Ruprecht-Karls-Universität Heidelberg
Fakultät für Physik und Astronomie
Max - Planck - Institut für Kernphysik

IHEP 97-05
HD-ASIC-31-0197

Der ATLAS Level-1 Trigger

Auslese des Frontends

Diplomarbeit

von

Cornelius Schumacher

ASIC-Labor http://www.ihep.uni-heidelberg.de/Asic/
Schröderstraße 90 D-69120 Heidelberg

Der ATLAS Level-1 Trigger
Auslese des Front-Ends

Die Diplomarbeit wurde von Cornelius Schumacher ausgeführt am
Institut für Hochenergiephysik der Universität Heidelberg
unter der Betreuung von
Herrn Prof. Dr. K. Meier

Inhaltsverzeichnis

Einleitung

Auf der Suche nach den Bausteinen der Materie und der Natur der zwischen ihnen wirkenden Kräften bedient sich die Physik immer größerer Teilchenbeschleuniger, um Streuexperimente bei immer höheren Energien durchzuführen. Einer der größten geplanten Beschleuniger ist der *Large Hadron Collider* (LHC). Man hofft, mit seiner Hilfe unter anderem das Higgs-Boson zu finden, das letzte unentdeckte Teilchen des Standardmodells der Elementarteilchenphysik.

Bei Beschleuniger-Experimenten werden hochenergetische Teilchen zur Kollision gebracht. Je höher die zu Verfügung stehende Energie ist, desto kleinere Strukturen können beobachtet und desto schwerere Teilchen erzeugt werden. Die Prozesse, die man beobachten möchte, sind sehr selten. Deshalb müssen sehr viele Reaktionen stattfinden, um verwertbare Ergebnisse zu erzielen.

Neben der Erzeugung und Kontrolle des Teilchenstrahls bei sehr hoher Energie gibt es bei diesen Experimenten eine große Schwierigkeit. Aus der Vielzahl der stattfindenden Reaktionen müssen diejenigen isoliert werden, die das zu untersuchende physikalische Ereignis enthalten. Ein enormer Hintergrund an Ereignissen muß in Echtzeit unterdrückt werden, um eine Aufnahme der Daten zu ermöglichen. Das System, das diese Aufgabe erfüllt, ist das *Trigger-System.*

Die technische Realisierung des Auslese- und Trigger-Systems eines Hochenergieexperiments erfordert kompakte, schnelle Elektronik und darauf angepaßte Software. In dieser Diplomarbeit soll Elektronik- und Software-Entwicklung vorgestellt werden, die im Rahmen eines Demonstrations-Systems für das LHC-Experiment ATLAS durchgeführt wurde. Es handelt sich dabei um ein Demonstrationsmodul für das Front-End des Level-1-Triggers, das *Front-End-Modul* (FEM).

Diese Arbeit ist folgendermaßen aufgeteilt: Kapitel 1 gibt eine Einführung in die Physik, die am LHC untersucht wird, eine Beschreibung des Beschleunigers, des ATLAS-Detektors und des Trigger-Systems. In Kapitel 2 wird das Demonstrationssystem und das Front-End-Modul beschrieben. In den folgenden Kapiteln 3 und 4 wird die Entwicklung des *Front-End-Controllers* (FEC) beschrieben, der einen wesentlichen Bestandteil des FEM darstellt. Er dient hauptsächlich der Auslese und Konfiguration des FEM. Kapitel 3 gibt einen allgemeinen Überblick über die beim Design des FEC verwendeten Methoden, während Kapitel 4 die konkrete Realisierung beschreibt. Kapitel 5 beschreibt die Software zur Ansteuerung des FEM. Der erste Test des Moduls ist in Kapitel 6 dokumentiert.

Kapitel 1

Das LHC-Experiment ATLAS

Im Jahre 2005 soll am europäischen Kernforschungszentrum CERN in Genf im vorhandenen Tunnel des LEP (*Large Electron Positron Collider*) ein neuer Teilchenbeschleuniger in Betrieb gehen, der LHC (*Large Hadron Collider*). Mit ihm wird es möglich sein, in bisher unerreichbare Regionen der experimentellen Elementarteilchenphysik vorzudringen. ATLAS (*A Toroidal LHC Aparatus*) ist eins der beiden großen Experimente, das am LHC betrieben werden soll.

1.1 Der Large Hadron Collider

Der LHC ist ein *Proton-Proton-Collider*. Protonen werden auf gegenläufigen Kreisbahnen bis auf eine kinetische Energie von jeweils 7 TeV beschleunigt. Dann werden sie in den Wechselwirkungszonen der Experimente zur Kollision gebracht. Für die dort stattfindenden Reaktionen steht eine Schwerpunktsenergie von $\sqrt{s} = 14$ TeV zu Verfügung.

Am LEP wurden bisher gegenläufige Elektronen und Positronen (Masse: je 511 keV/c^2) benutzt. Da die abgegebene Synchrotron-Strahlung die Energie der beschleunigten Teilchen begrenzt, ist es vorteilhafter, schwerere Teilchen zu benutzen. Deshalb verwendet man beim LHC Protonen (Masse: 938 MeV/c^2).

Der vorhandene Tunnel hat einen Umfang von 26,7 km. Um die Protonen bei der angestrebten Energie auf der Kreisbahn zu halten, sind starke Magnetfelder mit einer Feldstärke von über 8 T nötig, die mit Hilfe supraleitender Dipol-Magnete erzeugt werden.

Der Protonen-Strahl des LHC ist in Teilchenpakete oder *Bunches* aufgeteilt, die alle 25 ns im Detektor kollidieren. Die Kollison dieser Pakete wird *Bunch-Crossing* genannt.

