

Einbindung eines Transientenrekorders in das COMPASS-Datennahmesystem

Louis Philipp Lauser



Physikalisches Institut
Albert-Ludwigs-Universität Freiburg

Einbindung eines Transientenrekorders in das COMPASS-Datennahmesystem

Diplomarbeit

vorgelegt
von

Louis Philipp Lauser

Physikalisches Institut
Albert-Ludwigs-Universität
Freiburg

April 2009

Inhaltsverzeichnis

1	Einleitung	1
2	Theoretische Grundlagen	3
2.1	Aufbau der Nukleonen	3
2.1.1	Das Quark-Modell	3
2.1.2	Quarks, Hadronen und die starke Kernkraft	3
2.1.3	Die tiefinelastische Streuung	4
2.1.4	Das Partonmodell	6
2.1.5	Die Impulsverteilung im Nukleon	8
2.2	Der Spin des Nukleons	9
2.2.1	Der Spin der Quarks	10
2.2.2	Der Spin der Gluonen	11
2.2.3	Der Beitrag des Drehimpulses	12
3	Das COMPASS-Experiment	16
3.1	Der Strahl und das Target	16
3.2	Die Spurdetektoren	17
3.3	Die Detektoren zur Teilchenidentifikation	18
3.4	Der Einbau des Rückstoß-Detektors	18
4	Das Datennahmesystem am COMPASS Experiment	22
4.1	Der Aufbau des Datennahmesystems	22
4.2	Der Triggergenerierung	23
4.3	Das Trigger Control System	24
5	Das GANDALF-Modul	26
5.1	Die Schnittstellen der Baugruppe	27
5.1.1	VME64x	27
5.1.2	VXS	29
5.1.3	Die S-Link Schnittstelle	30
5.1.4	Die USB-Schnittstelle	31
5.1.5	Die JTAG-Schnittstellen	31
5.1.6	Die Compact-Flash Karte	32
5.1.7	Die I ² C-Schnittstellen	32
5.2	Die Funktionen der Baugruppe	33

5.2.1	Die Überwachung der Spannungsversorgung	33
5.2.2	Die Gimli-Karte	33
5.2.3	Die Taktverteilung	34
5.2.4	Die Verarbeitung der Daten	34
5.3	Der GANDALF-Transientenrekorder	35
5.3.1	Die AMC-Aufsteckkarte	36
5.3.2	Die Güte der Digitalisierung	40
5.3.3	Die Verarbeitung der digitalisierten Daten	41
6	Die GANDALF-Kontrolleinheit	43
6.1	Chiptechnologie und Hardwarebeschreibung	44
6.1.1	Der Aufbau eines programmierbaren Logikbausteines	44
6.1.2	Die Entwicklung der Firmware mit VHDL	46
6.2	Die VME-Schnittstelle	50
6.2.1	Die VME Board-Identifikation	52
6.2.2	Die VME-Adressierung	52
6.2.3	Die Address Modifier Codes	54
6.2.4	Die VME-Leseroutinen	55
6.2.5	Die VME-Schreibroutinen	58
6.3	Die Konfiguration der FPGAs über VME	63
6.3.1	Die Slave Serial Konfiguration	63
6.3.2	Die SelectMAP Konfiguration	64
6.4	Die SystemACE-Umgebung	66
6.4.1	Die Konfiguration der FPGAs	66
6.4.2	Datenspeicherung auf Compact-Flash	67
6.5	Die I ² C-Schnittstelle	70
7	Zusammenfassung	73
A	Hardwaredesign des Controller-CPLDs	75
A.1	Die Dekodierung der VME-Adresse	75
A.2	Die schematische Übersicht	76
A.3	Der Controller	77
A.4	Die FPGA-Konfiguration	90
A.5	SystemACE	94
A.6	Die I ² C-Schnittstelle	98
A.7	Die GANDALF VME-Routinen	102
B	Die VME64x-Steckverbinder bei GANDALF	103
	Abbildungsverzeichnis	106
	Tabellenverzeichnis	108

Kapitel 1

Einleitung

Mit der tiefinelastischen Streuung von Leptonen an den Kernbausteinen konnten in den vergangenen Jahrzehnten grundlegende Erkenntnisse über den Aufbau von Proton und Neutron gewonnen werden. Demnach besteht das Nukleon aus punktförmigen Konstituenten, den Quarks und den Gluonen. Die Gluonen sind Austauschpartikel der starken Wechselwirkung und halten die Quarks zusammen. Zudem wurde gezeigt, dass der Impuls des Nukleons von den Quarks und den Gluonen zu gleichen Teilen getragen wird. Daraufhin wurde begonnen, über die inelastische Streuung mit polarisierten Strahlteilchen und Targetmaterialien die Zusammensetzung des Nukleonspins zu ergründen. Die Elektron-Myon-Kollaboration (EMC) am CERN untersuchte zu Beginn der 1980er Jahre die Spinbeiträge der Quarks zum Spin des Nukleons. Es konnte gezeigt werden, dass der Anteil der Quarks $\Delta\Sigma$ nicht wesentlich zum Gesamtspin beiträgt. Dieses Ergebnis war im Vergleich zur Impulsverteilung im Nukleon überraschend.

Mit der Helizität der Gluonen ΔG und den Beiträgen des Drehimpulses von Quarks und Gluonen L_q und L_g kann die Zusammensetzung des Nukleonspins über eine Summenregel dargestellt werden:

$$\frac{1}{2}\Delta\Sigma + L_q + \Delta G + L_g = J_q + J_g \quad (1.1)$$

Zugang zum Gesamtdrehimpuls J_q und J_g der Quarks bzw. Gluonen eröffnen Generalisierte Partonverteilungen (GPD). Sie können aus verschiedenen Asymmetrie- und Wirkungsquerschnittsmessungen in der tiefvirtuellen Compton-Streuung (DVCS) und harten exklusiven Meson-Produktion (HEMP) bestimmt werden. Mit einem Rückstoßproton-Detektor (RPD) verfügt das COMPASS-Experiment über die notwendige Anordnung, um eine exklusive Messung der Endzustände von DVCS zu gewährleisten. Über die Messung der Flugzeit und des spezifischen Energieverlustes (dE/dx) von geladenen Teilchen soll das Proton, das im Zuge dieses Streuprozesses rückgestoßen wird, identifiziert werden. Zur Unterscheidung von Protonen und Pionen sind Messungen mit hoher Zeitauflösung notwendig. Mit kurzen Anstiegsflanken der Detektorsignale von 3 bis 5 ns Dauer und einem hohen Untergrund durch Myonen sind höchste Anforderungen an die Ausleseelektronik gestellt. Ein GANDALF-Transientenrekorder ist entwickelt worden, um die Signalverarbeitung am COMPASS-RPD durchzuführen und eine selbstgetriggerte Auslese zu ermöglichen.

Die Einbindung von GANDALF in das COMPASS-Datennahmesystem erfordert die Implementierung einer VME-Schnittstelle. So kann die zur Verarbeitung der Daten notwendige Firmware an eine Baugruppe übertragen werden und Messdaten sowie Statusinformationen können ausgelesen werden. Zur Konfiguration der intelligenten Bausteine sowie zur Ansteuerung und Auslese von Registern einzelner Bauelemente des Moduls werden weitere Schnittstellen und Protokolle benötigt. Die zentrale und wechselseitige Steuerung sämtlicher Protokolle in einem Hardwarebaustein gewährleistet die volle Funktionalität von GANDALF.

Im Rahmen dieser Diplomarbeit wurden die Protokolle und Funktionen, die den Einsatz des Moduls am COMPASS-Experiment sicherstellen, in eine Hardwarebeschreibung umgesetzt. Als zentrale Einheit zur Steuerung der Datenströme wird bei GANDALF ein CPLD (*Complex Programmable Logic Device*) eingesetzt. Der Austausch von Daten zwischen einzelnen Schnittstellen wird durch einen geregelten zeitlichen Ablauf der Protokolle erreicht. Mit dem Ausschöpfen der protokollspezifischen Möglichkeiten erfolgt eine Optimierung der Datenraten.

Die Arbeit ist in sieben Kapitel unterteilt. In Kapitel 2 wird der theoretische Hintergrund von inelastischen Streuexperimenten bis zu den aktuellen Fragestellungen beleuchtet. In Kapitel 3 wird das COMPASS-Experiment und die Erweiterung des Spektrometers mit dem Rückstoß-Detektor erläutert, bevor in Kapitel 4 auf das COMPASS-Datennahmesystem eingegangen wird. In Kapitel 5 werden die Einsatzmöglichkeiten von GANDALF mit seinen Schnittstellen und Funktionen beschrieben. Außerdem wird der GANDALF-Transientenrekorder vorgestellt und die Digitalisierung der Signale sowie die Verarbeitung der Daten zur Bestimmung der Flugzeit von Teilchen erläutert. Kapitel 6 enthält die Beschreibung der im CPLD realisierten Schnittstellen und Funktionen, die eine Einbindung von GANDALF in das Datennahmesystem am COMPASS-Experiment ermöglichen: Ausgehend von grundlegenden VME-Lese- und Schreibroutinen können mit VME-Blocktransfers die Datenraten in beide Richtungen zwischen Anwender und Baugruppe erhöht werden. GANDALF enthält mehrere Konfigurationsoptionen und Schnittstellen, die vom CPLD gesteuert werden sollen. Schließlich ist ein eigenständiger Einsatz des Moduls mit der *SystemACE*-Konfigurationsumgebung zu gewährleisten. Eine Zusammenfassung der Arbeit wird in Kapitel 7 gegeben.

Kapitel 2

Theoretische Grundlagen

2.1 Aufbau der Nukleonen

2.1.1 Das Quark-Modell

Lange Zeit ging man davon aus, dass Protonen, Neutronen, Elektronen, Myonen und Neutrinos sowie Pionen und Photonen die einzigen existierenden elementaren Teilchen sind. Doch bei dem Versuch, mit Teilchenbeschleunigern die Kräfte zwischen den Nukleonen zu verstehen, entdeckte man zahlreiche neue instabile Teilchen, die als Hadronen bezeichnet wurden. Um Ordnung in diesen Teilchenzoo zu bringen, versuchte man die Teilchen nach möglichst allgemeinen Gesichtspunkten wie Masse oder Spin zu klassifizieren. Dies versuchten Murray Gell-Mann und George Zweig unabhängig voneinander [1]. Sie fanden einzelne Gruppen von Hadronen mit gleichem Spin und gleicher Parität bei ähnlichen Massen. Daraus folgerten sie, dass es weitere, neue elementare Bausteine geben müsse. Bekräftigt wurde diese Hypothese durch vorhandene Anregungszustände vieler Hadronen sowie durch die Existenz eines anomalen magnetischen Moments des Neutrons. Letzteres ist ausschließlich dadurch erklärbar, dass das Neutron aus elementaren Teilchen aufgebaut ist, deren Ladungen sich in der Summe zu Null kompensieren. Demzufolge wurde ein Modell mit drei elementaren Teilchen entwickelt, die sich in der elektrischen Ladung Q , im Isospin T und seiner Komponente T_3 und in der Seltsamkeit S unterscheiden. Diese Teilchen tragen Spin $1/2$ und werden allesamt als *Quarks* bezeichnet. Einzelne Typen von Quarks werden mit der Bezeichnung *Up*, *Down* und *Strange* unterschieden. Mit der Einführung des Isospins wurde der Tatsache Rechnung getragen, dass sich Proton und Neutron in erster Linie in ihrer Ladung unterscheiden und demnach nur bezüglich der Coulomb-Wechselwirkung als verschiedene Teilchen gelten. In Analogie zum Spin des Elektrons im Atom mit zwei möglichen Einstellungen erfolgte die Entwicklung des Isospin-Formalismus. Mit diesem Modell konnten alle Hadronen aus Quarks und Antiquarks aufgebaut werden. Die Eigenschaften der zusammengesetzten Teilchen werden dabei aus den elementaren Bausteinen abgeleitet.

2.1.2 Quarks, Hadronen und die starke Kernkraft

Insgesamt gibt es zwei Klassen von Hadronen. Als Baryonen werden die Teilchen bezeichnet, die aus drei Quarks aufgebaut sind, während die Mesonen aus einem Quark und einem Antiquark bestehen. Die Baryonzahl erlaubt eine Unterscheidung in Baryon ($B = 1$), Antibaryon

($B = -1$) und Meson ($B = 0$). Im Jahr 1974 wurde bei e^+, e^- -Reaktionen und Proton-Proton Kollisionen eine sehr scharfe Resonanz gemessen [1]. Weitere Messungen zeigten, dass die Breite $\Gamma = 63 \text{ KeV}$ beträgt, was einer Lebensdauer von 10^{-20} s entspricht. Aufgrund seiner großen Masse konnte es nicht aus den bereits existierenden Quarks aufgebaut werden. Ein weiteres Quark, das *Charm-Quark* wurde mit der dazugehörigen Quantenzahl c eingeführt. So konnte das entdeckte Teilchen aus der Kombination von Charm und Anticharm zusammengesetzt werden. Eine große Bewährungsprobe für das Quark-Modell war die Entdeckung des Ω^- Baryons und des Δ^{++} Teilchens. Diese Teilchen mit Gesamtspin $3/2$ können ausschließlich aus drei gleichen Quarks mit parallelem Spin aufgebaut sein und würden so das Spin-Statistik-Theorem von Pauli (1940) verletzen. Mit der Einführung der Farbe als weitere Quantenzahl konnten die Teilchen in dem Modell existieren. Die drei Quarks werden mit den Farben rot, grün und blau unterschieden, sodass sich keine identischen Fermionen im gleichen Quantenzustand befinden. Somit ist das Pauli-Prinzip im Quarkmodell integriert und darüberhinaus die Ladung -hier als Farbladung bezeichnet- eingeführt. Sie ist notwendig, um Quarks über die starke Farbkraft und dem damit verbundenen Austausch von Gluonen zusammenzuhalten. Die der starken Wechselwirkung zwischen Quarks zugrundeliegende Theorie wird als Quantenchromodynamik bezeichnet (QCD). Gluonen sind masselose Bosonen mit Spin 1, sogenannte Vektorbosonen.

2.1.3 Die tiefinelastische Streuung

Streuexperimente sind in vielerlei Hinsicht hilfreich, um Einblicke in die Struktur der Materie zu gewinnen, kleinste Bausteine zu erkennen und die Wechselwirkungen zu verstehen. Dabei wird ein Strahl von Teilchen auf ein Objekt geschossen. Mit geeigneten Detektoranordnungen werden am Streuzentrum (engl.: *target*) abgelenkte Teilchen und neu entstandene Reaktionsprodukte aufgenommen. Die Messungen der Reaktionswahrscheinlichkeit werden sowohl winkelabhängig als auch in Abhängigkeit von Energien und Massen durchgeführt. Bei der inelastischen Streuung wird das Targetteilchen durch die vom Streuteilchen abgegebene kinetische Energie angeregt und neue Teilchen können entstehen. Eine wichtige Größe zur Beschreibung und Interpretation dieser Reaktionen ist der Wirkungsquerschnitt σ . Er ist ein Maß für die Wahrscheinlichkeit einer Reaktion zwischen zwei Stoßpartnern. Die inelastische Streuung etablierte sich in den 1960er Jahren, als die dafür notwendigen Energien im Bereich einiger GeV zur Verfügung standen. Allgemein streut hierbei ein Lepton an einem Hadron, das ausgetauschte Photon überträgt im Laborsystem den eigenen Viererimpuls $q = (E - E'/c, \vec{q})$ auf den des einlaufenden Protons $P = (Mc, \vec{0})$ (siehe Abbildung 2.1).

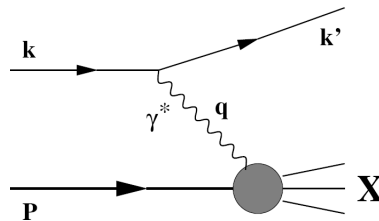


Abbildung 2.1: Der inelastische Streuprozess. Ein Lepton streut an einem Hadron, das Austauschphoton überträgt den Viererimpuls q auf das einlaufende Proton P .

Mit der invarianten Masse W und $\nu = \frac{Pq}{M} = E - E'$,

$$W^2 c^2 = P' = (P + q)^2 = M^2 c^2 + 2Pq + q^2 = M^2 c^2 + 2M\nu - Q^2, \quad (2.1)$$

findet sich die Beziehung

$$2M\nu - Q^2 > 0, \quad (2.2)$$

mit $W > M$ im inelastischen Fall. Der Wirkungsquerschnitt ist

$$\frac{d^2\sigma}{d\Omega dE'} \propto \left[W_2(Q^2, \nu) + 2W_1(Q^2, \nu) \tan^2 \frac{\theta}{2} \right]. \quad (2.3)$$

Die in (2.3) enthaltenen Strukturfunktionen W_1 und W_2 sind damit von Q^2 und ν abhängig. Überraschenderweise ging aus den ersten tiefinelastischen Streuexperimenten hervor, dass hin zu niedriger Streuenergie bzw. hohen Werten von W eine Q^2 -Abhängigkeit des Wirkungsquerschnitts kaum mehr auszumachen ist: Die Strukturfunktionen sind in diesem Bereich unabhängig von Q^2 (siehe Abbildung 2.2). Um dieses Verhalten zu erfassen, wurde die dimensionslose Bjorkensche Skalenvariable x eingeführt:

$$x := \frac{Q^2}{2Pq} = \frac{Q^2}{2M\nu}, \quad (2.4)$$

womit aus (2.2) $0 < x < 1$ folgt und dimensionslose Strukturfunktionen eingeführt werden können:

$$F_1(x, Q^2) = Mc^2 W_1(Q^2, \nu) \quad (2.5)$$

$$F_2(x, Q^2) = \nu W_2(Q^2, \nu) \quad (2.6)$$

Die Tatsache, dass die Strukturfunktionen lediglich von der dimensionslosen Größe x ab-

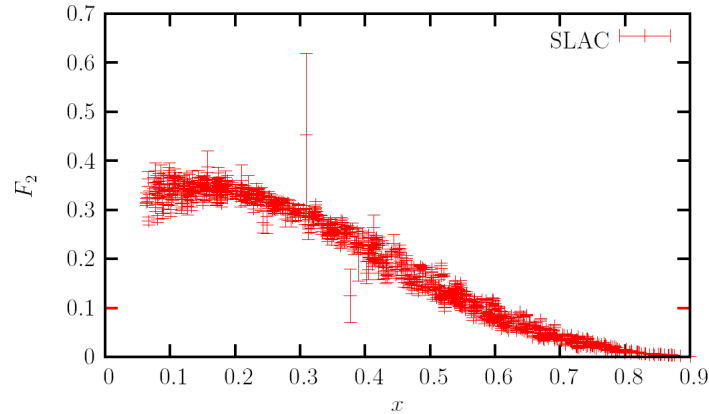


Abbildung 2.2: AM SLAC gemessene Strukturfunktion $F_2(x)$ für verschiedene Werte von Q^2 [2].

hängen, was in Analogie zur Elektron-Kern Streuung einer quasi-elastischen Streuung entspricht, wird auch als Bjorkensche Skaleninvarianz bezeichnet [3]. Dieses Skalenverhalten ist ausschließlich für den kleinen Bereich von (Q^2, x) , der in Fixed-Target-Experimenten zugänglich ist, gültig. Feynman war es, der im Jahre 1968 mit der Entwicklung des Partonmodells eine intuitive Erklärung dieser Beobachtung geben konnte.

2.1.4 Das Partonmodell

Um physikalische Prozesse zu veranschaulichen, kann der Wechsel in ein geeignetes Bezugssystem sinnvoll sein. Feynman konnte so einen Zusammenhang zwischen dem Impulsübertrag und der inneren Struktur des Protons aufzeigen [4]: Betrachtet man das Proton in einem Bezugssystem, in dem die transversalen Impulse und die Ruhmassen der Konstituenten vernachlässigt werden können, ist die innere Struktur des Protons durch die longitudinalen Impulsanteile der Bausteine gegeben. Diese Teilchen werden Partonen genannt. Eine inelastische Streuung an Protonen kann als elastische Streuung an Partonen betrachtet werden, sofern die Abstände dieser inneren Teilchen gering sind und die Wechselwirkungszeit zwischen Austauschphoton und Parton als kurz angenommen wird. Dies entspricht einer Stoßnäherung, -die in diesem Bezugssystem mit $Q^2 \gg M^2 c^2$ angewendet- eine anschauliche Deutung der Skalenvariablen x ermöglicht. Sie entspricht dem Impulsanteil des Protons, der vor der Streuung von einem Parton getragen wurde (siehe Abbildung 2.3). Nimmt man an, dass es sich um eine elastische Streuung am Parton vom Typ p handelt, das den Bruchteil η des Partonimpulses trägt, so ist

$$(q + \eta P)^2 = 2\eta Pq - Q^2 = 0. \quad (2.7)$$

Damit gilt $\eta = x$ und x ist der Impulsbruchteil des Protons, der von einem Parton getragen wird.

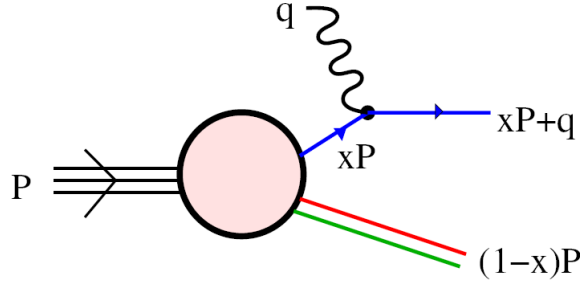


Abbildung 2.3: Tief inelastische Streuung im Parton Modell. Der Impulsbruchteil x des Protons wird von einem Parton getragen.

So kann in diesem Bezugssystem, das auch als *infinite momentum frame* bezeichnet wird, eine Erklärung für die Skaleninvarianz der Strukturfunktion $F_{1,2}$ gegeben werden. Wie aus Abbildung 2.4 hervorgeht, steht der normierte Wirkungsquerschnitt von inelastischen Streuvorgängen bei verschiedenen invarianten Massen im Gegensatz zu den Messwerten, die man bei einer elastischen Streuung am Nukleon zu erwarten hätte. Der von Q^2 nahezu unabhängige Verlauf der Streuwahrscheinlichkeit entspricht einer Streuung an einer Punktladung. Diese Punktladungen werden in diesem Modell mit geladenen Partonen identifiziert. Der Einfluss des Eigendrehimpulses dieser punktförmigen Teilchen auf den Wirkungsquerschnitt kann aus der spinabhängigen Streuung erhalten werden. Da die Strukturfunktionen F_1 und F_2 die magnetische bzw. elektrische Streuung eindeutig repräsentieren, verschwindet die Strukturfunktion F_1 für Teilchen ohne Spin. Für punktförmige Streupartner mit Spin 1/2 und normalen magnetischen Momenten sind die Strukturfunktionen über die Callan-Gross Beziehung

$$\frac{2xF_1(x)}{F_2(x)} = 1 \quad (2.8)$$

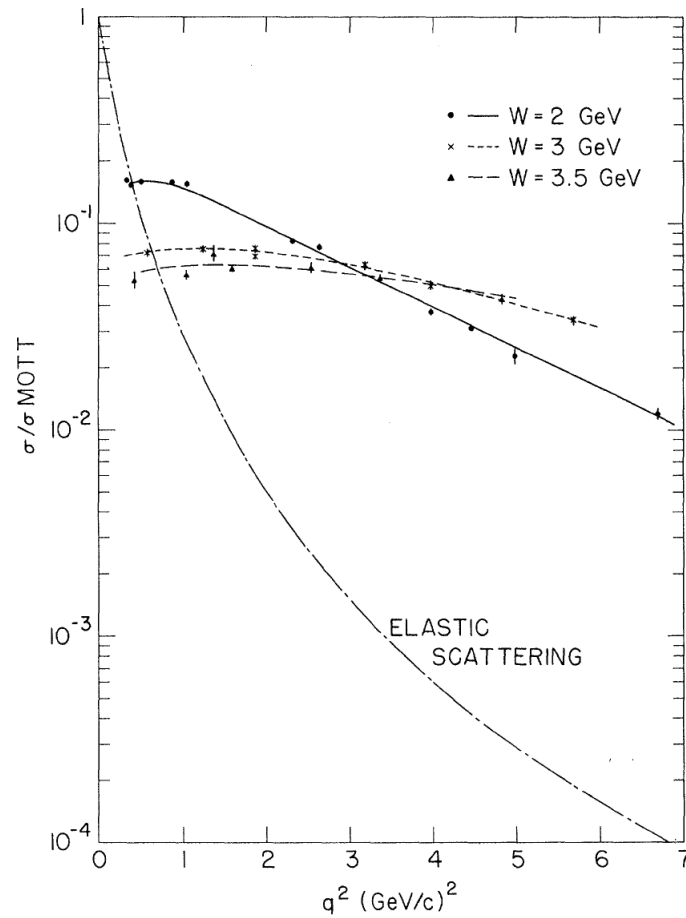


Abbildung 2.4: Normierte Wirkungsquerschnitte der Elektron-Proton Streuung. Der bei einer elastischen Streuung gegebene Abfall mit $\frac{1}{Q^8}$ steht im Gegensatz zu der schwachen Q^2 -Abhängigkeit bei verschiedenen Werten der invarianten Masse [5].

verknüpft. Elektron-Streu-Experimente am SLAC haben die Relation (2.8) in einem weiten Bereich von x und Q^2 vermessen. Das Resultat ist mit Eins verträglich und deutet darauf hin, dass Partonen ausschließlich Spin-1/2 Teilchen sind. Ein Experiment am CERN mit Neutrinos als Streuteilchen in der Gargamelle-Blasenkammer zeigten, dass Neutrinos diesselbe Art von Substruktur innerhalb des Nukleons auflösen und Partonen drittelzahlige Ladungen tragen. Quarks und Partonen haben mit Spin, Ladung und verschwindender Ausdehnung gemeinsame inhärente Eigenschaften, womit eine Identifizierung der Quarks mit geladenen Partonen gerechtfertigt werden kann [3].

2.1.5 Die Impulsverteilung im Nukleon

Aus der Verteilung der Impulsanteile der Partonen kann der Gesamtimpuls bestimmt werden, der von Partonen getragen wird. Zudem geben diese Verteilungen Aufschluss über die innere Struktur des Protons. Im Wirkungsquerschnitt der tiefinelastischen Streuung treten im Grenzwert $Q^2 \rightarrow \infty$ bei festem x_B die Quarkverteilungsfunktionen $q_f(x_B, Q^2)$ als Summe über die Quarktypen f auf:

$$\sigma(x_B, Q^2) \propto \sum_f z_f^2 q_f(x_B, Q^2). \quad (2.9)$$

Der elektromagnetische Streuquerschnitt ist proportional zum Quadrat der jeweiligen Ladungen der Quarks z_f . Demzufolge werden die Strukturfunktion als Summe über die mit x gewichteten Quark- und Antiquarkverteilungen $q_f(x)$ und $\bar{q}_f(x)$ angegeben:

$$F_2(x) = x \cdot \sum_f z_f^2 (q_f(x) + \bar{q}_f(x)) \quad (2.10)$$

Informationen über verschieden geladene Quarktypen können durch die Neutrino-Streuung erhalten werden, eine Unterscheidung zwischen Quarks und Antiquarks ist möglich. Die Kombination mit Antineutrinos erlaubt zudem eine Trennung zwischen Valenzquarks und Seequarks. Die in Abbildung 2.5 skizzierten Partonverteilungsfunktionen zeigen, dass Seequarks lediglich bei kleinen x Werten zur Strukturfunktion beitragen. Das Maximum der Strukturfunktion der Valenzquarks liegt bei $x \approx 0,2$ und fällt mit $x \rightarrow 1$ auf 0 ab. Mit den hier als $u(x)$, $d(x)$ und $s(x)$ bezeichneten Verteilungsfunktionen der Quarktypen kann der Anteil des Gesamtimpulses eines bestimmten Quarktyps durch Multiplizieren mit x erhalten werden. Aus

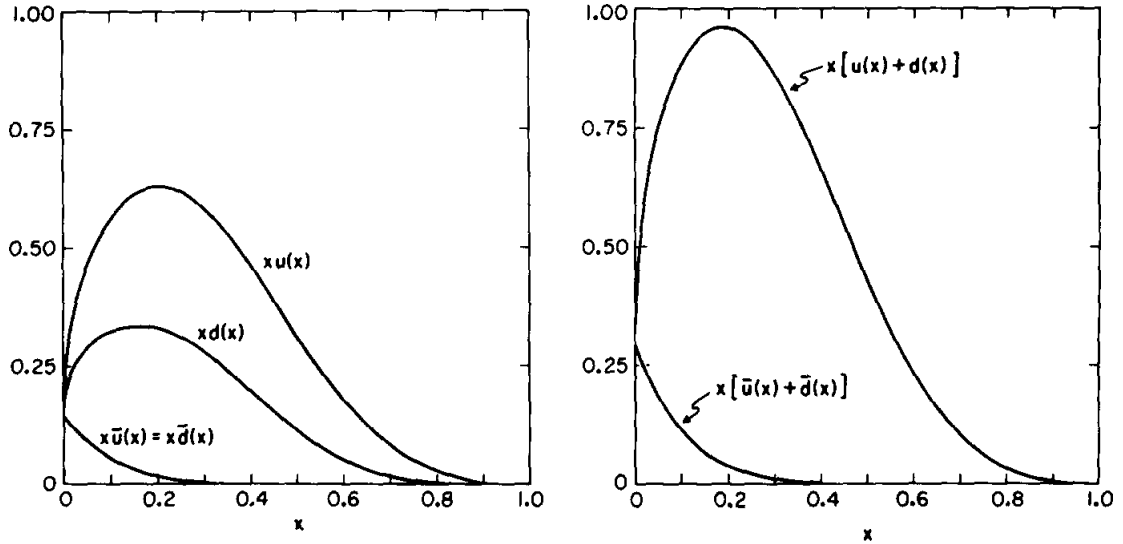


Abbildung 2.5: Partonverteilungsfunktionen im Nukleon. See-Quarks sind für Werte von $x > 0,3$ vernachlässigbar, bei höheren x -Werten dominieren u -Quarks [6].

der Impulserhaltung folgt

$$\int_0^1 dx x \sum_f (q_f(x) + \bar{q}_f(x)) = 1 - \epsilon, \quad (2.11)$$

mit ϵ als der Anteil des Gesamtimpulses, der von elektrisch neutralen Teilchen, den Gluonen, getragen wird. Die in Gleichung (2.11) auftretende Summe kann als Summe über die Strukturfunktionen $F_2(x)$ aus Elektron- und Neutrinostreuung aufgefasst werden. Damit konnte experimentell bestimmt werden, dass der Gesamtimpuls zu jeweils annähernd 50 % von Quarks und Gluonen getragen wird [7].

2.2 Der Spin des Nukleons

Um Partonverteilungsfunktionen zu bestimmen, hat sich die tiefinelastische Streuung mit Leptonen als Streuteilchen als bewährte Methode erwiesen. Helizitätsverteilungen $\Delta q_f(x)$ erlauben Aussagen über den Spin von Partonen. Dazu müssen sowohl Strahl als auch Target longitudinal polarisiert sein. Gemessen wird die Differenz von Wirkungsquerschnitten zweier Streuprozesse: Bei einer Reaktion entspricht die Polarisationsrichtung des Targetmaterials derjenigen des Strahls. Bei der anderen Reaktion zeigt die Polarisation des Targets in entgegengesetzte Richtung zur Polarisationsrichtung des Strahls, was im folgenden mit einem Exponenten $\pm$ dargestellt wird:

$$\sigma_{LL}(x) = \frac{1}{2} [\sigma^+(x) - \sigma^-(x)] \quad (2.12)$$

Aus den Daten werden sogenannte *double-spin*-Asymmetrien

$$A(x) = \frac{\sigma_{LL}(x)}{\sigma_{UU}(x)} \quad (2.13)$$

ermittelt, wobei L/U die longitudinale Polarisation bzw. unpolarisierte Teilchen sowohl des Strahls als auch des Targets bezeichnen,

$$\sigma_{UU}(x) = \frac{1}{2} [\sigma^+(x) + \sigma^-(x)] \quad (2.14)$$

beschreibt die Polarisation der Gesamtheit der Teilchen. Über die Messung von Asymmetrien wird mit den bekannten Werten σ_{LL} bestimmt. Bei einer inklusiven tiefinelastischen Streuung werden die entstandenen Hadronen nicht untersucht. In diesem Fall ist der Wirkungsquerschnitt σ_{LL} , der aus der Messung des Lepton-Streuteilchens folgt, proportional zur Strukturfunktion $g_1(x)$. Für ein Nukleon kann die Strukturfunktion als Dichteverteilung von Helizitätszuständen für einen Quarktyp f betrachtet werden:

$$g_1(x) = \frac{1}{2} \sum_f z_f^2 (\Delta q_f(x) - \Delta \bar{q}_f(x)). \quad (2.15)$$

Im folgenden werden die einzelnen Beiträge der Quarks und Gluonen zum Gesamtspin erläutert, die aus Streuexperimenten mit polarisiertem Strahl und Target bestimmt werden können. Die Erkenntnis, wonach Quarks nur zu einem geringen Anteil zum Gesamtspin des Nukleons beitragen ($\Delta\Sigma \approx 0,3$), führte zu Beginn der 1980er Jahre auf die Frage nach weiteren Einflussgrößen. Wie neuere Resultate zeigen, ist in Analogie zum Quarkspin der Spin der Gluonen nicht in entscheidender Weise am Gesamtspin des Nukleons beteiligt. Es wird vermutet, dass der Drehimpuls $\mathcal{L}$ von Quarks und Gluonen in entscheidender Weise zum Spin des Nukleons beiträgt. Das von dieser Entwicklung ausgehende Verständnis der Spinstruktur kann gemäß

einer Summenregel, die aus den Gesetzen der Quantenchromodynamik folgt, beschrieben werden [8]:

$$\frac{1}{2} = \frac{1}{2}\Delta\Sigma + \Delta G + \mathcal{L} \quad (2.16)$$

Um den Einfluss des Drehimpulses auf den Gesamtspin experimentell zu bestimmen, werden in naher Zukunft Messungen zu HEMP (engl.: *Hard exclusive Meson Production*) und DVCS (engl.: *Deep Virtual Compton Scattering*) durchgeführt. Sie erlauben die Bestimmung von generalisierten Partonverteilungen und damit die Bestimmung des Gesamtdrehimpulses von Quarks. Die Zusammensetzung des Nukleonspins aus Valenzquarks, Gluonenbeiträgen und dem Drehimpuls der Quarks ist in Abbildung 2.6 aufgezeigt.

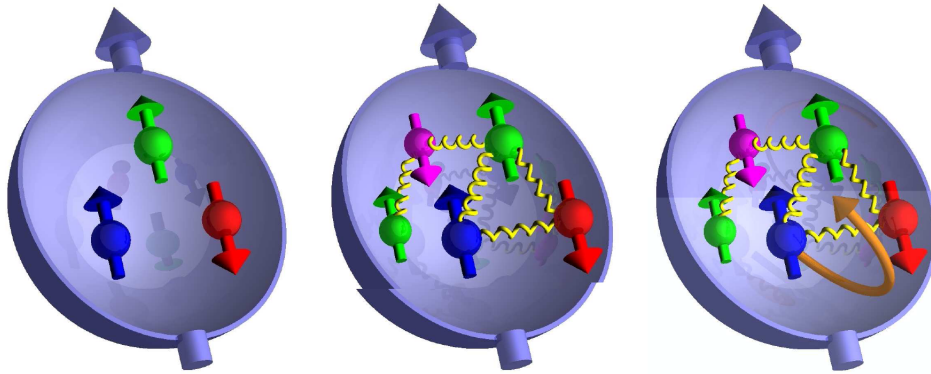


Abbildung 2.6: Die Struktur des Nukleons und seine Bausteine, die zum Gesamtspin beitragen [9]. Das Verständnis entwickelte sich über Valenzquarks (links) und Gluonen sowie Quark-Antiquark Paare (Mitte). Das Bild auf der rechten Seite zeigt das Resultat der Entwicklung, wonach der Drehimpuls von Quarks und Gluonen miteinbezogen wird.

2.2.1 Der Spin der Quarks

Mit der tiefinelastischen Streuung eines polarisierten Strahls an einem polarisierten Target ist es möglich, ausschließlich Quarks einer bestimmten Helizität als Streuteilchen auszuwählen. Die Erhaltung der Helizität besagt, dass Quarks mit der Ausrichtung des Spins in einer zum Spin des virtuellen Photons entgegengesetzter Richtung das Photon absorbieren können. Abbildung 2.7 zeigt die eindeutige Selektion der Streuteilchen durch den Spin bei longitudinaler Polarisierung der Protonen mit der Strahlachse als Quantisierungsachse. Aus der Differenz $\sigma_{LL}(x)$ der Wirkungsquerschnitte mit Polarisierung von Strahl und Target in gleicher bzw. entgegengesetzter Richtung werden die Verteilungsfunktionen der Quarks mit gleicher sowie entgegengesetzter Helizität relativ zu einem Proton im Target ermittelt. Bei inklusiver Streuung ist die spinabhängige Strukturfunktion $g_1(x)$ proportional zum Wirkungsquerschnitt $\sigma_{LL}(x)$. Ein auf diese Weise von der EMC Kollaboration am CERN gemessenes Ergebnis besagt, dass der Beitrag des Quarkspins am Gesamtspin des Protons klein ist [10]. Heute liegen große Mengen an Daten von weiteren Experimenten am CERN, SLAC und DESY vor, die zu den in Abbildung 2.8 (links) gezeigten Resultaten führen. Die rechte Seite in Abbildung 2.8 zeigt die Strukturfunktion des Protons in Abhängigkeit von Q^2 für verschiedene Werte

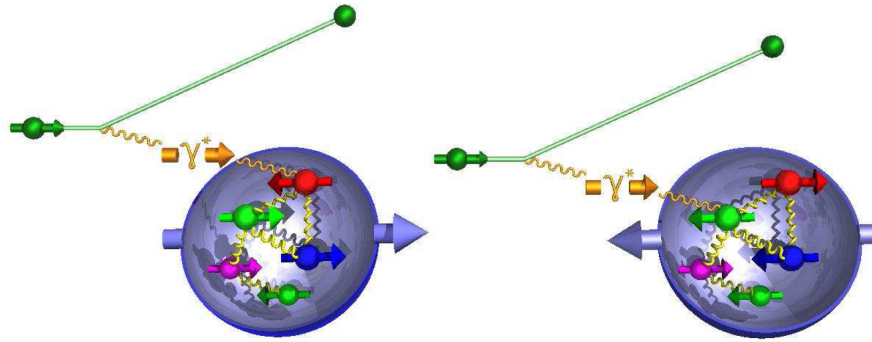
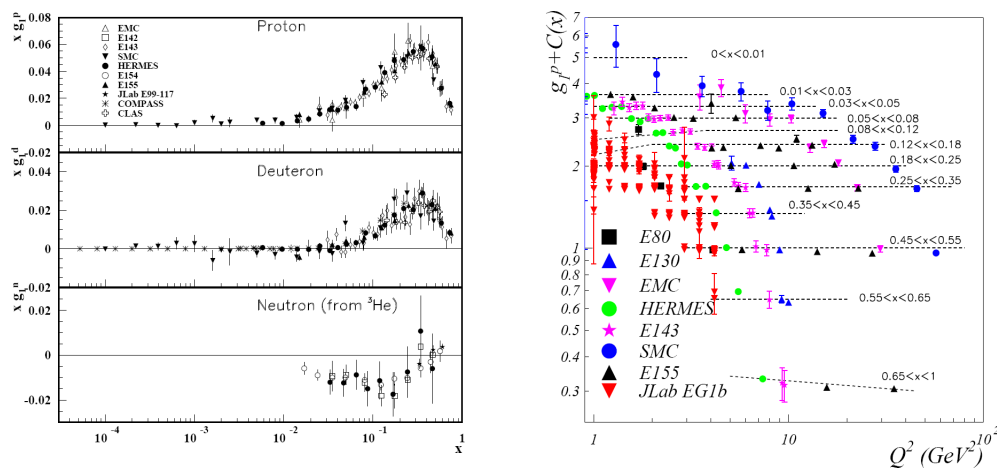


Abbildung 2.7: Die durch die Erhaltung der Helizität bedingte Auswahl von Streuteilchen [9].

