dr. B., (1985-1987a). FADC Crate DL300 System Guide. Technical Manual.

t'H76 t'Hooft, G., (1976). Computation of the quantum effects due to a four-dimensional pseudoparticle. *Phys. Rev. D.*, 14:3432.

Tho06 Thoma, U. et al., (2006). Veröffentlichung in Vorbereitung, N^* and Δ^* decays into N $\pi^0\pi^0$. .

van03 van Pee, H., (2003). Untersuchung der Reaktion $\gamma\mathrm{p} \rightarrow \mathrm{p}\,\pi^0$ für Photonenergien von 0.45 bis 1.3 GeV mit dem Crystal-Barrel-Detektor an ELSA. Dissertation, Universität Bonn.

Vek45 Veksler, V., (1945). A new method of acceleration of relativistic particles. *Journal of Physics UdSSR*, 9:153-158.

Vra00 Vrana, T.P. and Dytman, S.A. and Lee, T.S.H., (2000). Baryon Resonance Extraction from Pion Data using a Unitary Multichannel Model. *Physics Reports*, 328:181.

Wen04 Wendel, C., (2004). Entwicklung eines Szintillations-Detektors zur Identifikation geladener Teilchen im Crystal-Barrel Vorwärtsdetektor. Diplomarbeit, Universität Bonn.

Wil02 Wille, Klaus, (2002). *Physik der Teilchenbeschleuniger und Synchrotronstrahlungsquellen*. Teubner, Wiesbaden.

Wit95 Wittmack, K., (1995). Entwicklung, Bau und Test eines VME-Moduls zur Auslese des SAPHIR-Taggingsystems. Diplomarbeit, Universität Bonn.

Wol00 Wolf, M. et al., (2000). Photoproduction of neutral pion pairs from the proton. *Eur. Phys. J.*, A9:5-8.

F Abbildungen

G Tabellen

H Quellcode

Danksagung

Zum Abschluss möchte ich mich bei all denen bedanken, die mit zum Gelingen dieser Arbeit beigetragen haben.

Bei Prof. Eberhard Klempt möchte ich mich sehr für die Betreuung meiner Arbeit und die Unterstützung bedanken. Seine Anregungen haben wesentlich zum Gelingen dieser Arbeit beigetragen. Bei Prof. Ulrike Thoma möchte ich mich für die vielen wichtigen Impulse und Hilfestellungen, sowie für ihre Geduld und Mühe bedanken.

Ganz besonderer Dank gilt Dr. Hartmut Kalinowsky, der mir mit sehr viel Geduld bei Fragen oder Problemen stets zur Seite stand, um gemeinsam mit mir eine Lösung für neue auftauchende Probleme zu finden. Von ihm habe ich in den letzten Jahren viel lernen können, insbesondere dann, wenn die Situation wiedereinmal besonders verworren und „zum Mäuse melken“ war. Danke!

Desweiteren möchte ich mich bei Jan-Keno-Janssen, Eric Gutz, Anja Purvinskis und Kathrin Valerius bedanken, die bei der Suche nach inhaltlichen und grammatikalischen Fehlern geholfen haben.

Den Kollegen aus der Arbeitsgruppe und den Mitarbeitern des HISKP möchte ich für die sehr angenehme Arbeitsatmosphäre danken. Hierbei sind insbesondere meine Studienkollegen Christian Funke und Eric Gutz zu nennen, die mich seit dem ersten Semester im Studium und dann auch später in der Arbeitsgruppe begleitet haben. Bedanken möchte ich mich auch bei Frau Mosblech, Frau Balci und Frau Schwenk, die mir immer bei allen organisatorischen Dingen freundlich geholfen haben.

Bei meiner langjährigen Freundin und Lebensgefährtin Kathrin möchte ich mich für ihr Verständnis und ihre Geduld während meiner Promotion und insbesondere bei Strahlzeiten bedanken.

Meinen Eltern danke ich für ihre Unterstützung während meines gesamten Studiums.

Bei der CB-ELSA und CB-TAPS Kollaboration möchte ich für die gute und angenehme Zusammenarbeit bedanken.