Die Reaktionsrate R, die man am Wechselwirkungspunkt erhält, setzt sich aus einem Anteil, der vom Beschleuniger bestimmt wird, der sogenannten *Luminosität* L, und einem durch die stattfindende Reaktion bestimmten Anteil zusammen, dem *Wirkungsquerschnitt* σ. Es gilt

$$R = \sigma L.$$

Die Luminosität ist nach [1] definiert als

$$L = n_1 n_2 f / A$$

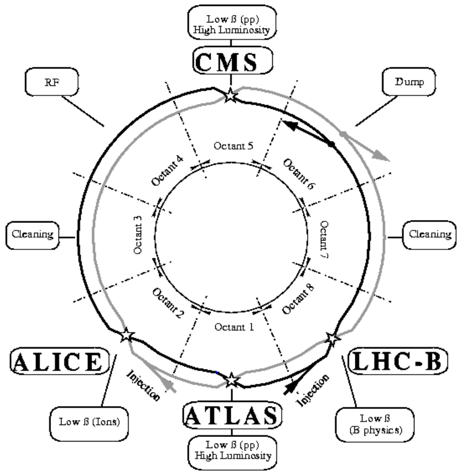


Abbildung 1.1: Der LHC [2]

(n_1, n_2,A: Teilchenzahlen bzw. Fläche der zur Kollision gebrachten Teilchenpakete, f: Zahl der umlaufenden Teilchenpakete pro Zeiteinheit). Bei vielen der am LHC zu untersuchenden Prozesse handelt es sich um sehr seltene Prozesse, das heißt sie haben einen kleinen Wirkunsquerschnitt. Deshalb benötigt man eine große Luminosität, um eine ausreichende Reaktionsrate zu erzielen. Die Luminosität soll beim LHC anfangs $10^{33}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ betragen und bis auf $10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ gesteigert werden.

Die *integrierte Luminosität*

$$\int L\, dt$$

ist ein Maß für die insgesamt stattgefundenen Reaktionen und somit auch ein Maß für die Grundlage, auf der statistische Aussagen getroffen werden können. Für sehr seltene Prozesse und genaue Messungen ist es erforderlich, eine hohe integrierte Luminosität zu erreichen.

Am LHC sollen vier große Experimente betrieben werden (Abbildung 1.1): Die beiden Universaldetektoren ATLAS und CMS[1] sowie LHC-B, das sich mit der Physik der B-Mesonen, insbesondere der CP-Verletzung befaßt, und ALICE, für das statt Protonen schwere Ionen bei Schwerpunktsenergien von über 1000 TeV zur Kollision gebracht werden sollen, um bei sehr hoher Materiedichte das sogenannte *Quark-Gluon-Plasma* zu untersuchen.

1.2 Physik am LHC

Die Grundlage der heutigen Elementarteilchenphysik ist das sogenannte *Standardmodell.* In ihm wird sämtliche Materie als aus fundamentalen Teilchen zusammengesetzt beschrieben. Es gibt zwei Arten fundamentaler Teilchen, Quarks und Leptonen. Sie kommen

---

[1] *Compact Muon Solenoid*

	Familien		
Quarks	u (up) d (down)	c (charme) s (strange)	t (top) b (bottom)
Leptonen	e^- ν_e	μ^- ν_μ	τ^- ν_τ

Tabelle 1.1: Die fundamentalen Teilchen des Standardmodells

jeweils in drei Familien vor (Tabelle 1.1). Zu jeder Familie gehören zwei Quarks, ein elektron-artiges Teilchen und ein dazugehöriges Neutrino. Kräfte zwischen den fundamentalen Teilchen werden durch drei Wechselwirkungen vermittelt: Die starke Wechselwirkung, die elektroschwache Wechselwirkung und die Gravitation. Man kann die Vermittlung der Kräfte durch den Austausch von speziellen Teilchen beschreiben. Für die starke Wechselwirkung sind dies die *Gluonen (g)*, für die elektomagnetische das *Photon (γ)*, für die schwache die intermediären Vektorbosonen (W^+,W^- und Z^0) und für die Gravitation das *Graviton*.

Die Theorie zur Vereinheitlichung von elektromagnetischer und schwacher zur elektroschwachen Wechselwirkung verlangt ein neutrales, skalares Teilchen, das als einziges des Standardmodells bisher noch nicht experimentell nachgewiesen werden konnte, das Higgs-Boson. Das Higgs-Boson entsteht in der Theorie als Überbleibsel des Mechanismus der spontanen Symmetriebrechung, der die Massen der intermediären Vektorbosonen erklärt. Dies ist der einzige Ansatz innerhalb des Standardmodells, der eine Erklärung von Massen bietet. Die Suche nach dem Higgs-Boson ist somit eines der vorrangigen Ziele der experimentellen Elementarteilchenphysik. Da die Theorie eine obere Grenze von $m_\mathrm{H} = 1$ TeV für die Masse des Higgs-Bosons setzt [3], sollte es am LHC erzeugt werden können. Aus bisherigen Experimenten, dem Studium des Z-Zerfalls am LEP, ist eine untere Grenze der Masse von $m_\mathrm{H} > 60$ Gev [4] bekannt.

Das Standardmodell, so genau es auch viele Phänomene beschreibt, enthält doch einige unbefriedigende Aspekte. So gibt es eine Vielzahl von freien Parametern, zum Beispiel die Massen aller Quarks und Leptonen. Außerdem wäre es ästhetisch und philosophisch reizvoll, nach dem Beispiel der elektoschwachen Wechselwirkung eine vollständige Vereinheitlichung der Wechselwirkungen zu erreichen. Neue Physik jenseits des Standardmodells stellt daher einen Schwerpunkt der Forschungen am LHC dar.