Abbildung 2.8: Das aus mehreren Experimenten erhaltene Datenvolumen zur Bestimmung der polarisierten Strukturfunktion $xg_1(x)$ für das Proton, das Deuteron und das Neutron. Auf der rechten Seite sind die Daten, die aus der Streuung an Protonen erhalten wurden, zusätzlich für verschiedene x -Bereiche aufgetragen.

von x . Die schwache Abweichung im Skalenverhalten kann auf den Einfluss der polarisierten Gluondichte zurückgeführt werden [11].

2.2.2 Der Spin der Gluonen

Während das virtuelle Photon in der tiefinelastischen Streuung von Leptonen über die elektrische Ladung direkt an ein Quark koppelt, sind Gluonen nur indirekt zugänglich. Ihr Einfluss ist in dem schwachen, logarithmischen Verhalten der Quarkdichten auszumachen. Die mit dieser Abweichung verbundene Verletzung des Bjorkenschen Skalenverhaltens kann verwendet werden, um den Beitrag ΔG der Gluonen zum Impuls und Spin des Nukleons zu bestimmen: Messungen am COMPASS-Experiment bis zu hohen Werten von Q^2 ($Q^2 \approx 100 \text{ GeV}^2$) und Deuterium als Targetmaterial konnten die Genauigkeit von g_1 deutlich verbessern [12].

Der Einfluss der Gluonen auf den Gesamtspin des Nukleons kann auf direkte Weise aus der

Photon-Gluon Fusion

$$\gamma^* g \rightarrow q\bar{q}$$

bestimmt werden. Dabei entsendet das Gluon ein Quark-Antiquark Paar, bei dem jedes Quark die Helizität des Gluons trägt. Das Quark absorbiert das virtuelle Photon, wenn beide Teilchen identische Helizität besitzen.

In Übereinstimmung sämtlicher Daten kann der Beitrag der Gluonen zum Spin des Nukleons mit $|\Delta G| \approx 0,2$ angegeben werden [13]. Eine hochpräzise Angabe erfordert Messungen in einem ausgedehnten Bereich von x [11].

2.2.3 Der Beitrag des Drehimpulses

Zur Ermittlung der Drehimpulse von Quarks und Gluonen, die das Spin-Budget des Nukleons vervollständigen, werden aus den Daten kommender Messungen der tiefvirtuellen Compton Streuung sowie der exklusiven Mesonproduktion Generalisierte Partonverteilungsfunktionen (GPD) ermittelt. Aus Ihnen können sehr allgemeine und tiefgehende Informationen über die Struktur im Nukleon erhalten werden. Sie ermöglichen bei spezieller Betrachtung Aussagen über Formfaktoren, Partondichtevertellungen oder Reaktionswahrscheinlichkeiten. Eine implizite Eigenschaft einer Generalisierten Partonverteilung H ist, dass für festen Impulsübertrag t auf das Target, daraus die Formfaktoren bezüglich des mittleren Impulsanteiles x des betrachteten Quarktyps f bestimmt werden können:

$$\int dx H^f(x, \xi, t) = F_1^f(t) \quad (2.17)$$

Während H^f zum Dirac-Formfaktor $F_1^f(t)$ führt, liefern die Quark GPDs E^f , $\tilde{H}^f$ und $\tilde{E}^f$ die einzelnen Beiträge zum Pauli-Formfaktor, zum axialen sowie zum pseudoskalaren Formfaktor. Zur Berücksichtigung der Richtungsabhängigkeit des Impulsübertrages dient die Variable ξ . Im Falle von verschwindenden t und ξ reduzieren sich die helizitätsunabhängige GPD $H^f(x, 0, 0)$ und helizitätsabhängige GPD $\tilde{H}^f(x, 0, 0)$ auf die entsprechenden Partonverteilungsfunktionen $q_f(x)$ sowie $\Delta q_f(x)$. Anwendung erlangten die GPDs bei der Einführung der Ji -Summenregel, nach der ein zweites Moment von GPDs zum Gesamtdrehimpuls der Quarks vom Typ f oder von Gluonen führt:

$$\frac{1}{2} = \sum_f J_f + J_G \quad (2.18)$$

mit

$$J_f = \frac{1}{2} \lim_{t \rightarrow 0} \int_{-1}^1 dx x [H^f(x, \xi, t) + E^f(x, \xi, t)], \quad (2.19)$$

$$J_G = \frac{1}{2} \lim_{t \rightarrow 0} \int_0^1 dx [H^G(x, \xi, t) + E^G(x, \xi, t)] \quad (2.20)$$

Somit kann der vollständige Beitrag der Quarks und Gluonen zum Nukleonspin aus einer Messung der GPDs extrahiert werden. Gleichung (2.19) ermöglicht über

$$J_f = \frac{1}{2}(\Delta q_f + \Delta \bar{q}_f) + L_f \quad (2.21)$$

den Zugang zum Drehimpulsanteil der Quarktypen f . Für einen experimentellen Zugang stehen die beiden Kanäle HEMP und DVCS zur Verfügung, in denen GPDs die Verbindung zwischen Anfangszustand und Endzustand des im Streuprozess involvierten Quarks darstellen.

Exklusive Meson-Produktion

Die Produktion eines Mesons aus der Streuung eines virtuellen Photons an einem Quark kann durch ein Handbag-Diagramm beschrieben werden (Abbildung 2.9). Ein Handbag-Diagramm stellt das Quadrat der Streuamplitude in Vorwärtsrichtung dar. Diese Größe ist über das optische Theorem mit dem Wirkungsquerschnitt verknüpft. Die Wechselwirkung des virtuellen Photons mit dem Parton kann dabei vom Nukleon getrennt betrachtet werden. Durch die Streuung des virtuellen Photons an einem Parton mit longitudinalem Impulsanteil $x + \xi$ entsteht ein Meson. Das Parton trägt im Anschluss einen verringerten Impulsanteil $x - \xi$ und kehrt damit in den Endzustand des Nukleons zurück. Der Zugang zu den Generalisierten Partonverteilungsfunktionen kann über die Streuamplituden der beiden in diesem Prozess auftretenden Zustände erhalten werden. Voraussetzung für die Faktorisierung in einen weichen und harten Anteil ist die Streuung mit longitudinal polarisierten Photonen [14]. Die Entste-

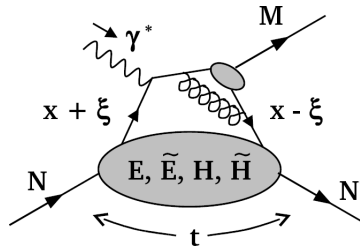


Abbildung 2.9: Das Handbag-Diagramm zur exklusiven Meson-Produktion. Die dazugehörige Amplitude ist durch die obere Fläche gekennzeichnet, die darunterliegende Fläche entspricht den Generalisierten Partonverteilungen. Der Impulsübertrag zwischen dem Anfangs- und Endzustand des Nukleons $\Delta = P' - P$ enthält den longitudinalen Anteil $\xi = \frac{x}{2-x}$ und das Quadrat des Impulsübertrags ist mit $t = \Delta^2$ gekennzeichnet.

hung des Mesons lässt sich auf den Zerfall eines aus der Streuung resultierenden Gluons in ein Quark-Antiquark Paar zurückführen. Das Antiquark rekombiniert mit dem Streuquark zu einem Meson. Bei dieser Meson-Produktion durch longitudinale Photonen an einem unpolarisierten Target können einzelne Paare von Quark-GPDs extrahiert werden. Die Produktion von Vektormesonen wie dem ρ^0 führt zu den GPDs H^f und E^f , während aus der Produktion pseudoskalarer Mesonen wie zum Beispiel Pionen die GPDs $\tilde{H}^f$ und $\tilde{E}^f$ erhalten werden. Diese Paare von Quark-GPDs erhalten bzw. vertauschen die Helizität des Nukleons [15]. Um den Wirkungsquerschnitt in Anteile longitudinaler und transversaler Photonen zu trennen, verwendet man die Erhaltung der Helizität im s-Kanal. Demzufolge erhält das Meson die Helizität des Photons und ist nach obiger Argumentation longitudinal polarisiert. Aus der Winkelverteilung des Zerfalls eines Vektormesons können Informationen über seine Helizität erhalten werden. Auf diese Weise ist es möglich, von den Wirkungsquerschnitten polarisierter Mesonen auf Wirkungsquerschnitte longitudinaler und transversaler Photonen zu schließen.

und damit zu Informationen über GPDs zu gelangen [16].

Tief-virtuelle Compton-Streuung

Bei der Streuung eines Leptons an einem Quark entsteht ein reales Photon. Analog zur exklusiven Mesonproduktion kann die Streuung des virtuellen Photons an einem Quark vom Nukleon getrennt betrachtet werden. Der longitudinale Impulsanteil $x + \xi$ des Quarks wird durch die Wechselwirkung mit dem Photon auf den Anteil $x - \xi$ verringert und ein dabei produziertes reales Photon kann detektiert werden. Bei festen Werten des Impulsübertrages t zwischen Anfangs- und Endzustand des Nukleons finden sich Informationen über die Quark GPDs ($H, \tilde{H}, E, \tilde{E}$) in der Korrelation der beiden Zustände. Die t -Abhängigkeit der Quark GPDs, insbesondere der Verlauf mit $t \rightarrow 0$ kann mit DVCS bestimmt werden. Damit erfolgt die Berechnung des zweiten Momentes der Summe $H^f + E^f$ bezüglich x und die Bestimmung des Gesamtdrehimpulses J_f nach Gleichung (2.19). Abbildung 2.10(a) zeigt das dazugehörige *Handbag*-Diagramm. Für eine exklusive Messung ist es erforderlich, neben dem produzierten Photon auch das rückgestoßene Proton zu detektieren. Darüberhinaus ist der zu DVCS gehörende Endzustand nicht eindeutig. Im Bethe-Heitler Prozess wird ein reales Photon von dem einlaufenden oder auslaufenden Lepton produziert. Aus Abbildung 2.10 geht hervor, dass sowohl in DVCS als auch im Bethe-Heitler Prozess ein Photon im Endzustand vorliegt. Der Bethe-Heitler Prozess ist ein elastischer Streuprozess, der unter Abstrahlung von Ener-

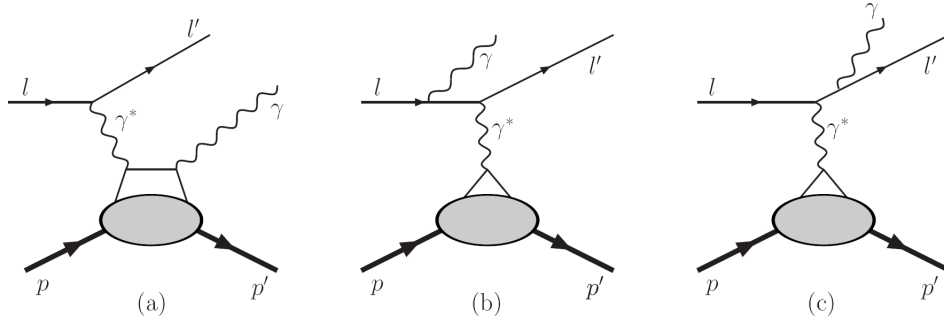


Abbildung 2.10: Produktion realer Photonen aus der Lepton Streuung. Tiefvirtuelle Compton Streuung(a). Bethe-Heitler Prozess (b, c)

gie stattfindet, und mit in die Streuamplituden eingeht. Demzufolge wird der differenzielle Wirkungsquerschnitt für die Produktion realer Photonen durch Leptonen ausgedrückt durch [15]

$$\frac{d\sigma(ep \rightarrow ep\gamma)}{dx dQ^2 d|t| d\phi} \propto |\tau_{BH}|^2 + |\tau_{DVCS}|^2 + \underbrace{\tau_{DVCS}\tau_{BH}^* + \tau_{DVCS}^*\tau_{BH}}_I. \quad (2.22)$$

Die ununterscheidbaren Endzustände führen zu einem Interferenzterm I . Der Winkel ϕ ist der azimuthale Winkel zwischen Ebenen, die durch die jeweiligen Lepton- und Photonflugrichtungen aufgespannt werden. Die Amplitude des Bethe-Heitler Prozesses kann berechnet werden, und der reine DVCS Anteil ist durch Integration über die azimuthale Winkelabhängigkeit des Wirkungsquerschnittes zugänglich. Die experimentelle Bestimmung der komplexwertigen Amplitude des DVCS Prozesses kann über die Messung der azimuthalen Winkelabhängigkeit

von I vollzogen werden. Somit dient der Bethe-Heitler Prozess als Referenz für die DVCS Prozessamplitude und schafft durch seine große Reaktionswahrscheinlichkeit eine Grundlage, den DVCS-Prozess in der erforderlichen Präzision zu untersuchen [15].

Kapitel 3

Das COMPASS-Experiment

COMPASS ist ein Experiment am CERN mit festem Target. Sowohl Hadronen als auch Myonen als Strahlteilchen treffen mit Energien im Bereich von 100–200 GeV auf ein polarisierbares Target. Das Spektrometer ist aus zwei Teilen aufgebaut (siehe Abbildung 3.1). Das *Large Angle Spectrometer* (LAS) befindet sich strahlabwärts unmittelbar hinter dem Target und überdeckt einen Polarwinkelbereich bis 180 mrad. Im Anschluss werden mit dem *Small Angle Spectrometer* Teilchen innerhalb eines Polarwinkels von ± 30 mrad detektiert. In beiden Abschnitten befindet sich jeweils ein Dipolmagnet (SM1 und SM2), Detektorsysteme zur Spurrekonstruktion sowie die zur Teilchenidentifikation eingesetzten elektromagnetischen (ECAL1, ECAL2) bzw. hadronischen Kalorimeter (HCAL1, HCAL2) und Myonfilter. Zur Identifikation von Hadronen wird ein RICH-Detektor eingesetzt. Der RICH-Detektor befindet sich nur im LAS und ermöglicht eine Trennung von Pionen und Kaonen im gesamten Akzeptanzbereich bis zu einer Energie von 50 GeV.

3.1 Der Strahl und das Target

Aus dem Super Proton Synchrotron (SPS) des CERN werden hochenergetische Protonen ($E \approx 400$ GeV) auf ein Beryllium Target geleitet. Der Fluss der Teilchen erreicht die Größenordnung von $1,2 \cdot 10^{13}$ Protonen pro 4,8 s Extraktionsdauer (Spill). Die Periodendauer eines Spillzyklus beträgt typisch 16,8 s. Beim Auftreffen auf das Target entstehen Pionen und zu einem Anteil von 3,6 % Kaonen. Durch die Wahl der Dicke des teilchenerzeugenden Targets kann der effektive sekundäre Teilchenfluss variiert werden. Die neu entstandenen Teilchen werden über Quadrupol- und Dipolmagnete fokussiert und durch einen 600 m langen Kanal mit fokussierenden und defokussierenden Elementen zum Experiment geleitet. Zu Beginn dieses Transports zerfällt ein Teil der Reaktionsprodukte jeweils in ein Myon und ein Neutrino. Eine mit diesem schwachen Zerfall verbundene Paritätsverletzung führt zur longitudinalen Polarisation der geladenen Teilchen [18]. Bei Messungen mit einem $\mu^\pm$ -Strahl werden die verbleibenden Hadronen mit Absorbern aus dem Strahl entfernt. Paare von Cherenkov-Zählern dienen zur Identifikation der im Strahl enthaltenen Pionen, Protonen und Kaonen.

Die Polarisation des Targets wird durch *Dynamic Nuclear Polarisation* (DNP) erreicht: Hierbei übertragen Mikrowellen die Polarisation der Elektronen auf den Kernspin [19]. Um den Einfluss der Dipolfelder auf den systematischen Fehler zu kompensieren, wird die Polarisation des Targets regelmäßig umgekehrt. Bei den Messungen kommen verschiedene Targetmaterialien

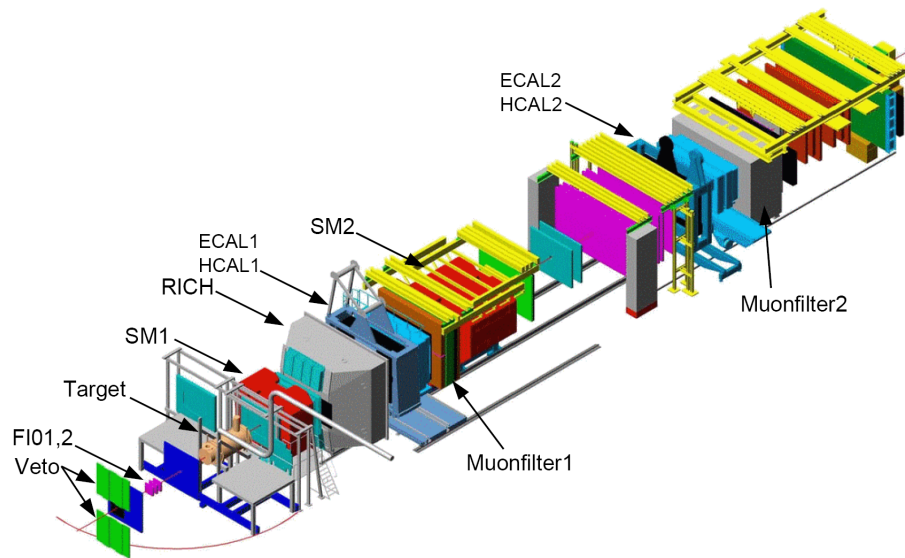


Abbildung 3.1: Das COMPASS-Spektrometer [17]. Der Myonenstrahl trifft von unten links auf das polarisierte Target. Die gestreuten und produzierten Teilchen werden in einer zweistufigen Detektoranordnung registriert. Teilchen mit großen Streuwinkeln werden im ersten Teil des Spektrometers (SM1) detektiert. Der zweite Teil (SM2) ist für Teilchen bestimmt, die unter kleinen Winkeln vom Target ausgehen.

lien zum Einsatz: ${}^6\text{LiD}$ kann als ein Spin-0 ${}^4\text{He}$ -Kern mit zwei Deuteronen betrachtet werden und weist einen Anteil des polarisierbaren Materials von 0,35 auf. ${}^6\text{LiD}$ kann typisch bis ca. 50 % polarisiert werden. Das Festkörpertargetmaterial NH_3 hat einen Anteil an polarisierbarem Material von lediglich 15% ($f \approx 0.15$). Aufgrund seines überaus hohen Polarisationsgrades ($> 80\%$) stellt dieses Material trotzdem eine genügende Zahl von polarisierten Protonen bereit [20, 21]. DVCS-Messungen erfordern ein möglichst reines Protontarget. Bei COMPASS wird zu diesem Zweck flüssiger Wasserstoff eingesetzt, eine Optimierung der Luminosität wird durch eine Targetlänge von 2,5 m erreicht.

3.2 Die Spurdetektoren

Die unterschiedlichen lokalen Gegebenheiten im Experiment erfordern Detektoren, die für einen speziellen Einsatzbereich optimiert sind. Während Detektoren, die sich im Strahl sowie in seiner unmittelbaren Nähe befinden, bei verhältnismäßig großem Teilchenstrom eine sehr präzise Winkelauflösung bereitstellen müssen, sind die Anforderungen in weiter entfernt liegenden Bereichen weniger eng gesteckt. Am COMPASS Experiment werden drei Kategorien von Spurdetektoren eingesetzt, die wie folgt zusammengefasst werden können:

- *Very Small Area Trackers* (VSAT). Acht Einheiten von szintillierenden Faserdetektoren befinden sich in einem kleinen Bereich in und um den Strahl. Vor dem Target sind Siliziumstreifen angebracht. Zur Vermeidung von sekundären Streuungen ist es notwendig, den Materialumfang in diesen Bereichen möglichst gering zu halten.

- *Small Area Trackers* (SAT). In Bereichen, die mehr als 2,5 cm vom Rand des Strahles entfernt liegen, werden Detektoren eingesetzt, die einen mittleren Größenumfang nicht übersteigen und eine hohe Ortsauflösung ermöglichen. Sowohl Micromegas als auch GEM Detektoren sind für diesen Einsatz geeignet. Inaktive Detektorregionen werden durch Elemente der VSAT abgedeckt.
- *Large Area Trackers* (LAT). Für große Winkelbereiche werden Detektoren eingesetzt, die eine gute räumliche Auflösung aufweisen und Flächen überdecken, die durch die Akzeptanz des Aufbaus begrenzt werden. Die hierbei verwendeten Driftkammern, Violdrahtkammern (MWPC) und *Straw Tube Tracker* sind um die beiden Spektrometermagneten angeordnet.

3.3 Die Detektoren zur Teilchenidentifikation

Die beiden Teilbereiche des COMPASS Spektrometers LAS und SAS enthalten jeweils ein elektromagnetisches Kalorimeter (ECAL) und ein hadronisches Kalorimeter (HCAL). Die elektromagnetischen Kalorimeter bestehen hauptsächlich aus Blöcken von Bleiglas. Sie garantieren selbst bei hohen Raten eine Rekonstruktion von Photonen mit guter Energie- und Ortsauflösung. Die aus Blei und Szintillatormodulen aufgebauten Shashlik-Module weisen eine erhöhte Winkel- und Energieauflösung auf. Hadronenkalorimeter messen die Energie von Hadronen. Darüberhinaus erzeugen sie bei inelastischer Streuung von Myonen ein Triggersignal. Um eine gute Auflösung zu gewährleisten sind sie aus übereinanderliegenden Schichten von Eisen und Szintillatorplatten aufgebaut. Zudem dient ein RICH Detektor im LAS zur Identifizierung von Pionen, Kaonen und Protonen im gesamten Winkelbereich bis zu einer Teilchenenergie von 50 GeV. Im Anschluss an die Hadronkalorimeter befindet sich in beiden Abschnitten ein Myonfilter, der Hadronen absorbiert. Im LAS wird dazu ein 60 cm dicker Eisenabsorber eingesetzt, das SAS enthält einen Betonabsorber mit einer Dicke von 2,4 m. Diese Absorber sind auf beiden Seiten umgeben von Detektorebenen zur Rekonstruktion der Teilchenspuren und ein Myon gilt als nachgewiesen, wenn eine Spur in beiden Detektoranordnungen oberhalb und unterhalb des Filters rekonstruiert werden kann.

3.4 Der Einbau des Rückstoß-Detektors

Studien haben ergeben, dass Messungen von tiefvirtueller Comptonstreuung zu weitreichenden Informationen über GPDs führen, sodass aus ihnen Rückschlüsse auf den Gesamtdrehimpuls eines Quarktyps gezogen werden können. Dieser Prozess beinhaltet die Streuung eines Leptons an einem Targetproton, woraus im Endzustand neben dem Myon und einem erzeugten Photon das gestreute Proton zu erwarten ist:

$$\mu p \rightarrow \mu p \gamma$$

Um exklusive Messungen von DVCS zu gewährleisten, muss das rückgestreute Proton detektiert und die Erzeugung weiterer Teilchen durch ein Veto signalisiert werden. Dies wird durch die Erweiterung des COMPASS-Spektrometers durch einen Rückstoß-Detektor ermöglicht. Der Prototyp dieses Detektors wurde für einen Betrieb im Hadron-Experiment konzipiert und ist aus zwei Ringen von jeweils mehreren Szintillatorstreifen aufgebaut, sodass die Flugzeit

der Teilchen mit der erforderlichen Auflösung von $\sigma = 200$ ps gemessen werden kann. Damit konnten erste DVCS-Testmessungen durchgeführt werden. Der RPD im Myon-Experiment ist deutlich größer und wird mit einem 2,4 m langem Target betrieben. In Verbindung mit Energieverlustmessungen im Szintillatormaterial ist es möglich, Protonen von Pionen und Deltaelektronen sowie Halo-Myonen zu unterscheiden und daraus einen Protontrigger zu generieren. In Abbildung 3.2 ist ein vertikaler Schnitt durch den Detektor skizziert. Im Zentrum des Detektors befindet sich das Target, an dem die Streuung stattfindet. Der innere Ring A besteht derzeit aus 12 Szintillatorstreifen, die in einem Abstand von 120 mm zur Strahlachse angebracht sind. Der äußere Ring B liegt 775 mm von der Strahlachse entfernt. Die Anordnung deckt dabei den Winkel zwischen Strahl und der dazu senkrecht stehenden Achse im Bereich von 55° bis 90° ab, in dem das rückgestoßene Proton erwartet wird. Ein Trigger wird in Koinzidenz von einem Streifen im inneren Ring mit einem von drei möglichen Streifen im äußeren Ring erzeugt [22]. Geladene Teilchen geben beim Durchgang durch Materie kinetische

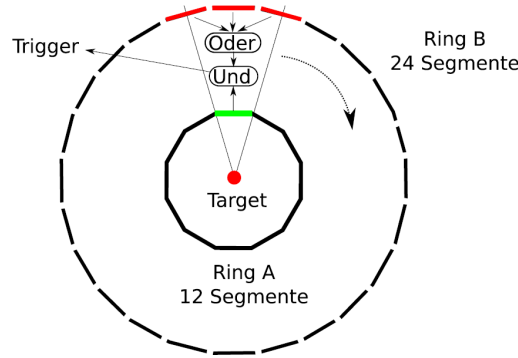


Abbildung 3.2: Die Anordnung der Szintillatorstreifen im Rückstoß-Detektor.

Energie ab und regen dadurch gebundene Elektronen an oder ionisieren Atome. Der mittlere Energieverlust dE eines geladenen Teilchens pro zurückgelegter Strecke dx in einem Medium berechnet sich nach Bethe und Bloch wie folgt [23]:

$$-\frac{dE}{dx} = 4\pi N_A r_e^2 m_e c^2 z^2 \frac{Z}{A} \frac{1}{\beta^2} \left[\ln \left(\frac{2m_e c^2 \gamma^2 \beta^2}{I} \right) - \beta^2 - \frac{\delta}{2} \right] \quad (3.1)$$

mit

z – Ladung des einfallenden Teilchens in Einheiten der Elementarladung

Z, A – Kernladungszahl und Massenzahl des Absorbers

$\beta c = v$ Geschwindigkeit des einfallenden Teilchens

m_e – Elektronmasse

r_e – klassischer Elektronenradius

N_A – Avogadro Zahl

I – eine für das Material typische Ionisationskonstante

δ – Parameter des Dichteeffekts. Abschirmung des transversalen elektrischen Feldes der einfallenden Teilchen durch die Ladungsdichte der Atomelektronen

Misst man den Energieverlust von Teilchen an mehreren Orten entlang ihrer Flugbahn, erhält man für jede Teilchensorte einen charakteristischen Verlauf der abgegebenen Energien. Abbildung 3.3 zeigt den Verlauf der Energien im Rückstoß-Detektor für Protonen, Pionen und Elektronen. Mit wachsender Geschwindigkeit fällt der Energieverlust eines Teilchens nach

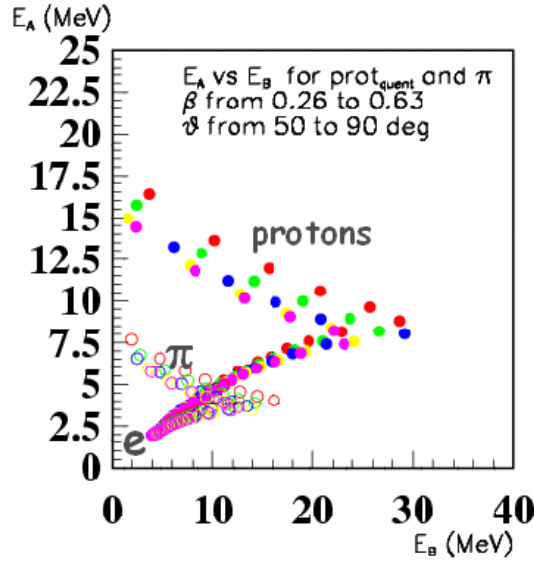


Abbildung 3.3: Aus Simulationen erhaltener Energieverlust von Teilchen in den Detektorringen [24]. Damit ist eine Identifizierung von Protonen durch das Setzen von entsprechenden Triggerschwellen garantiert.

Bethe und Bloch auf ein Minimum ab. Durchquert ein Teilchen den Detektor mit hoher Geschwindigkeit, gibt es demnach nur wenig Energie im Szintillatormaterial ab. Je langsamer ein Teilchen ist, desto mehr Energie ist im Detektor zu erwarten. Bei einer bestimmten Geschwindigkeit hat ein Teilchen im ersten Szintillatorelement bereits einen Großteil seiner Energie verloren, sodass es im zweiten Element absorbiert wird. Im äußersten Fall findet eine Absorption bereits im ersten Szintillator statt. Die bei der Wechselwirkung eines Teilchens mit dem Szintillatormaterial entstandene Energie im äußeren Ring B wurde in Abhängigkeit der Flugzeit berechnet. Abbildung 3.4 zeigt die Energieaufnahme des Detektorrings für verschiedene geladene Teilchen in Abhängigkeit der aus der Flugzeit erhaltenen Geschwindigkeit [24].

In [25] wurden verschiedene Strategien zur Bestimmung des Zeitnullpunkts eines Signals untersucht. Dabei empfahl sich die *analog Constant Fraction Discrimination* (aCFD) als die am besten geeignete Methode. Sie erlaubt die Bestimmung des Zeitnullpunktes mit einer Auflösung von < 50 ps und ermöglicht darüberhinaus die Bestimmung von Zeiten zweier sich überlagernder Pulse.

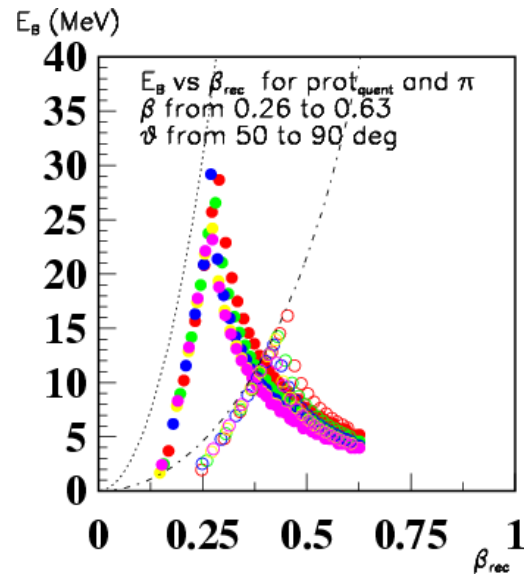


Abbildung 3.4: Berechneter Energieverlust im äußeren Ring B des Rückstoßdetektors in Abhängigkeit von der Teilchengeschwindigkeit, die aus den Flugzeitmessungen erhalten wird.

Kapitel 4

Das Datennahmesystem am COMPASS Experiment

Eine hohe Intensität des Strahls sowie Triggerraten von 10 KHz führen zu großen Mengen an Daten. Das Datennahmesystem am COMPASS-Experiment ermöglicht die Weiterleitung der Daten unter diesen Bedingungen. In diesem Kapitel wird der Aufbau des Datennahmesystems und die Erzeugung eines experimentweiten Triggersignales beschrieben. Um die Trigger an die Ausleseelektronik zu verteilen und die Module zu synchronisieren, wird ein Trigger-Verteilungssystem eingesetzt, dessen Eigenschaften an dieser Stelle ebenfalls betrachtet werden.

4.1 Der Aufbau des Datennahmesystems

Das Datennahmesystem wurde ausgehend von den erwarteten hohen Datenraten unter der Berücksichtigung der Spillstruktur des Strahls und der großen Zahl der Detektorkanäle entwickelt. Die Daten werden deswegen bereits detektornah digitalisiert und mit einer Ereignisnummer gekennzeichnet. Eine detektorspezifische Auslese ist mit wenigen verschiedenen Front-End Einheiten realisiert. Zur Sortierung, Markierung und Weiterleitung der Experimentdaten dienen die in Freiburg entwickelten und gebauten CATCH-Module sowie in einigen Fällen GeSiCA-Module. Diese leiten die Triggerinformationen an die Detektoren weiter und empfangen die Daten von sämtlichen am Experiment eingesetzten Front-End Karten. Diese Daten werden in Ereignis-Fragmenten zusammen mit der Identifikationsnummer des Ereignisses über die S-Link Schnittstelle an Zwischenspeicher (engl.: *readout buffer*, ROB) weitergeleitet (siehe Abbildung 4.1). Die ROB-PCs enthalten PCI Spillbuffer-Karten, die während der Extraktionsdauer ankommende Ereignis-Fragmente zwischenspeichern und in der Spillpause weiterleiten. Ein Gigabit-Ethernet Netzwerk führt die Daten über ein Switch-System zu den *event builder* PCs, die die gesamten Daten eines Ereignisses zu einem abgeschlossenen Ereignisblock kombinieren und zu einem zentralen Datenspeichersystem weiterleiten. Die zur Steuerung des Datenstroms eingesetzte Software DATE beinhaltet sowohl die Bündelung der Daten (*event building*) und die Kontrolle der Aufnahmezyklen (*run control*) als auch die Bereitstellung von Laufzeitinformationen (*logging*) und die Sortierung der Ereignisse. Als ein Maß für das Datenvolumen kann die Datenmenge eines Spills herangezogen werden. Sie umfasst ungefähr 10 GB,

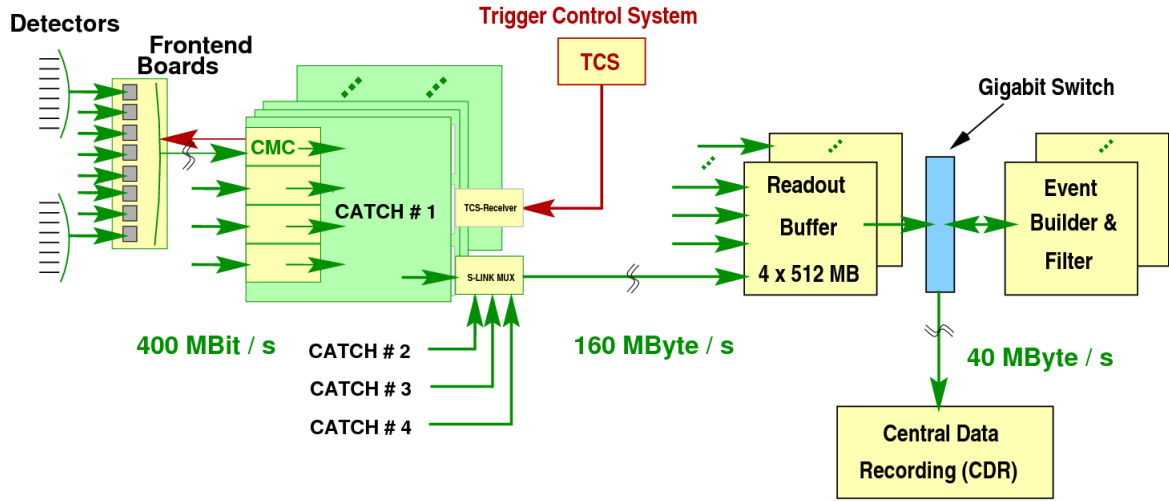


Abbildung 4.1: Die Architektur des COMPASS-Datennahmesystems [26].

sodass > 600 TB Daten pro Jahr an die Speichereinheiten übergeben werden.

4.2 Der Triggergenerierung

Die ereignisgesteuerte Auslese am COMPASS-Experiment beruht auf der Detektierung des gestreuten Strahlteilchens. Dies ist mit Messungen der Energie von produzierten Hadronen verbunden sowie der Unterdrückung der Datennahme bei Teilchen aus dem weiteren Umfeld des Strahls (engl.: *beam halo*). Dabei wird zwischen Ereignissen der tiefinelastischen Streuung und der Produktion eines quasi-realen Photons mit einem Impulsübertrag von $Q^2 < 0,5 \text{ (GeV/c)}^2$ unterschieden. Letzteres wird mit drei Detektorsystemen realisiert (Inner, Ladder, Middle), die aus paarweise angeordneten Hodoskopen bestehen und ein am Target gestreutes Myon registrieren. Die Anordnung erlaubt eine Messung des Energieverlustes der gestreuten Myonen bei einer Dauer bis zur Triggerentscheidung von $< 500 \text{ ns}$ unter Berücksichtigung eines lokal verschiedenen Teilchenflusses. Ein in Abbildung 4.2 skizziertes Hodoskopsystem misst den Abstand des gestreuten Teilchens von der Strachlache an zwei verschiedenen Positionen x_4 und x_5 . Diese Informationen werden zur Berechnung des Winkels θ_x und des durch die Magnete hervorgerufenen Ablenkwinkels α herangezogen. Damit erfolgt die Berechnung der Energie des gestreuten Myons innerhalb des Energiebereiches, der durch die gegebene Kombination der beiden Szintillatoren bestimmt ist [27].

Darüberhinaus verlieren Myonen auch durch Streuung an Elektronen oder durch Strahlung einen Teil ihrer Energie. Die resultierende Restenergie dieser Teilchen entspricht einem Energiebereich der gestreuten Myonen. Aus diesem Grund ist eine ereignisgesteuerte Datennahme ausschließlich auf Basis gestreuter Myonen nicht möglich. Der kalorimetrische Trigger wird durch erzeugte Hadronen bei Erreichen einer bestimmten Energieschwelle ausgelöst. In Koinzidenzschaltung mit den Hodoskopsystemen garantiert er eine effiziente Triggergenerierung mit hoher Trefferquote.

Die kinematische Region $Q^2 > 0,5 \text{ (GeV/c)}^2$ wird durch das Middle System sowie ein weiteres Hodoskopsystem (Outer) überdeckt, das Myonen bis zu $Q^2 \approx 20 \text{ (GeV/c)}^2$ berücksichtigt. Eine

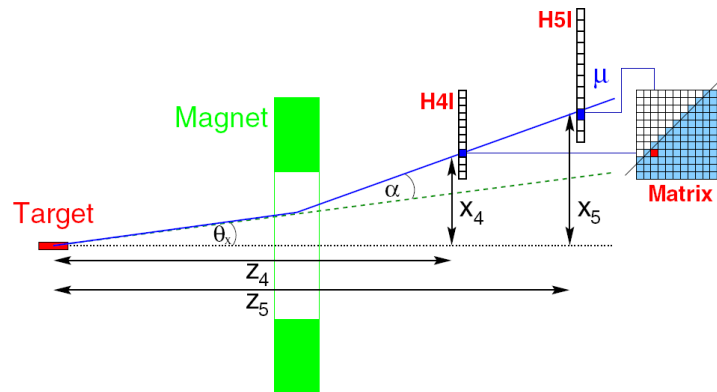


Abbildung 4.2: Ein Hodoskopsystem zur Berechnung der Energie des ausgetauschten Photons. Das aus den beiden Spektrometermagneten resultierende Magnetfeld ist durch einen effektiven Magneten dargestellt.