Ein Beispiel für neue Physik ist die Theorie der Super-Symmetrie (SUSY). In ihr wird zu jedem Fermion des Standardmodells ein bosonischer und zu jedem Boson ein fermionischer Partner gefordert. Damit wird die Theorie symmetrisch bezüglich des Spektrums von Fermionen und Bosonen und es kann eine Vereintlichung der Kräfte erreicht werden. Von der großen Zahl an postulierten Teilchen der SUSY könnten einige im Massenbereich liegen, der dem LHC zugänglich ist.

Weitere Forschungsschwerpunkte im Bereich neuer Physik stellen eine mögliche Substruktur der Quarks oder Leptonen und die eventuelle Existenz von angeregten Vektorbosonen dar.

Neben der Suche nach neuer Physik ist die nähere Untersuchung bekannter Phänomene interessant. Das Top-Quark, das mit heutigen Beschleunigern nur mit sehr geringen Raten erzeugt werden kann, erschließt sich so einer genauen Beobachtung. Neue Erkenntnisse, unter anderem über die Struktur der CP-Verletzung, erwartet man bei der sogenannten B-Physik, der Physik der Mesonen, die ein Bottom-Quark enthalten.

1.2.1 Proton-Proton-Streuung

Am LHC findet an den Wechselwirkungspunkten der beiden großen Experimente ATLAS und CMS Proton-Proton-Streuung bei einer Schwerpunktsenergie von bis zu 14 TeV statt. Dieser Streuprozeß wird von der starken Wechselwirkung dominiert. Er muß deshalb mit Hilfe der *Quantenchromodynamik* (QCD), der Theorie der starken Wechselwirkung, analysiert werden.

Quantenchromodynamik

Die QCD führt für Quarks eine ladungsartige Eigenschaft ein, die Farbe. Über den Austausch von Gluonen wechselwirken Farbladungen miteinander. Gluonen tragen selbst Farbe und können so untereinander wechselwirken.

Dies führt dazu, daß die Stärke der Wechselwirkung zwischen Quarks mit wachsendem Abstand zunimmt. Die potentielle Energie des Farbfeldes wächst, bis sie ausreicht, ein Quark-Antiquark-Paar mit der Farbe zu erzeugen, die nötig ist, um die Farbladung abzuschirmen. So entstehen nach außen farbneutrale Konfigurationen von Quarks, die *Hadronen*. Den Vorgang der Bildung neuer Hadronen beim Versuch, ein einzelnes Quark aus einer farbneutralen Struktur zu entfernen, nennt man auch *Hadronisierung* des Quarks [5].

Bei Rechnungen, die im Rahmen der QCD durchgeführt werden, taucht das Problem auf, daß bei größerem Abstand der stark wechselwirkenden Teilchen die Störungstheorie nicht mehr anwendbar ist. Sie ist gewöhnlich in Feldtheorien, zu denen auch die QCD gehört, die Methode, mit der man zum Beispiel Wirkungsquerschnitte berechnet. Prozesse, bei denen sich die stark wechselwirkenden Teilchen nicht sehr nah beieinander befinden, lassen sich somit nicht exakt auf Basis der QCD berechnen, sondern nur phänomenologisch beschreiben.

Das Proton läßt sich im Rahmen der QCD als aus *Partonen*, das heißt Quarks und Gluonen, zusammengesetzt betrachten. Der Impuls des Protons verteilt sich zu verschiedenen Anteilen auf die Partonen.

Der Streuprozeß

Bei der Proton-Proton-Streuung können verschiedene Einzelprozesse stattfinden. Welche Prozesse das sind, hängt von der Energie der Stoßpartner, vom übertragenen Impuls und den Wirkungsquerschnitten für die Einzelprozesse ab.

Bei einer elastischen Streuung bleibt die Struktur der stoßenden Protonen erhalten, es wird bevorzugt nur wenig Impuls übertragen, und die Protonen werden nur wenig abgelenkt. Bei inelastischen Streuprozessen bricht die Proton-Struktur auf und es entstehen durch Hadronisierung weitere Teilchen.

Bei Prozessen, die durch einen großen Impulsübertrag gekennzeichnet sind, wechselwirken praktisch einzelne Partonen der stoßenden Protonen miteinander. Sie werden in

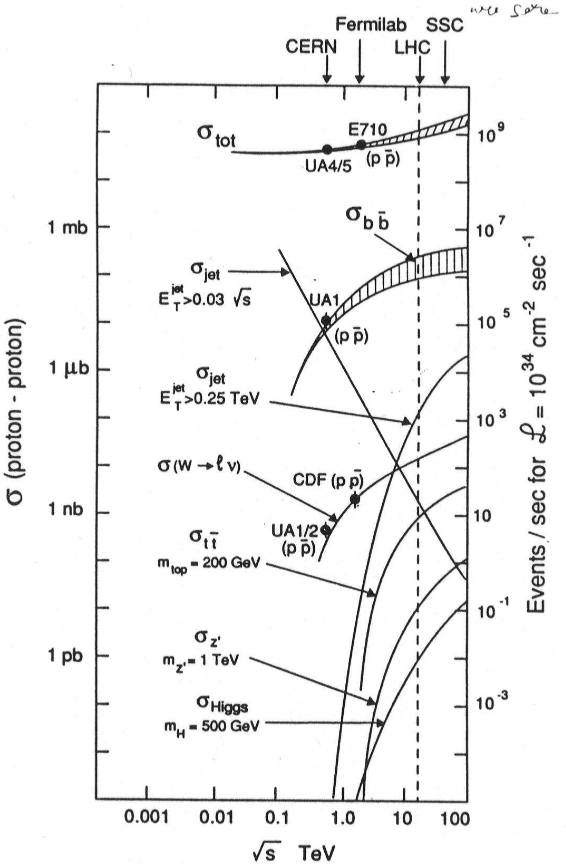


Abbildung 1.2: Wirkungsquerschnitte für bei Proton-Proton-Streuung stattfindende Prozesse [6]