Reduktion des Datenvolumens wird am COMPASS Experiment mit einem speziellen Veto-System erreicht, das Myonen unterdrückt, die aus dem äußeren Ring des Strahls stammen [28].

4.3 Das Trigger Control System

Das Trigger Control System ist ein zweikanaliges optisches Übertragungssystem, das Informationen aus einer Quelle an mehrere hundert Front-End Module verteilt. Diese Informationen setzen sich zusammen aus ereignisspezifischen Triggern, Signalen zur Kontrolle von elektronischen Modulen und Detektoren sowie synchronen Zeitreferenzen mit einer Genauigkeit von 50 ps RMS [29]. Die Übertragung des Triggersignals unterliegt dabei einer festen Latenzzeit im Bereich einer Mikrosekunde in Bezug auf den Zeitpunkt, zu dem ein Teilchen das Target durchquert. Damit das Datennahmesystem die Sortierung und Zusammenfassung der Ereignisse vornehmen kann, leitet das TCS die Spill- und Ereignisnummern zusammen mit dem Ereignistyp bei jedem Trigger an die CATCH-Module weiter. Dort werden diese Information an jedes Datenpaket angehängt. Mit einer einstellbaren Totzeit hält das TCS Trigger zurück, welche nicht von der Front-End Elektronik angenommen werden können. Hierbei kann sowohl die Zeit nach einem Trigger als auch die Anzahl der Triggerereignisse in einem bestimmten Zeitfenster eingestellt werden. Das TCS erlaubt zudem eine Triggerung aus Teilbereichen der Detektoren und die Übertragung von Zufallstriggern zu Test- und Kalibrationszwecken.

Das zentrale Element des Trigger Control Systems ist der TCS-Controller, der Trigger über Kanal A weiterleitet, die Identifikation der Ereignisse vornimmt und die Totzeiten generiert. Auch der für Kontrollsignale vorgesehene Kanal B wird vom Controller angesteuert. Diese Signale gehen vom TCS Server aus, der den TCS Controller überwacht und eine Kommandoschnittstelle zwischen der TCS Hardware und der Datennahme bzw. der dazugehörigen Software darstellt. Das am CERN entwickelte Verschlüsselungssystem zur lasergesteuerten Datenübertragung TTC (*Trigger, Timing and Control*) [30] verarbeitet die vom TCS Controller ausgehenden Daten und leitet sie an ein optisches Netzwerk weiter.

Ein CATCH erhält alle Informationen, die das Trigger Control System betreffen, von einem

TCS Receiver, der über die Rückseiten der VME-Trägergehäuse an das Modul angeschlossen wird. Der TCS Receiver empfängt die Daten über das optische Netzwerk, erzeugt aus ihnen einen 38,88 MHz Takt, dekodiert die Daten und Triggerinformationen und sendet sie zusammen mit den Kontrollsignalen zu den Auslesemodulen. Abbildung 4.3 zeigt sämtliche Bestandteile des TCS und deren Anbindung an Detektoren und Front-End Module.

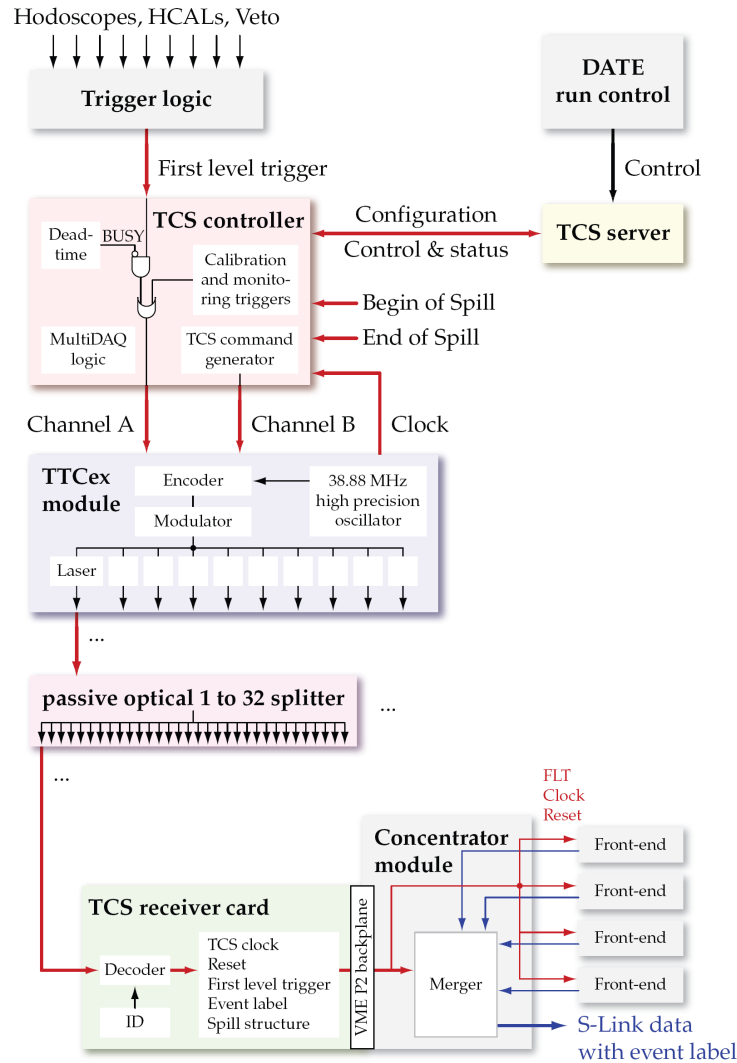


Abbildung 4.3: Die Architektur des Trigger Control Systems am COMPASS-Experiment [31].

Kapitel 5

Das GANDALF-Modul

Die vom COMPASS-Rückstoß-Detektor ausgehenden analogen Signale erfordern eine Verarbeitungstechnik, die über die Möglichkeiten der bisher am Experiment eingesetzten Elektronik hinausgeht. Zur Messung der Flugzeiten der Teilchen im Detektor muss der Zeitpunkt der Signale bestimmt werden. Aus dem Energieverlust eines Teilchens in den Szintillatoren in Verbindung mit deren geometrischer Anordnung kann ein Triggersignal erzeugt werden. Die Bestimmung des Energieverlustes muss deshalb unmittelbar am Detektor erfolgen. Das GANDALF-Modul (*General Advanced Numerical Device for Analytic and Logic Functions*) wurde entwickelt, um diese Berechnungen durchzuführen und die Weiterleitung des Triggersignales zu ermöglichen. Mit einer aufsteckbaren Digitalisierungseinheit wird GANDALF als Transientenrekorder die Signale des RPD digitalisieren und eine Identifizierung von Rückstoss-Protonen gewährleisten.

Die Verarbeitung der Daten erfolgt auf dem Modul in intelligenten Bausteinen, sogenannten FPGAs (*Field Programmable Gate Arrays*). Ein flexibler Einsatz des Moduls ist durch die Verwendung von bis zu zwei austauschbaren Aufsteckkarten gewährleistet. Damit kann GANDALF vielseitig eingesetzt werden, um Daten zu erfassen und zu verarbeiten. In naher Zukunft wird GANDALF als TDC- und Scaler-Modul sowie als Trigger-Matrix-Modul, zur Auslese von Front-End-Einheiten und als Transientenrekorder eingesetzt. In diesem Kapitel wird auf die einzelnen Schnittstellen und Funktionen der GANDALF-Baugruppe eingegangen (Abschnitt 5.1 und 5.2) und die Funktionsweise von GANDALF als Transientenrekorder beschrieben (Abschnitt 5.3).

Der schematische Aufbau von GANDALF geht aus Abbildung 5.1 hervor. GANDALF ist ein 6U VME/VXS-Modul¹ und über Stecker mit der Rückseite des VME-Trägergehäuses verbunden (im Bild rechts). Mit der Anbindung an die VME-Schnittstelle ist eine bidirektionale Datenübertragung zwischen Anwender und Baugruppe sichergestellt, über VXS kann eine schnelle Übertragung von Triggersignalen an ein Trigger-Switch erfolgen. Über den P2 Stecker besteht ein Anschluss an die S-Link-Schnittstelle zur Übertragung von Daten an die Speichereinheiten. Ein USB-Anschluss sowie eine Compact-Flash-Speicherkarte ermöglichen einen eigenständigen Einsatz des Moduls bei voller Funktionalität. Eine Gimli-Aufsteckkarte stellt die Verbindung an ein externes Triggersystem her und leitet ein Taktsignal auf die Baugruppe (in Abbildung 5.1 mit den Funktionen *Trigger* & *Clock* gekennzeichnet). Beim Einsatz von GANDALF als Transientenrekorder empfängt eine AMC-Aufsteckkarte bis zu 8 Kanä-

¹VME: VERSAmodule Eurocard - VXS: VMEbus Switched Serial

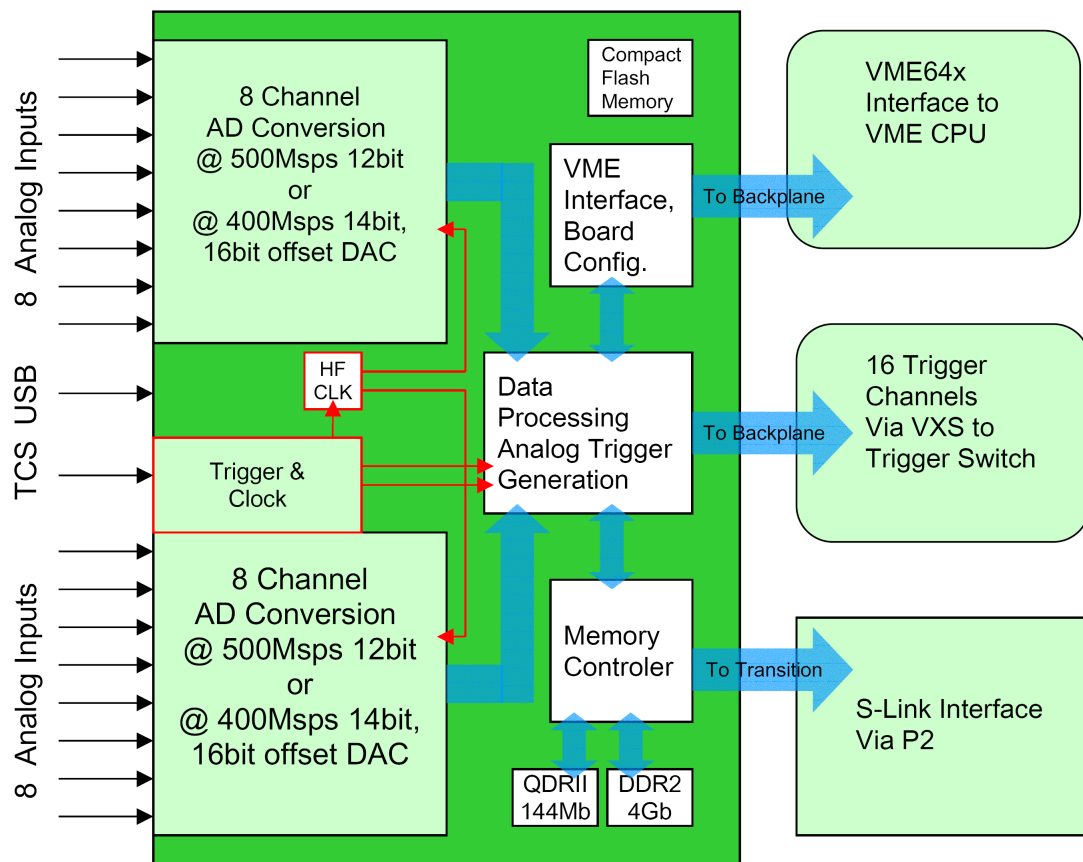


Abbildung 5.1: Der Aufbau des GANDALF-Moduls als Transientenrekorder. Die Detektorsignale werden auf AMC-Steckkarten digitalisiert und zur weiteren Verarbeitung auf die Baugruppe geleitet.

le. Die auf der Karte integrierten Analog-Digital-Konverter (ADCs) führen eine hochpräzise Digitalisierung der Signale durch. Im Anschluss werden die Daten über breite LVDS-Busse einem FPGA des GANDALF-Moduls zugeführt. Abbildung 5.2 zeigt GANDALF als Transientenrekorder mit zwei AMC-Aufsteckkarten und der Gimli-Karte, die zwischen den beiden Steckplätzen der Aufsteckkarten angebracht wird.

5.1 Die Schnittstellen der Baugruppe

5.1.1 VME64x

Um Daten zwischen dem Anwender und der GANDALF-Einheit zu übertragen, wird die VME-Schnittstelle eingesetzt. Sie ist nach dem Master/Slave-Prinzip aufgebaut. Einzelne Slave-Module werden von einem Master-Modul, das auch als SBC bezeichnet wird, angesteuert. Bis zu 20 Module befinden sich mit dem obligatorischen Steuerungsmodul (SBC) in einem Trägergehäuse und sind über zwei Steckverbinder P1 und P2 mit der Rückseite des Gehäuses,

der sogenannten *Backplane* verbunden (siehe Abbildung 5.2). Die darin enthaltenen VME-Signalleitungen setzen sich aus 31 Adressleitungen, 32 Datenleitungen und mehreren Steuerungsleitungen zusammen (die VME-Steckerbelegungen bei GANDALF sind in Anhang B aufgelistet). Die grundlegenden Lese- und Schreibroutinen zur Datenübertragung bei VME beinhalten ein Adresswort zur Ansteuerung bestimmter Module und deren Register sowie ein Datenwort. Eine gebündelte Datenübertragung erfolgt über einen Blocktransfer, indem bis zu 256 Datenworte nacheinander bei nur einem Adresswort versendet werden. Das GANDALF-Modul ist ohne den zentralen P0 Stecker als VME64x-Modul einsetzbar, sodass über die bereits angegebenen Transfermodi hinausgehend eine Übertragung von 64 bit breiten Adress- und Datenworten möglich ist. Dabei werden die Worte im Time-Multiplex Modus abwechselnd über sämtliche Daten- und Adressleitungen und eine Steuerungsleitung übertragen.

5.1.2 VXS

Ein Datentransfer bei sehr hohen Raten wird über VXS erreicht [32]. Der Standard umfasst ein verschaltetes Netz von seriellen Verbindungen zwischen einzelnen Modulen, die in diesem Zusammenhang als *Payload*-Module bezeichnet werden. Die Punkt-zu-Punkt Verbindungen zwischen einzelnen Baugruppen werden über bis zu zwei Switch-Module gesteuert, die sich in dem passenden Trägergehäuse befinden und auf denen sämtliche aktive Schaltelemente angebracht sind. Der in dem VME64x-Standard optional verwendbare Stecker P0 wird bei VXS durch einen RT2-Stecker ersetzt, der eine differenzielle Signalübertragung auf bis zu 16 Leitungspaaren pro Modul gestattet. Die Switch-Module sind über einen RT-Stecker mit der Backplane verbunden. Letztere sind in verschiedenen Architekturen erhältlich: Bei der *Single Star*-Backplane ist jedes Payload-Modul mit einem einzigen Switch-Modul verbunden, die *Dual Star*-Architektur ermöglicht die Anbindung aller Payload-Module an zwei miteinander verbundene Switch-Module (Abbildung 5.3). Eine *Daisy Chain*-Verbindung der

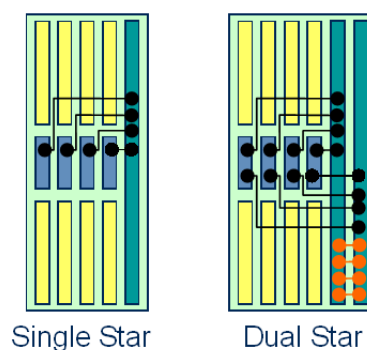


Abbildung 5.3: VXS ermöglicht die Verbindung von bis zu 18 Payload-Modulen (im Bild gelb) mit einem Switch-Modul (im Bild grün) über eine Single Star-Backplane und die Verbindung von bis zu 18 Payload-Modulen mit zwei Switch Modulen (im Bild grün) über die Dual Star-Backplane.

Payload-Boards zu ihren nächsten Nachbarn sowie eine Vernetzung von bis zu drei Modulen (*Mesh*) kann mit einer geeigneten Backplane ohne Switch erreicht werden. Zur Übertragung von Daten können mehrere Protokolle verwendet werden. *InfiniBand*, *Serial RapidIO*, *Gigabit*

Ethernet oder *PCI Express* garantieren Datenraten von bis zu 3,125 Gbit/s pro Leitungspaar. Bei GANDALF dient dieser Bus zur Übertragung eines Realtime-Triggers und wird dazu mit 500 MHz getaktet. So kann die GANDALF-Einheit als Element eines Triggersystems betrachtet werden, in dem bis zu zwei zentrale Trigger-Switch Module physikalische Daten von den GANDALF-Modulen empfangen und über einstellbare Koinzidenzschaltungen Triggerentscheidungen veranlassen.

5.1.3 Die S-Link Schnittstelle

Als S-Link wird eine Schnittstellenverbindung (*Simple Link Interface*) bezeichnet, die verwendet werden kann, um beliebige Stufen der Front-End Elektronik an die nächste Stufe der Datenauslese zu koppeln. Dieser Standard wurde für das ATLAS-Experiment am CERN entwickelt und beinhaltet neben den Datentransfers mit Steuersignalen auch die Möglichkeit zur Fehlerkorrektur. Zudem ist bei der Duplex-Version ein Rückgabekanal zur Kontrolle des Datenflusses enthalten sowie eine Selbsttestfunktion. Die Spezifikation [33] sieht vor, dass ein Front-end Motherboard (FEMB) mit der *Link Source Card* (LSC) verbunden ist und von dort der Vorwärtskanal über die physikalische Schicht zur *Link Destination Card* (LDC) geleitet wird (siehe Abbildung 5.4). Hier werden die Daten über die dazugehörige Schnittstelle an das Auslese-Motherboard (ROMB) gelenkt, an dem sie die S-Link Schnittstelle verlassen und weiter verarbeitet werden können. Eine 32 bit-breite Datenleitung mit einer maximalen Taktrate von 40 MHz ergibt eine maximale Datendurchsatzrate von 160 MByte/s. Das Protokoll verarbeitet die Daten sowohl auf Sender- als auch auf Empfängerseite nach dem Prinzip eines FIFOs. In der Duplex Version ist ein Rückgabekanal vorhanden, sodass Informationen zur

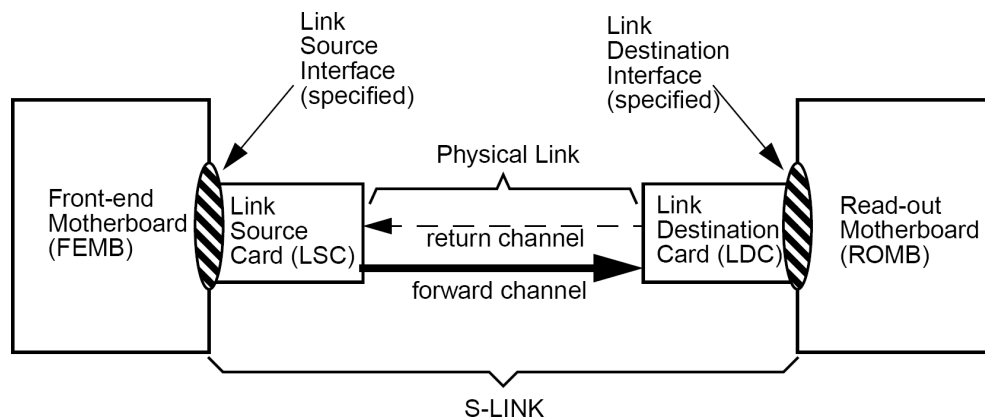


Abbildung 5.4: Datenauslese über die S-Link Schnittstelle [33] (forward channel). In der Duplex-Version ist ein Kanal zur Kommunikation zwischen LDC und LSC vorgesehen (return channel).

Datenquelle geleitet werden können. Sobald das Motherboard dieser Seite zum Lesen bereit ist, können über den ROMB Kommandos zur Kontrolle des Datenstroms gesendet werden. Um einen Datenverlust zu verhindern, ist das Senden der Daten vom Front-End Motherboard zum Zielort solange unterbunden bis das Auslese-Motherboard erreichbar ist.

Anwendung

Die Link Source Card (LSC) wird auf eine Adapterkarte (FEMB) gesteckt, die an der hinteren Seite des VME-Trägergehäuses mit der Backplane verbunden ist. Die Backplane stellt eine Verbindung mit dem P2 Stecker des entsprechenden vorderen Einschubfaches her. Beide FPGAs eines GANDALF-Moduls, das sich in diesem Einschubfach befindet, können auf diesem Weg Daten an die LSC weiterleiten. Diese Daten werden von der LSC empfangen und über optische Fasern zu den Link Destination Cards (LDC) gesendet, die in Readout Buffer PCs eingebaut sind. Dort werden die Daten auf Spillbufferkarten zwischengespeichert. Am COMPASS-Experiment wird S-Link im Duplex Modus mit einem Kanal in Vorwärtsrichtung und einem Kanal mit Richtung zum Front-End Modul betrieben. Zur Datenübertragung werden doppelte Faserleitungen eingesetzt.

Um unterschiedliche Datenraten auszugleichen und die gegebenen Bandbreiten der einzelnen S-Link Verbindungen vollständig zu nutzen, können Daten von bis zu vier Front-End Modulen mit einem S-Link Multiplexer zu einem Datenstrom zusammengeführt werden. Der Multiplexer wird mit einem GANDALF-Modul verbunden und kann mit Multiplex-Adaptern und entsprechenden Kabeln die Daten von bis zu drei weiteren Modulen auslesen.

5.1.4 Die USB-Schnittstelle

Sämtliche auf einer GANDALF-Baugruppe enthaltenen Messdaten und Statusinformationen können über die USB-Schnittstelle ausgelesen werden. Die Schnittstelle kann zudem zur Konfiguration der FPGAs verwendet werden. Somit steht mit dem USB-Anschluss die volle Funktionalität von GANDALF zur Verfügung. Der EZ-USB SX2 USB-Kontroller der Firma Cypress [34] erlaubt eine Datenübertragung auf Basis von USB 2.0 in High-Speed (480Mbit/s) und Full-Speed (12 Mbit/s). Die Anbindung an externe Geräte erfolgt über einen Mini-USB Anschluss.

5.1.5 Die JTAG-Schnittstellen

Über eine Verbindung zwischen einem JTAG-Steuermodul und der JTAG-Schnittstelle einzelner Bauelemente können im Rahmen des Standards IEEE 1149.1 umfangreiche Funktionstests durchgeführt werden. Dabei befinden sich ein oder mehrere Bausteine in einer JTAG-Kette, in der Daten von einem Baustein an *TDI* empfangen und über *TDO* ausgegeben werden, die Takt- und Steuersignale *TCK* und *TMS* sind parallel mit den Modulen verschaltet (siehe Abbildung 5.5). Der *Test Access Port* (TAP) eines JTAG-fähigen Bausteins ermöglicht einen seriellen *Boundary Scan*: Sämtliche Aus- und Eingänge eines Bausteins können testweise auf bestimmte Spannungspegel gesetzt bzw. abgefragt werden. Üblicherweise sind neben dem *Boundary-Scan* Register die TAP-Register *Instruction*, *Bypass* und *Identification* vorhanden. So können weitere Anweisungen an Bausteine gesendet oder einzelne Bauelemente in einer JTAG-Kette deaktiviert bzw. identifiziert werden. Über die JTAG-Schnittstelle und dem Standard IEEE 1532 ist es zudem möglich, eine serielle Konfiguration von Hardwarebauelementen vorzunehmen (*In-System Configuration*, ISC). In Tabelle 5.1 sind die JTAG-Schnittstellen von GANDALF aufgelistet, die zum Boundary Scan Test oder zur Konfiguration der Hardwareelemente verwendet werden können.

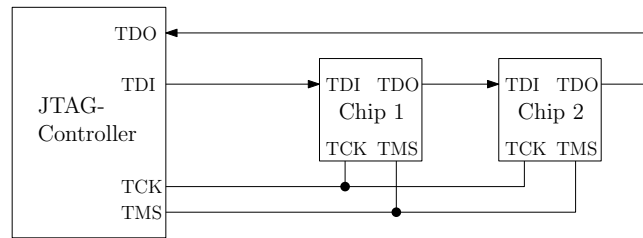


Abbildung 5.5: Über einen JTAG-Controller werden die Bausteine einer JTAG-Kette angesteuert. Die Daten werden über die serielle Leitung TDI an den ersten Baustein gegeben, der sie an den nächsten Baustein über TDO weiterleitet. Das Taktsignal TCK sowie das Steuersignal TMS sind parallel mit den Bausteinen verschaltet.

Tabelle 5.1: Verfügbare JTAG-Ketten bei der GANDALF-Einheit.

JTAG-Schnittstelle	Inhalt der JTAG-Kette	Test/Konfiguration
CFGJTAG	DSP-FPGA, QDR-FPGA	Test, Konfiguration
CPLDJTAG	CPLD	Test, Konfiguration
TSTJTAG	SystemACE	Test
QDRJTAG	2 QDR-II SRAMs	Test
AMCJTAG	siehe eingesetzte AMC-Karte	

5.1.6 Die Compact-Flash Karte

Für einen Einsatz der Baugruppe ohne die Anbindung an VME und USB ist die SystemACE Umgebung (ACE - *Advanced Configuration Environment*) vorgesehen. Die Compact-Flash Karte ist Bestandteil dieser Umgebung und stellt die zur Konfiguration erforderliche Firmware bereit. Dabei kann aus bis zu acht möglichen Versionen ausgewählt werden. Zudem bietet die Karte Speicherplatz für Einstellungen, Statusinformationen und Messdaten. Die Compact-Flash Schnittstelle unterstützt einen Großteil der gängigen Compact-Flash (Typ I oder II) Karten bis zu 8 GB sowie Hitachi Microdrives bis zu 6 GB.

5.1.7 Die I²C-Schnittstellen

Um Register eines Bausteines zu beschreiben oder auszulesen wird häufig die I²C-Schnittstelle eingesetzt. Dieser Bus befindet sich zwischen einzelnen Bausteinen und wird von einem Master gesteuert. Das als Master eingesetzte Bauelement hat Zugang zu allen Bausteinen, die sich in einer I²C-Kette befinden. Bei GANDALF sind vier I²C-Ketten vorhanden, die jeweils von einem Bauelement gesteuert werden und darüberhinaus über Steckverbinder an externe Steuereinheiten angeschlossen werden können. In Tabelle 5.2 sind die einzelnen I²C-Ketten zusammengefasst.

Tabelle 5.2: Die bei GANDALF eingesetzten I²C-Schnittstellen.

I ² C-Kette	Master-Bauelement	Slave-Bausteine der I ² C-Kette
SI_M	DSP-FPGA	<i>Si5326</i> , in der Kette integrierte Bausteine auf beiden AMC-Karten
GP_A	DSP-FPGA	in der Kette integrierte Bausteine auf der AMC-Karte A
GP_B	DSP-FPGA	in der Kette integrierte Bausteine auf der AMC-Karte B
UCD	CPLD	<i>CDCE949</i> , <i>UCD9081</i>

5.2 Die Funktionen der Baugruppe

5.2.1 Die Überwachung der Spannungsversorgung

Der bei GANDALF eingesetzte Spannungsprüfer *UCD9081* von Texas Instruments dient der gesteuerten Inbetriebnahme einzelner Bauteile und zur Überwachung von Spannungswerten verschiedener Versorgungsnetze [35]. Die mit dem *UCD9081* verbundenen Netze (*rails*) werden nach einem definierten Ablauf aktiviert. Im Betrieb werden die Spannungswerte in regelmäßigen Abständen von einem 10 bit ADC digitalisiert und im Anschluss in internen Registern gespeichert. Bei einer externen Referenzspannung von 3,3 V beträgt die Auflösung damit 3,22 mV. Das Setzen einer internen Referenzspannung von 2,5 V erhöht die Auflösung entsprechend auf 2,44 mV. Boardspannungen, deren Werte die eingesetzte Referenzspannung übersteigen, können ebenfalls überwacht werden, indem ein Spannungsteiler eingesetzt wird (siehe dazu die *Application Information* im Datenblatt des *UCD9081* [35]).

Die Bedingungen für die Aktivierung eines Netzes sind an zeitliche Ereignisse bzw. an die Spannungswerte anderer Netze geknüpft. Außerdem ist eine Abschaltung einzelner Netze in Abhängigkeit von Spannungswerten möglich. Die auf einem GANDALF-Modul überwachten Spannungsleitungen sind in Tabelle 5.3 zusammengefasst. Die Register des *UCD9081* werden über eine I²C-Schnittstelle ausgelesen und beschrieben.

5.2.2 Die Gimli-Karte

Eine Verbindung zwischen dem GANDALF-Modul und einem externen Trigger-System kann durch die Gimli-Karte [36] hergestellt werden. Um einen weitreichenden Einsatzbereich von GANDALF sicherzustellen, ist die Gimli-Karte in drei Versionen verfügbar: Version 1 ist für den Einsatz im Bereich des COMPASS-TCS bestimmt und besitzt einen optischen Empfänger, der mit der Glasfaser des Trigger Control Systems verbunden wird. Auf der Karte werden die optischen Signale in elektrische Signale umgewandelt und der Referenztakt des TCS wird aus den ankommenden Daten erzeugt. Daraufhin erreichen die experimentweiten Taktsignale und Triggersignale den DSP-FPGA auf der GANDALF-Einheit, der aus den Triggersignalen die Trigger und Ereignisnummern extrahiert. Werden die TCS-Signale über Koaxialkabel übertragen, kann Version 1 auch mit einem LEMO-Stecker bestückt werden. Die Versionen 2 und 3 ermöglichen einen eigenständigen Einsatz von GANDALF ohne TCS: Bei beiden Versionen

Tabelle 5.3: Die GANDALF-Versorgungsspannungen. Sie werden durch den *UCD9081* gesteuert und überwacht.

Spannung [V]	Eingang am <i>UCD9081</i>
1,0 VCC	Rail 1
1,8 VCC	Rail 2
2,5 VCC	Rail 3
5,0 VCA	Rail 4
3,3 VCA	Rail 5
3,3 VCC	Rail 6
2,5 VCCAUX	Rail 7
3,3 VCPLD -12,0 VCC	Rail 8

werden Triggersignale über LEMO-Stecker empfangen. In Version 2 wird ein Referenztakt über einen weiteren LEMO-Eingang empfangen. Version 3 enthält einen 20 MHz Quarzoszillator, dessen Taktsignal auf die GANDALF-Einheit geleitet wird und dort als Referenztakt verwendet werden kann.

5.2.3 Die Taktverteilung

Zur Bereitstellung verschiedener Taktsignale, die für den Betrieb der Bauteile auf dem Modul notwendig sind, wird der von der Gimli-Karte ausgehende Takt sowohl verteilt als auch den erforderlichen Frequenzen angepasst. Zudem dienen zwei Quarzoszillatoren auf der Baugruppe zur Bereitstellung fester Taktfrequenzen. Abbildung 5.6 zeigt eine schematische Darstellung der Taktverteilung bei GANDALF. Der Baustein *MC100EP210S* dient als dualer 1-zu-5 Treiber und gibt den 155 MHz Takt weiter an den DSP-FPGA, an die AMC-Karten und den Taktvervielfacher *Si5326*. Letzterer erzeugt daraus ein 500 MHz Taktsignal und ein weiteres Taktsignal, das von der Aurora-Schnittstelle verwendet wird. Das 500 MHz-Taktsignal wird an den zweiten Eingang des 1-zu-5 Treibers gegeben, um beide Virtex5-FPGAs mit der maximal möglichen Taktrate auszustatten. Ein 40 MHz Oszillator, dessen Taktsignal an den Controller-CPLD geleitet wird, garantiert eine grundlegende Ansteuerung des GANDALF-Moduls über den CPLD. Weitere einstellbare Frequenzen werden von dem programmierbaren Takt-Synthesizer *CDCE949* erzeugt, der den Referenztakt von einem 27 MHz Oszillator erhält.

5.2.4 Die Verarbeitung der Daten

GANDALF ist mit zwei FPGAs ausgestattet, die über eine bidirektionale *Aurora*-Schnittstelle miteinander verbunden sind. Über *Aurora* werden Daten auf 16 Leitungen übertragen. Die Datenrate beträgt bis zu 3,125 Gbit/s pro Leitung. Der als DSP-FPGA bezeichnete *XC5V SX95T* enthält neben 640 DSP-Schichten (*slices*) zur Verarbeitung von Daten 60000 Flip-Flops sowie 8 Mbit Block RAM und wird mit einer Taktrate von bis zu 500 MHz betrieben. Der hier als QDR-FPGA bezeichnete *XC5VLX30T* mit 32 DSP-Slices, 20000 Flip-Flops und 1,2 Mbit

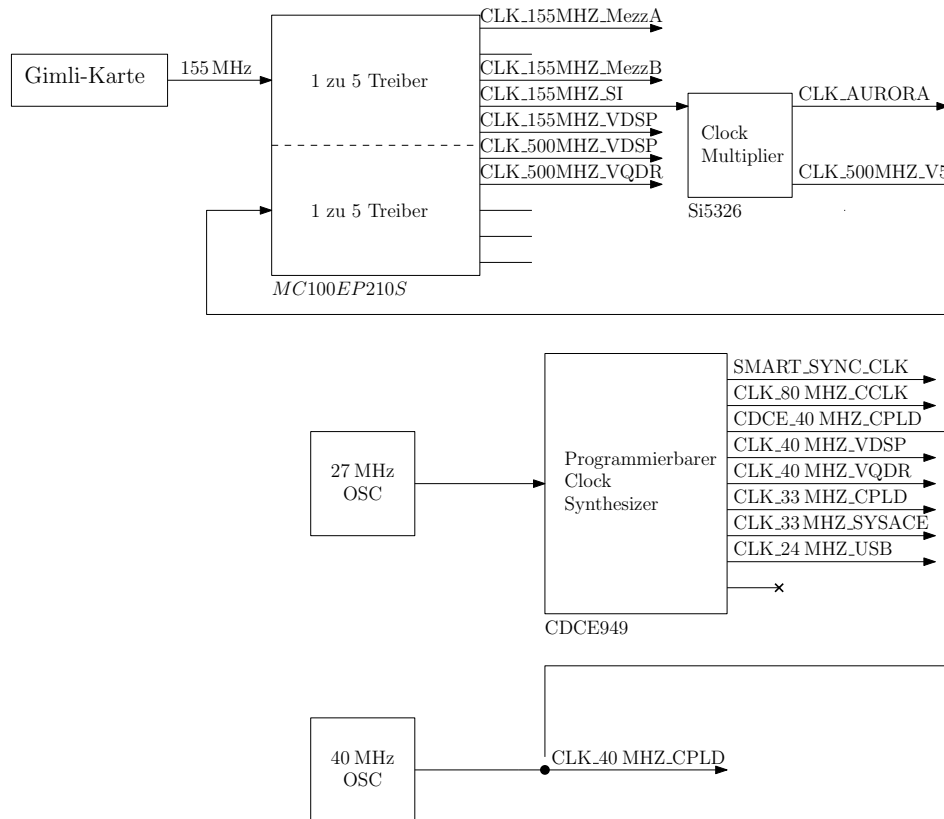


Abbildung 5.6: Die Erzeugung und Verteilung der Takte auf der GANDALF-Baugruppe.

Block RAM ermöglicht eine Datenverarbeitung ebenfalls bei 500 MHz. Diesem FPGA stehen jeweils zwei QDRII+ und DDR2 Speicherbausteine zur Verfügung, sodass ein L2 Cache von 144 Mbit und ein L3 Cache von 4 Gbit entsteht.

Für die Konfiguration der FPGAs, die nach jedem Trennen von der Spannungsversorgung oder nach Aktualisierung der entsprechenden Firmware neu vorgenommen werden muss, stehen mehrere Optionen zur Verfügung. Generell liegt die Firmware der beiden FPGAs in einer Datei vor. Diese kann standardmäßig über die VME-Schnittstelle geladen werden. Darüber hinaus können die FPGAs über USB und zu Testzwecken über einen JTAG-Stecker geladen werden.

5.3 Der GANDALF-Transientenrekorder

Die analogen Signale des RPD weisen sehr kurze Anstiegsflanken zwischen 3 und 5 ns im gesamten dynamischen Bereich des Photomultipliers auf. Um den Zeitpunkt und die Amplitude dieser Signale bestimmen zu können, muss die Digitalisierung der Signale mit hoher Auflösung bei maximalen Taktraten erfolgen. Zudem ist eine Anpassung der Signale eines Kanals an den Eingangspegel des verwendeten ADC notwendig. Eine AMC-Karte ist entwickelt worden, die mehrere ADCs und die dazugehörigen Eingangsschaltungen enthält. Maximal 8 Kanäle können an eine Karte angeschlossen werden. Darüberhinaus können zwei ADCs im Interleaved-Modus betrieben werden: Sie erhalten die Signale eines Kanals und führen die

Digitalisierung mit einem phasenverschobenem Taktsignal aus. Die AMC-Karte ist als Aufsteckkarte für GANDALF anwendbar und erlaubt so die Weiterleitung der Datenworte und Steuersignale an die GANDALF-Baugruppe. So kann eine Weiterverarbeitung der Daten unmittelbar nach der Digitalisierung erfolgen. Damit ist ein Transientenrekorder verfügbar, der höchsten Anforderungen an die Güte der Digitalisierung erfüllt und die Verarbeitung der Daten mit anwendungsspezifischen Firmware-Versionen durchführen kann.

5.3.1 Die AMC-Aufsteckkarte

Eine AMC-Karte enthält 8 ADCs und die jeweiligen Eingangsschaltungen. Außerdem sind auf der Karte Bauteile zur Erzeugung der zur Digitalisierung verwendeten Taktsignale und ein I²C-Bus vorhanden. Dieser Bus wird zur Ansteuerung eines Digital-Analog-Konverters (DAC), eines Temperatursensors und eines EEPROMs verwendet. Der DAC dient zur Einstellung eines Offsets. Eine AMC-Karte kann mit zwei verschiedenen ADCs der Firma *Texas Instruments* bestückt werden, die zueinander pin-kompatibel sind. Auf diese Weise kann eine Digitalisierung bei maximaler Auflösung bzw. bei maximaler Abtastrate erfolgen: Der *ADS5463* digitalisiert mit einer Auflösung von 12 bit bei einer maximalen Abtastrate von 500 MHz, der *ADS5474* weist eine Auflösung von 14 bit auf und arbeitet mit einer maximalen Abtastrate von 400 MHz. Im Interleaved-Modus können demzufolge Signale mit einer effektiven Abtastrate von bis zu 1 GHz digitalisiert werden. Die digitalen Worte werden über parallele Verbindungen auf das GANDALF-Modul zu einem FPGA geleitet, der die Daten weiterverarbeitet und Triggersignale generieren kann. Im folgenden wird die Eingangsschaltung beschrieben, bevor auf die Eigenschaften der eingesetzten ADCs eingegangen wird. Dies soll am Beispiel des *ADS5463* erfolgen.

Die Analog-Eingangsschaltung

Bei einem Spannungsbereich von -4 bis 0 V der PMT-Signale am Analogeingang führt die Quantisierung zu einer theoretischen Unschärfe von $0,977$ mV. Eine analoge Eingangsschaltung, die auf jeden SMC-Eingang auf der AMC-Karte folgt, wandelt ein Signal des Photomultipliers in ein differenzielles Signal mit einem Mittelwert der Eingangsspannungen (engl.: *Common-Mode Voltage*) von $2,4$ V um und legt den Spannungsbereich auf $2,4 \pm 0,55$ V fest [36]. Die differenzielle Spannung $U_{\text{ADC}}^{\text{diff}} = U_{\text{ADC}}^+ - U_{\text{ADC}}^-$ befindet sich im Bereich zwischen $-1,1$ V und $+1,1$ V und es folgt damit ein differenzieller Eingangsbereich des ADCs von $2,2$ V_{pp} wie es die Spezifikation des *ADS5463* [37] vorsieht. Die Verwendung des gesamten Digitalisierungsbereiches des ADCs wird durch eine Verschiebung der Signale erreicht: Um den Bereich von 0 bis -4 V mit DC auf 0 V in den ADC abzubilden, wird 0 V auf $+1,1$ V und -4 V auf $-1,1$ V projiziert.

Der ADS5463

In einem N bit ADC wird der analoge Wertebereich in $2^N - 1$ Teilbereiche gleicher Größe unterteilt, die jeweils durch ein eindeutiges digitales Wort der Länge N dargestellt werden. Je höher die Auflösung eines ADC ist, desto präziser ist die Identifizierung eines analogen Wertes mit einem digitalen Wort. Die Konvertierung einer natürlichen Zahl in das Binärsystem erfolgt gemäß

$$\text{Zahl}_{10} = 2^{N-1}a_{N-1} + 2^{N-2}a_{N-2} + \dots + 2^1a_1 + 2^0a_0 \quad (5.1)$$

Die Anordnung der bits in der Reihenfolge $a_{N-1} \dots a_0$ bildet das digitale Wort. Dabei wird das links stehende bit entsprechend seiner Gewichtung als MSB (*Most Significant Bit*) bezeichnet, das bit mit dem kleinsten Vorfaktor (rechts) als LSB (*Least Significant Bit*). Um sowohl positive wie auch negative analoge Größen mit der Binärkodierung darstellen zu können, wird am ADS5463 die *Offset-Binary*-Kodierung eingesetzt, bei der der Nulldurchgang eines Signals durch logisch Eins am MSB und logisch Null an allen anderen bits dargestellt wird. In Abbildung 5.7 ist die Transferfunktion eines 3 bit-ADCs dargestellt. Sie gibt die Offset-Binary Datenworte als Funktion des analogen Eingangs an. Die Auflösung des ADS5463 beträgt 12 bit,

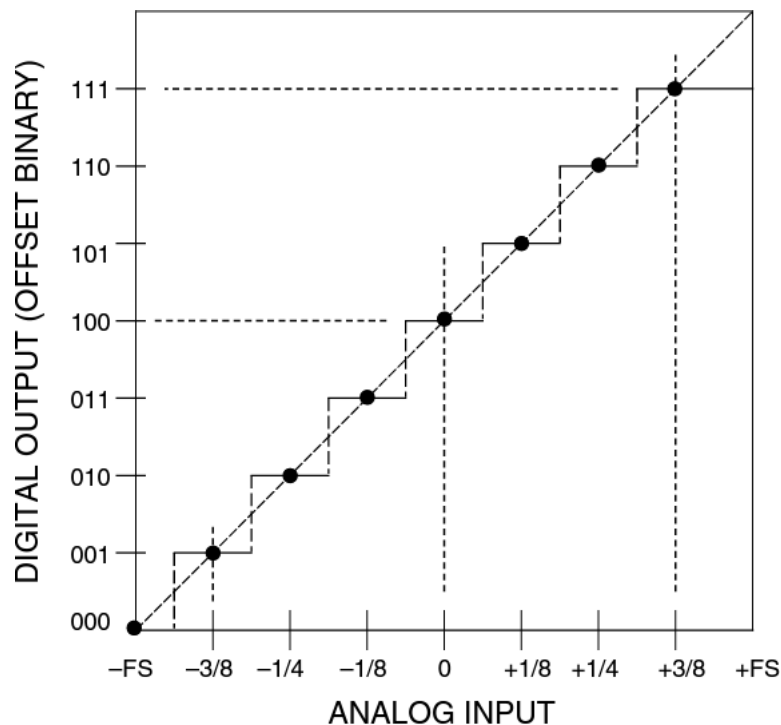


Abbildung 5.7: Die Transferfunktion eines idealen 3 bit-ADCs in Offset-Binary-Kodierung mit einem maximalen Spannungswert FS (engl.: *Full Scale*) [38].

sodass $2^{12} - 1 = 4095$ Teilbereiche zur Darstellung des analogen Signalwerts zur Verfügung stehen.

Der Pipeline-ADC. Eine hohe Auflösung bei schnellen Digitalisierungszeiten kann mit dem Pipeline-Modus realisiert werden: In einer Stufe wird das Eingangssignal auf eine *track-and-hold* Leitung (TH) und auf eine ADC-DAC Kombination gegeben und die Differenz zwischen Original- und dem nach der Digitalisierung zurückkonvertierten Wert wird ermittelt. Dieser Wert wird nach einem halben Taktzyklus zur Messung der Spannung mit feiner Auflösung an die nächste Stufe weitergeleitet und verstärkt. Abbildung 5.8 zeigt die Pipeline-Architektur

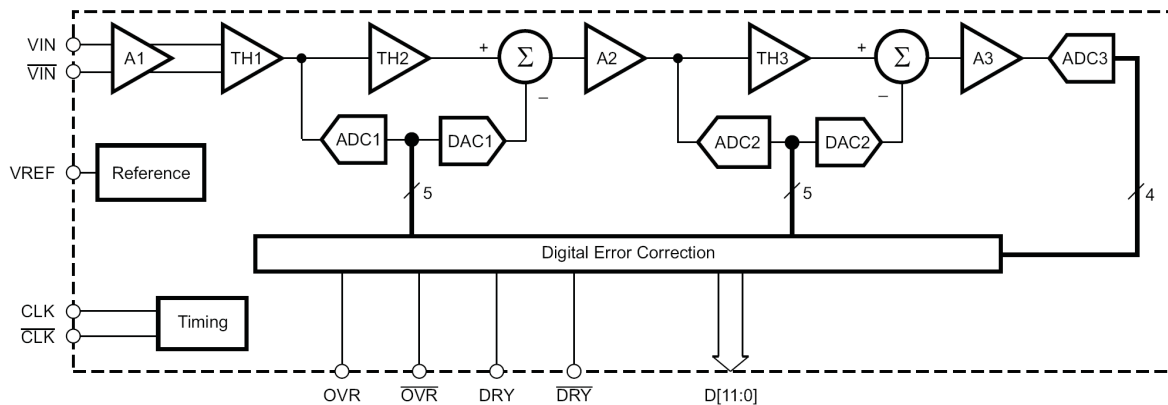


Abbildung 5.8: Die Pipeline-Architektur des *ADS5463* mit Fehlerkorrektur [37].

des *ADS5463*.

Damit die aus der Digitalisierung erhaltenen Werte die logischen Elemente zur Fehlerkorrektur zur selben Zeit erreichen, sind die Ausgänge der Stufen mit der passenden Anzahl von Schieberegistern versehen. Nach 3,5 Taktperioden hat ein Signal sämtliche Digitalisierungsstufen des ADC durchlaufen und ist als 12 bit-Wort am Ausgang abrufbar.

Die Quantisierung. Eine ideale Quantisierung wie sie durch die Transferfunktion 5.7 ausgedrückt ist, ist bei einem realen Analog-Digital Wandler nicht gegeben. Mehrere mögliche Fehlerquellen können zu einem Versatz dieser Geraden oder darüberhinaus zur Entstehung einer nicht-linearen oder nicht-monoton steigenden Kurve führen. Diese Abweichungen werden als Offset Fehler, Gain Fehler (Gain - engl.: *Verstärkung*) oder als Fehler in der Linearität bezeichnet. Der Offset Fehler führt zu einem Versatz der Transferfunktion auf dem Achsenabschnitt der analogen Größe, wohingegen der Verstärkungsfehler die Abweichung der Steigung von ihrem idealen Wert kennzeichnet (siehe Abbildung 5.9). Der *ADS5463* weist einen Versatz der Transferfunktion durch einen Verstärkungsfehler von max. $\pm 5\%$ des vollen Eingangsspannungsbereiches bei einem Fehler im Offset von max. $\pm 11\text{ mV}$ auf. Weiterhin kann die Transferfunktion von einer idealen Geraden abweichen. Die integrale Nichtlinearität (INL) als Ausdruck dieses Verhaltens ist definiert als die maximale Abweichung der Funktion von einer Geraden, die durch einen Fit nach der Methode der kleinsten Quadrate an die für den spezifischen ADC charakteristische Transferfunktion ermittelt wird. So ist die zu jedem Eingangswert zugehörige INL die Differenz zwischen realer und idealer Transferfunktion in Einheiten eines LSB [37]. Sie beträgt beim *ADS5463* max. $\pm 2,5\text{ LSB}$. Ein Bezug zum realen Konverter verschafft die differenzielle Nichtlinearität (DNL), die die Abweichung eines Übergangs von einem quantisierten Bereich in den nächsten bezüglich des idealen Verhaltens darstellt. Auch hier wird als Einheit das LSB gewählt, ein fehlendes Wort (*missing code*) kann bei einer $\text{DNL} \geq 1$ auftreten. Mit dem im Datenblatt des *ADS5463* angegebenen maximalen Wert von $\pm 0,95\text{ LSB}$ ist eine verlustfreie Funktion dieses ADC gewährleistet.

Durch die Quantisierung werden verschiedene analoge Werte einem bestimmten Kodierungswort zugeordnet, sofern die Differenz dieser Signalwerte den von einem LSB repräsentierten Wert q nicht übersteigt. Dieser Quantisierungsfehler halbiert sich mit jedem zusätzlichen bit eines ADCs. Das daraus resultierende theoretische Quantisierungsrauschen Q als RMS des

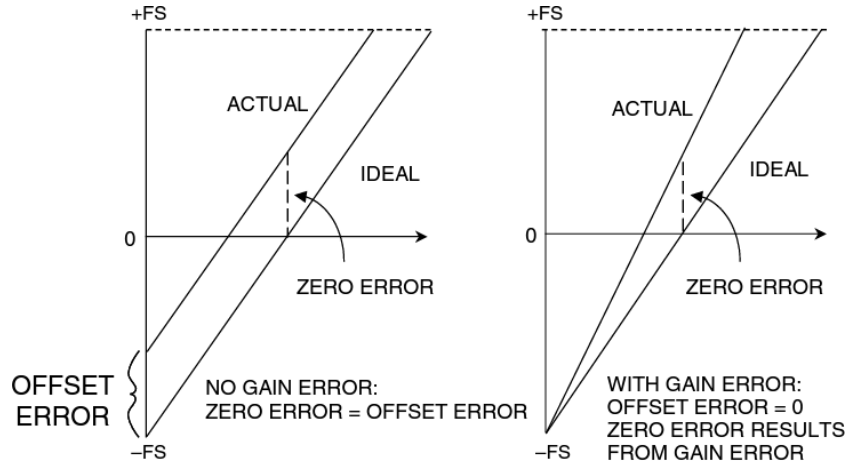


Abbildung 5.9: Abweichungen der linearen Transferfunktion von einem idealen Verhalten werden durch einen Offset- und Verstärkungsfehler (links bzw. rechts im Bild) ausgedrückt [38].

Quantisierungsfehler kann in Abhängigkeit des Wertes eines LSBs berechnet werden [38] und lautet

$$Q = \frac{q}{\sqrt{12}}. \quad (5.2)$$

Mit der Annahme, dass am Eingang des ADC ein Sinussignal über den vollen Bereich der Analogspannung anliegt,

$$v(t) = \frac{q2^N}{2} \sin(2\pi ft) \quad (5.3)$$

folgt als RMS Wert dieses Signals über den gesamten Wertebereich

$$P = \frac{q2^N}{2\sqrt{2}}. \quad (5.4)$$

Das Signal-Rausch-Verhältnis (SNR) eines idealen N bit AD-Wandlers ist damit

$$\text{SNR} = 20 \log_{10} \frac{P}{Q} = 20 \log_{10} \left[\frac{q2^N/2\sqrt{2}}{q/\sqrt{12}} \right] = (6,02 N + 1,76) \text{ dB} \quad (5.5)$$

mit $0 \leq f \leq f_s$. Das Nyquist Kriterium besagt, dass die maximale Frequenz eines Eingangssignals bei Konvertierung nur dann erhalten bleibt, falls die Abtastrate f_s den doppelten Wert dieser Frequenz übersteigt. Laut der Spezifikation liegt das Signal-Rausch-Verhältnis des ADS5463 bei Eingangsfrequenzen von 10 und 300 MHz bei 65,4 bzw. 65,0 dbFS.

Neben dem Quantisierungsrauschen hat das elektronische Rauschen ebenfalls einen Einfluss auf die Güte des digitalisierten Signals. Das Kodierungswort eines analogen Wertes fluktuiert bei einem realen ADC generell um den nominalen Wert eines idealen ADC und führt zu einem von der Quantisierung unabhängigen Rauschen. Ein Maß für diese Schwankung ist die effektive Anzahl der bits (ENOB, engl.: *effective number of bits*): Sie gibt die Auflösung eines idealen

Wandlers an, mit welcher theoretisch dasselbe Ergebnis zu erzielen wäre. Eine Bestimmung der ENOBs kann über das Signal-Rausch-Verhältnis mit

$$\text{ENOB} = \frac{\text{SNR} - 1,76 \text{ dB}}{6,02 \text{ dB}} \quad (5.6)$$

vorgenommen werden [39]. Für Eingangsfrequenzen $f_{\text{in}} \leq 300 \text{ MHz}$ beträgt die im Datenblatt des *ADS5463* angegebene effektive Anzahl der bits $\text{ENOB}_{\text{spec}} = 10,4 \text{ bit}$.

5.3.2 Die Güte der Digitalisierung

Um die Genauigkeit der Digitalisierung zu bestimmen, wurden mit dem Signalgenerator *Tektronix AFG3251* erzeugte Sinus-Signale digitalisiert. Die Analyse der Daten erfolgte über eine *Fast-Fourier-Transformation* (FFT) in der Software *VisualAnalog* [40]. Um Rauschen und Fremdfrequenzen in den erzeugten Signalen weitgehend zu unterdrücken, wurden die analogen Signale über koaxiale Bandpassfilter SIF und SBP der Firma *Mini-Circuits* auf den Eingang der AMC-Karte geleitet. Abbildung 5.10 zeigt das Resultat der Digitalisierung eines 23 MHz Sinus-Signals mit einer Abtastrate von 504 MHz. Das Spektrum umfasst 8000 Datenworte, die von der AMC-Karte kommend über SystemACE auf die Compact-Flash Karte gespeichert wurden. Mit weiteren Messungen über einen Frequenzbereich zwischen 23 MHz und

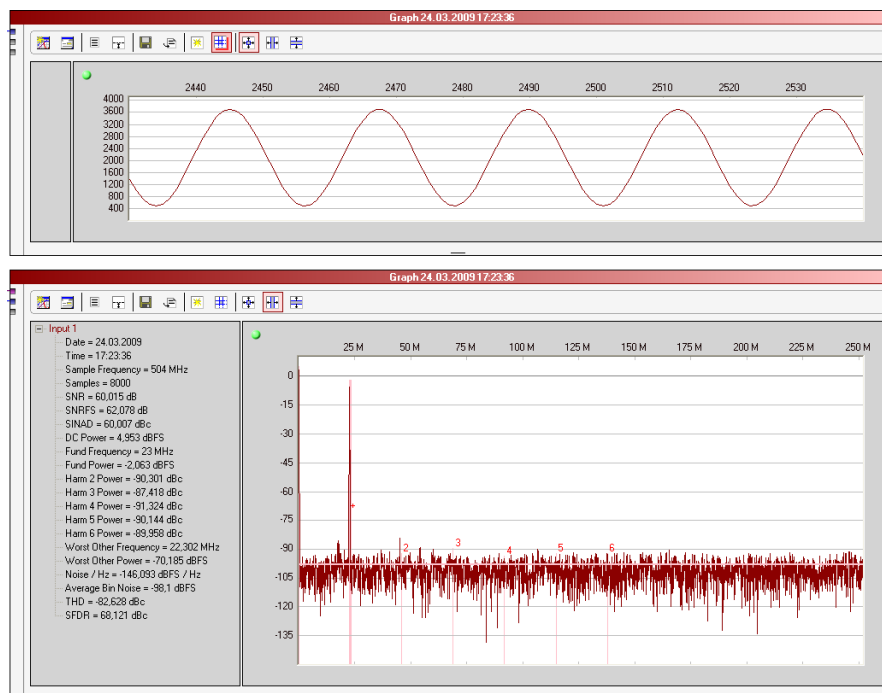


Abbildung 5.10: Messung zur Bestimmung der Leistung der Digitalisierungseinheit. Die analogen Signale werden über einen Bandpassfilter auf den Eingang gegeben und können in digitalisierter Form wahlweise über die S-Link- oder VME-Schnittstelle ausgelesen werden. In dieser Messung wurden die erhaltenen Daten in den Controller-CPLD eingelesen und auf der Compact-Flash-Karte gespeichert.

240 MHz wurde gezeigt, dass das Signal-Rausch-Verhältnis über dem vollen Bereich generell den Wert $\text{SNR}_{\text{FS}} = 62 \text{ dB}$ übersteigt (Abbildung 5.11). Somit beträgt bei einer Abtastrate von 500 MHz die mit dem GANDALF-Transientenrekorder erreichte effektive Auflösung $\text{ENOB}_{\text{meas}} = 10,1 \text{ bit}$.

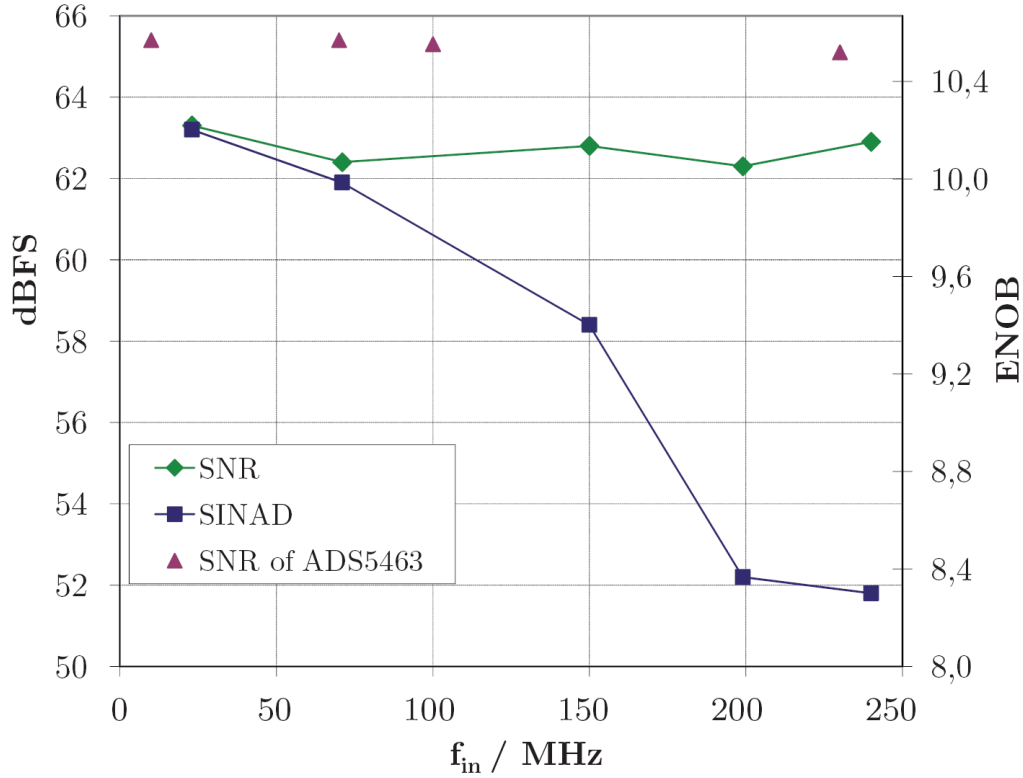


Abbildung 5.11: Gemessene Werte des SNR sowie von SINAD (*Signal to Noise and Distortion*) über den vollen Bereich des ADCs (FS, *Full-Scale*) [36]. Zum Vergleich sind die aus der Spezifikation des ADS5463 [37] entnommenen Werte ebenfalls eingezeichnet.

5.3.3 Die Verarbeitung der digitalisierten Daten

Am COMPASS-RPD soll der GANDALF-Transientenrekorder die Flugzeit eines Protons bestimmen und den Energieverlust der Teilchen im Szintillator ermitteln, aus dem schließlich ein Triggersignal erzeugt wird. Dabei ist es von großer Relevanz, möglichst viele Werte der Anstiegsflanke des Signals aufzunehmen, um aus ihnen einen Zeitpunkt zu berechnen. Aufgrund der kurzen Anstiegszeit kann die Verwendung des Interleaved-Modus notwendig werden. Sowohl mit festen als auch variablen Anstiegszeiten ist es möglich, das Signal analytisch durch eine Moyal-Kurve zu beschreiben (Abbildung 5.12).

Die Studien zur Bestimmung des Zeitpunktes eines Signals in [25] haben folgendes ergeben: Unter Berücksichtigung von Rauschen und Jitter kann ein geeigneter Zeitpunkt mit der Methode der *analog Constant Fraction Discrimination* (aCFD) bestimmt werden, sodass bei Amplituden von $> 80 \text{ mV}$ die Zeitauflösung mindestens 50 ps beträgt. Zudem ist es mit der

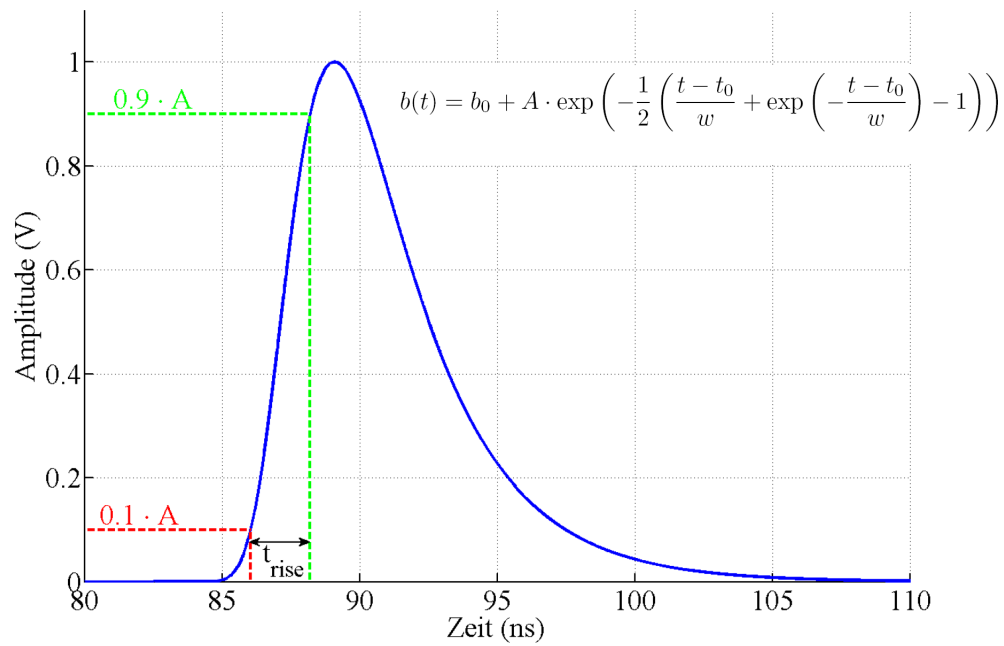


Abbildung 5.12: Die mit einem Offset b_0 an die Signale des Rückstoßdetektors angepasste Moyalkurve [25]. Dabei entspricht t_0 dem Zeitpunkt der Amplitude und w der Kurvenbreite.

aCFD-Methode möglich, Zeitpunkte von zwei sich überlagernden Pulsen zu bestimmen: Da die Anstiegszeit einer Moyal-Kurve linear abhängig von der Amplitude ist, können die Einzelpulse durch die Bestimmung der beiden Amplituden mathematisch erfasst werden.

Kapitel 6

Die GANDALF-Kontrolleinheit

Eine sichere Ansteuerung der einzelnen GANDALF-Baugruppen durch den Anwender ist für die Überwachung und Konfiguration der Module unerlässlich. Dies setzt zum einen eine Schnittstelle zum Versenden von Befehlen und Daten voraus. Zum anderen sind boardspezifische Prozeduren notwendig wie die Ansteuerung der Bausteine sowie die Befehls- und Datenübertragung zwischen Bausteinen. Damit eine von weiteren Boardfunktionen unabhängige Ansteuerung der Module gewährleistet ist, wird bei GANDALF ein CPLD (*Complex Programmable Logic Device*) eingesetzt, der sowohl die dazugehörigen Protokolle als auch die Steuerung der Prozeduren gleichermaßen beinhaltet. Dabei bleiben sämtliche Informationen, die durch die einmalige Programmierung an den Chip übertragen wurden, auch nach einer Trennung von der Spannungsversorgung erhalten.

Im Rahmen dieser Diplomarbeit wurde das Hardwaredesign und die dazugehörige Firmware für den GANDALF Controller-CPLD entwickelt und eine PC-seitige Ansteuerung der GANDALF-Module aufgebaut. Als Schnittstelle zwischen Anwender und Baugruppe kommt bei GANDALF der VME-Bus zum Einsatz. Es wurden mehrere Lese- und Schreibroutinen implementiert, um sowohl Befehle und Daten an ein Modul zu übertragen als auch das Auslesen von Experimentdaten und Statusinformationen zu ermöglichen. Zur Konfiguration der FPGAs wurden drei Methoden in den Controller CPLD implementiert. Eine on-board Datenspeicherung auf einer Compact-Flash Karte wurde realisiert und die Anbindung an einen I²C-Bus integriert. Über diesen Bus kann ein Datentransfer zwischen dem CPLD und Bauteilen stattfinden, die Statusinformationen aufzeichnen oder programmierbare Register enthalten. Die entwickelten Prozeduren wurden auf den Betrieb bei hohen Taktraten bis zu der maximal möglichen Rate des Controller-CPLDs von 80 MHz optimiert.

Die Entwicklung einer Firmware zur Beschreibung von kombinatorischen und getakteten Funktionen setzt weitreichende Kenntnisse der Chiptechnologie voraus. Nur so können die zur Verfügung stehenden Ressourcen effektiv verwendet werden. Zudem erlaubt ein paralleler Ablauf von Prozessen eine Verarbeitung von Signalen und Befehlen in kürzester Zeit und damit eine optimale Auslastung des CPLD. Dafür ist ein geübter Umgang mit der eingesetzten Hardwarebeschreibungssprache VHDL notwendig.

Zwar enthält der eingesetzte CoolrunnerII XC2C512 eine vergleichsweise große Anzahl an Makrozellen, die damit verbundene hohe Kapazität an programmierbaren und getakteten Einheiten wurde jedoch bereits im Verlauf der Entwicklungsarbeit ausgeschöpft. Aus diesem Grund wurde daraufhin eine Programmiertechnik verwendet, die sich durch sparsamen Umgang mit Ressourcen auszeichnet und über die eine vollständige Implementierung aller notwendigen

Funktionen erreicht werden konnte. In den nun folgenden Abschnitten werden die bei dieser Entwicklungsarbeit verwendeten Werkzeuge, Technologien und Methoden erklärt sowie die implementierten Schnittstellen und Prozeduren im Detail erklärt und die resultierenden Datenraten angegeben. Falls nicht anders vermerkt, sind die eingesetzten Signale active-high (ein aktiver Zustand bzw. der Bitwert 1 wird durch den höherwertigen Spannungspegel des jeweiligen Signalsstandards angezeigt).

6.1 Chiptechnologie und Hardwarebeschreibung

Programmierbare Bausteine enthalten eine feste Anzahl von Gattern, Registern und getakteten Flip-Flops. Mit Entwicklungsumgebungen, die in der Regel durch den Hersteller des Bausteines zur Verfügung gestellt werden, kann eine anwendungsspezifische Firmware auf Grundlage der vorhandenen Ressourcen erstellt werden. Mehrere Schnittstellen stehen zur Verfügung, um die Firmware auf den Baustein zu übertragen. Zur Erstellung des Hardwaredesigns werden sogenannte Hardwarebeschreibungssprachen verwendet, beispielsweise ABEL¹ oder das heutzutage vermehrt anzutreffende VHDL².

6.1.1 Der Aufbau eines programmierbaren Logikbausteines

Logische Verknüpfungen werden in sogenannten PLAs (engl.: *Programmable Logic Array*) realisiert, die sowohl in einer UND Matrix als auch in einer ODER Matrix programmiert werden können (Abbildung 6.1). Die Eingänge können auf beliebige Weise negiert oder nicht-negiert sein und werden über eine programmierbare UND Matrix verschaltet, bevor die ODER Matrix die Konjunktionen auf einen programmierbaren Ausgang leitet. Diese Verknüpfungen werden jeweils als *Produktterm* bezeichnet und sind das Ergebnis einer Logikminimierung des in der Entwicklungsumgebung enthaltenen Synthesewerkzeuges. Ausgehend von den PLAs wurden im Verlauf der vergangenen 50 Jahre Bausteine mit wachsender Komplexität entwickelt. Heutzutage werden mehrere, untereinander verschaltbare Funktionsblöcke in einem Chip eingesetzt. Diese komplexen, programmierbaren Logikbausteine tragen die Bezeichnung CPLD. Ein Funktionsblock enthält zahlreiche Produktterme und mehrere Elemente zur synchronen Verschaltung der einzelnen Produktterme. Diese sogenannten Makrozellen werden entweder über einen von außen an den Baustein gegebenen Takt gesteuert oder ein interner Takt, der aus einer Verschaltung der Produktterme erzeugt werden kann, wird auf das Flip-Flop der Makrozelle gegeben. Eine zentrale Schaltmatrix verbindet die einzelnen Funktionsblöcke untereinander, sodass deren Ausgänge entweder mit der Schaltmatrix oder mit den Ein- und Ausgängen des Bausteins verbunden werden können. Diese Bauweise ermöglicht die Realisierung von komplexen und zeitkritischen Vernetzungen, ohne dass Verzögerungszeiten in Betracht gezogen werden müssen. Auf diese Weise können im Zuge einer globalen Verbindung der lokalen PLA Strukturen Hardwarebeschreibungen mit tausenden Logikgattern realisiert werden [41].

Der bei GANDALF eingesetzte XC2C512 CoolRunner-II CPLD enthält eine PLA-Architektur und ermöglicht uneingeschränkte Verbindungen der einzelnen Produktterme innerhalb eines Funktionsblocks (siehe Abbildung 6.2). Dieser CPLD enthält die für einen Coolrunner-II größtmögliche Anzahl von 512 Makrozellen, die auf insgesamt 32 Funktionsblöcke verteilt sind. Die

¹Advanced Boolean Equation Language

²VHSIC (Very High Speed Integrated Circuits) Hardware Description Language

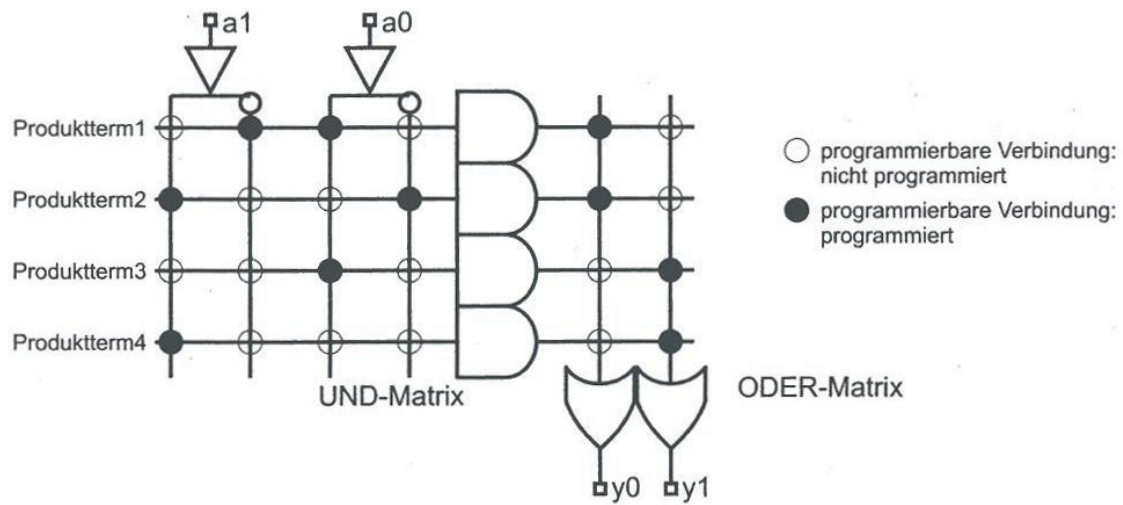


Abbildung 6.1: In einem PLA ist die UND-Matrix beliebig programmierbar. Die ODER-Matrix weist die Produktterme auf einen Ausgang zu. [41]

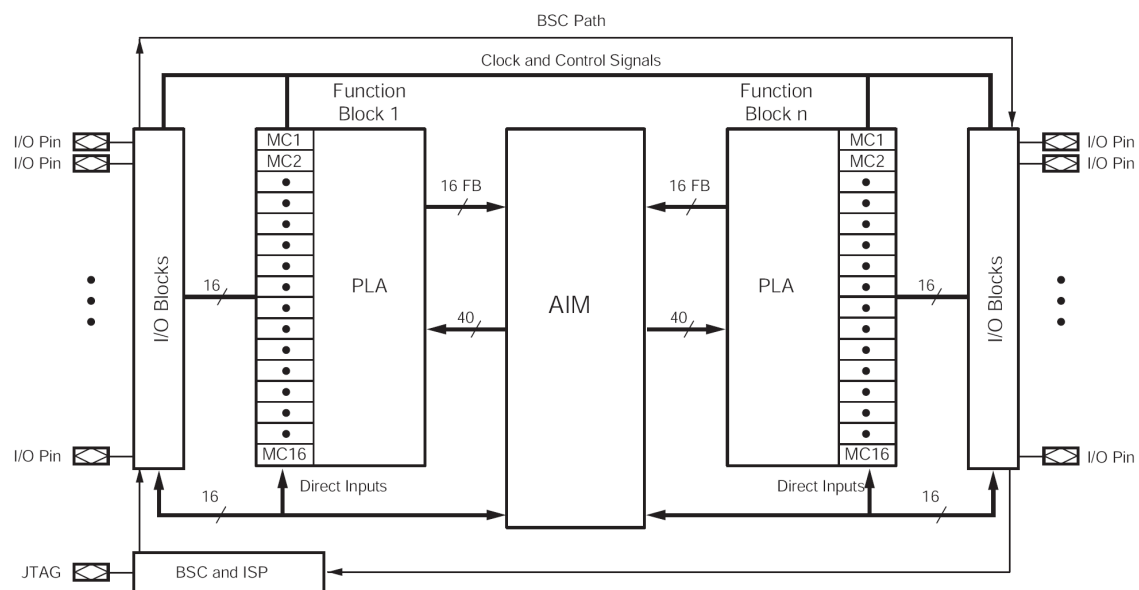


Abbildung 6.2: Der Aufbau des Coolrunner-II CPLDs [42]. Die als AIM bezeichnete Schaltmatrix verbindet die einzelnen Funktionsblöcke, die jeweils sowohl die logischen Verknüpfungen enthalten als auch die synchrone Weiterleitung der Signale durchführen.

Schaltmatrix, hier mit AIM (engl.: *Advanced Interconnect Matrix*) bezeichnet, leitet 40 negierte und nicht negierte Eingänge auf jeden Funktionsblock weiter, der jeweils 16 Makrozellen und 56 Produktterme enthält. Die Anzahl der Logikgatter beläuft sich dadurch auf 12000 [42].

6.1.2 Die Entwicklung der Firmware mit VHDL

Sämtliche in einem Logikbaustein zur Verfügung stehenden Gatter, Register und Flip-Flops können für die Programmierung von Protokollen und Prozeduren herangezogen werden. Das synchrone und kombinatorische Verhalten eines intelligenten Bausteines wird mit einer Hardwarebeschreibungssprache programmiert. In der hier eingesetzten Hardwarebeschreibungssprache VHDL wird in Abhängigkeit von Signalwerten eine Reaktion des CPLD ausgelöst: Prozesse werden dann angestoßen und sequenziell abgearbeitet, falls sich Signale, die in der Sensitivitätsliste eines Prozesses enthalten sind, ändern. Dabei kann ein Prozess sensitiv auf ein Taktsignal und auf ein Reset-Signal sein, ein weiterer Prozess kann von mehreren asynchronen Signalen abhängig sein.

Prozesse stellen ein wesentliches Modellierungselement in VHDL dar und können nebenläufig betrieben werden. Folgt beispielsweise aus einem ablaufenden Prozess eine Signaländerung, so kann damit ein weiterer Prozess angestoßen werden. Listing 6.1 zeigt einen Prozess zur Beschreibung eines Flip-Flops mit asynchronem Reset: Falls das mit höchster Priorität versehene RESET nicht auf logisch Eins gesetzt ist, wird bei einer steigenden Flanke des Taktsignales CLK das Eingangssignal auf den Ausgang gegeben ($DOUT \leq DIN$).

Listing 6.1: Ohne ein aktives Resetsignal wird bei einer steigenden Taktflanke der Eingang des Bausteines `dflip` auf den Ausgang gelegt. Durch diesen Prozess ist ein Flip-Flop mit asynchronem Reset vollständig beschrieben.

```
1  entity dflip is
2      Port ( CLK : in  STD_LOGIC;
3            RESET : in  STD_LOGIC;
4            DIN  : in  STD_LOGIC;
5            DOUT : out  STD_LOGIC);
6  end dflip;
7
8  architecture flipflop of dflip is
9  begin
10
11  ff: process(CLK,RESET) is
12  begin
13
14  if (RESET='1') then
15  DOUT<='1';
16
17  elsif(CLK'event and CLK='1')
18  then
19  DOUT<=DIN;
20  end if;
21
22  end process ff;
```

Die Zustandsmaschine

Zur Steuerung einer Befehls- und Datenübertragung werden sogenannte Zustandsmaschinen eingesetzt. Dieser in der englischen Sprache als *finite state machine* (FSM) bezeichnete Kreislauf definiert diskrete Zustände, die der Chip in Abhängigkeit von Eingangssignalen einnimmt. Ist die Voraussetzung für die Aktivierung eines Zustandes erfüllt, so wird der Zustand mit der

folgenden Taktflanke eingenommen und es werden die innerhalb dieses Zustandes programmierten Signalzuweisungen gültig. Ein weiterer Zustandswechsel ist nun an Bedingungen, die in dem aktuellen Zustand formuliert sind, geknüpft. Auf diese Weise können protokollspezifische Randbedingungen wie ein bestimmtes Zeitverhalten oder die Abfolge von verschiedenen Steuersignalen eingehalten werden. Eine Zustandsmaschine erlaubt zudem das gesteuerte Einlesen von Daten, sodass Informationen zum Zeitpunkt des Versendens bereits vorliegen. Die Zustandsmaschine in Abbildung 6.3 beschreibt die Auslese von Informationen eines Moduls, die in zwei Zuständen enthalten sind. Der Zustand *Board ID* enthält die Identifikationsnummer des Moduls und der Zustand *Board Status* Informationen zum Status der Baugruppe. Sobald ein Befehl zum Auslesen der jeweiligen Information als Adresse gesendet wird, erfolgt die Dekodierung des Befehls im Grundzustand *idle* und der dem Befehl entsprechende Zustand wird eingenommen. Nachdem die Information übermittelt worden ist, erreicht die Zustandsmaschine wieder den Grundzustand und ein weiterer Befehl kann angenommen werden.