großem Winkel zur Strahlrichtung abgelenkt und erzeugen durch Hadronisierung Bündel von Teilchen, die in eine gemeinsame Richtung abgestrahlt werden. Diese Bündel von hadronischen Teilchen werden *Jets* genannt. Bei einem harten Stoß zwischen zwei Partonen treten sie paarweise in entgegengesetzten Richtungen auf.

Aus der Analyse von Jets mit hohem transversalem Impuls kann man Rückschlüsse auf die Ausdehnung der Quarks ziehen. Abweichungen von der durch die QCD vorhergesagten Punktförmigkeit würden eine mögliche Substruktur der Quarks zeigen.

Die meisten bei der Proton-Proton-Streuung stattfindenden Prozesse sind einfache Prozesse, bei denen wenig Impuls übertragen wird. Sie leisten den Hauptbeitrag zum totalen Wirkungsquerschnitt für die Proton-Proton-Streuung. Prozesse, bei denen schwere Quarks oder Jets hohen transversalen Impulses erzeugt werden, sind selten, ebenso, nach den theoretischen Voraussagen, die Produktion bisher im Experiment nicht nachgewiesener Teilchen wie angeregter Vektorbosonen oder Higgs-Bosonen. In Abbildung 1.2 sind die voraussichtlichen Wirkungsquerschnitte beziehungsweise Ereignisraten bei voller LHC-Luminosität für einige Prozesse dargestellt [6].

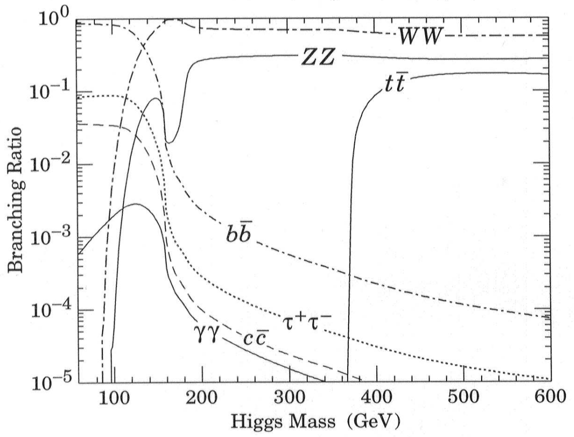


Abbildung 1.3: Verzweigungsverhältnis der Zerfallskanäle des Higgs-Bosons [4]

Der totale Wirkungsquerschnitt für die Proton-Proton-Streuung beträgt bei voller LHC-Energie etwa 100 mb[2]. Das heißt, bei voller Luminosität finden etwa 10^9 Proton-Proton-Stöße pro Sekunde statt. Das sind im Durchschnitt 25 Ereignisse pro Bunch-Crossing. Der theoretisch vorhergesagte Wirkungsquerschnitt für beispielsweise die Produktion eines Higgs-Bosons mit einer Masse von 500 GeV liegt dagegen bei etwa 1 pb. Das heißt, eines von 10^{11} Ereignissen könnte die Produktion eines solchen Higgs-Bosons sein. Um solche Ereignisse trotz ihrer Seltenheit beobachten und mit ausreichender statistischer Grundlage untersuchen zu können, ist die hohe Luminosität erforderlich, die der LHC zu Verfügung stellen soll.

Die Schwierigkeit beim Nachweis seltener Ereignisse besteht darin, daß andere weit häufiger stattfindende Prozesse einen Hintergrund erzeugen, vor dem der gesuchte Prozeß identifiziert werden muß. Da wegen ihrer Menge nicht die Daten aller Ereignisse aufgenommen werden können, ist es erforderlich, an Hand typischer Merkmale in Echtzeit die Ereignisse auszuwählen, die den gesuchten Prozeß enthalten könnten. Diese Aufgabe erfüllt das *Trigger-System.*

1.2.2 Nachweis des Higgs-Bosons

Am Beispiel des Higgs-Bosons soll erläutert werden, wie ein in einem seltenen Prozess erzeugtes Teilchen anhand seiner Zerfallsprodukte nachgewiesen werden kann.

Es gibt verschiedene Mechanismen zur Higgs-Erzeugung [7]. Die häufigsten sind Gluon-Fusion (gg→H) und W/Z-Fusion (WW→H, ZZ→H). Daneben kann das Higgs auch in Verbindung mit weiteren Teilchen produziert werden, entweder einem $t\bar{t}$-Paar in den Reaktionen $gg \rightarrow t\bar{t}H$ und $q\bar{q} \rightarrow t\bar{t}H$, oder einem W bzw. Z ($q\bar{q} \rightarrow WH$, $q\bar{q} \rightarrow ZH$).