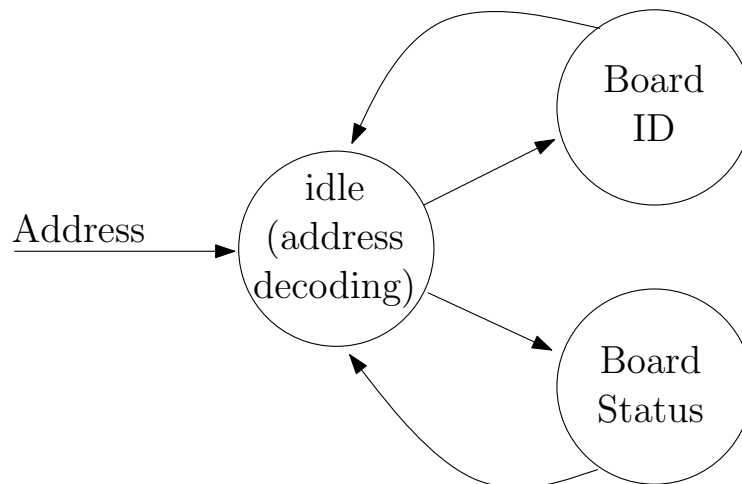


Abbildung 6.3: Die Zustandsmaschine zur Auslese der Boardidentifikation und des Boardstatus. Die Dekodierung der Adresse erfolgt im Grundzustand, nach erfolgter Datenübertragung erreicht der Kreislauf wieder den Ausgangszustand

Die zwei-Prozess Methode

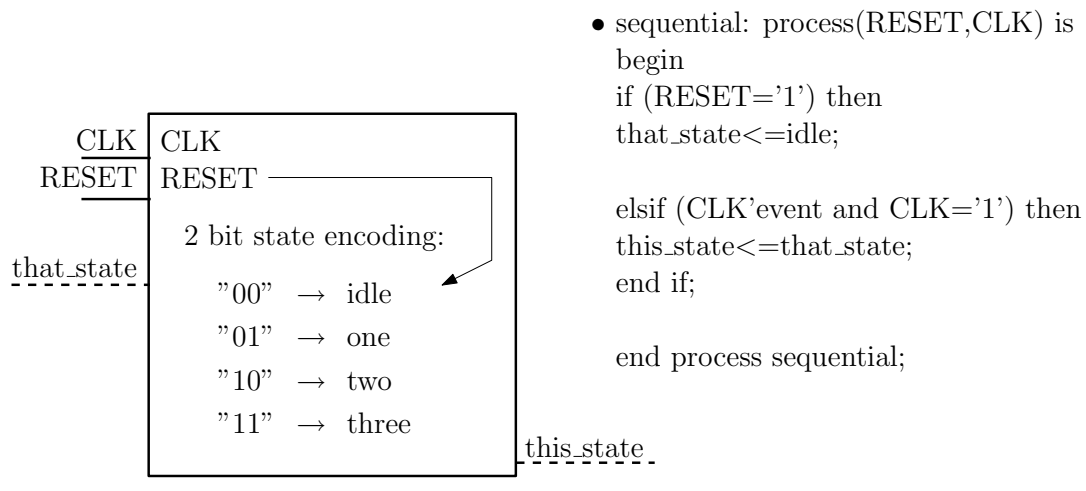
Zur Verarbeitung von Befehlen und Daten wird ein Taktsignal benötigt, damit ein synchroner Zustandswechsel erfolgen kann. Ein auf logischen Verknüpfungen beruhender Zustandswechsel würde zu einem undurchschaubaren Zeitverhalten führen und damit zu einem unkontrolliertem Ablauf der implementierten Protokolle und Funktionen. Im Verlauf der Entwicklung der Firmware hat sich gezeigt, dass mit einem einzigen getakteten Prozess zur Beschreibung der Zustandsmaschine eine große Anzahl der verfügbaren synchronen Logikelemente verbraucht werden.

Jede Zuweisung in einem Prozess, dessen Sensitivitätsliste ausschließlich ein *RESET*- und ein Taktsignal (*CLK*) enthalten ist, wird vom Synthesewerkzeug als einzelnes Flip-Flop mit einem *Enable*-Signal realisiert. Dieses *Enable*-Signal wird aktiv, sobald der Zustand, in dem sich diese Zuweisung befindet, eingenommen wird. Im Allgemeinen werden durch diese Program-

miertechnik bei einer großen Anzahl von bedingten Zuweisungen eine übermäßige Anzahl von Makrozellen verbraucht.

Mit einer Programmieretechnik, durch die ausschließlich der Zustandswechsel betaktet wird und sämtliche Signalzuweisungen durch logische Gatter realisiert werden, können Flip-Flops eingespart werden. In dieser auch als zwei-Prozess Methode bekannten Programmierweise wird die Zustandsmaschine durch zwei nebenläufige Prozesse beschrieben. Dabei ist einer der beiden Prozesse betaktet, wohingegen der zweite Prozess lediglich über logische Verknüpfungen von Signalen eine Zuweisung veranlasst.

Die Definition der Zustandsmaschine erfolgt nun in dem allein auf logischen Verknüpfungen beruhenden Prozess, wohingegen der Zustandswechsel im sequenziellen Prozess vorgenommen wird. Abbildung 6.4 zeigt die Beschreibung einer Zustandsmaschine mit der zwei-Prozess-Methode anhand eines Beispiels mit vier Zuständen. Die einzelnen Zustände sind im Prozess *combined* enthalten, der Zustandswechsel erfolgt im Prozess *sequential*. Durch ein aktives *RESET*-Signal wird die Zustandsmaschine auf den *idle*-Zustand gesetzt und logisch Null an die beiden Ausgänge *a* und *b* gegeben. Gleichzeitig erfolgt die Zuweisung für einen Zustandswechsel ($that_state \leq one$). Der Zustand *one* wird jedoch erst erreicht, sobald das *RESET*-Signal deaktiviert wird und mit der nächsten steigenden Flanke im sequenziellen Prozess der Übergang $this_state \leq that_state$ vollzogen wird. Eine Änderung von *this_state* hat eine Aktivierung des kombinatorischen Prozesses zur Folge und damit tritt Zustand *one* in Kraft. Der Wechsel der Zustände läuft ab sofort synchron zur steigenden Taktflanke ab. Die Rückkehr in den Grundzustand *idle* ist an ein aktives *STROBE*-Signal geknüpft.



- combined: process(this_state,STROBE) is

```

begin
  case this_state is

```

```

    when idle =>

```

```

      a<='0';
      b<='0';
      that_state<=one;

```

```

    when one =>

```

```

      a<='0';
      b<='1';
      that_state<=two;

```

```

    when two =>

```

```

      a<='1';
      b<='0';
      that_state<=three;

```

```

    when three =>

```

```

      a<='1';
      b<='1';
      if (STROBE='1') then
        that_state<=idle;
      end if;

```

```

    end case;

```

```

  end process combined;

```

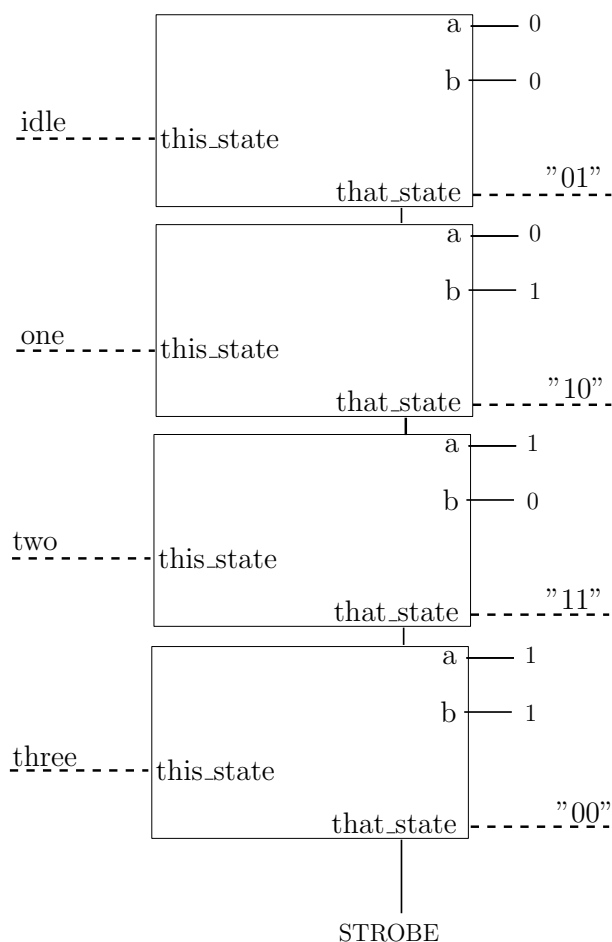


Abbildung 6.4: Eine Zustandsmaschine, die über die zwei-Prozess-Methode gesteuert wird. Im betakteten Prozess (oben) findet bei einem inaktiven *RESET*-Signal der Zustandswechsel statt. Die einzelnen Zustände sind im kombinatorischen Prozess (unten) definiert. Eine Zustandskodierung wird durch das Synthesewerkzeug vorgenommen.

6.2 Die VME-Schnittstelle

Die Übertragung von Adressen und Daten auf dem VME-Bus wird durch das Setzen von Lese- und Schreibbefehlen auf dem SBC veranlasst. Bei Lesebefehlen empfängt der VME-Master ein Datenwort, bei Schreibbefehlen wird das zu versendende Datenwort direkt an den Befehl angehängt. Der als Master eingesetzte SBC enthält ein Linux-Betriebssystem, die nötigen VME Geräte und die dazugehörige VME-Treibersoftware. Abgesehen von der 64 bit breiten Adress- und Datenübertragung sind bei GANDALF sämtliche in 5.1.1 beschriebene Routinen implementiert.

Die grundlegenden Befehls- und Datentransfers von VME werden asynchron über ein sogenanntes *handshaking*-System gesteuert: Sowohl Master als auch Slave werden unabhängig voneinander getaktet, aktuelle Adressen und Daten werden einem Slave durch *Strobe*-Signale mitgeteilt und der Slave bestätigt den Empfang oder das Versenden von Daten mit einem *Data Acknowledge*. Des weiteren ist ein Signal zur Kennzeichnung der Übertragungsrichtung vorhanden, ein *Address Modifier Code* gibt die Art des Datentransfers vor.

In der für die GANDALF-Einheit realisierten VME-Schnittstelle ist das Adresswort analog zur Adressdekodierung bei CATCH aufgebaut [43]. Das CATCH-Modul wurde vor einigen Jahren in unserer Arbeitsgruppe entwickelt und kann über die VME-Schnittstelle angesteuert und ausgelesen werden. Die Beschreibung des dabei eingesetzten VME-Protokolls wurde in ABEL vorgenommen und diente so als Ausgangspunkt für die Entwicklung der GANDALF VME-Ansteuerung. Das GANDALF Adresswort enthält die GANDALF- und Board-Identifikation sowie eine eindeutige Registeradresse. Die mit dem Senden eines Befehls verbundenen verschiedenen Lese- und Schreibroutinen können mit den in Tabelle 6.1 gelisteten Signalen vollständig ausgeführt werden.

Die Taktrate des SBC beträgt 10 MHz, wohingegen die Verarbeitung von Signalen im Controller-CPLD bei einer Taktrate von 80 MHz erfolgt. Um Signale zwischen einzelnen Modulen zu übertragen, wird in der VME-Technologie der TTL-Signalstandard eingesetzt. VME-Steuersignale sind active-low, der jeweilige Vorgang wird durch den niederwertigen Spannungspegel aktiviert (dies wird im folgenden durch * gekennzeichnet).

Tabelle 6.1: Die bei einem VME-Transfer eingesetzten Daten-, Adress- und Steuersignale.

Signal	Bezeichnung gemäss der VME-Spezifikation [44]	Bedeutung bei GANDALF
A(31:1)	Address Bus	GANDALF- und Board-Identifikation, Art des Transfers, Registeradressen
D(31:0)	Data	Konfigurationsdaten, Experimentdaten, Boardstatus, Registerinhalte ...
AM(5:0)	Address Modifier Code	Dieses Wort wird durch den Master beim Senden einer Adresse gesetzt und enthält Grösse und Typ der Adresse: Extended (32 bit) Supervisory /non-privileged Daten- und Blocktransfer
AS*	Address Strobe	Der Master signalisiert gültige AM(5:0) und A(31:1)
DS0*	Data Strobe 0	Daten werden vom Master geschrieben oder angefordert, beim Single Transfer kombiniert mit DS1,
DS1*	Data Strobe 1	Daten werden vom Master geschrieben oder angefordert, beim Single Transfer kombiniert mit DS0,
DTACK*	Data Acknowledge	Der Slave signalisiert dem Master die Aufnahme oder das Versenden eines Datenwortes
BSYSRES*	SYSRES	kann von jedem Modul zur Anzeige eines Resets aktiviert werden
LWORD*	Long Word	Zeigt mit DS0, DS1 und A01 die Grösse des Datentransfers an /Dient darüberhinaus in 64 bit Transfers als Adress- und Datenleitung eines bits
IACK*	Interrupt Ack.	Mit LWORD und A01 Voraussetzung zur Boardansteuerung
WRITE*	Read/Write	Der Master signalisiert die Richtung eines Datenstroms

6.2.1 Die VME Board-Identifikation

Damit ein VME-Befehl ausschließlich das GANDALF-Modul erreicht, das angesteuert werden soll, sind im jeweiligen Adresswort zwei Kennungen zur Identifikation einer GANDALF-Baugruppe enthalten. Ein an ein GANDALF gerichteter Befehl wird erst dann dekodiert und verarbeitet, wenn sowohl die GANDALF-Kennung $0xE0^3$ als auch die jeweilige Board-ID eines Moduls mit der in der Adresse enthaltenen Bitfolge übereinstimmt. Während die allgemeine GANDALF-Kennung bereits voreingestellt ist und eine Verwechslung mit anderen Modulen im Trägergehäuse ausschließt, kann die Board-ID über zwei Drehschalter (*DIP-Switches*), die sich auf jedem GANDALF befinden, eingestellt werden. Abbildung 6.5 zeigt eine Abfrage der Adresse zu Identifikationszwecken wie sie im Controller-CPLD realisiert ist.

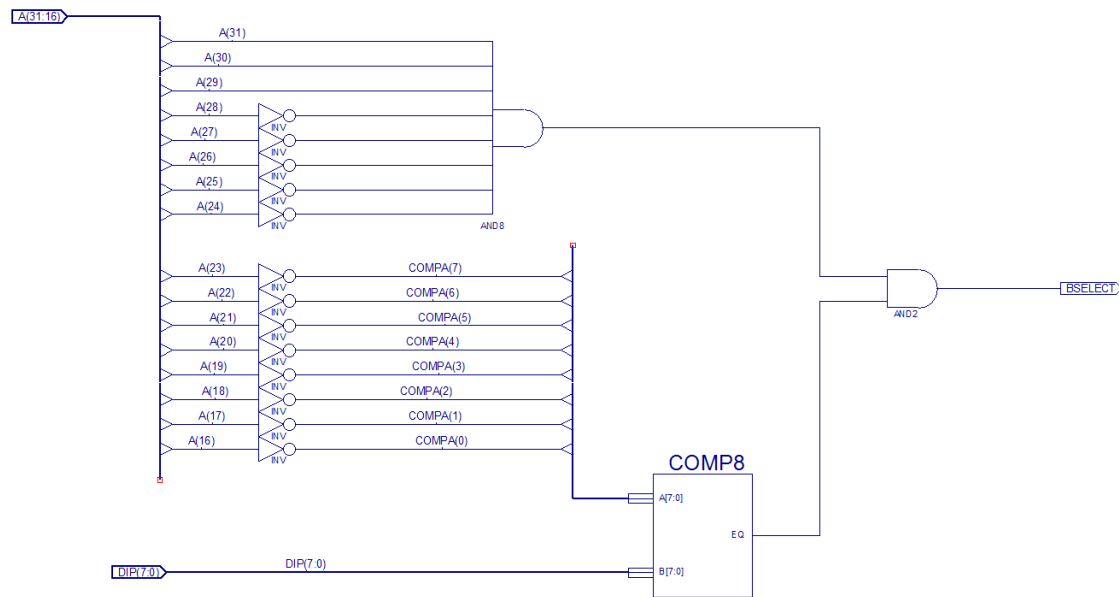


Abbildung 6.5: Beim Empfang einer VME Adresse wird der obere Adressbereich A(31:16) mit den board-eigenen Kennungen verglichen. Stimmen sämtliche bits überein wird *BSELECT* aktiviert. Erst dann kann eine weitere Verarbeitung der Adresse und die Ausführung eines Befehls erfolgen.

6.2.2 Die VME-Adressierung

Das 32 bit breite Adresswort enthält die Identifikationsmerkmale eines Moduls sowie die Kennung des anzusteuern Registers. Dieses Adresswort wird nach erfolgter Identifikation mit der fallenden Flanke des *Address Strobe*-Signals AS^* dekodiert, womit die Aktivierung eines Zustandes veranlasst wird. Die bits A(15:2) sind laut VME-Spezifikation [44] frei verfügbar. Für CPLD-Register wurde der Adressbereich A(9:2) reserviert. Für die Ansteuerung der FPGAs einer GANDALF-Einheit mit VME-Befehlen wurde der Adressbereich A(13:10)

³ $0xE0$ wurde bereits als CATCH-Kennung eingesetzt. Da die GANDALF-Module in einem 6U VME64x/VXS-Trägergehäuse betrieben werden, ist eine Verwechslung mit einem CATCH-Modul ausgeschlossen.

vorgesehen. Der Typ der Datenübertragung kann vom Anwender mit den bits A(15:14) eingestellt werden. In Abbildung 6.6 ist der Aufbau eines Adresswortes bei GANDALF gezeigt. Die Kommandos zur Ansteuerung der FPGAs über die VME-Schnittstelle gehen aus der Firmware-Version hervor, mit der die FPGAs konfiguriert wurden. Um eine GANDALF-Baugruppe über

A31	GANDALF-ID $0xE0$
A24	
A23	BOARD-ID $0x00$ bis $0xFF$
A16	
A15	Art des Transfers
A14	
A13	FPGA-Register
A10	
A09	CPLD-Register
A02	
A01	'0'
A00	'0'

Abbildung 6.6: Das allgemeine Adressmuster zur Ansteuerung einer GANDALF-Baugruppe über die 32 bit VME-Adressierung (*extended addressing*).

die VME-Schnittstelle anzusteuern, werden die in Tabelle 6.2 aufgelisteten CPLD-Register verwendet.

Tabelle 6.2: Die VME-Register im Controller-CPLD und die zu einer Schreibroutine zugehörigen Datenwörter in hexadezimaler Schreibweise. Die jeweiligen Prozesse enden nach ihrem Ablauf im Grundzustand.

A(15:0)	R/W	Datenwort	Beschreibung
0x00FC	R	-	Lese Identifikationsmerkmale
0x0004	R	-	Lese Boardstatus
0x0100	R	-	Lese aktuelle Spannungswerte der Netze 1-3
0x0110	R	-	Lese aktuelle Spannungswerte der Netze 4-6
0x0120	R	-	Lese aktuelle Spannungswerte der Netze 7-8
0xC000	R	-	Lese blockweise Daten aus, schreibe auf CF-Karte
0x0010	W	0x00000040	Setze PROGRAM_B Pin der FPGAs auf 0
0x0010	W	0x00000080	Setze PROGRAM_B Pin der FPGAs auf 1
0x8000	W	Konfigurationsdaten	wort- und blockweise FPGA Konfiguration
0x0070	W	ACE Verzeichnis	FPGA-Konfiguration über SystemACE
0x0000	R	-	Grundzustand

6.2.3 Die Address Modifier Codes

Zur Übermittlung von Informationen bezüglich des aktuellen Transfertyps sendet der Master beim Senden eines Befehls automatisch einen eindeutigen *Address Modifier Code* (AM) an den Slave. Dieser Code ist laut VME-Spezifikation [44] Voraussetzung für einen gültigen VME-Transfer und wird zusammen mit der Adresse im Controller-CPLD dekodiert. So ist eine Kontrolle des Slave-Moduls durch die VME-Steuerlogik des SBC auch während einer Datenübertragung gewährleistet. In Tabelle 6.3 sind die bei GANDALF eingesetzten Address Modifier Codes aufgelistet.

Tabelle 6.3: Die VME-*Address Modifier Codes*.

Hierarchie	Art des Transfers	Kodierung
supervisory	data access	0x0D
non-privileged		0x09
supervisory	block transfer	0x0F
non-privileged		0x0B

6.2.4 Die VME-Leseroutinen

Um Datenworte eines GANDALF-Moduls über die VME-Schnittstelle auszulesen, werden Lesebefehle am SBC ausgeführt. Dabei kommen zwei Routinen zum Einsatz, mit denen ein einzelnes Datenwort (einfacher Transfer) bzw. mehrere Datenworte pro Befehl (Blocktransfer) an den SBC übertragen werden. Die einzelnen Befehle werden grundsätzlich von einem Adresswort begleitet, das im wesentlichen die Identifikationsmerkmale des Moduls und die Registeradresse beinhaltet (die einzelnen Lesebefehle sind im Anhang A.7 aufgelistet). Abbildung 6.7 zeigt den zeitlichen Ablauf der Board-Adressierung des SBC, der bei der Entwicklung der Firmware eingesetzt wurde. Der Master gibt die Adresse auf die Adressleitung und signalisiert mit fallender Flanke an AS^* ihre Gültigkeit. Sofern die in der Adresse enthaltenen Identifikationsmerkmale mit denjenigen des Moduls übereinstimmen, wird die Registeradresse mit fallender Flanke des AS^* -Signals im Controller-CPLD dekodiert. Zur Kennzeichnung der Richtung des Datenstroms bleibt $WRITE^*$ inaktiv.

Während eines einfachen Transfers wird der *Address Modifier Code* auf *Data Access* gesetzt.

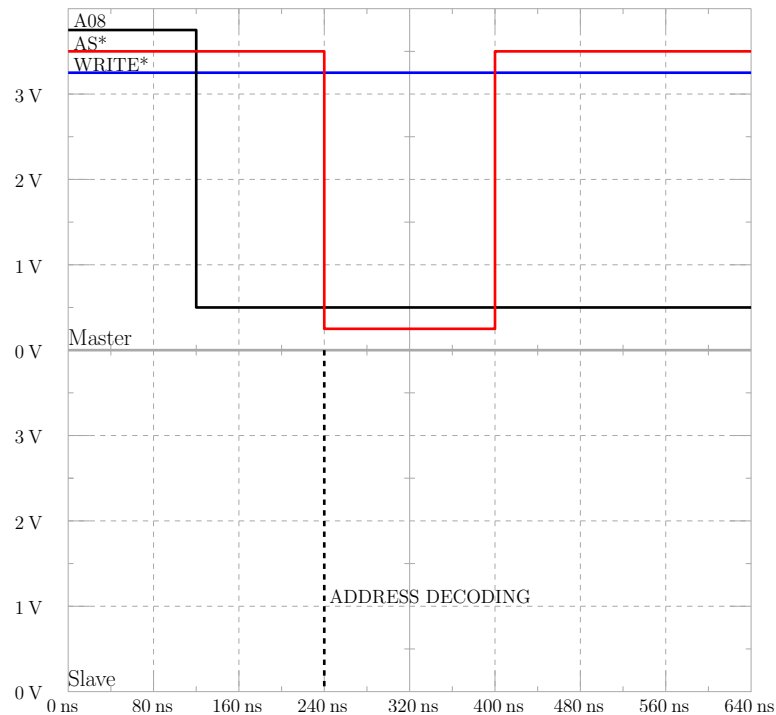


Abbildung 6.7: Der SBC beginnt die einfache Leseroutine mit dem Anlegen der Adresse A . Die Richtung der Daten wird mit $WRITE^*$ angegeben, mit fallender Flanke an AS^* ist die Adresse gültig.

Daten aus dem angesteuerten Register werden mit zwei *Data Strobe*-Signalen angefordert ($DS0^*$, $DS1^*$). Sobald der SBC beide Signale aktiviert, erreicht die Zustandsmaschine im Controller-CPLD den aus der Adressdekodierung erhaltenen Zustand. Das in diesem Zustand enthaltene Datenwort wird parallel auf die Datenleitung gesetzt und mit $DTACK^*$ validiert. Sobald das Datenwort den VME-Master erreicht hat, signalisiert dieser einen erfolgreichen Empfang mit dem Rücksetzen der *Data Strobes*. Eine einfache Leseroutine wird durch den Sla-

ve beendet: Sobald die *Data Strobes* inaktiv sind, deaktiviert er *DTACK**. Abbildung 6.8 gibt den zeitlichen Verlauf der Datenübertragung im einfachen Lesemodus wieder. Die Leseroutine

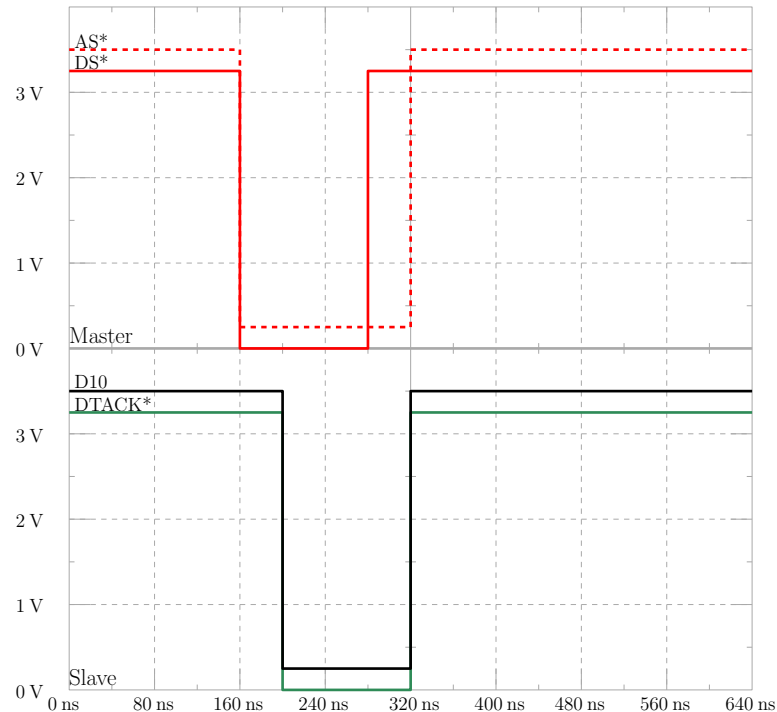


Abbildung 6.8: Die Übertragung eines Datenwortes an den SBC im einfachen Transfer. Der Master fordert mit den *Data Strobes* ein Datenwort an. Der Slave sendet das Wort und aktiviert *DTACK**. Inaktive *Data Strobes* signalisieren die Ankunft des Datenwortes am SBC, woraufhin der Slave *DTACK** deaktiviert.

wird mit dem Setzen der Adresse durch den Master eingeleitet und mit dem Rücksetzen von *DTACK** durch den Slave beendet. Das dazwischenliegende Zeitintervall $t_{\text{single_read}} = 280 \text{ ns}$ gibt die Zeit an, die für das Lesen eines Datenwortes aus dem angesteuerten Register benötigt wird. Während dieser Zeit ist es nicht möglich, weitere Module, die mit dem SBC in Verbindung stehen, anzusteuern.

Im Blocktransfer übergibt der Master die Adresse analog zum einfachen Transfer (siehe Abbildung 6.7) an die Module. *Address Strobe* und *Data Strobes* signalisieren eine gültige Adresse sowie die Bereitschaft des Masters, Daten zu empfangen. Nachdem das angesteuerte Modul die Registeradresse verarbeitet hat, wird das erste Datenwort ausgelesen.

Im Gegensatz zum einfachen Lesemodus wird dieser Zustand nicht verlassen: Der Master signalisiert den Blockmodus durch den entsprechenden *Address Modifier Code* und erhält dadurch den Zustand aufreht. Sobald der Master erneut bereit ist, Daten zu empfangen, aktiviert er *DS0**. Der Slave versendet ein weiteres Datenwort, wartet auf das Rücksetzen von *DS0** und setzt im Anschluss daran *DTACK** zurück. Ein erneutes Setzen von *DS0** durch den Master erfolgt, falls die Anzahl der auszulesenden Worte noch nicht erreicht ist. Zum Abschluss des Blocktransfers löscht der VME Master den *Address Modifier Code* und deaktiviert damit den Blockmodus. Der Controller-CPLD setzt ein letztes Mal *DTACK** zurück und nimmt

den Grundzustand ein. Abbildung 6.9 zeigt die Aufnahme dieser gebündelten Übertragung mit dem Oszilloskop. Über ein aktives $DS0^*$ wird ein Datenwort angefordert, das hier durch $D10$ repräsentiert wird. Mit aktivem $DTACK^*$ bestätigt GANDALF das Versenden des Datenwortes. Über ein inaktives $DTACK^*$ signalisiert GANDALF, dass ein weiteres Datenwort empfangen werden kann. Die Dauer der Bus-Belegung durch das blockweise Lesen ist abhängig

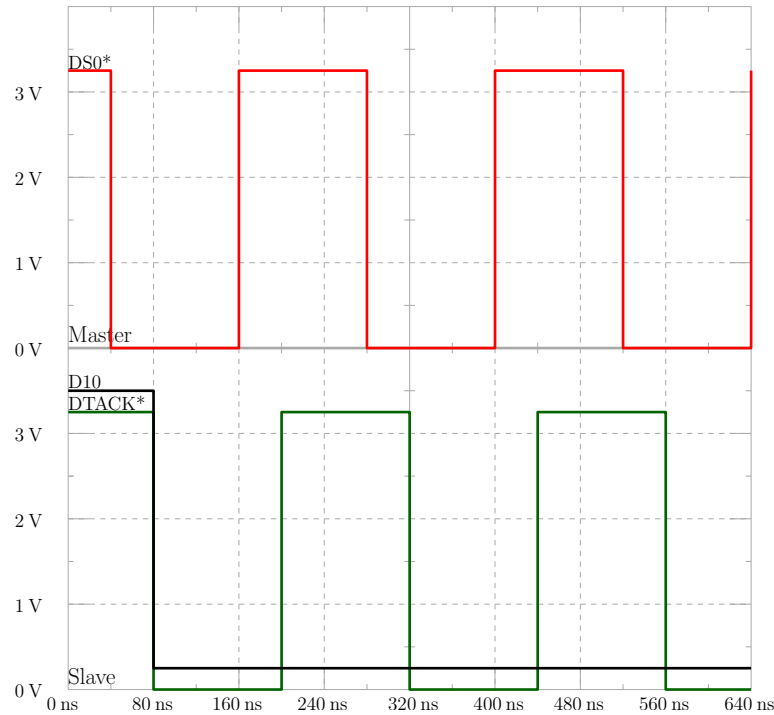


Abbildung 6.9: Eine Oszilloskopaufnahme zu Beginn einer gebündelten Übertragung von Daten an den SBC.

von der Anzahl der übertragenen Worte. Als Maß für die maximale Länge eines Blocktransfers kann das Zeitintervall zwischen dem Anlegen der Adresse A und dem letzten Rücksetzen von $DTACK^*$ durch den Slave dienen. Die effektive Dauer eines Blocktransfers wächst linear mit der Anzahl der übertragenen Worte, eine Übertragung von 256 Worten im Blockmodus ist nach $t_{\text{block_read}} = 53,60 \mu\text{s}$ beendet. Die Dauer eines Datentransfers mit 256 direkt hintereinander ausgeführten, einzelnen Lese-prozeduren beträgt $t_{\text{single_read}} \cdot 256 = 71,68 \mu\text{s}$. Demnach ist die Datenrate im Blockmodus gegenüber der einfachen Leseroutine um den Faktor 1,34 größer. Mit der Anzahl der Datenworte zwischen der Aktivierung der Adresse und dem (letztmaligem) Rücksetzen von $DTACK^*$ entsprechen diese Zeiten den folgenden Datenraten:

$$f_{\text{single_read}} = 14,3 \text{ MB/s} \quad (6.1)$$

$$f_{\text{block_read}} = 19,1 \text{ MB/s} \quad (6.2)$$

So konnten bis zu 48 % der theoretischen Übertragungsrate von $f_{\text{max}} = 40 \text{ MB/s}$ [44] erreicht werden. Eine weitere Steigerung der tatsächlichen Raten erfordert eine höhere Datenrate zu den Speichereinheiten im SBC und ein beschleunigtes Setzen des *Address-Strobe*-Signals (kürzere Setup-Time) nach dem Anlegen einer Adresse.

6.2.5 Die VME-Schreibroutinen

Werte von Registern sowie ganze Konfigurationsdateien werden mit VME-Schreibroutinen übertragen [44]. GANDALF bietet zwei Möglichkeiten, Daten zu empfangen: In einem einfachen Transfer wird ein einzelnes Datenwort in den Controller-CPLD eines GANDALF geschrieben. In einem Blocktransfer werden mehrere Worte pro Befehl an ein GANDALF-Modul gesendet. Die einzelnen Befehle sind in Anhang A.7 beschrieben. Im Allgemeinen wird durch die Dekodierung der Adresse im Controller-CPLD bei aktiven *Data Strobes* ein Zustand aktiviert. Der CPLD ist daraufhin bereit, ein oder mehrere Datenworte zu empfangen. Der Slave signalisiert eine abgeschlossene Weiterverarbeitung der Daten, indem er *DTACK** deaktiviert, woraufhin er den Grundzustand einnimmt. Bei einer gebündelten Datenübertragung wird in Abhängigkeit des *Address Modifier Code* ein neues Datenwort erwartet. Sobald alle zu schreibenden Worte an das Modul übertragen worden sind, wird im Controller-CPLD der Grundzustand aktiviert.

Die Adressierung einer GANDALF-Einheit durch den SBC erfolgt in einem einfachen Transfer nach dem in Abbildung 6.10 gezeigten Verlauf. Die Gültigkeit der Adresse *A* wird mit fallender Flanke an *AS** signalisiert, *WRITE** gibt die Richtung des Datenstroms vor. Im Controller-CPLD wird die Adresse mit der fallenden Flanke an *AS** dekodiert. Nachdem der

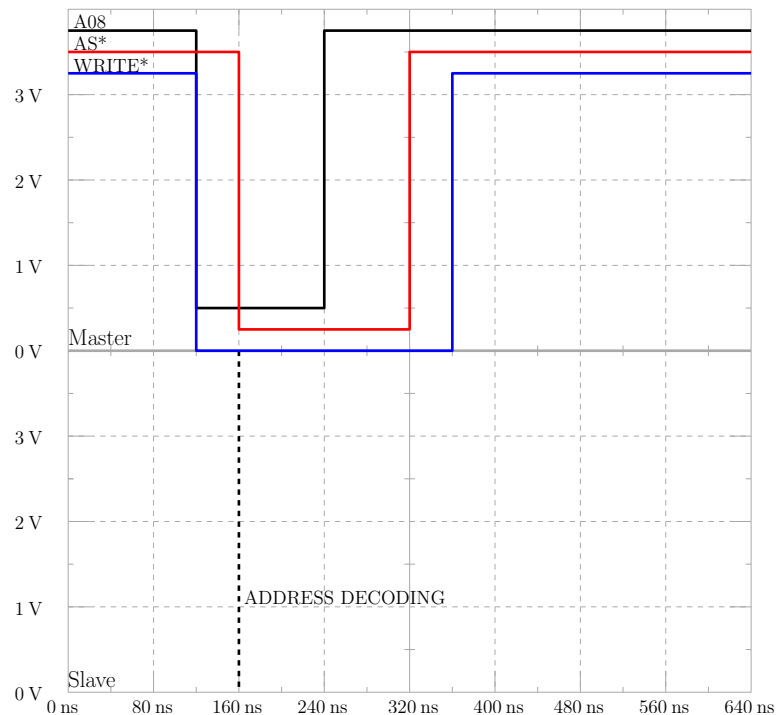


Abbildung 6.10: Der SBC beginnt die Schreibroutine beim einfachen Transfer mit dem Anlegen der Adresse *A*. Die Richtung der Daten wird mit *WRITE** angegeben, mit fallender Flanke an *AS** ist die Adresse gültig.

angesteuerte Zustand ermittelt worden ist und der SBC die *Data Strobes* (*DS0**, *DS1**) aktiviert hat, nimmt der Controller-CPLD diesen Zustand ein. Er ist solange aktiviert bis der Slave den Bus mit logisch Null an *DTACK** freigibt. Die Zeit zwischen dem Anlegen der Adresse

durch den Master und der Freigabe von $DTACK^*$ durch den Slave definiert die Dauer einer Schreibroutine. Der Abschluss der Übertragung durch das $DTACK^*$ -Signal kann jedoch erst nach der Verarbeitung der Daten im CPLD erfolgen. Bei GANDALF werden Datenworte zur Konfiguration von Bausteinen sowohl seriell weitergeleitet als auch auf einem 8 bit breitem Bus. Die dadurch entstehenden verschiedenen Verarbeitungszeiten führen zu einer unterschiedlichen Dauer einer Schreibroutine. Abbildung 6.11 zeigt den zeitlichen Verlauf einer einfachen Datenübertragung mit serieller Weiterleitung. Das einmalige Schreiben eines Datenwortes mit anschließender Serialisierung ist nach $t_{\text{single_write_serial}} = 580 \text{ ns}$ abgeschlossen. Die

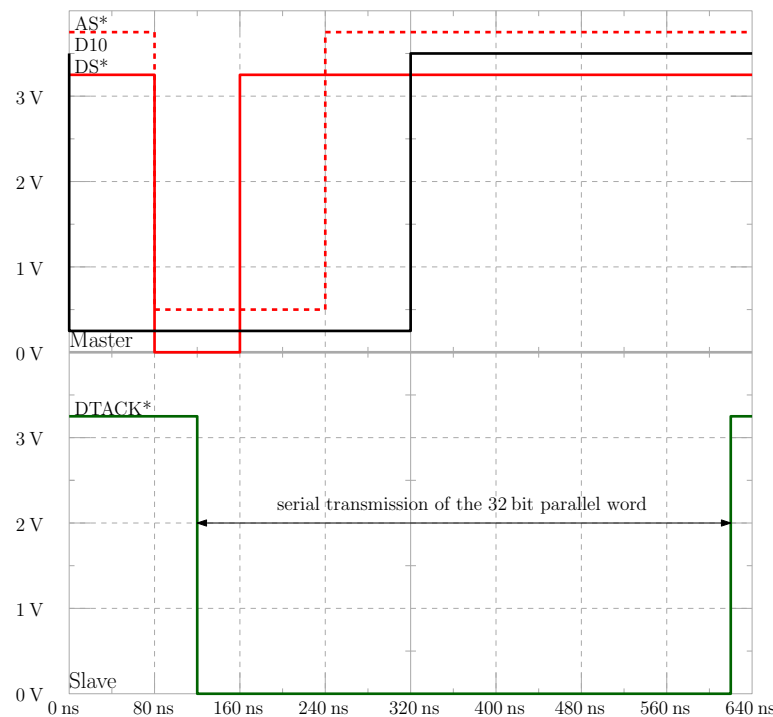


Abbildung 6.11: Die Übertragung eines Datenwortes an GANDALF, bei der die Daten im Controller-CPLD in einzelne bits aufgeteilt und weitergeleitet werden.

Übertragungszeit eines 32 bit-Datenwortes mit einer Weiterleitung der Daten in Bytes könnte auf $t_{\text{single_write_byte}} = 280 \text{ ns}$ verringert werden. Der in Abbildung 6.12 gezeigte Verlauf der Datenübertragung zeigt ein im Vergleich zur seriellen Weiterleitung verkürztes $DTACK^*$.

Das blockweise Schreiben von Daten wird zur Übertragung von Konfigurationsdateien verwendet, die in hexadezimalen Format vorliegen. Abgesehen von einem länger anhaltenden $WRITE^*$ entspricht der zeitliche Verlauf der Adressierung eines Moduls im Blocktransfer demjenigen beim einzelnen Transfer 6.10. Wie auch beim blockweisen Lesetransfer wird die Datenübertragung mit $DS0^*$ vom Master aktiviert und mit $DTACK^*$ vom Slave bestätigt. Abbildung 6.13 zeigt den Beginn einer Datenübertragung bei GANDALF mit einer Serialisierung der einzelnen Worte im Controller-CPLD. Innerhalb der für die Serialisierung eines Wortes benötigten Zeit $32 \cdot 12,5 \text{ ns} = 400 \text{ ns}$ muss $DTACK^*$ auf logisch Eins gehalten werden. Eine Aufteilung der über VME gesendeten Worte in Bytes beschleunigt den Datentransfer, da die Anpassung eines Datenwortes nun innerhalb von 8 Taktzyklen erfolgt.

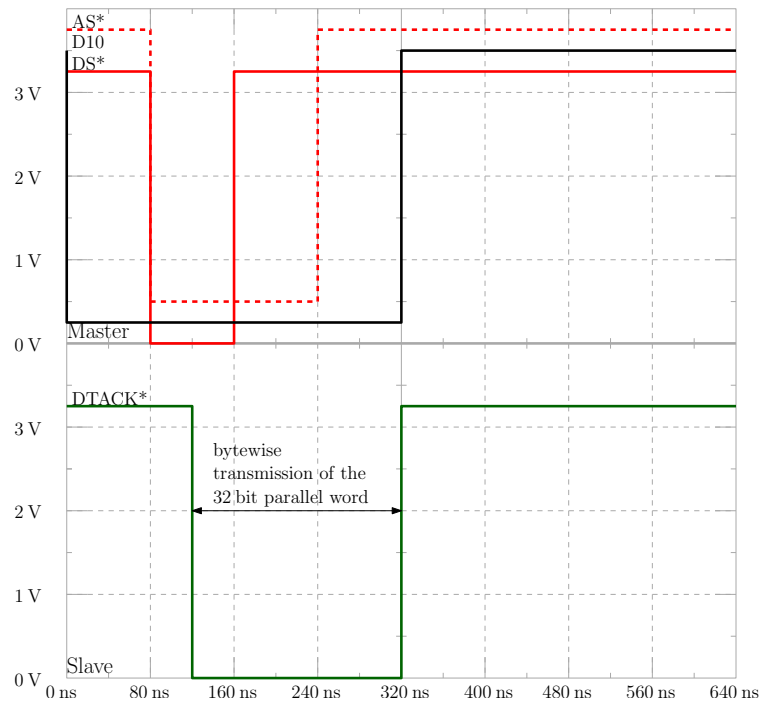


Abbildung 6.12: Die Übertragung eines einzelnen Datenwortes an GANDALF, bei der das Datenwort im Controller-CPLD byteweise aufgeteilt und weitergeleitet wird.

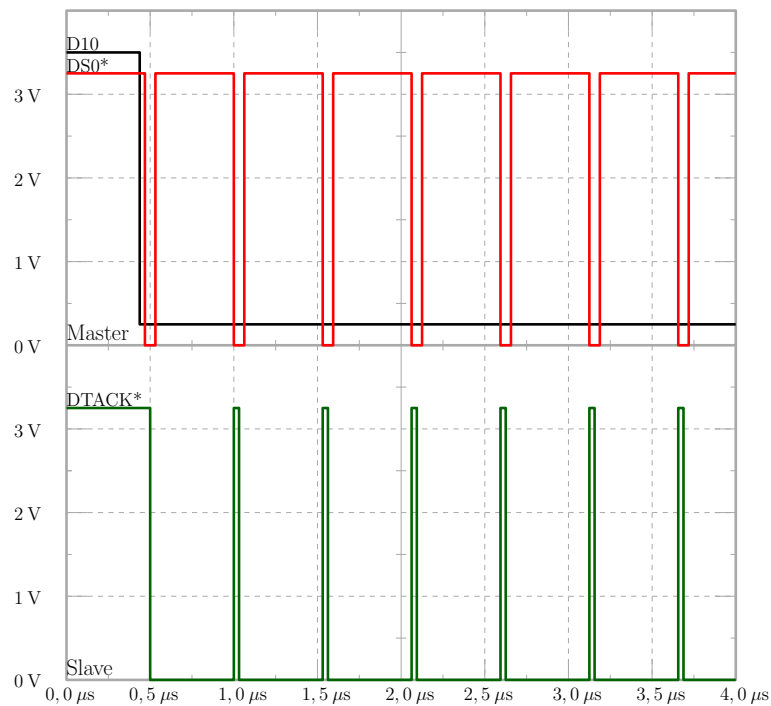


Abbildung 6.13: Die Übertragung von mehreren Datenworten im Blocktransfer, bei der die Datenworte im Controller-CPLD in einzelne bits aufgeteilt und weitergeleitet werden.

Bei einer Taktperiode im Controller-CPLD von $12,5 \text{ ns}$ beträgt die zur Verarbeitung notwendige Zeit $8 \cdot 12,5 \text{ ns} = 100 \text{ ns}$ pro Datenwort. Der Verlauf der Übertragung ist in Abbildung 6.14 zu sehen. Logisch Eins an $DTACK^*$ stellt sicher, dass der Master bis zur vollständigen Verarbeitung kein weiteres Datenwort sendet. Im Anschluss an einen einzelnen Transfer wird mit logisch Eins an $DS0^*$ das nächste Datenwort gesendet. Eine experimentell bestimmte Verzögerung von 4 Taktperioden bis zur Freigabe von $DTACK^*$ ist notwendig, um dem Zeitverhalten des Masters Rechnung zu tragen: Da dieser mit 10 MHz getaktet wird ist anzunehmen, dass er eine steigende Flanke an $DTACK^*$ nach bereits $4 \cdot 12,5 \text{ ns} = 50 \text{ ns}$ nicht erkennen kann. Erst in der darauffolgenden Taktperiode wird ein Signal an $DTACK^*$ erwartet. Abbildung 6.15 zeigt anhand von Geradensteigungen die unterschiedlichen Datenraten.

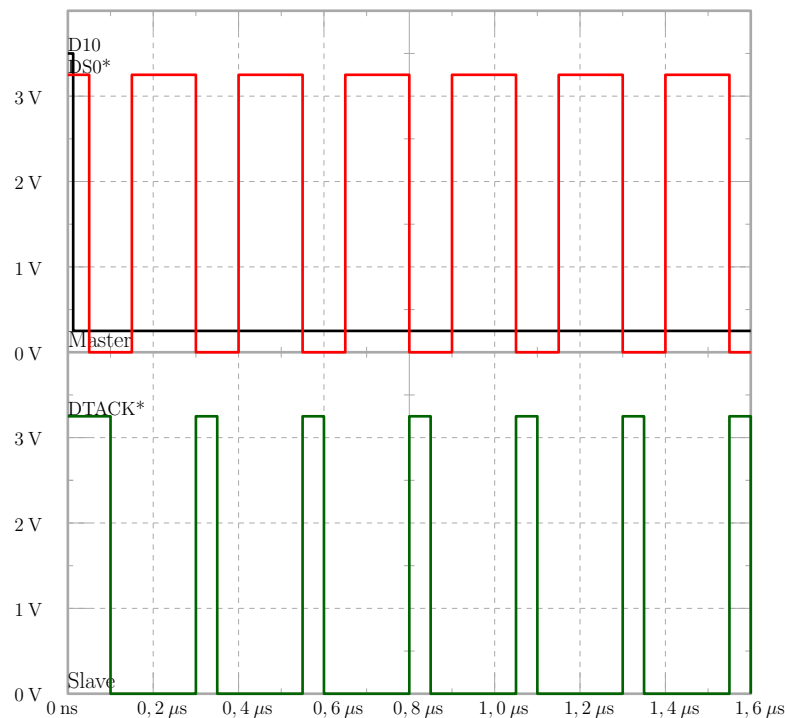


Abbildung 6.14: Die Übertragung von Datenworten im Blocktransfer, bei der die Datenworte im Controller-CPLD in einzelne Bytes aufgeteilt und weitergeleitet werden.

Ein Transfer von 256 Worten mit serieller Weiterleitung ist nach $t_{\text{block_serial}} = 138,80 \mu\text{s}$ abgeschlossen, während die Übertragung eines Datenblockes gleicher Größe mit einer Weiterleitung in Bytes lediglich $t_{\text{block_byte}} = 62,60 \mu\text{s}$ andauert. Mit der entwickelten Firmware werden in den einfachen Schreibroutinen folgende Datenraten erreicht:

$$f_{\text{single_serial}} = 6,9 \text{ MB/s} \quad (6.3)$$

$$f_{\text{single_byte}} = 14,3 \text{ MB/s.} \quad (6.4)$$

Die Datenraten im Blockmodus berechnen sich aus den gemessenen Zeiten zu:

$$f_{\text{block_serial}} = 7,4 \text{ MB/s} \quad (6.5)$$

$$f_{\text{block_byte}} = 16,4 \text{ MB/s.} \quad (6.6)$$

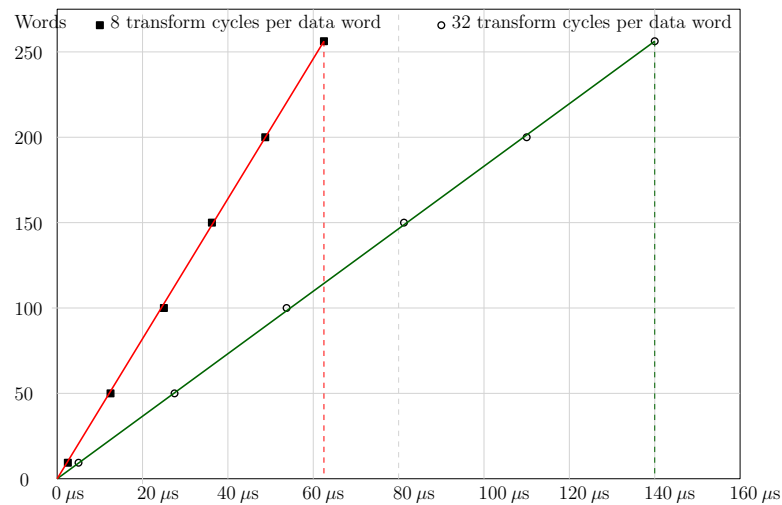


Abbildung 6.15: Die Übertragungszeit eines Datenblocks hängt von der Verarbeitungsdauer des angesprochenen Zustandes im Controller-CPLD ab. Werden die Datenworte nach dem Empfang seriell weitergeleitet, ist die Übertragung nach 138,80 μs beendet. Die Aufteilung der Daten in Bytes reduziert die Dauer der Übertragung auf 62,60 μs .

Die Datenrate bei Serialisierung der Worte wird durch die Verarbeitungszeit im Controller-CPLD bestimmt, während die Datenrate bei einer byteweisen Aufteilung der Worte durch die Taktrate des VME-Master limitiert ist. Von der theoretisch möglichen Übertragungsrate von $f_{\text{max}} = 40 \text{ MB/s}$ werden bis zu 41 % erreicht. Die Begrenzung der tatsächlichen Datenrate kann mit der geringen Taktfrequenz des SBC begründet werden. Das Setzen und Abfragen mehrerer Steuersignale zu versetzten Zeiten kann bei einer Taktung von 10 MHz ausschließlich innerhalb von Zeiten $\geq 200 \text{ ns}$ erfolgen, was einer Datenrate von $\leq 20 \text{ MB/s}$ entspricht.

6.3 Die Konfiguration der FPGAs über VME

Eine Konfigurationsdatei für die FPGAs wird mit der Entwicklungsumgebung *ISE* des Chipherstellers Xilinx erzeugt. Die Firmware des DSP-FPGA und die Firmware des QDR-FPGAs werden in dieser Reihenfolge in den *PROM File Formatter* der Software *iMPACT* geladen. Anschließend wird die Konfigurationsdatei im *PROM File Format* HEX erstellt. Dabei ist zu beachten, dass die Datei in unveränderter Bitfolge (*not swapped*) erstellt wird. Die Daten zur Konfiguration können über einfache Schreibroutinen oder in Paketen (blockweises Schreiben) an den Controller-CPLD übergeben werden. Dieser ist mit den Konfigurationsketten *Slave Serial* und *SelectMAP* verbunden. Die Daten werden im CPLD an die Breite des Datenbus der Konfigurationskette angepasst und synchron zum Konfigurationstakt an die FPGAs übermittelt. Zur Konfiguration stehen zwei Skripte zur Verfügung (siehe Tabelle 6.4), die einfache bzw. blockweise Datentransfers beinhalten. Dazu lesen die Skripte die angehängte Datei *2fpga.hex* ein und senden die Daten an ein GANDALF-Modul mit der hexadezimalen Identifikationsnummer ID. Die Größe einer GANDALF-Konfigurationsdatei beträgt mit der *ASCII*-Kodierung

Tabelle 6.4: Die Skripte zur Konfiguration einer GANDALF-Baugruppe über die VME-Schnittstelle. Bei (2) führt die große Anzahl der Datenworte in einer Routine zu einer Erhöhung der Datenrate.

	Befehl	Art der Übermittlung
1	<code>progga 2fpga.hex (ID)</code>	Einzelne 32 bit-Datentransfers
2	<code>blockwrite E0(ID)8000 2fpga.hex</code>	Blocktransfer mit 256 Datenworten zu je 32 bit

10,75 MB. Der Umfang der Datei ist unabhängig von den verwendeten Hardwareressourcen der FPGAs und generell gleich groß. Bei GANDALF erfolgt das Versenden der Daten über VME im hexadezimalen Format, sodass die effektive Größe der Datei mit 5,38 MB anzugeben ist. Eine Umwandlung der hostspezifischen Byteordnung des SBC in das Netzwerk-Format (*big endian*), welche das GANDALF-Modul verwendet, ist durch eine ebenfalls in den Skripten implementierte *host to network* Funktion realisiert.

Der Controller-CPLD empfängt die Daten bei beiden Transfermodi im selben Anfangszustand. Im Blockmodus wird zum erneuten Einlesen von Daten ein Zustandskreislauf durchlaufen, ohne dass eine Rückkehr in den Grundzustand erfolgen muss. Generell kann die Konfiguration sowohl über die *Slave Serial*- als auch über die *SelectMAP*-Kette mit beiden Programmen erfolgen. Die Datenrate ist bei der *Slave Serial*-Konfiguration durch die Verarbeitungsdauer im Controller-CPLD begrenzt, wohingegen die Dauer einer *SelectMAP*-Konfiguration von der Taktrate des SBC bestimmt wird.

6.3.1 Die Slave Serial Konfiguration

Die *Slave Serial*-Konfiguration ermöglicht eine serielle Konfiguration der FPGAs [45], die durch den Controller-CPLD gesteuert wird. Parallel verschaltete Steuerleitungen zwischen den FPGAs und eine gemeinsame Taktleitung garantieren eine zusammenhängende Programmierung. Der 80 MHz Takt des CPLDs wird mit einem intern erzeugten *Enable*-Signal als Konfigurationstakt verwendet. Zu Beginn wird der erste FPGA in der Kette programmiert. Nachdem dieser alle Konfigurationsdaten erhalten hat, leitet er den Datenstrom an den zweiten

FPGA weiter.

Das ebenfalls zum Ablauf der Konfiguration (siehe Abbildung 6.16) notwendige Setzen und Abfragen von Signalwerten wird von dem eingesetzten Skript vollständig gesteuert: Die Steuersignale *PROGRAM_B* und *INIT_B* sind *active-low*. Logisch Eins an *PROGRAM_B* setzt die FPGAs in einen Reset-Zustand. Die Antwort der FPGAs erfolgt mit logisch Eins an *INIT_B*. Eine Deaktivierung von *PROGRAM_B* versetzt die FPGAs in den Konfigurationszustand. Bei inaktivem *INIT_B* versendet der Controller-CPLD die Datenbits an *Slave D_IN* synchron zum Takt an *CCLK*. Die erfolgreiche Programmierung der FPGAs wird mit logisch Eins an *DONE* angezeigt. Damit ist die GANDALF-Einheit in vollem Umfang einsatzbereit. Mit der Firmware des Controller-CPLDs beträgt die Dauer der Konfiguration eines GANDALF-Moduls im *Slave Serial*-Modus

$$t_{\text{progga}} = 7,10 \text{ s} \quad (6.7)$$

$$t_{\text{blockwrite}} = 0,88 \text{ s.} \quad (6.8)$$

Diese Zeiten sind unabhängig vom Steckplatz im Trägergehäuse.

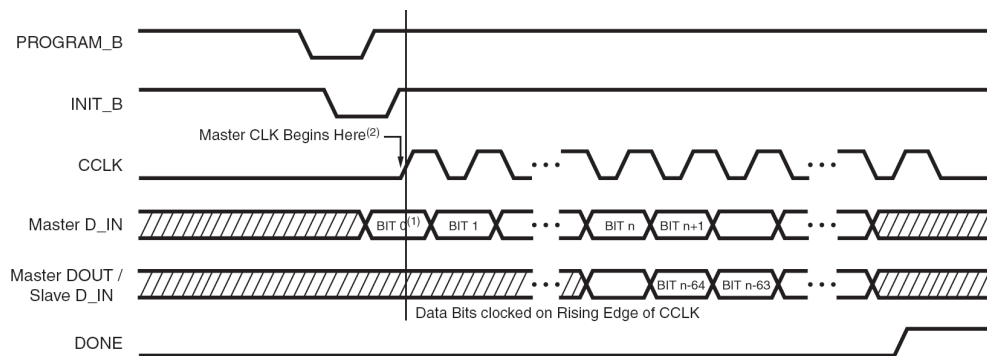


Abbildung 6.16: Die Programmierung der FPGAs im *Slave Serial*-Konfigurationsmodus [45]. Über VME-Befehle wird *PROGRAM_B* gesetzt und der Status von *INIT_B* und *DONE* abgefragt. Der Controller-CPLD taktet die Konfiguration mit dem maximal möglichen Takt von 80 MHz und sendet die Daten synchron zur steigenden Flanke an *CCLK* über *Slave D_IN* an die FPGAs.

6.3.2 Die SelectMAP Konfiguration

Die Steuerung der *SelectMAP*-Konfigurationsroutine wird ebenfalls vom Controller-CPLD ausgeführt. Der zeitliche Verlauf entspricht dem der *Slave Serial*-Konfiguration [45] (siehe Abbildung 6.16). Die Steuersignale *WRITE*, *INIT* und *PROGRAM* sind *active-low*. Mit einem VME-Befehl wird *PROGRAM* gesteuert, sodass damit die Konfiguration gestartet werden kann. Sobald *INIT* logisch Null erreicht hat, gibt der Controller-CPLD die 8 bit breiten Datenworte zusammen mit dem Takt (*CCLK*) an den Datenbus *DATA(7:0)* weiter. Im Gegensatz zu *Slave Serial* ist die 8 bit breite Datenleitung zwischen den beiden FPGAs parallel verschaltet. Auch eine Datenübertragung von einem FPGA zum Controller-CPLD ist möglich. Die Richtung des Datenstroms wird über das *RDWR_B*-Signal bestimmt, ein Chip wird über *CS_B* ausgewählt (siehe Abbildung 6.17).

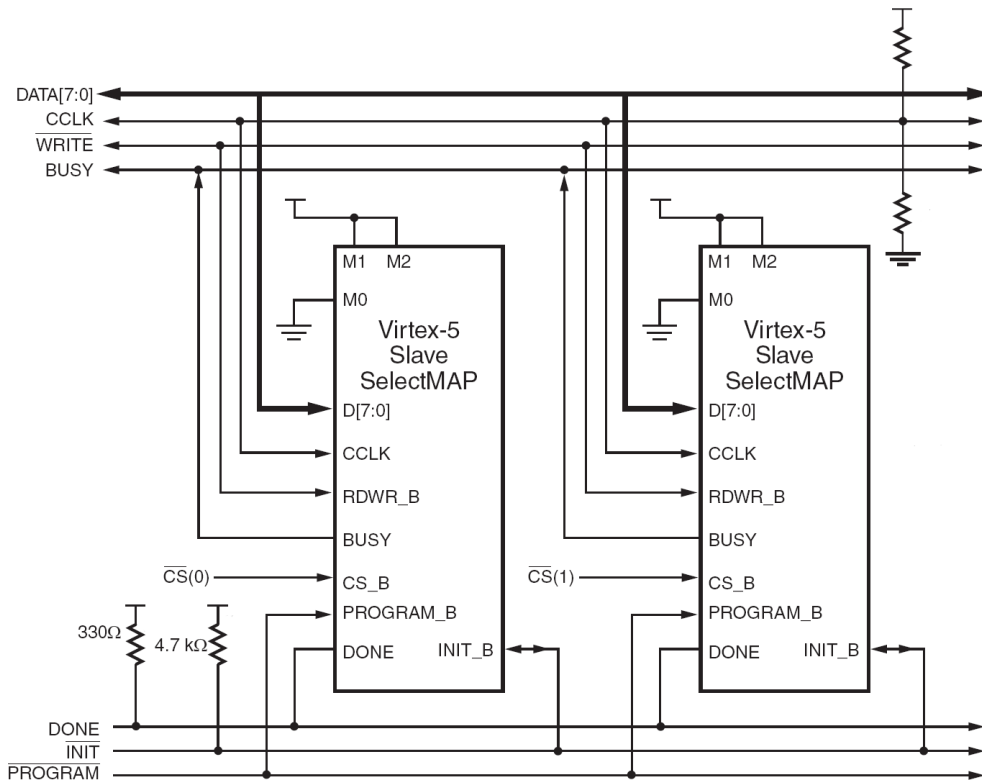


Abbildung 6.17: Die für eine Konfiguration über den SelectMAP-Modus notwendigen Daten-, Takt-, und Steuersignale [45]. Sämtliche Netze stehen in Verbindung mit dem Controller-CPLD, der die Konfiguration der beiden Virtex-5 FPGA steuert. Die beiden FPGAs werden hintereinander konfiguriert, eine Umschaltung der *Chip-Select* bits (CS_B) wird bei Ankunft eines in der Konfigurationsdatei enthaltenen eindeutigen Datenwortes im Controller-CPLD vorgenommen. Er gibt auch die für eine SelectMAP-Konfiguration notwendige Einstellung der M[2:0]-Pins vor, die aus der Abbildung hervorgeht.

Eine zusammenhängende Konfiguration beider FPGAs kann erfolgen, wenn die FPGAs im Zuge der Datenübertragung dynamisch aktiviert und deaktiviert werden. Die Datei beinhaltet das Datenwort ($0x500477F4$), das direkt im Anschluss der für den DSP-FPGA bestimmten Daten gesendet wird und das Ende der Konfiguration des ersten FPGAs signalisiert. Demzufolge können über die Abfrage der am Controller-CPLD übertragenen Worte die Werte der beiden CS -Signale gesteuert werden. Sobald die Konfiguration des ersten FPGAs in der Kette abgeschlossen ist, kann ohne Unterbrechung des Datenstroms der QDR-FPGA programmiert werden. Die Zeit für eine vollständige Konfiguration eines GANDALF-Moduls im *SelectMAP*-Modus beträgt

$$t_{\text{progga}} = 6,8 \text{ s} \quad (6.9)$$

$$t_{\text{blockwrite}} = 0,44 \text{ s}, \quad (6.10)$$

unabhängig vom Steckplatz im Trägergehäuse.

6.4 Die SystemACE-Umgebung

Ein Einsatz von GANDALF ohne aktive Schnittstellen zwischen Baugruppe und Anwender (VME) sowie zwischen Baugruppe und Speichereinheiten (S-Link) ist durch die Konfigurationsumgebung *SystemACE* (ACE: *Advanced Configuration Environment*) sichergestellt. Sie beinhaltet einen SystemACE-Controller, eine Compact-Flash-Karte und zwei JTAG-Schnittstellen. Die Steuerung von SystemACE geht vom Controller-CPLD aus, der Befehle und Daten an den SystemACE-Controller sendet. Ohne VME-Schnittstelle erfolgt die Konfiguration automatisch nach Einstecken der Compact-Flash Karte in das für sie vorgesehene Steckfach, wodurch eine auf der Karte gespeicherte Konfigurationsdatei geladen wird. Ist die GANDALF-Einheit mit einer VME-Schnittstelle verbunden, können im Rahmen einer VME-Schreibroutine mit der entsprechenden Ordernummer verschiedene Konfigurationsdateien ausgewählt werden (siehe Tabelle 6.2). Ein weiterer Befehl führt die Speicherung von Daten auf der Compact-Flash Karte aus. Mit der eingebauten Zustandsmaschine ist eine zusammenhängende Datennahme von 131 KB möglich. Der zur Zwischenspeicherung der Daten eingesetzte Fifo muss bis zum erneuten Erreichen der Schreibroutinen für $75 \cdot 30 \text{ ns} = 2,25 \mu\text{s}$ auf *write disabled* gesetzt werden.

6.4.1 Die Konfiguration der FPGAs

Die Verzeichnisstruktur der eingesetzten Compact-Flash Karte geht aus Abbildung 6.18 hervor. Jeder einzelne Ordner *cfgi* ($i = 0, \dots, 7$) enthält eine komplette Firmware-Version. Der Ordner der Konfigurationsdatei kann über *CFGADDR(2:0)* ausgewählt werden und ist mit (000) voreingestellt. Die einzelnen Verzeichnisse werden in der Datei *xilinx.sys* über *dir = ML50X* sowie *cfgaddr0 = cfg0* angegeben. Die Datei ist an oberster Stelle auf der Compact-Flash Karte gespeichert.

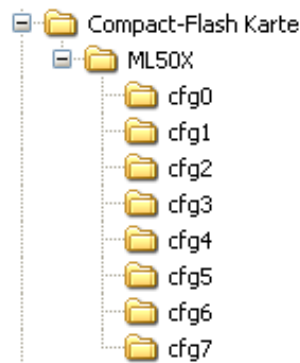


Abbildung 6.18: Die Verzeichnisstruktur auf der Compact-Flash Karte. Jedes Verzeichnis *cfg0* bis *cfg7* kann eine Firmware-Version enthalten.

Zur Konfiguration versetzt der Controller-CPLD den SystemACE-Controller mit logisch Eins an *CFGMODEPIN* in den Konfigurationsmodus. Die Konfiguration startet nach einem Reset an *RESET_B*. Sobald eine *Compact-Flash* Karte über das Steckfach mit GANDALF verbunden worden ist, führt der Controller-CPLD die Konfiguration mit dem Setzen dieser Werte aus. Standardmäßig wird so die in *cfg0* gespeicherte Datei geladen. Der Controller-CPLD ermöglicht zudem eine Konfiguration der FPGAs über SystemACE, die von einem VME-Befehl

eingeleitet wird. So kann der Anwender die Konfiguration mit Dateien aus einem der acht möglichen Ordner vornehmen. Hier werden aus der Adresse und dem Datenwort die Werte der Mode-Pins und der Ordneradressen ermittelt. Die Konfiguration wird mit der Datei aus dem angegebenen Ordner unmittelbar nach der Dekodierung der VME-Adresse durchgeführt. Dazu übergibt der aktivierte Zustand die bits $D(2:0)$ des Datenwortes an den SystemACE-Controller.

Eine rot blinkende Status-LED (*LED1*) zeigt den Konfigurationsprozess an. Eine fehlende Konfigurationsdatei oder ein Fehler in der Datenübertragung wird durch eine rot leuchtende Error-LED (*LED2*) angezeigt. Die hierbei eingesetzte JTAG-Konfigurationskette weist einen Datendurchsatz von 16,7Mbit/s auf, die Grösse der Konfigurationsdatei beträgt 5,38 MB. Demnach ist der Prozess nach ca. 2,5 s abgeschlossen und die FPGAs sind einsatzbereit. Dies wird durch eine dauerhaft rot leuchtende *Status-LED* angezeigt. *LED1* und *LED2* befinden sich auf der Baugruppe direkt neben dem SystemACE-Controller.

6.4.2 Datenspeicherung auf Compact-Flash

Die SystemACE-Spezifikation [46] sieht das Schreiben von Daten auf die Compact-Flash Karte vor. Dabei werden 8 bzw. 16 bit breite Datenworte an einzelne Sektoren des Flash-Speichers übertragen. Ein Sektor fasst 512 Bytes und ist einzeln adressierbar. Die Kommunikation zwischen Controller-CPLD und SystemACE-Controller erfolgt durch das Senden von Adressen und Daten in einzelnen Lese- und Schreibroutinen zusammen mit den drei Steuersignalen *MPCE*, *MPWE* und *MPOE*. Das Setzen der Signale erfolgt synchron zu einem 33 MHz-Takt, mit dem sowohl der SystemACE-Controller getaktet wird als auch der für eine Kommunikation mit SystemACE zuständige Bereich im Controller-CPLD. Ein Datenwort *MPD* wird über die in Abbildung 6.19 gezeigte Abfolge von Adress-, Daten- und Steuersignalen in ein Register übertragen (siehe auch [47]). Die Steuersignale sind active-low und werden in einem einzelnen Schreibroutinen wie folgt gesetzt: Logisch Eins an *MPCE* aktiviert den SystemACE-Controller. Um Daten zu schreiben, wird *MPWE* auf logisch Eins gesetzt. Logisch Null an *MPOE* stellt sicher, dass keine Daten am Ausgang des Controllers anliegen.

Abbildung 6.20 zeigt einen abgeschlossenen Prozess zur Übertragung von Daten auf die Compact-Flash Karte. Dieser Transferzyklus ist in den Controller-CPLD implementiert und garantiert eine Übertragung von $256 \cdot 512 \text{ bytes} = 131,072 \text{ KB}$ auf die Speicherkarte. Mit einem parallel zu diesem Prozess gesteuerten Lesetakt (*rdclk*) werden Daten aus einem Fifo des DSP-FPGA ausgelesen und zum geeigneten Zeitpunkt in den Prozess eingefügt. Die Datenübertragung an die Compact-Flash Karte ist in mehrere einzelne Unterprozesse aufgeteilt, die sequenziell ablaufen (siehe Abbildung 6.20). Dabei werden die zur Datenübertragung erforderlichen Registerwerte gesetzt und abgefragt und die Datenworte werden an die Register des Datenspeichers gesendet. Diese Prozedur läuft wie folgt ab (siehe dazu auch Abbildung 6.20): Damit der Controller-CPLD die Compact-Flash Karte ansteuern kann, muss die Compact-Flash Karte auf eine Kommunikation mit dem CPLD vorbereitet werden (*CF-Lock*). Ihre Bereitschaft wird mit einem (*Ready for Command*) signalisiert. Im Anschluss daran wird dem SystemACE-Controller die Adresse des Sektors mitgeteilt, in den die ersten 512 Bytes der Daten geschrieben werden sollen. Diese 28 bit breite Adresse wird auch als logische Blockadresse (*LBA*) eines Sektors bezeichnet. Die Anzahl der zu beschreibenden Sektoren kann zwischen 0 und 256 variieren und wird in das Register *SECCNTCMDREG* geschrieben, an das auch das *WriteMemCardData*-Wort gesendet wird.

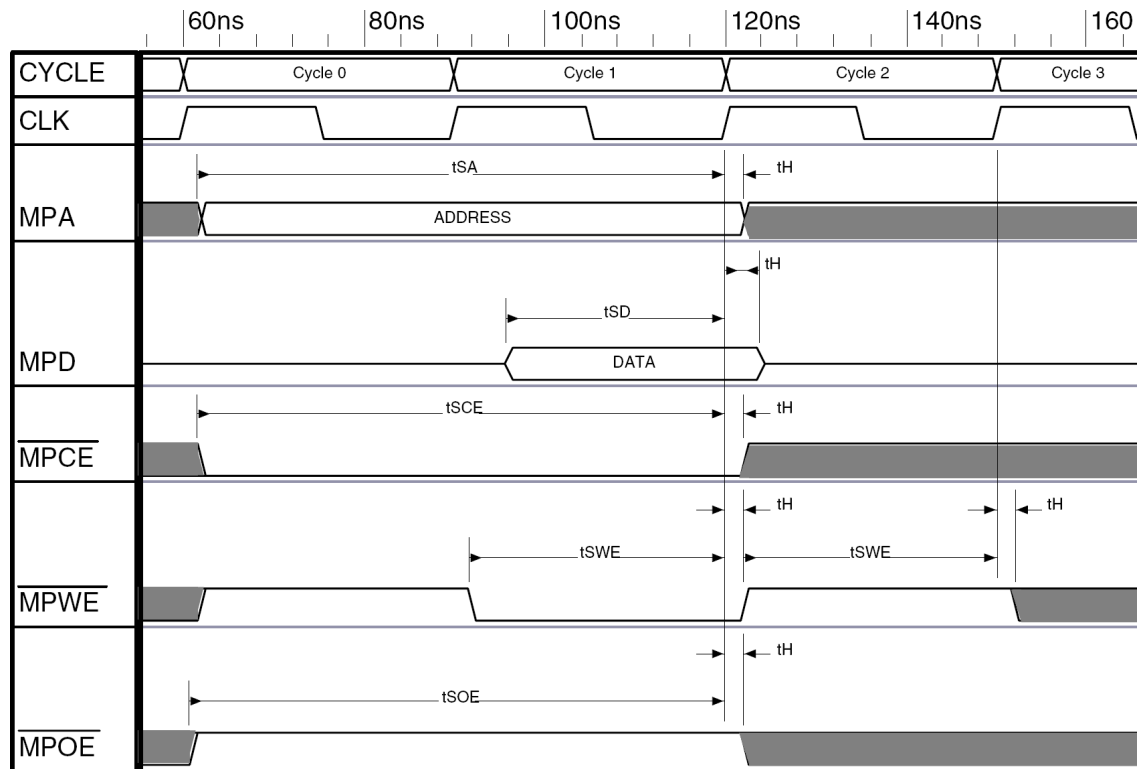


Abbildung 6.19: Die Übertragung eines Datenwortes an ein Register des SystemACE-Controllers [46]. Bei GANDALF beträgt die Breite eines Datenwortes 16 bit. In 5 bit breite Register werden neben den Datenworten auch Werte übertragen, durch die die Datenübertragung an die Compact-Flash Karte gesteuert werden kann.

Nach einem *Reset* des *CFGJTAG*-Controllers wird die zu übertragende Datenmenge im CPLD in jeweils 32 Byte große *Data Buffer* aufgeteilt und an die Controller-Einheit versendet. Die Daten werden dabei über einen 16 Worte breiten Adressbereich an den Fifo gesendet, indem nach dem Versenden eines Datenwortes im Controller-CPLD die Adresse um ein bit inkrementiert wird. Ein Paket von 32 Bytes entspricht dem Fassungsvermögen des SystemACE-Fifos, der nach Erhalt des letzten Datenwortes seinen gesamten Inhalt an die Compact-Flash Karte überträgt. Sind weitere *Data Buffer* Pakete vorhanden, so wird der *Write Data Buffer*-Zyklus erneut ausgeführt. Am Ende der Datenübertragung wird der Wert des *CFGRESET*-bits und des *LOCKREQ*-bits in dem entsprechenden Register gelöscht. Somit ist das Reset des *CFGJTAG*-Controllers aufgehoben und die Compact-Flash Karte ist freigegeben.

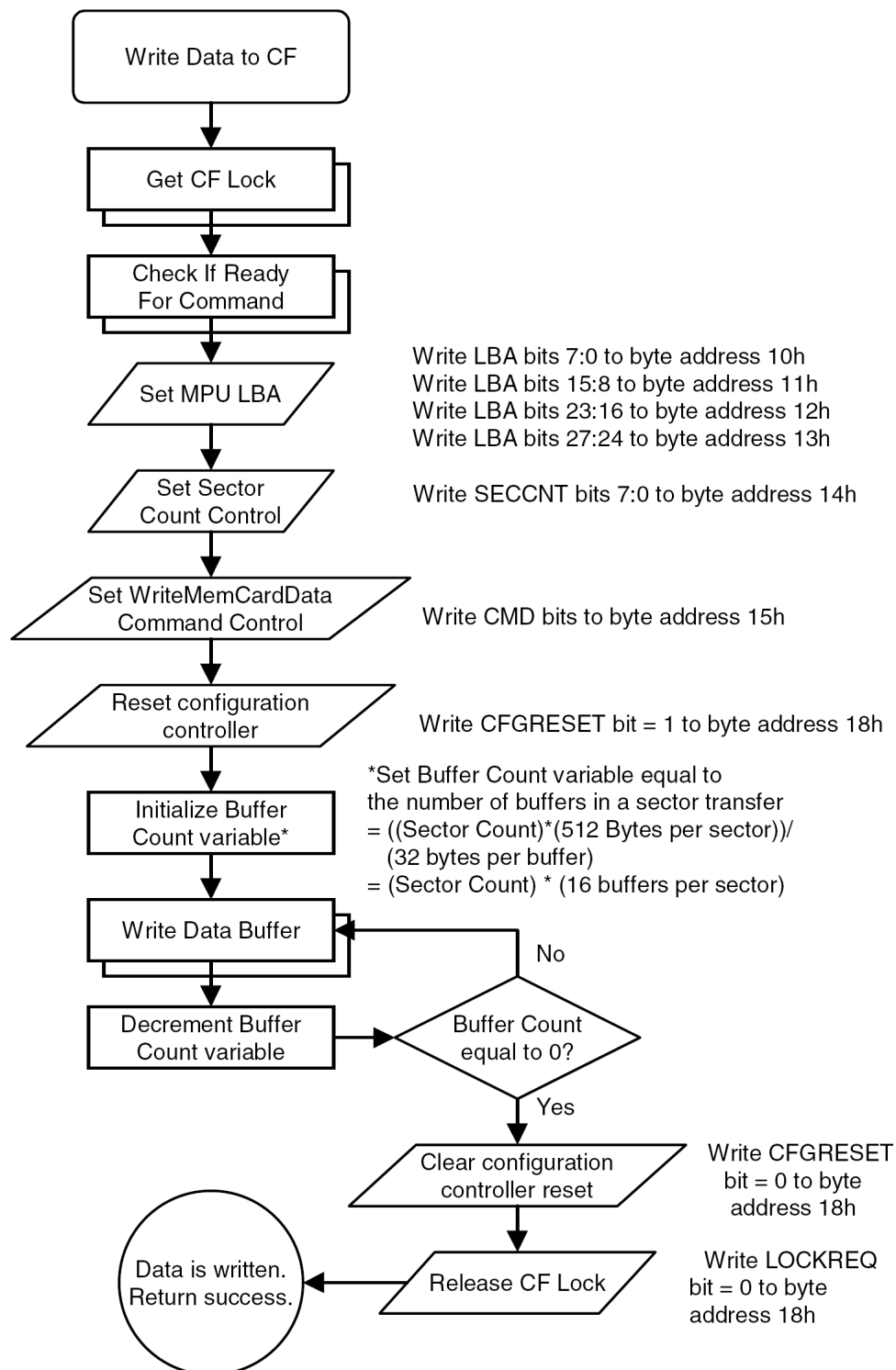


Abbildung 6.20: Der im Controller-CPLD ablaufende Transferzyklus zur Übertragung von Experimentdaten an die Compact-Flash-Karte [46]. Die Daten werden auf der Speicherkarte zu Beginn des über die logische Blockadresse (LBA) adressierten Sektors geschrieben und fortlaufend über die in *SECCNT* angegebene Anzahl von Sektoren gespeichert.