Der Nachweis des Higgs-Bosons erfolgt über seine Zerfallsprodukte. Es existieren verschiedene Zerfallskanäle [4]. Ihr Verzweigungsverhältnis in Abhängigkeit von der Higgs-

---

[2] 1 barn = 1 b = 10^{-24} cm^2

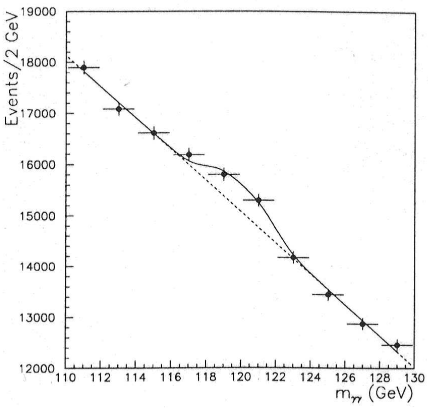


Abbildung 1.4: Erwartetes H → $\gamma\gamma$ Signal für $m_{\rm H} = 120$ GeV über $\gamma\gamma$-Hintergrund bei einer angenommenen integrierten Luminosität von 10^5 pb^{-1} [8]

Masse ist in Abbildung 1.3 dargestellt. Jeder Zerfallskanal erzeugt eine spezifische Signatur, die es gilt, vor dem Hintergrund anderer Prozesse zu identifizieren [8].

Unterhalb der Schwelle, an der das Higgs-Boson schwer genug ist, um in zwei Z- oder W-Bosonen zu zerfallen, also in einem Bereich der Higgs-Masse von etwa 80 GeV $< m_{\rm H} <$ 120 GeV, ist H → b$\bar{\rm b}$ der dominierende Zerfallskanal. b$\bar{\rm b}$-Paare werden jedoch bei der p-p-Streuung auch direkt, ohne vorherige Produktion eines Higgs-Bosons, in großer Zahl erzeugt (siehe Abbildung 1.2). Um diese b$\bar{\rm b}$-Paare von den beim Higgs-Zerfall entstehenden zu unterscheiden, beschränkt man die Suche auf diejenigen Higgs-Bosonen, die gemeinsam mit einem W oder t$\bar{\rm t}$ produziert werden. Dann wird neben dem b$\bar{\rm b}$-Paar noch ein isoliertes Lepton aus dem leptonischen Zerfall des W oder t$\bar{\rm t}$ erzeugt. Dieses Lepton kann vom Trigger erkannt werden und dient als Indiz dafür, daß das b$\bar{\rm b}$-Paar aus einem Higgs-Zerfall stammt.

Ein weiterer Zerfallskanal des Higgs-Bosons in einem Massenbereich von 80 GeV $< m_{\rm H} <$ 120 GeV ist der Prozeß H → $\gamma\gamma$. Er wird von einem starken Hintergrund überlagert, der aus Photon-Paaren besteht, die bei der Proton-Proton-Streuung direkt erzeugt werden. Um hier ein gegenüber dem Hintergrund signifikantes Signal zu erhalten (siehe Abbildung 1.4), werden nur Photonen ab einem gewissen transversalen Impuls berücksichtigt. Zudem muß die Energieauflösung des Kalorimeters sehr hoch sein und es ist eine große integrierte Luminosität erforderlich, um statistisch aussagekräftige Ergebnisse zu erzielen. Eine weitere Schwierigkeit ist, daß die elektromagnetischen Energiedepositionen von Jets als Photonen mißdeutet werden können. Da Zwei-Jet-Ereignisse in sehr hoher Zahl stattfinden, würde eine schlechte Trennung von Photon- und Jet-Ereignissen einen sehr starken Hintergrund zur Photon-Paar-Produktion erzeugen, der die Detektion des Higgs-Zerfalls in diesem Kanal unmöglich machen würde.

Ist das Higgs-Boson schwer genug, um in zwei Z-Bosonen zu zerfallen, erhält man durch den Zerfall H → ZZ → 4ℓ eine sehr klare Signatur von vier Leptonen. Für große Higgs-Massen ist auch der Zerfall H → ZZ → $\ell\ell\nu\bar{\nu}$ interessant. Hierbei besteht jedoch die Schwierigkeit, daß die Energie der Neutrinos nicht direkt gemessen werden kann. Man kann jedoch über die Impulserhaltung auf fehlenden, im Detektor nicht gemessenen, Transversalimpuls zurückschließen, der den Neutrinos zugeordnet werden kann.

Am Beispiel der Suche nach dem Higgs-Boson sieht man, daß es wichtig ist, Reaktionen an Hand markanter Signaturen zu identifizieren. Der Detektion und Unterscheidung von Leptonen, Photonen und Jets sowie der Messung fehlenden transversalen Impulses kommt dabei besondere Bedeutung zu. In Abschnitt 1.3.2 wird beschrieben, wie der Trigger von ATLAS dieser Aufgabe nachkommt.

1.3 Der ATLAS-Detektor

ATLAS ist ein Universaldetektor. Er ist auf keine spezielle Teilchenreaktion ausgelegt. Seine Aufgabe ist die Identifikation der bei der Proton-Proton-Streuung entstehenden Teilchen, die Messung ihrer Energie und ihres Impulses sowie die Rekonstruktion ihrer Spur. Um diese Aufgabe zu erfüllen, setzt sich der ATLAS-Detektor aus verschiedenen Subdetektoren zusammen (siehe Abbildung 1.5).

Direkt um den Wechselwirkungspunkt, an dem die Kollison der beiden Proton-Strahlen stattfindet, liegt der innere Detektor. Er besteht aus Pixeldetektoren, Siliziumstreifenzählern und Drahtkammern. Diese Detektoren stellen Informationen über den Ort durchlaufender Teilchen zu Verfügung und machen so eine Rekonstruktion der Teilchenspuren möglich. Der innere Detektor befindet sich in einem Magnetfeld solenoidaler[3] Geometrie. Durch die Ablenkung geladener Teilchen in diesem Magnetfeld kann der Teilchenimpuls bestimmt werden. Aus den vom inneren Detektor gelieferten Daten können Informationen zur Identifiktion der Teilchen und zu ihren Entstehungspunkten, den *Vertices*, gewonnen werden.

Um den inneren Detektor befindet sich das Kalorimeter von ATLAS. Es besteht aus einem inneren elektromagnetischen Teil und einem äußeren hadronischen Teil. Der innere Teil dient zur Messung der Energie von Elektronen und Photonen, der äußere Teil zur Messung der Energie von hadronischen Jets.

Myonen passieren aufgrund ihrer im Vergleich zu Elektronen hohen Masse inneren Detektor und Kalorimeter weitgehend ungehindert. Deswegen wird das Kalorimeter von einem Myon-Spektrometer umgeben. Es befindet sich in einem toroidalen[4] Magnetfeld, das die Messung der Myon-Impulse ermöglicht.

Abbildung 1.5 zeigt eine Ansicht des gesamten Detektors. Er hat eine Höhe von 22 m und wiegt 7000 t.

Als Koordinaten für den Detektor benutzt man die beiden Winkel eines spärischen Koordinatensystem mit Ursprung im Wechselwirkungspunkt, dessen z-Achse in Strahlrichtung liegt. Es wird der Azimutalwinkel ϕ und statt des Polarwinkels θ die Pseudorapidität η verwendet. Die Pseudorapidität ist definiert als

$$\eta = -\ln\tan\frac{\theta}{2}.$$

---

[3] Die magnetischen Feldlinien verlaufen parallel zum Strahl

[4] Die magnetischen Feldlinien verlaufen in konzentrischen Kreisen um den Strahl und liegen in Ebenen, die seknrecht auf dem Strahl stehen.

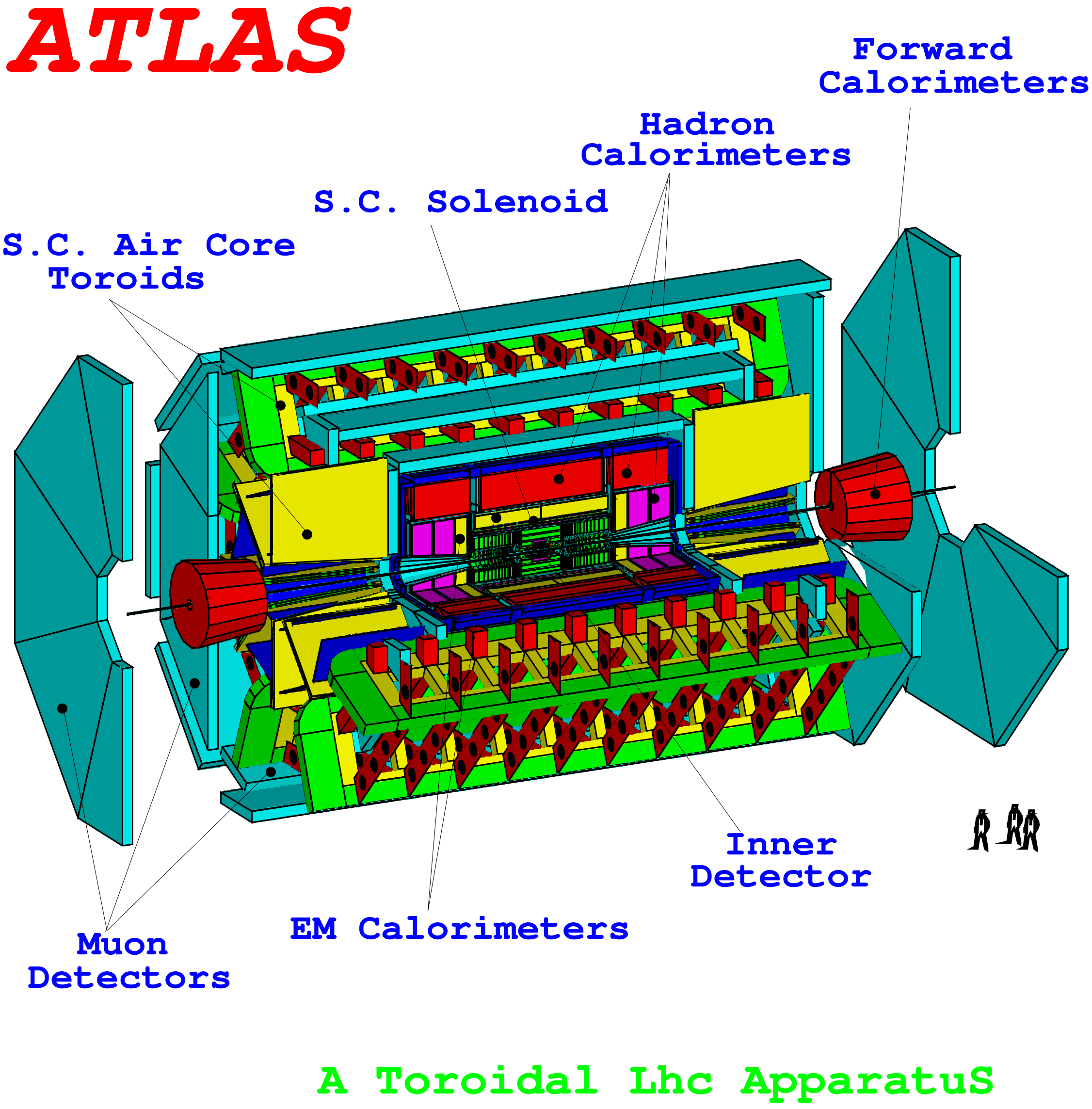


Abbildung 1.5: Der ATLAS-Detektor [9]