Die im Controller-CPLD der GANDALF-Baugruppe enthaltene Routine beschreibt 256 Sektoren, beginnend mit dem 145000. Sektor. In binärer Kodierung stellt diese Nummer die logische Block-Adresse *LBA* dar. Mit einem Editor-Programm zur Darstellung der Sektorinhalte kann die Speicherung der Daten verifiziert werden (siehe Abbildung 6.21). Die Datenrate der

Offset(h)	00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E	
046CCFE0	8AFA	87E0	17C3	CB9B	AF01	DDEA	3790	BB5B	DECA	3C90	0609	0A71	9E7B	726A	6A34	9A46	
046CD000	099D	0E25	0E52	0E55	0D3A	0C1A	0BB4	0981	036B	060C	04C0	0320	027D	0161	014F	0284	Sektor 145000
046CD020	03A7	04E6	0687	0847	0A85	0B1F	0C04	0DED	0EF1	0E68	0D6D	0C07	0BB1	0A24	08FD	0619	
046CD040	0120	039B	0205	01FB	01D0	0246	03A3	045F	0160	0724	09EB	0B14	0828	0D1D	0EC2	0E7C	
046CD060	0E0A	0D6D	0C30	0A38	092E	0741	0504	0411	02CA	0221	01EC	0214	02DC	0364	05C0	0700	
046CD080	0859	0A7D	0C88	0D44	0E08	0E7D	0E5A	0D38	0C1D	0B4C	09AE	0761	02A5	0490	0324	025A	
046CD0A0	0143	0171	02E8	0371	006E	0661	0888	0964	0B2C	0CDF	0984	0E65	0ED8	0E01	0D37	0B52	
046CD0C0	0A18	0880	0641	051D	0382	0223	02A6	01DB	0200	030E	046E	054C	0707	094C	0AC4	0C68	
046CD0E0	09A7	0E46	0EC8	0E38	0D7A	0C50	0A7E	0921	03C3	052D	0421	02F9	022F	01D8	022B	02B6	
046CD100	034F	053E	06CC	0828	0AEO	0BE0	0D23	0E09	0E75	0E64	0D70	0C41	0BCE	09C6	0822	0641	
046CD120	0006	035C	02E9	016D	016E	026C	03EF	042B	025D	0806	0946	0B6D	08C3	0D48	0ECD	0E64	
046CD140	0E1E	0D34	0B4F	0A59	08AC	0664	05A5	03C9	0233	020B	01E2	0220	02E0	0429	05BF	0764	
046CD160	09AA	0A55	0C68	0D7D	0E12	0E79	0E68	0D0D	0C41	0AE3	096E	0780	0144	0441	0388	023D	
046CD180	014B	0204	02A4	0336	01A6	06C3	081D	0A42	0BC3	0D1B	0E93	0E71	0E44	0D59	0C65	0B81	
046CD1A0	09DC	0821	06E5	0445	0361	0279	0171	0168	02F6	0343	041B	0627	07D3	0925	0BE2	0CAC	
046CD1C0	090B	0E5C	0EFF	0E19	0D71	0C01	0A02	0847	0384	0555	0344	0245	02A5	01DE	0214	02E7	
046CD1E0	0022	058E	0741	0906	0A00	0C39	0D6F	0E29	0E64	0E4A	0D3C	0C00	0B1E	0968	0704	0568	
046CD200	04E1	0321	0260	0162	014D	0293	03BB	04F8	069F	0862	0A9F	0B32	0B0C	0DF3	0E44	0E6B	Sektor 145001

Abbildung 6.21: Mit dem Transientenrekorder GANDALF erhaltene Experimentdaten, die auf der Compact-Flash Karte abgespeichert wurden. Daten einzelner Sektoren werden mit einem Hex-Editor angezeigt. In diesem Beispiel kommt der HxD-Hexeditor [48] zum Einsatz.

Schreibroutine berechnet sich folgendermaßen:

Für einen Transfer von 2 Bytes werden 8 Taktzyklen benötigt. Zudem werden pro *Data Buffer* weitere 8 Zyklen für die Statusabfrage des Fifos und das Inkrementieren der Bufferzahl benötigt. Somit werden Daten vom Controller-CPLD zur Compact-Flash Karte mit der effektiven Transferrate von

$$f_{\text{Write_on_CF}} = 7,7 \text{ MB/s} \quad (6.11)$$

übertragen.

6.5 Die I²C-Schnittstelle

Über I²C werden die Register des Power Supply Sequencers *UCD9081* regelmäßig in einstellbaren Zeitabständen durch den Controller-CPLD ausgelesen. Damit diese Informationen über VME an den Anwender übertragen werden können, werden sie im Controller-CPLD zwischengespeichert und können mit VME-Befehlen ausgelesen werden.

Die serielle I²C Schnittstelle wurde in den 1990er Jahren entwickelt. Sie ermöglicht eine serielle Verbindung und damit eine Datenübertragung zwischen einzelnen Mikrokontrollern nach dem *Master/Slave*-Prinzip (*Inter-IC*). Der Bus besteht aus zwei bidirektionalen Leitungen (siehe Abbildung 6.22): Wenn das Taktsignal an *SCL* auf Logisch Null steht werden einzelne Adress- und Datenbits über die Datenleitung *SDA* gesendet. Die Daten sind während einer '1' an *SCL* gültig und während dieser Zeit konstant (zur Programmierung der Schnittstelle siehe auch [47]). Die Übertragung von Adressen und Daten erfolgt bei 200 KHz in Paketen von 8 bits.

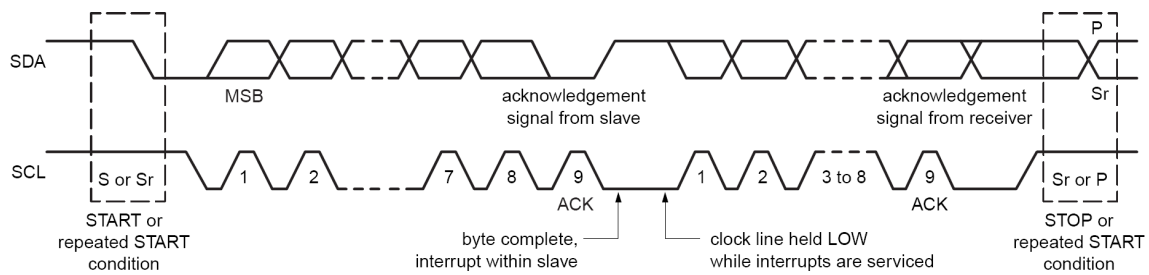


Abbildung 6.22: Serielle Übertragung von Adressen und Daten über den I²C-Bus in Datenpaketen von jeweils 8 bits [49]. Der Sender erwartet eine Bestätigung des Empfängers an *ACK* nach jedem Byte.

Die Übertragung von den einzelnen 7 bit-Adressen wird generell von einem *write*-bit begleitet. Zudem sind sämtliche Übertragungen an *START* und *STOP*-Bedingungen geknüpft. Der Empfänger teilt dem Sender über ein *ACK*-Signal den vollständigen und korrekten Empfang eines Bytes mit.

Ein Register des *UCD9081* enthält ein 8 bit breites Datenwort. Der Spannungswert eines Netzes i ($i = 1, \dots, 8$) setzt sich aus den Werten von jeweils zwei Registern *RAIL(i)H* und *RAIL(i)L* zusammen [35]. Über die Adressierung des Registers *RAIL(i)H* wird durch automatisches Inkrementieren im *UCD9081* das Register *Rail(i)L* ebenfalls erreicht. Die Ansteuerung des *UCD9081* und die Datenübertragung zum Controller-CPLD erfolgt im *I²C Combined Format*, in dem zu Beginn die Chip-Adresse *0x6F* zusammen mit der Registeradresse an den *UCD9081* gesendet wird (siehe Abbildung 6.23).

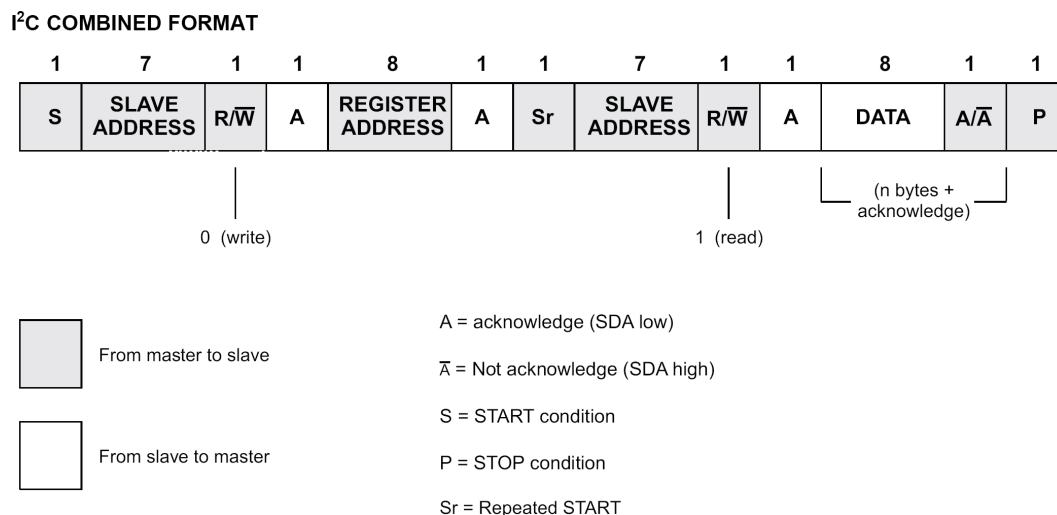


Abbildung 6.23: Das kombinierte Auslesen mehrerer Register des *UCD9081* [35]. Durch die Ansteuerung eines Registers wird die Adresse automatisch um eins erhöht. Dadurch können Daten von Registern ausgelesen werden, ohne dass die Register einzeln angesteuert werden müssen.

Kapitel 7

Zusammenfassung

Generalisierte Partonverteilungen geben Aufschluss über die Drehimpulse der Partonen in Nukleonen. Durch die Erweiterung des COMPASS-Spektrometers mit einem Rückstoß-Detektor sollen in naher Zukunft GPDs aus der tiefvirtuellen Compton-Streuung sowie aus der harten exklusiven Meson-Produktion erhalten werden. Dazu wird der GANDALF-Transientenrekorder die Signale des Detektors verarbeiten und Triggersignale setzen.

Im Rahmen dieser Diplomarbeit wurde ein Hardwaredesign entwickelt, mit dem ein CPLD den Einsatz von GANDALF als eigenständiges Modul im COMPASS-Datennahmesystem sicherstellt. Für die Ansteuerung der einzelnen Baugruppen wurde eine VME-Schnittstelle implementiert. Sie ermöglicht eine befehlsgesteuerte Übertragung von Datenworten zwischen Anwender und Modul. Zur Konfiguration weiterer intelligenter Bausteine wurde die Steuerung von zwei Konfigurationsketten in den CPLD integriert. Außerdem ist er an eine I²C-Schnittstelle angebunden, über die Statusinformationen wie die Registerinhalte eines Spannungsprüfers ausgelesen und Taktfrequenzen eingestellt werden. Ein flexibler Einsatz von GANDALF wurde mit der Konfigurationsumgebung *SystemACE* in Verbindung mit einer Compact-Flash-Karte sichergestellt: Die Speicherkarte kann mehrere Firmware-Versionen enthalten, die zur Konfiguration des Moduls eingesetzt werden können und eine Speicherung von Messdaten und Boardinformationen ist ebenfalls möglich.

Während in einfachen VME-Lese- und Schreibroutinen ein Datenwort pro Befehl übertragen wird, sind im VME-Blockmodus bis zu 256 Worte in einem Transferzyklus enthalten. Die im Blockmodus erzielten Datenraten führen zu einer verringerten Belegungsdauer des VME-Bus:

Das blockweise Auslesen von Daten einer GANDALF-Einheit erhöht die Datenrate um 34 % auf

$$f_{\text{block_read}} = 19,1 \text{ MB/s.}$$

Die Rate, mit der Daten an eine GANDALF-Einheit übertragen werden, ist im Blockmodus um 15 % erhöht und beträgt

$$f_{\text{block_write}} = 16,4 \text{ MB/s.}$$

Eine Konfigurationsdatei umfasst generell 5,38 MB und wird entweder über einfache Schreibroutinen oder im Blocktransfer an die GANDALF-Baugruppe übertragen. Die minimale Zeit

für eine serielle Konfiguration der FPGAs beträgt

$$t_{\text{serial}} = 0,88 \text{ s.}$$

Die Konfiguration über einen 8 bit breiten Datenbus dauert bei maximalen Datenraten

$$t_{\text{byte}} = 0,44 \text{ s}$$

an, womit die Möglichkeiten der eingesetzten VME-CPU ausgeschöpft wurden.

Die Digitalisierungseinheit des Transientenrekorders enthält ADCs, die gemäß den Herstellerangaben bei Eingangsfrequenzen von $f_{\text{in}} \leq 300 \text{ MHz}$ und maximaler Abtastrate eine effektive Digitalisierungsbreite von $\text{ENOB}_{\text{spec}} = 10,4 \text{ bit}$ aufweisen. Mit Sinussignalen konnte die Güte der Digitalisierungseinheit $\text{ENOB}_{\text{meas}}$ bestimmt werden: Der Wert $\text{ENOB}_{\text{meas}} = 10,1 \text{ bit}$ zeigt, dass der GANDALF-Transientenrekorder sehr rauscharm arbeitet.

Mit den über die VME-Routinen ansteuerbaren Registern und den verschiedenen Konfigurationsmöglichkeiten sowie den internen Schnittstellen ist die GANDALF-Baugruppe für den Einsatz am COMPASS Experiment gerüstet. Der Anwender hat stets die Möglichkeit, das Modul mit der aktuellsten Firmware-Version zu konfigurieren, Informationen über den Status des Moduls einzuholen und Messdaten anzufordern. Somit ermöglicht das GANDALF-Modul in Verbindung mit der Digitalisierungseinheit und den Algorithmen zur Signalverarbeitung die Detektierung von Protonen mit dem COMPASS-RPD. Die daraus resultierenden Erkenntnisse werden in naher Zukunft zu einem tiefgehenden Verständnis der Spinstruktur des Nukleons führen und die Frage nach weiteren Spinbeiträgen zum Gesamtspin des Nukleons beantworten.

A.2 Die schematische Übersicht

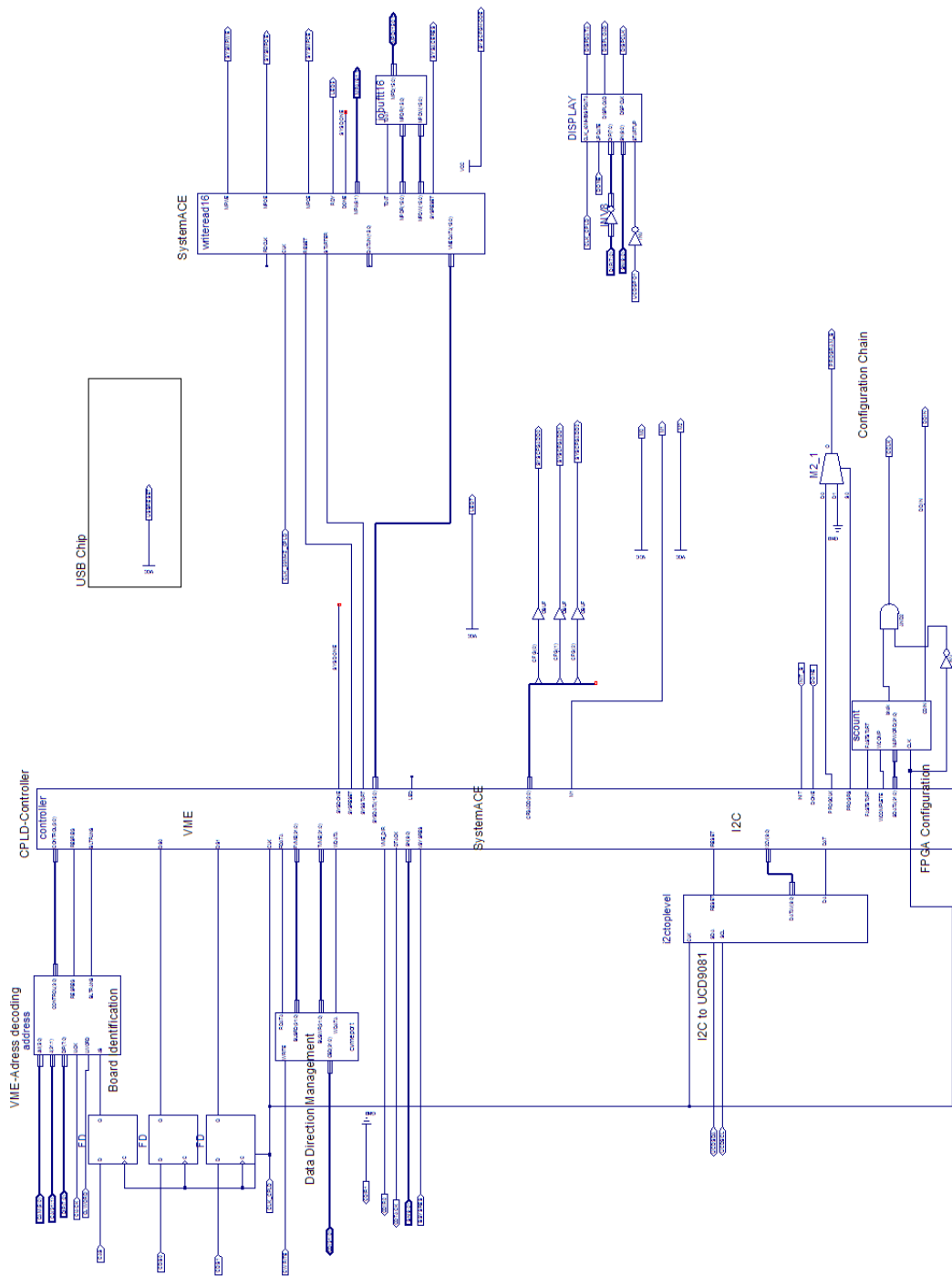


Abbildung A.2: Das Toplevel des CPLD-Controllers.

A.3 Der Controller

Listing A.1: Der Controller

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;

6  entity controller is
7      Port ( CONTROL : in  STD_LOGIC_VECTOR (3 downto 0);
8            DSO : in  STD_LOGIC;
9            DS1 : in  STD_LOGIC;
10           DTACK : out STD_LOGIC;
11           RESREG : out STD_LOGIC;
12           INIT : in STD_LOGIC;
13           DONE : in STD_LOGIC;
14           PROGCLK : out STD_LOGIC;
15           PROGRS : out STD_LOGIC;
16           SDATA : out STD_LOGIC_VECTOR(31 downto 0);
17           FASTSTART : out STD_LOGIC;
18           SYSDATA : in STD_LOGIC_VECTOR(15 downto 0);
19           SYSRESET : out STD_LOGIC;
20           SYSSTART : out STD_LOGIC;
21           SYSDONE : in STD_LOGIC;
22           WCOMPLETE : in STD_LOGIC;
23           CLK : in STD_LOGIC;
24           CFGADD : out STD_LOGIC_VECTOR(2 downto 0);
25           M1 : out STD_LOGIC := '0';
26           I2CV : in STD_LOGIC_VECTOR(9 downto 0);
27           DAT : in STD_LOGIC;
28           IRESET : out STD_LOGIC;
29           LED : out STD_LOGIC;
30           RDATA : out STD_LOGIC;
31           WDATA : out STD_LOGIC;
32           VME_DIR : out STD_LOGIC; --1 place data on bus
33           TVME : out  STD_LOGIC_VECTOR (31 downto 0);
34           FVME : in  STD_LOGIC_VECTOR (31 downto 0);
35           BLTRANS : in STD_LOGIC;
36           VSYSRES : in STD_LOGIC;
37           SN : in STD_LOGIC_VECTOR(9 downto 0)
38           );
39  end controller;

41  architecture Behavioral of controller is

43      constant RESET_PROG : std_logic_vector(7 downto 0) := X"80"; --high
44      constant SET_PROG : std_logic_vector(7 downto 0) := X"40"; --low

46      signal daco : integer range 7 downto 0;
47      signal i2c1, i2c2 : std_logic_vector(29 downto 0);
48      signal i2c3 : std_logic_vector(19 downto 0);
49      signal ass : std_logic;
50      signal SYSRES : std_logic;

```

```

51  signal modeset: std_logic_vector(1 downto 0):="00";
52  signal bcount: unsigned(7 downto 0):=(others => '0');
53  signal icount: unsigned(23 downto 0):=(others => '0');
54  signal mcount: integer range 0 to 4096;

56  constant id_read: std_logic_vector(3 downto 0):="1000";
57  constant up: std_logic_vector(23 downto 0) := X"FFFFFF";
58  constant zero: unsigned(27 downto 0) := (others => '0');

61  TYPE stypes is (idle,idle1,idread,idread1,bstatus,bstatus1,oe,oe1,oelf,oe2,oe2f,
    bwrite0,bwrite1,bwrite2,bwrite3,bread0,bread1,i2c00,i2c01,i2c10,i2c11,i2c20,
    i2c21,syswr,syswr1,syswr2);

63  signal thisstate,thatstate: stypes;

65  begin

67  SYSSTART<='0';
68  SYSRESET<=SYSRES;

70  mach: process(thisstate,CONTROL,DS0,DS1,SN,FVME,INIT,DONE,WCOMPLETE,BLTRANS,i2c1
    ,i2c2,i2c3,bcount,SYSDONE,SYSDATA) is
71  begin
72      SYSRES<='0';
73      modeset<="00";
74      VME_DIR<='0';
75      RDATA<='0';
76      WDATA<='0';
77      DTACK<='1';
78      RESREG<='0';
79      PROGCLK<='1';
80      PROGRS<='0';
81      SDATA<=FVME;
82      FASTSTART<='1';
83      TVME<=X"00000000";
84      thatstate<=thisstate;

86  case thisstate is

88  when idle1 =>
89      SYSRES<='0';
90      modeset<="00";
91      VME_DIR<='0';
92      RDATA<='0';
93      WDATA<='0';
94      DTACK<='1';
95      RESREG<='0';
96      PROGCLK<='1';
97      PROGRS<='0';
98      SDATA<=FVME;
99      FASTSTART<='1';
100     TVME<=X"00000000";
101     if ((DS0='1' and DS1='1') and CONTROL="0000") then

```

```

102     thatstate<=idle;
103     end if;
104     when idle =>
105         SYSRES<='0';
106         modeset<="00";
107         VME_DIR<='0';
108         RDATA<='0';
109         WDATA<='0';
110         DTACK<='1';
111         RESREG<='0';
112         PROGCLK<='1';
113         PROGRS<='0';
114         SDATA<=X"00000000";
115         FASTSTART<='1';
116         TVME<=X"00000000";
117         --ENTERING STATES
118         if (DS0='0' or DS1='0') then
119             case CONTROL is
120                 when "1000" =>
121                     thatstate<=idread;
122                 when "0011" =>
123                     thatstate<=bstatus;
124                 when "0001" =>
125                     thatstate<=oe;
126                 when "0110" =>
127                     thatstate<=bwrite0;
128                 when "1110" =>
129                     thatstate<=bread0;
130                 when "1001" =>
131                     thatstate<=i2c00;
132                 when "1010" =>
133                     thatstate<=i2c10;
134                 when "1011" =>
135                     thatstate<=i2c20;
136                 when "0100" =>
137                     thatstate<=syswr;
138                 when others =>
139                     thatstate<=idle1;
140             end case;
141         end if;
142     when idread =>
143         SYSRES<='0';
144         modeset<="11";
145         VME_DIR<='1';
146         RDATA<='1';
147         WDATA<='0';
148         RESREG<='0';
149         PROGCLK<='1';
150         PROGRS<='0';
151         DTACK<='0';
152         SDATA<=FVME;
153         FASTSTART<='1';
154         TVME(31)<='1';
155         TVME(30)<='1';

```

```

156     TVME(29) <= '1';
157     TVME(28) <= '1';
158     TVME(27) <= '1';
159     TVME(26) <= '1';
160     TVME(25) <= '1';
161     TVME(24 downto 19) <= "000000";
162     TVME(18) <= '1';
163     TVME(17 downto 10) <= "00000000";
164     TVME(9 downto 0) <= not SN;
165     if ((DS0='1' and DS1='1') and CONTROL="1000") then
166         thatstate <= idread1;
167     end if;
168     when idread1 =>
169         SYSRES <= '0';
170         modeset <= "00";
171         VME_DIR <= '1';
172         RDATA <= '1';
173         WDATA <= '0';
174         DTACK <= '1';
175         RESREG <= '1';
176         PROGCLK <= '1';
177         PROGRS <= '0';
178         SDATA <= FVME;
179         FASTSTART <= '1';
180         TVME(31) <= '1';
181         TVME(30) <= '1';
182         TVME(29) <= '1';
183         TVME(28) <= '1';
184         TVME(27) <= '1';
185         TVME(26) <= '1';
186         TVME(25) <= '1';
187         TVME(24 downto 19) <= "000000";
188         TVME(18) <= '1';
189         TVME(17 downto 10) <= "00000000";
190         TVME(9 downto 0) <= not SN;
191         if ((DS0='1' and DS1='1') and CONTROL="0000") then
192             thatstate <= idle;
193         end if;
194     when bstatus =>
195         SYSRES <= '0';
196         modeset <= "00";
197         VME_DIR <= '1';
198         RDATA <= '1';
199         WDATA <= '0';
200         DTACK <= '0';
201         RESREG <= '0';
202         PROGCLK <= '1';
203         PROGRS <= '0';
204         SDATA <= FVME;
205         FASTSTART <= '1';
206         TVME(31 downto 24) <= "00000000";
207         TVME(23) <= INIT;
208         TVME(22) <= DONE;
209         TVME(21 downto 20) <= "00";

```

```

210     TVME(19 downto 0) <= X"00000";
211     if (DS0='1' and DS1='1' and CONTROL="0011") then
212         thatstate <= bstatus1;
213     end if;
214 when bstatus1 =>
215     SYSRES <= '0';
216     modeset <= "00";
217     VME_DIR <= '1';
218     RDATA <= '1';
219     WDATA <= '0';
220     DTACK <= '1';
221     RESREG <= '1';
222     PROGCLK <= '1';
223     PROGRS <= '0';
224     SDATA <= FVME;
225     FASTSTART <= '1';
226     TVME(31 downto 24) <= "00000000";
227     TVME(23) <= INIT;
228     TVME(22) <= DONE;
229     TVME(21 downto 20) <= "00";
230     TVME(19 downto 0) <= X"00000";
231     if (DS0='1' and DS1='1' and CONTROL="0000") then
232         thatstate <= idle;
233     end if;
234 when oe =>
235     SYSRES <= '0';
236     modeset <= "00";
237     VME_DIR <= '0';
238     RDATA <= '0';
239     WDATA <= '1';
240     DTACK <= '1';
241     RESREG <= '0';
242     PROGCLK <= '0'; -- 1
243     PROGRS <= '1'; -- 0
244     SDATA <= FVME;
245     FASTSTART <= '1';
246     TVME <= X"00000000";
247     if ((DS0='0' and DS1='0') and CONTROL="0001") then
248         thatstate <= oe1;
249     end if;
250 when oe1 =>
251     SYSRES <= '0';
252     modeset <= "00";
253     VME_DIR <= '0';
254     RDATA <= '0';
255     WDATA <= '1';
256     DTACK <= '0';
257     RESREG <= '0';
258     SDATA <= FVME;
259     FASTSTART <= '1';
260     TVME <= X"00000000";
261     PROGCLK <= '0';
262     PROGRS <= '1';
263     if ((DS0='1' and DS1='1') and CONTROL="0001") then

```

```

264     thatstate<=oe2;
265     end if;
266 when oe2 =>
267     SYSRES<='0';
268     modeset<="00";
269     VME_DIR<='0';
270     RDATA<='0';
271     WDATA<='1';
272     DTACK<='0';
273     RESREG<='0';
274     SDATA<=FVME;
275     FASTSTART<='1';
276     TVME<=X"00000000";
277     PROGCLK<='0';
278     PROGRS<='1';
279     if (FVME(7 downto 0)=SET_PROG and CONTROL="0001") then
280         thatstate<=oelf;
281     elsif (FVME(7 downto 0)=RESET_PROG and CONTROL="0001") then
282         thatstate<=oe2f;
283     else
284         thatstate<=idle;
285     end if;
286 when oelf =>
287     SYSRES<='0';
288     modeset<="00";
289     VME_DIR<='0';
290     RDATA<='0';
291     WDATA<='1';
292     SDATA<=FVME;
293     FASTSTART<='1';
294     TVME<=X"00000000";
295     PROGCLK<='0'; --f
296     PROGRS<='1';
297     DTACK<='1';
298     RESREG<='1';
299     if ((DS0='1' and DS1='1') and CONTROL="0000" and INIT='0') then
300         thatstate<=oe2; --not yet idle, prog has to stay low for another cycle
301     end if;
302 when oe2f =>
303     SYSRES<='0';
304     modeset<="00";
305     VME_DIR<='0';
306     RDATA<='0';
307     WDATA<='1';
308     DTACK<='1';
309     RESREG<='1';
310     SDATA<=FVME;
311     FASTSTART<='1';
312     TVME<=X"00000000";
313     PROGCLK<='1';
314     PROGRS<='0';
315     if ((DS0='1' and DS1='1') and CONTROL="0000") then
316         thatstate<=idle;
317     end if;

```

```

318  when bwrite0=>
319      SYSRES<='0';
320      modeset<="00";
321      VME_DIR<='0';
322      RDATA<='0';
323      WDATA<='1';
324      SDATA<=FVME;
325      FASTSTART<='0';  --active low
326      TVME<=X"00000000";
327      PROGCLK<='1';
328      PROGRS<='0';
329      DTACK<='0';
330      RESREG<='0';
331      if (WCOMPLETE='1' and CONTROL="0110") then
332          thatstate<=bwrite1;
333      end if;
334  when bwrite1 =>
335      SYSRES<='0';
336      modeset<="00";
337      VME_DIR<='0';
338      RDATA<='0';
339      WDATA<='0';
340      SDATA<=X"00000000";
341      FASTSTART<='1';
342      TVME<=X"00000000";
343      PROGCLK<='1';
344      PROGRS<='0';
345      DTACK<='0';
346      RESREG<='0';
347      if (DS0='1' and DS1='1' and CONTROL="0110") then
348          thatstate<=bwrite2;
349      end if;
350  when bwrite2 =>
351      SYSRES<='0';
352      modeset<="00";
353      VME_DIR<='0';
354      RDATA<='0';
355      WDATA<='0';
356      SDATA<=X"00000000";
357      FASTSTART<='1';
358      TVME<=X"00000000";
359      PROGCLK<='1';
360      PROGRS<='0';
361      DTACK<='1';
362      RESREG<='0';
363      if (BLTRANS='1' and DS0='0') then
364          thatstate<=bwrite0;
365      elsif(BLTRANS='0' and CONTROL="0110") then
366          thatstate<=bwrite3;
367      end if;
368  when bwrite3 =>
369      SYSRES<='0';
370      modeset<="00";
371      VME_DIR<='0';

```

```

372  RDATA<='0';
373  WDATA<='0';
374  SDATA<=FVME;
375  FASTSTART<='1';
376  PROGCLK<='1';
377  PROGRS<='0';
378  DTACK<='1';
379  RESREG<='1';
380  if ((DS0='1' and DS1='1') and CONTROL="0000") then
381    thatstate<=idle;
382  end if;
383  when bread0 =>
384    SYSRES<='0';
385    modeset<="00";
386    VME_DIR<='1';
387    RDATA<='1';
388    WDATA<='0';
389    SDATA<=FVME;
390    FASTSTART<='1';
391    TVME(31 downto 24)<=X"ff";
392    TVME(23 downto 16)<=X"ac";
393    TVME(15 downto 0)<=SYSDATA;
394    PROGCLK<='1';
395    PROGRS<='0';
396    DTACK<='0';
397    RESREG<='0';
398    if ((DS0='1' and DS1='1') and CONTROL="1110") then
399      thatstate<=bread1;
400    end if;
401  when bread1 =>
402    SYSRES<='0';
403    modeset<="00";
404    VME_DIR<='1';
405    RDATA<='1';
406    WDATA<='0';
407    SDATA<=FVME;
408    FASTSTART<='1';
409    TVME(31 downto 24)<=X"ff";
410    TVME(23 downto 16)<=X"ac";
411    TVME(15 downto 0)<=SYSDATA;
412    PROGCLK<='1';
413    PROGRS<='0';
414    DTACK<='1';
415    RESREG<='0';
416    if (BLTRANS='1' and DS0='0') then
417      thatstate<=bread0;
418    elsif (BLTRANS='0' and CONTROL="1110") then
419      thatstate<=bwrite3;
420    end if;
421  when i2c00 =>
422    SYSRES<='0';
423    modeset<="00";
424    VME_DIR<='1';
425    RDATA<='1';

```

```

426   WDATA<='0';
427   RESREG<='0';
428   PROGCLK<='1';
429   PROGRS<='0';
430   DTACK<='0';
431   SDATA<=FVME;
432   FASTSTART<='1';
433   TVME(31 downto 30)<="00";
434   TVME(29 downto 0)<=i2c1;
435   if ((DS0='1' and DS1='1') and CONTROL="1001") then
436     thatstate<=i2c01;
437   end if;
438   when i2c01 =>
439     SYSRES<='0';
440     modeset<="00";
441     VME_DIR<='1';
442     RDATA<='1';
443     WDATA<='0';
444     DTACK<='1';
445     RESREG<='1';
446     PROGCLK<='1';
447     PROGRS<='0';
448     SDATA<=FVME;
449     FASTSTART<='1';
450     TVME(31 downto 30)<="00";
451     TVME(29 downto 0)<=i2c1;
452     if ((DS0='1' and DS1='1') and CONTROL="0000") then
453       thatstate<=idle;
454     end if;
455   when i2c10 =>
456     SYSRES<='0';
457     modeset<="00";
458     VME_DIR<='1';
459     RDATA<='1';
460     WDATA<='0';
461     RESREG<='0';
462     PROGCLK<='1';
463     PROGRS<='0';
464     DTACK<='0';
465     SDATA<=FVME;
466     FASTSTART<='1';
467     TVME(31 downto 30)<="00";
468     TVME(29 downto 0)<=i2c2;
469     if ((DS0='1' and DS1='1') and CONTROL="1010") then
470       thatstate<=i2c11;
471     end if;
472   when i2c11 =>
473     SYSRES<='0';
474     modeset<="00";
475     VME_DIR<='1';
476     RDATA<='1';
477     WDATA<='0';
478     DTACK<='1';
479     RESREG<='1';

```

```

480  PROGCLK<='1';
481  PROGRS<='0';
482  SDATA<=FVME;
483  FASTSTART<='1';
484  TVME(31 downto 30)<="00";
485  TVME(29 downto 0)<=i2c2;
486  if ((DS0='1' and DS1='1') and CONTROL="0000") then
487    thatstate<=idle;
488  end if;
489  when i2c20 =>
490    SYSRES<='0';
491    modeset<="00";
492    VME_DIR<='1';
493    RDATA<='1';
494    WDATA<='0';
495    RESREG<='0';
496    PROGCLK<='1';
497    PROGRS<='0';
498    DTACK<='0';
499    SDATA<=FVME;
500    FASTSTART<='1';
501    TVME(31 downto 20)<="000000000000";
502    TVME(19 downto 0)<=i2c3;
503    if ((DS0='1' and DS1='1') and CONTROL="1011") then
504      thatstate<=i2c21;
505    end if;
506  when i2c21 =>
507    SYSRES<='0';
508    modeset<="00";
509    VME_DIR<='1';
510    RDATA<='1';
511    WDATA<='0';
512    DTACK<='1';
513    RESREG<='1';
514    PROGCLK<='1';
515    PROGRS<='0';
516    SDATA<=FVME;
517    FASTSTART<='1';
518    TVME(31 downto 20)<="000000000000";
519    TVME(19 downto 0)<=i2c3;
520    if ((DS0='1' and DS1='1') and CONTROL="0000") then
521      thatstate<=idle;
522    end if;
523  when syswr =>
524    SYSRES<='1';
525    modeset<="00";
526    VME_DIR<='0';
527    RDATA<='0';
528    WDATA<='1';
529    DTACK<='1';
530    RESREG<='0';
531    PROGCLK<='1';
532    PROGRS<='0';
533    SDATA<=FVME;

```

```

534     FASTSTART<='1';
535     TVME<=X"00000000";
536     if (CONTROL="0100") then
537         thatstate<=syswr1;
538     end if;
539 when syswr1 =>
540     SYSRES<='1';
541     modeset<="00";
542     VME_DIR<='0';
543     RDATA<='0';
544     WDATA<='0';
545     DTACK<='0';
546     RESREG<='0';
547     PROGCLK<='1';
548     PROGRS<='0';
549     SDATA<=FVME;
550     FASTSTART<='1';
551     TVME<=X"00000000";
552     if ((DS0='1' and DS1='1') and CONTROL="0100") then
553         thatstate<=syswr2;
554     end if;
555 when syswr2 =>
556     SYSRES<='1';
557     modeset<="00";
558     VME_DIR<='0';
559     RDATA<='0';
560     WDATA<='0';
561     DTACK<='1';
562     RESREG<='1';
563     SDATA<=FVME;
564     FASTSTART<='1';
565     TVME<=X"00000000";
566     PROGCLK<='1';
567     PROGRS<='0';
568     if ((DS0='1' and DS1='1') and CONTROL="0000") then
569         thatstate<=idle;
570     end if;
571 when others=>
572 end case;