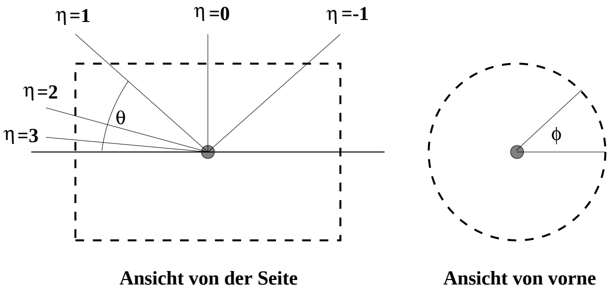


Abbildung 1.6: Das Koordinatensystem des Detektors

Sie ist für $\beta = \frac{v}{c} \rightarrow 1$ bis auf eine additive Konstante invariant unter Lorentz-Boosts. In Abbildung 1.6 ist das Koordinatensystem veranschaulicht.

1.3.1 Das Kalorimeter

Die Identifizierung von Photonen, Elektronen und hadronischen Jets sowie die Messung ihrer Energie ist ein wichtiger Teil der Durchführung von Proton-Proton-Streuexperimenten. Die Vielfalt der veschiedenartigen zu beobachtenden Reaktionen erfordert eine in jeder Beziehung hohe Leistunsfähigkeit des Kalorimeters. Im einzelnen sind dies:

- Hohe Energieauflösung
- Hoher dynamischer Bereich der Energiemessung von Teilchen, die Energien zwischen einigen hundert MeV bis zu Energien im TeV-Bereich haben.
- Hohe Ortsauflösung
- Überdeckung eines großen η-Bereichs

Die technische Realisierung des Kalorimeters erfolgt als sogenanntes *Sampling*-Kalorimeter. Das bedeutet, daß es aus sich abwechselnden Schichten aus Absorbermaterial bzw. sensitivem Material aufgebaut ist. In den Absorberschichten erzeugen durchlaufende Teilchen Schauer sekundärer Teilchen und verlieren dadurch Energie, die in den Schichten sensitiven Materials durch Ionisation nachgewiesen wird. Das Kalorimeter ist in eine große Zahl von Zellen gegliedert. Dadurch ist es möglich, den Ort der Energiedeposition zu bestimmen. Ein Pulsformer erzeugt aus der im sensitiven Bereich gemessenen Ionisationsladung ein elektrisches Signal, dessen Höhe proportional zu der in der entsprechenden Zelle deponierten Energie ist. Dieses Signal wird zur Ausleseelektronik weitergeleitet. In Abbildung 1.7 ist der geometrische Aufbau der verschiedenen Kalorimeterteile dargestellt.

Im elektromagnetischen Teil des Kalorimeters sind Bremsstrahlung, Paarbildung und Ionisation die vorherrschenden Mechanismen, durch die durchlaufende Teilchen Energie abgeben. Als Absorbermaterial wird deshalb Blei verwendet, das einen hohen Wirkungsquerschnitt für diese Mechanismen besitzt. Als sensitives Material wird flüssiges Argon verwendet. Die Ausdehnung des elektromagnetischen Kalorimeters ist so gewählt, daß auch hochenergetische Elektronen und Photonen ihre Energie dort nahezu vollständig

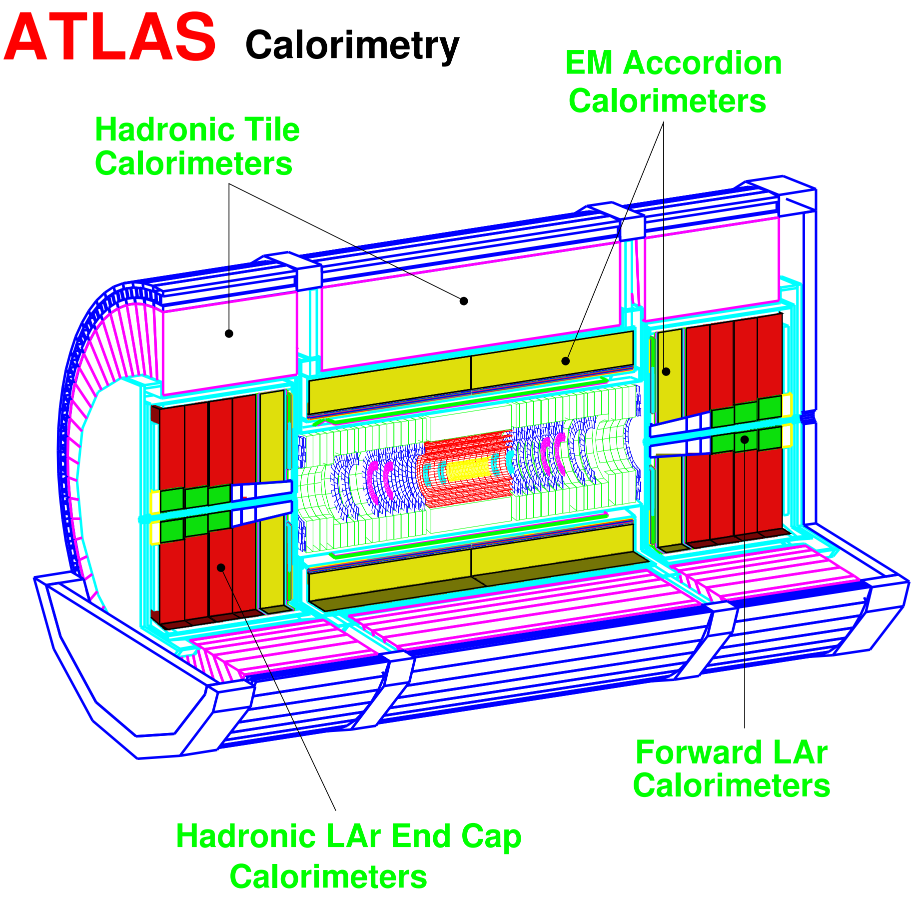


Abbildung 1.7: Das Kalorimetersystem von ATLAS [9]