574 end process;

578 takter: process(VSYSRES,CLK) is
579 begin
580 if (VSYSRES='0') then
581 LED<='1';
582 ass<='0';
583 daco<=7;
584 IRESET<='1';

586 thisstate<=idle;
587 elsif (CLK'event and CLK='1') then

```

```

588  --SET MODE PINS
589  if(SYSRES='0' and modeset="11") then
590    M1<='1';
591  elsif (SYSRES='1') then
592    CFGADD<=FVME(2 downto 0);
593    M1<='0';
594  end if;
595  thisstate<=thatstate;  -- CHANGE OF STATE
596  bcount<=bcount+1;
597  icount<=icount+1;
598  if(icount=unsigned(up)) then
599    IRESET<='1';
600    ass<='0';
601    daco<=7;
602  else
603    IRESET<='0';
604  end if;

607  --READ voltage rail values of UC9081
608  if(DAT='1') then
609    case daco is
610    when 7 =>
611      i2c1(9 downto 0)<=I2CV;
612      ass<='1';
613    when 6 =>
614      i2c1(19 downto 10)<=I2CV;
615      ass<='1';
616    when 5 =>
617      i2c1(29 downto 20)<=I2CV;
618      ass<='1';
619    when 4 =>
620      i2c2(9 downto 0)<=I2CV;
621      ass<='1';
622    when 3 =>
623      i2c2(19 downto 10)<=I2CV;
624      ass<='1';
625    when 2 =>
626      i2c2(29 downto 20)<=I2CV;
627      ass<='1';
628    when 1 =>
629      i2c3(9 downto 0)<=I2CV;
630      ass<='1';
631    when 0 =>
632      i2c3(19 downto 10)<=I2CV;
633      ass<='0';
634    when others =>
635      end case;
636  elsif(DAT='0' and ass='1') then
637    daco<=daco-1;
638    ass<='0';
639  else
640    end if;

```

```
642  end if;  
643  end process takter;  
  
649  end Behavioral;
```

A.4 Die FPGA-Konfiguration

Listing A.2: Die Serialisierung eines VME-Datenwortes

```

2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use IEEE.STD_LOGIC_ARITH.ALL;
5  use IEEE.STD_LOGIC_UNSIGNED.ALL;

8  entity scount is

11  PORT (
12      CLK : in STD_LOGIC;
13      FASTSTART: in STD_LOGIC;
14      MAPWORD: in STD_LOGIC_VECTOR(31 downto 0);
15      CDIN: out STD_LOGIC;
16      bitclk: out STD_LOGIC;
17      WCOMP: out STD_LOGIC
18  );

20  end scount;
21  architecture Behavioral of scount is

23      signal count: unsigned(5 downto 0);
24      signal ccount: std_logic_vector(5 downto 0);
25      constant b: std_logic_vector(5 downto 0):="000010";
26      constant c: std_logic_vector(5 downto 0):="100010";
27      constant d: std_logic_vector(5 downto 0):="110000";
28      begin
29          ccount<=std_logic_vector(count);
30          counter: process(CLK,FASTSTART)
31          begin
32              if (FASTSTART='1') then
33                  WCOMP<='0';
34                  count<=unsigned(b);
35                  bitclk<='0';
36              elsif (CLK'event and CLK='1') then

38                  count <= count+1;

40                  if (count<unsigned(c)) then
41                      bitclk<='1';
42                  elsif(count=unsigned(c)) then
43                      WCOMP<='1';
44                      bitclk<='0';
45                  end if;

47          case ccount is
48              when "000010" =>
49                  CDIN<=MAPWORD(31);
50              when "000011" =>

```

```
51  CDIN<=MAPWORD(30);
52  when "000100" =>
53  CDIN<=MAPWORD(29);
54  when "000101" =>
55  CDIN<=MAPWORD(28);
56  when "000110" =>
57  CDIN<=MAPWORD(27);
58  when "000111" =>
59  CDIN<=MAPWORD(26);
60  when "001000" =>
61  CDIN<=MAPWORD(25);
62  when "001001" =>
63  CDIN<=MAPWORD(24);
64  when "001010" =>
65  CDIN<=MAPWORD(23);
66  when "001011" =>
67  CDIN<=MAPWORD(22);
68  when "001100" =>
69  CDIN<=MAPWORD(21);
70  when "001101" =>
71  CDIN<=MAPWORD(20);
72  when "001110" =>
73  CDIN<=MAPWORD(19);
74  when "001111" =>
75  CDIN<=MAPWORD(18);
76  when "010000" =>
77  CDIN<=MAPWORD(17);
78  when "010001" =>
79  CDIN<=MAPWORD(16);
80  when "010010" =>
81  CDIN<=MAPWORD(15);
82  when "010011" =>
83  CDIN<=MAPWORD(14);
84  when "010100" =>
85  CDIN<=MAPWORD(13);
86  when "010101" =>
87  CDIN<=MAPWORD(12);
88  when "010110" =>
89  CDIN<=MAPWORD(11);
90  when "010111" =>
91  CDIN<=MAPWORD(10);
92  when "011000" =>
93  CDIN<=MAPWORD(9);
94  when "011001" =>
95  CDIN<=MAPWORD(8);
96  when "011010" =>
97  CDIN<=MAPWORD(7);
98  when "011011" =>
99  CDIN<=MAPWORD(6);
100 when "011100" =>
101 CDIN<=MAPWORD(5);
102 when "011101" =>
103 CDIN<=MAPWORD(4);
104 when "011110" =>
```

```
105  CDIN<=MAPWORD(3);
106    when "011111" =>
107  CDIN<=MAPWORD(2);
108    when "100000" =>
109  CDIN<=MAPWORD(1);
110    when "100001" =>
111  CDIN<=MAPWORD(0);
112    when others =>
113  CDIN<='0';
114  end case;

118  end if; --CLK

120  end process;

122  end Behavioral;
```

Listing A.3: Byteweises Auslesen eines VME-Datenwortes

```
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use IEEE.STD_LOGIC_ARITH.ALL;
5  use IEEE.STD_LOGIC_UNSIGNED.ALL;

8  entity scount is

11  PORT (
12  CLK : in STD_LOGIC;
13  FASTSTART: in STD_LOGIC;
14  MAPWORD: in STD_LOGIC_VECTOR(31 downto 0);
15  SMBUS: out STD_LOGIC_VECTOR(7 downto 0);
16  bitclk: out STD_LOGIC;
17  WCOMP: out STD_LOGIC
18  );

20  end scount;
21  architecture Behavioral of scount is

23  signal count: std_logic_vector(4 downto 0);

25  begin

27  counter: process(CLK,FASTSTART)
28  begin
29  if (FASTSTART='1') then
30  WCOMP<='0';
```

```
31  count<="00010";
32  bitclk<='0';
33  elsif (CLK'event and CLK='1') then

35      count <= count+1;

37      if (count<"00110") then
38          bitclk<='1';
39          elsif(count="00110") then
40              WCOMP<='0';
41              bitclk<='0';
42              elsif (count="01010") then -- "01010" best possible value for vme-master's
                  timing constraints
43                  WCOMP<='1';
44                  bitclk<='0';
45              end if;

47      case count is
48      when "00010" =>
49          SMBUS<=MAPWORD(31 downto 24);
50      when "00011" =>
51          SMBUS<=MAPWORD(23 downto 16);
52      when "00100" =>
53          SMBUS<=MAPWORD(15 downto 8);
54      when "00101" =>
55          SMBUS<=MAPWORD(7 downto 0);
56      when others =>
57          SMBUS<=X"00";
58      end case;

62  end if; --CLK

64  end process;

66  end Behavioral;
```

A.5 SystemACE

Listing A.4: Die Steuerung der Datenübertragung an SystemACE

```

3  library STD;
4  use STD.TEXTIO.ALL;
5  library IEEE;
6  use IEEE.STD_LOGIC_1164.ALL;
7  use IEEE.STD_LOGIC_ARITH.ALL;
8  use IEEE.STD_LOGIC_SIGNED.ALL;
9  use work.RWCMD.ALL;

11 entity writeread16 is
12     Port ( MPA : out  STD_LOGIC_VECTOR (6 downto 1);
13           MPWE : out  STD_LOGIC;
14           MPOE : out  STD_LOGIC;
15           MPCE : out  STD_LOGIC;
16           MPDR : in  STD_LOGIC_VECTOR (15 downto 0);
17           MPDW : out STD_LOGIC_VECTOR (15 downto 0);
18           CLK : in  STD_LOGIC;
19           RESET: in  STD_LOGIC;
20           STARTER: in STD_LOGIC;
21           TDAT : out STD_LOGIC;
22           RDCLK : out STD_LOGIC;
23           FIFORESET: out STD_LOGIC;
24           SYSRESET: out STD_LOGIC;
25           VMEDATA: out STD_LOGIC_VECTOR(15 downto 0);
26           DATAIN: in  STD_LOGIC_VECTOR (15 downto 0);
27           RDY: out STD_LOGIC;
28           DONE: out STD_LOGIC);

30 end writeread16;

32 architecture Behavioral of writeread16 is

34     signal wave: mpuif;
35     signal wticker: smwr;
36     signal rticker: smrd;
37     signal dataread: STD_LOGIC_VECTOR(15 downto 0);

39     signal machine: integer range 0 to 30;

41     signal timecount: integer range 0 to 1000001;
42     signal buffill: natural range 0 to 16;    -- 2 x 16 bytes per buffer
43     signal bufcount: natural range 0 to 5000; -- buffer count variable ANPASSEN
44     signal data: std_logic_vector(15 downto 0);
45     signal readclk: std_logic;
46     signal amcdata: std_logic_vector(0 to 15);
47     signal address: unsigned (5 downto 0);
48     constant startaddress: unsigned(5 downto 0) := "100000";

```

```

51  --SET LOGICAL BLOCK ADDRESS (LBA) AS LOWEST SECTOR ADDRESS 00023688, PHYS
    SECTOR 145000
52  constant sectoraddress0: std_logic_vector(15 downto 0):=X"3688";
53  constant sectoraddress1: std_logic_vector(15 downto 0):=X"0002";
54  constant acebuffer: integer:=4097;      --32 bytes per acebuffer, 16 acebuffer
    per sector --insert effective number of acebuffers + variable bufcount
    ANPASSEN

57  begin

59  MPCE<=wave.MPCE;
60  MPWE<=wave.MPWE;
61  MPOE<=wave.MPOE;
62  MPDW<=wave.MPDout;
63  MPA<=wave.MPA;
64  TDAT<=wave.TRISTATE;
65  wave.MPDin<=MPDR;

69  test: process (CLK,RESET,STARTER) is

71  begin

74  if (RESET='1') then
75  DONE<='0';
76  RDY<='0';
77  wticker<=one;
78  rticker<=one;
79  wave.MPCE<='1';
80  wave.MPWE<='1';
81  wave.MPOE<='1';
82  machine<=0;
83  timecount<=0;
84  buffill<=0;
85  bufcount<=0;
86  FIFORESET<='1';
87  address<=startaddress;
88  RDCLK<='0';

90  elsif (CLK'event and CLK='1') then
91      SYSRESET<='1';
92      if (STARTER='1') then
93          RDY<='1';
94          FIFORESET<='0';
95          case machine is
96          when 0 =>
97              wr(wave,wticker,"000000",X"0001");
98              if (wticker=wdone) then
99                  machine<=1;
100             end if;
101  -- SET LOCK REQUEST

```

```

102     when 1 =>
103         wr(wave,wticker,"001100",X"0002");
104         if (wticker=wdone) then
105             machine<=2;
106         end if;
107     -- POLL LOCK
108     when 2 =>
109         rd(wave,rticker,"000010",dataread);
110         if (rticker=rdone) then
111             machine<=3;
112         end if;
113     when 3 =>
114         if (dataread(1)='1') then
115             timecount<=0;
116             machine<=4;
117         elsif(timecount<10000) then
118             timecount<=timecount+1;
119             machine<=2;
120         end if;
121     -- POLL READY FOR COMMAND
122     when 4 =>
123         rd(wave,rticker,"000010",dataread);
124         if (rticker=rdone) then
125             machine<=5;
126         end if;
127     when 5 =>
128         if (dataread(8)='1') then
129             timecount<=0;
130             machine<=6;
131         elsif(timecount<10000) then
132             timecount<=timecount+1;
133             machine<=4;
134         end if;
135     -- SET LOGICAL BLOCK ADDRESS (LBA) AS LOWEST SECTOR ADDRESS
136     when 6 =>
137         wr(wave,wticker,"001000",sectoraddress0);
138         if (wticker=wdone) then
139             machine<=7;
140         end if;
141     when 7 =>
142         wr(wave,wticker,"001001",sectoraddress1);
143         if (wticker=wdone) then
144             machine<=8;
145         end if;
146     -- SECTOR COUNT
147     when 8 =>
148         wr(wave,wticker,"001010",X"0000");
149         if (wticker=wdone) then
150             machine<=9;
151         end if;
152     -- SET WRITEMEMCARDATA
153     when 9 =>
154         wr(wave,wticker,"001010",X"0400");
155         if (wticker=wdone) then

```

```

156     machine<=10;
157     end if;
158 -- RESET CFGJTAG
159     when 10 =>
160         wr(wave,wticker,"001100",X"0082");
161         if (wticker=wdone) then
162             machine<=11;
163         end if;
164 --     WRITE 32 BYTES DATA BUFFER
165 --     POLL DATA BUFFER READY (SYSTEM ACE DATA BUFFER)
166     when 11 =>
167         rd(wave,rticker,"000010",dataread);
168         if (rticker=rdone) then
169             machine<=12;
170         end if;
171     when 12 =>
172         if(dataread(5)='1') then
173             timecount<=0;
174             machine<=19;
175         elsif (timecount<10001) then
176             timecount<=timecount+1;
177             machine<=11;
178         else
179             end if;
180     when 19 =>
181         RDCLK<='1';
182         machine<=20;
183         when 20 =>
184             machine<=13;
185             amcdata<=DATAIN;
186     when 13 =>
187
188         wr(wave,wticker,std_logic_vector(address),amcdata); --BB
189         if (wticker=wdone) then
190             buffill<=buffill+1;
191             address<=address + 1;
192             machine<=21;
193         end if;
194
195     when 21 =>
196         RDCLK<='0';
197         machine<=14;
198     when 14 =>
199         if (buffill<16) then
200             machine<=19;
201         elsif (buffill=16) then
202             bufcount<=bufcount+1;
203             machine<=15;
204         end if;
205 --     BUFFER COUNT STATUS
206     when 15 =>
207         if (bufcount<acebuffer) then
208             buffill<=0;
209             address<=startaddress;

```

```

210     machine<=11;
211     elsif (bufcount=acebuffer) then
212         machine<=16;
213     end if;
214 -- RESET CFG RESET
215     when 16 =>
216         wr(wave,wticker,"001100",X"0002");
217         if (wticker=wdone) then
218             machine<=17;
219         end if;
220 -- RESET LOCK REQUEST
221     when 17 =>
222         wr(wave,wticker,"001100",X"0000");
223         if (wticker=wdone) then
224             machine<=18;
225         end if;

227         when others =>
228             end case;

232 end if; --START

235 end if; --CLK

237 end process;

241 end Behavioral;

```

A.6 Die I²C-Schnittstelle

Listing A.5: Die Datenübertragung über die I²C-Schnittstelle

```

2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  --use IEEE.STD_LOGIC_ARITH.ALL;

6  use IEEE.STD_LOGIC_UNSIGNED.ALL;
7  use IEEE.numeric_std.ALL;
8  library work;
9  use work.i2ccmd.ALL;

11 entity i2c is
12     Port ( SCLIN : in  STD_LOGIC;
13           SDAIN  : in  STD_LOGIC;

```

```

14         SDAOUT: out STD_LOGIC;
15         SCLOUT: out STD_LOGIC;
16         DATAV: out STD_LOGIC_VECTOR(9 downto 0);
17         DA: out STD_LOGIC;
18         SDAOE: out STD_LOGIC;
19         SCLOE: out STD_LOGIC;
20         RESET: in STD_LOGIC;
21         CLK : in  STD_LOGIC);
22     end i2c;

24     architecture Behavioral of i2c is

26         constant zero: unsigned(8 downto 0) := (others => '0');
27         constant a: std_logic_vector(8 downto 0):="110010000";
28         signal clken: std_logic;
29         signal ports: i2csignals;
30         signal readdata: std_logic_vector (7 downto 0);
31         signal rbyte1: std_logic_vector(9 downto 0);
32         signal machine: integer range 0 to 9;
33         signal data: integer range 0 to 7;
34         signal count: unsigned (8 downto 0);

36     begin

38         SDAOUT<=not ports.SDAOE;
39         SCLOUT<=not ports.SCLOE;
40         SDAOE<=ports.SDAOE;
41         SCLOE<=ports.SCLOE;
42         ports.SDAIN<=SDAIN;
43         ports.SCLIN<=SCLIN;
44         DATAV<=rbyte1;

48         i2cp: process(RESET,CLK) is
49             begin
50                 if(RESET='1') then
51                     machine<=0;
52                     data<=0;
53                     ports.ircount<=0;
54                     ports.ctrbit<=8;
55                     ports.sclOE<='0';
56                     ports.sdaOE<='0';
57                     ports.ticker<=idle;

60                 elsif(CLK'event and CLK='1') then
61                     if(clken='1') then

63                         case machine is
64                             when 0 =>
65                                 start(ports);
66                                 if(ports.ticker=final) then
67                                     machine<=1;

```

```
68  end if;
69  when 1 =>
70  wr(ports,"11011110");  --slave address 0x6F + write bit
71  if(ports.ticker=final) then
72  machine<=2;
73  end if;
74  when 2 =>
75  wr(ports,X"00");
76  if(ports.ticker=final) then
77  machine<=3;
78  end if;
79  when 3 =>
80  start(ports);
81  if(ports.ticker=final) then
82  machine<=4;
83  end if;
84  when 4 =>
85  wr(ports,"11011111");  -- slave address 6F
86  if(ports.ticker=final) then
87  machine<=5;
88  end if;
89  when 5 =>
90  DA<='0';
91  --RAILL1
92  rd(ports,readdata);
93  if(ports.ticker=final) then
94  rbyte1(9 downto 8)<=readdata(1 downto 0);
95  machine<=6;
96  end if;
97  when 6 =>
98  rd(ports,readdata);
99  if(ports.ticker=final) then
100  rbyte1(7 downto 0)<=readdata;
101  machine<=7;
102  end if;
103  when 7 =>
104  DA<='1';
105  if(data<7) then
106  data<=data+1;
107  machine<=5;
108  elsif(data=7) then
109  machine<=8;
110  end if;
111  when 8 =>
112  DA<='0';
113  stop(ports);
114  if(ports.ticker=final) then
115  machine<=9;
116  end if;
117  when 9 =>
118  when others =>

120  end case;
121  end if;
```

```
122  end if;

125  end process i2cp;

129  i2ccount: process (RESET,CLK) is
130  begin
131  if (RESET='1') then
132  count<=zero;
133  clken<='0';
134  elsif(CLK'event and CLK='1') then
135  count<=count+1;
136  if(count=unsigned(a))then
137  count<=zero;
138  clken<='1';
139  else
140  clken<='0';
141  end if;

144  end if; --CLK

147  end process i2ccount;

150  end Behavioral;
```

A.7 Die GANDALF VME-Routinen

Tabelle A.1: Die GANDALF VME-Routinen. Während mit den Befehlen `vme_read` und `vme_write` jeweils ein Datenwort in einfachen Transfer Routinen übertragen wird, werden mit den Befehlen `blockread` und `blockwrite` jeweils bis zu 256 Datenworte in blockweisen Transfer Routinen übertragen. Dabei entspricht N der dezimalen Anzahl der zu lesenden Datenworte pro Block, `blockwrite` teilt eine angehängte Datei automatisch in einzelne Blöcke auf. Die VME-Adressen enthalten im Allgemeinen die Kennzeichnung der GANDALF Module `0xE0`, die zweistellige hexadezimale Board-ID eines Moduls (ID) sowie die vierstellige Adresse `0xHHHH` der Register.

Befehl	Anwendung
<code>vme_read</code> [32 bit Adresswort]	<code>vme_read E0(ID)HHHH</code>
<code>vme_write</code> [32 bit Adresswort] [32 bit Datenwort]	<code>vme_write E0(ID)HHHH</code>
<code>blockread</code> [32 bit Adresswort] N	<code>blockread E0(ID)C000 N</code>
<code>blockwrite</code> [32 bit Adresswort] [Datei]	<code>blockwrite E0(ID)8000 2fpga.hex</code>

Anhang B

Die VME64x-Steckverbinder bei GANDALF

Tabelle B.1: Pinbelegung des VME64x-Steckverbinders P1

	Z	A	B	C	D
1	-	D(0)	-	D(8)	+5 V
2	GND	D(1)	-	D(9)	GND
3	-	D(2)	-	D(10)	+12 V
4	GND	D(3)	-	D(11)	+12 V
5	-	D(4)	-	D(12)	-
6	GND	D(5)	-	D(13)	+12 V
7	-	D(6)	-	D(14)	+12 V
8	GND	D(7)	-	D(15)	-
9	-	GND	-	GND	GAP
10	GND	-	BG3IN	-	GA0
11	-	GND	BG3OUT	BERR	GA1
12	GND	DS1	-	SYSRES	+3,3 V
13	-	DS0	-	LWORD	GA2
14	GND	WRITE	-	AM(5)	+3,3 V
15	-	GND	-	A(23)	GA3
16	GND	DTACK	AM(0)	A(22)	+3,3 V
17	-	GND	AM(1)	A(21)	GA4
18	GND	AS	AM(2)	A(20)	+3,3 V
19	-	GND	AM(3)	A(19)	-
20	GND	IACK	GND	A(18)	+3,3 V
21	-	IACKIN	-	A(17)	-
22	GND	IACKOUT	-	A(16)	+3,3 V
23	-	AM(4)	GND	A(15)	-
24	GND	A(7)	-	A(14)	+3,3 V
25	-	A(6)	-	A(13)	-
26	GND	A(5)	-	A(12)	+3,3 V
27	-	A(4)	-	A(11)	LI/I
28	GND	A(3)	-	A(10)	+3,3 V
29	-	A(2)	-	A(9)	LI/O
30	GND	A(1)	-	A(8)	+3,3 V
31	-	-12 V	-	+12 V	GND
32	GND	+5 V	+5 V	+5 V	+5 V

Tabelle B.2: Pinbelegung des VME64x-Steckverbinders P2

	Z	A	B	C	D
1	-	SINIT	+5 V	SDONE	-
2	GND	SCLK	GND	SDIN	-
3	-	GND	-	SPROG	-
4	GND	UD(0)	A(24)	GND	-
5	-	UD(2)	A(25)	UD(1)	-
6	GND	UD(4)	A(26)	UD(3)	-
7	-	UD(6)	A(27)	UD(5)	-
8	GND	GND	A(28)	UD(7)	-
9	-	UD(8)	A(29)	GND	-
10	GND	UD(10)	A(30)	UD(9)	-
11	-	UD(12)	A(31)	UD(11)	-
12	GND	UD(14)	GND	UD(13)	-
13	-	GND	+5 V	UD(15)	-
14	GND	UD(16)	D(16)	UD(17)	-
15	-	UD(18)	D(17)	GND	-
16	GND	UD(20)	D(18)	UD(19)	-
17	-	UD(22)	D(19)	UD(21)	-
18	GND	GND	D(20)	UD(23)	-
19	-	UD(24)	D(21)	UD(25)	-
20	GND	UD(26)	D(22)	GND	-
21	-	UD(28)	D(23)	UD(27)	-
22	GND	UD(30)	GND	UD(29)	-
23	-	GND	D(24)	UD(31)	-
24	GND	UCTRL	D(25)	UDW0	-
25	-	UDW1	D(26)	UDTEST	-
26	GND	UDRESET	D(27)	GND	-
27	-	GND	D(28)	UDWEN	-
28	GND	UDCLK	D(29)	GND	-
29	-	GND	D(30)	LFF	-
30	GND	LDOWN	D(31)	SRESET	-
31	-	-	GND	-	GND
32	GND	-	+5 V	-	+5 V

Abbildungsverzeichnis

2.1	Der inelastische Streuprozess	4
2.2	Strukturfunktionen für verschiedene Werte von Q^2	5
2.3	Tiefinelastische Streuung im Parton-Modell	6
2.4	Vergleich der Wirkungsquerschnitte von elastischer und tiefinelastischer Elektron-Proton Streuung	7
2.5	Partonverteilungsfunktionen	8
2.6	Der Aufbau des Nukleons und die einzelnen Beiträge zum Gesamtspin	10
2.7	Die Erhaltung der Helizität und die spinabhängige Streuung	11
2.8	Die polarisierte Strukturfunktion $g_1(x)$	11
2.9	Die exklusive Meson-Produktion	13
2.10	Die tiefvirtuelle Compton-Streuung und der Bethe-Heitler Prozess	14
3.1	Das COMPASS-Spektrometer	17
3.2	Der ringförmige Aufbau des Rückstoß-Detektors	19
3.3	Energieverlust geladener Teilchen im Detektor	20
3.4	Energieverlust und Teilchengeschwindigkeit	21
4.1	Das COMPASS-Datennahmesystem	23
4.2	Die Energie eines gestreuten Teilchens	24
4.3	Die Architektur des Trigger Control Systems (TCS)	25
5.1	Der Aufbau des GANDALF-Moduls als Transientenrekorder	27
5.2	Das GANDALF-Modul als Transientenrekorder	28
5.3	VXS-Architekturen mit aktivem Switch	29
5.4	Die S-Link-Schnittstelle	30
5.5	Eine JTAG-Kette	32
5.6	Das Taktsystem der GANDALF-Baugruppe	35
5.7	Die ideale Transferfunktion eines ADCs	37
5.8	Die Pipeline-Architektur des <i>ADS5463</i>	38
5.9	Reale Transferfunktion eines ADCs	39
5.10	Auswertung der Daten des GANDALF-Transientenrekorders	40
5.11	Leistung des Transientenrekorders	41
5.12	Mathematische Beschreibung der Detektorsignale	42
6.1	Die Gatterstruktur eines Logikbausteins	45
6.2	Der Aufbau des Coolrunner-II CPLDs	45
6.3	Funktionsweise einer Zustandsmaschine	47

6.4	Die Zustandsmaschine in VHDL	49
6.5	Die VME Board-Identifikation	52
6.6	Allgemeines Adressmuster zur Ansteuerung einer GANDALF-Baugruppe über VME	53
6.7	Die VME-Leseroutinen: Die Board-Adressierung beim einfachen Transfer	55
6.8	Die VME-Leseroutinen: Die Übertragung eines einzelnen Datenwortes im einfachen Transfer	56
6.9	Die VME-Leseroutinen: Die Übertragung mehrerer Datenworte im Blocktransfer	57
6.10	Die VME-Schreibroutinen: Die Board-Adressierung beim einfachen Transfer . .	58
6.11	Die VME-Schreibroutinen: Die Übertragung eines einzelnen Datenwortes im einfachen Transfer, bei der das Datenwort vom CPLD seriell weitergeleitet wird	59
6.12	Die VME-Schreibroutinen: Die Übertragung eines einzelnen Datenwortes im einfachen Transfer, bei der das Datenwort vom CPLD bytewise weitergeleitet wird	60
6.13	Die VME-Schreibroutinen: Die Übertragung mehrerer Datenworte im Blocktransfer, bei der die Datenworte vom CPLD seriell weitergeleitet werden	60
6.14	Die VME-Schreibroutinen: Die Übertragung mehrerer Datenworte im Blocktransfer, bei der die Datenworte vom CPLD bytewise weitergeleitet werden . .	61
6.15	Ein Vergleich von Datenraten der VME-Schreibroutinen	62
6.16	Die <i>Slave Serial</i> -Konfiguration der FPGAs	64
6.17	Die <i>SelectMAP</i> -Konfiguration der FPGAs	65
6.18	Die Verzeichnisstruktur auf der Compact-Flash Karte	66
6.19	Zugriff auf SystemACE	68
6.20	Speicherung von Daten mit SystemACE	69
6.21	Sektorweises Speichern von Messdaten auf einem Flash-Speicher	70
6.22	Das I ² C-Protokoll	71
6.23	Ansteuerung des Spannungsprüfers über den I ² C-Bus	71
6.24	Die Zusammenführung von Registerwerten des <i>UCD9081</i> zur Berechnung des Spannungswertes eines Netzes	72
A.1	Die Dekodierung der VME-Adresse.	75
A.2	Das Toplevel des CPLD-Controllers.	76

Tabellenverzeichnis

5.1	Verfügbare JTAG-Ketten bei der GANDALF-Einheit.	32
5.2	Die bei GANDALF eingesetzten I ² C-Schnittstellen.	33
5.3	Die GANDALF-Versorgungsspannungen	34
6.1	Die bei einem VME-Transfer eingesetzten Daten-, Adress- und Steuersignale. .	51
6.2	Die VME-Register im Controller-CPLD	54
6.3	Die VME- <i>Adress Modifier Codes</i>	54
6.4	Die Skripte zur Konfiguration einer GANDALF-Baugruppe	63
A.1	Die GANDALF VME-Routinen	102
B.1	Pinbelegung des VME64x-Steckverbinders P1	104
B.2	Pinbelegung des VME64x-Steckverbinders P2	105

Literaturverzeichnis

- [1] DEMTRÖDER, W.: *Experimentalphysik. Bd.4 : Kern-, Teilchen- und Astrophysik.* Springer-Verlag Berlin Heidelberg New-York, 2004.
- [2] BLOOM, E. D., D. H. COWARD, H. DESTAEBLER et al.: *High-Energy Inelastic $e - p$ Scattering at 6° and 10° .* Phys. Rev. Lett., 23(16):930–934, 1969.
- [3] POVH, B., K RITH, C. SCHOLZ und F. ZETSCHKE: *Teilchen und Kerne.* Springer-Verlag Berlin Heidelberg New-York, 2004.
- [4] FEYNMAN, R. P.: *Very High-Energy Collisions of Hadrons.* Phys. Rev. Lett., 23(24):1415–1417, 1969.
- [5] BREIDENBACH, M., J. I. FRIEDMAN, H. W. KENDALL et al.: *Observed Behavior of Highly Inelastic Electron-Proton Scattering.* Phys. Rev. Lett., 23(16):935–939, 1969.
- [6] BARGER, V. und R. J. N. PHILLIPS: *Quark-parton model relations in deep inelastic lepton scattering.* Nuclear Physics B, 73(2):269 – 294, 1974.
- [7] CLOSE, F. E.: *An Introduction to Quarks and Partons.* Academic Press Inc., London, 1980.
- [8] JAFFE, R. L. und A. MANOHAR: *The g_1 problem: Deep inelastic electron scattering and the spin of the proton.* Nuclear Physics B, 337(3):509 – 546, 1990.
- [9] *Das HERMES-Experiment.* <http://www-hermes.desy.de/>.
- [10] ASHMAN, J., B. BADELEK, G. BAUM et al.: *An investigation of the spin structure of the proton in deep inelastic scattering of polarised muons on polarised protons.* Nuclear Physics B, 328:1–35, 1989. CERN EP 89-73.
- [11] KUHN, S. E., J.-P. CHEN und E. LEADER: *Spin Structure of the Nucleon - Status and Recent Results.* arXiv:0812.3535v2 [hep-ph], 2009.
- [12] ALEXAKHIN, V. Y., Y. A. ALEXANDROV, G. D. ALEXEEV et al.: *The deuteron spin-dependent structure function and its first moment.* Physics Letters B, 647(1):8 – 17, 2007.
- [13] PLATCHKOV, S.: *Determination of the gluon contribution to the nucleon spin with COMPASS.* Nuclear Physics A, 790(1-4):58c – 63c, 2007.
- [14] COLLINS, J. C., L. FRANKFURT und M. STRIKMAN: *Factorization for hard exclusive electroproduction of mesons in QCD.* Phys. Rev. D, 56(5):2982–3006, 1997.

- [15] BURKARDT, M., MILLER, A. und W. D. NOVAK: *Spin-polarized high-energy scattering of charged leptons on nucleons*. <http://www.citebase.org/abstract?id=oai:arXiv.org:0812.2208>, 2008.
- [16] BARWIND, J.: *Azimuthal Asymmetries in Polarized Vector-Meson Production at the COMPASS Experiment*. Diplomarbeit, Albert-Ludwigs-Universität Freiburg, 2008.
- [17] WOLLNY, H.: *Bestimmung des Myonenflusses am COMPASS-Experiment*. Diplomarbeit, Albert-Ludwigs-Universität Freiburg, 2007.
- [18] DOBLE, N., L. GATIGNON, G. VON HOLTEY und F. NOVOSKOLTSEV: *The upgraded muon beam at the SPS*. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 343(2-3):351 – 362, 1994.
- [19] A., ABRAGAM und L. C. HEBEL: *The Principles of Nuclear Magnetism*. American Journal of Physics, 29(12):860–861, 1961.
- [20] MALLOT, G. K.: *The COMPASS spectrometer at CERN*. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 518(1-2):121 – 124, 2004.
- [21] BALL, J., G. BAUM, P. BERGLUND et al.: *First results of the large COMPASS ^6LiD polarized target*. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 498(1-3):101 – 111, 2003.
- [22] BERNHARD, J. B.: *Aufbau des inneren Rings eines Recoildetektors am COMPASS-Experiment*. Diplomarbeit, Johannes-Gutenberg-Universität Mainz, 2007.
- [23] GRUPEN, C.: *Teilchendetektoren*. BI Wissenschaftsverlag, 1993.
- [24] D’HOSE, N.: *Electronic Design for a proton trigger and a Time of Flight measurement with the Recoil Proton Detector for the Hadron Program*. Internal COMPASS note, 2007.
- [25] BARTKNECHT, S. P.: *Untersuchungen von Strukturen zeitdiskreter Systeme in Experimenten der Teilchenphysik*. Diplomarbeit, Albert-Ludwigs-Universität Freiburg, 2009.
- [26] SCHMIDT, T.: *A Common Readout Driver for the COMPASS Experiment*. Doktorarbeit, Albert-Ludwigs-Universität Freiburg, 2002.
- [27] LEBERIG, M., A. BRAVAR, J. HANNAPPEL et al.: *The trigger system of the COMPASS experiment*. Nuclear Science Symposium Conference Record, 2003 IEEE, Seiten 297–301 Vol.1, Oct. 2003.
- [28] BERNET, C., A. BRAVAR, J. HANNAPPEL et al.: *The COMPASS trigger system for muon scattering*. Nuclear Instruments and Methods in Physics Research A, 550:217–240, September 2005.
- [29] KONOROV, I., H. ANGERER, B. GRUBE et al.: *The trigger control system for the COMPASS Experiment*. Nuclear Science Symposium Conference Record, 2001 IEEE, 1:98–99 vol.1, Nov. 2001.

-
- [30] *Timing, Trigger and Control (TTC) Systems for the LHC*. <http://ttc.web.cern.ch/TTC/>.
 - [31] GRUBE, B.: *A Trigger Control System for COMPASS and A Measurement of the Transverse Polarization of Λ and Ξ Hyperons from Quasi-Real Photo-Production*. Doktorarbeit, Technische Universität München, 2006.
 - [32] 41.0-2006, ANSI/VITA: *American National Standard for VXS, VITA*, 2006.
 - [33] BOYLE, O., R. McLAREN und E. VAN DER BIJ: *The S-Link Interface Specification*. Technischer Bericht, ECP Division CERN, 1997.
 - [34] SEMICONDUCTOR, CYPRESS: *CY7C68001 EZ-USB SX2 High-Speed USB Interface Device*, 2003.
 - [35] TEXAS INSTRUMENTS: *8-channel power supply sequencer and monitor with error logging*, 2008.
 - [36] SCHOPFERER, S.: *Entwicklung eines hochauflösenden Transientenrekorders*. Diplomarbeit, Albert-Ludwigs-Universität Freiburg, 2009.
 - [37] TEXAS INSTRUMENTS INC.: *12-Bit 500 / 550-MSPS Analog-to-Digital Converters ADS5463 / ADS54RF63*, 2008.
 - [38] KESTER, W.: *The Data Conversion Handbook*. Analog Devices Inc., 2004.
 - [39] WENZL, K.: *Konzeption und Planung eines Transientenrekorders*. Diplomarbeit, Albert-Ludwigs-Universität Freiburg, 2008.
 - [40] *VisualAnalog by Analog Devices*. <http://www.analog.com/en/index.html>.
 - [41] KESEL, F. und R. BARTHOLOMÄ: *Entwurf von digitalen Schaltungen und Systemen mit HDLs und FPGAs*. Oldenbourg Verlag München Wien, 2006.
 - [42] XILINX, INC.: *XC5C512 CoolRunner II CPLD*, 2008.
 - [43] BRAUN, G., H. FISCHER, J. FRANZ et al.: *CATCH and CMC-HOTLink Readout Driver Specification*. Technischer Bericht, COMPASS-Note 1999-15, 1999.
 - [44] PETERSON, W. D.: *The VMEbus Handbook, 4th edition*. Technischer Bericht, VITA, 2006.
 - [45] XILINX, INC.: *Virtex-5 FPGA Configuration User Guide*, 2008.
 - [46] XILINX, INC.: *System ACE Compact-Flash Solution*, 2008.
 - [47] MOLITOR, P. und J. RITTER: *VHDL Eine Einführung*. Pearson Studium, 2004.
 - [48] *HxD - Freeware Hex-Editor und Disk-Editor*. <http://mh-nexus.de/de/hxd/>.
 - [49] NXP SEMICONDUCTORS: *I2C-bus specification and user manual*, 2007.

Danksagung

An dieser Stelle möchte ich all denjenigen Menschen danken, die zum Gelingen dieser Arbeit beigetragen haben:

- Prof. Horst Fischer für die ausgezeichnete Betreuung während meiner Zeit als Hauptpraktikant und Diplomand.
- Prof. Kay Königsmann für die Aufnahme in seine Arbeitsgruppe.
- Florian Herrmann, Sebastian Schopferer und Stefan Bartknecht für die tolle Arbeitsatmosphäre im GANDALF-Team.
- Den Probelesern Christian Schill, Heiner Wollny, Florian und Sebastian.
- Allen COMPASS- und CAST-Mitgliedern der Freiburger Gruppe für Anregungen, Hinweise und den abwechslungsreichen Begegnungen: Kay Königsmann, Susanne Rombach-Mikl, Andreas M., Andreas W., Christian, Elisabeth, Florian, Frank, Fritz-Herbert, Gregor, Heiner, Horst, Julia, Jürgen, Khalil, Martin, Rainer, Sebastian, Stefan, Tillmann.
- Miriam für das Verständnis, dass manche Abende und zuletzt ganze Wochenenden der Physik gehörten.
- Meiner Familie für ihre liebe Unterstützung während der gesamten Dauer meines Studiums.

Erklärung

Diese Arbeit ist von mir selbständig verfasst worden und ich habe keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Louis Philipp Lauser, April 2009