abgeben. Die spezielle geometrische Anordnung der Schichten des elektromagnetischen Kalorimeters wird *Akkordeon*-Geometrie genannt.

Auch Hadronen geben einen Teil ihrer Energie im elektromagnetischen Kalorimeter ab. Aufgrund ihrer im Vergleich zu Elektronen großen Masse ist dies jedoch nur ein kleiner Teil ihrer gesamten Energie. Den größten Teil geben sie nach starker Wechselwirkung mit Atomkernen aus dem Absorbermaterial des hadronischen Kalorimeters ab. Für diesen Vorgang haben andere Materialien als Blei höhere Wirkunsquerschnitte. Deshalb kommen im äußeren Bereich Eisen, im in Strahlrichtung gelegenen Bereich Kupfer und Wolfram zur Anwendung. Als sensitives Material benutzt man im äußeren Bereich Plastikszintillatoren, im in Strahlrichtung gelegenen Bereich, wie im elektromagnetischen Teil des Kalorimeters, flüssiges Argon.

Der Grund für die Verwendung von flüssigem Argon als sensitive Kalorimeterkomponente liegt in der hohen Strahlenbelastung, der das Kalorimeter insbesondere in Strahlrichtung ausgesetzt ist. Die hohe Luminosität, die erforderlich ist, um seltene Prozesse beobachten zu können, erzeugt hier eine starke Belastung des Detektors. Flüssiges Argon erleidet im Gegensatz zu beispielsweise Plastikszintillatoren keine Strahlenschäden

Subdetektor	Kanalzahl	Datenvolumen [kByte]
Pixel Detektor	$1,4 \cdot 10^8$	50
Spurrekonstrunktion	$5,6 \cdot 10^6$	850
Kalorimeter	$2,3 \cdot 10^5$	180
Myon Detektor	$1,3 \cdot 10^6$	200
Gesamt		1280

Tabelle 1.2: Im Detektor pro Bunch-Crossing entstehende Datenvolumina

und bietet sich deshalb für die Teile des Detektors an, die einer hohen Strahlenbelastung ausgesetzt sind.

Im elektromagnetischen Teil des Kalorimeters muß es möglich sein, einzelne Elektronen oder Photonen von Jets zu unterscheiden, die eine Energiedeposition ähnlicher räumlicher Ausdehnung hinterlassen. Es ist deshalb sehr fein segmentiert. Während das hadronische Kalorimeter gut 20.000 Kanäle zur Auslese benötigt, sind für das elektromagnetische Kalorimeter über 200.000 Kanäle erforderlich.

Im Kalorimeter wird der transversale Energieanteil

$$E_{\mathrm{T}} = E \sin\theta$$

der produzierten Teilchen gemessen. Der longitudinale Anteil enthält keine verwertbare physikalische Aussage. Dies liegt daran, daß der Schwerpunkt-Impuls der stossenden Partonen in longitudinaler Richtung zum Zeitpunkt des Stoßes nicht bekannt ist, da ihr Anteil am longitudinalen Gesamtimpuls des jeweiligen Protons in der Regel verschieden ist. Der transversale Impuls der Protonen ist im Vergleich zum longitudinalen Impuls vernachlässigbar klein. Deshalb kann der Schwerpunkt-Impuls der stoßenden Partonen in transversaler Richtung als null angenommen werden.

Eine wichtige Meßgröße ist die fehlende transversale Energie $E_{\mathrm{T}}^{\mathrm{miss}}$. Addiert man vektoriell den transversalen Anteil der Impulse aller entstandenen Teilchen so muß die Summe aufgrund des Impulserhaltungssatzes verschwinden. Ist dies nicht der Fall, so haben Teilchen wie Neutrinos, die nur schwach wechselwirken, Energie aus dem Detektor getragen. Der nicht verschwindende Anteil dieser Summe ist der fehlende transversale Impuls bzw. die entsprechende fehlende transversale Energie. Um eine genaue Bestimmung von $E_{\mathrm{T}}^{\mathrm{miss}}$ zu ermöglichen, muß das Kalorimeter einen möglichst großen Teil des Raumwinkels abdecken, in den Teilchen emittiert werden.

1.3.2 Kalorimeter-Auslese und Trigger

Die Datenmengen, die im Detektor anfallen, sind sehr groß. An die Elektronik des Auslesesystems werden somit hohe Anforderungen gestellt. Um eine vollständige Rekonstruktion des stattgefundenen Ereignisses zu ermöglichen, muß der gesamte Detektor ausgelesen werden. In Tabelle 1.2 [8] sind die Datenvolumina dargestellt, die der ein komplettes Ereignis umfassende Datensatz enthält, der pro Bunch-Crossing anfällt. Bei einer Bunch-Cossing-Rate von 40 MHz und damit einer Datenrate von 51,2 TByte/s ist es unmöglich,

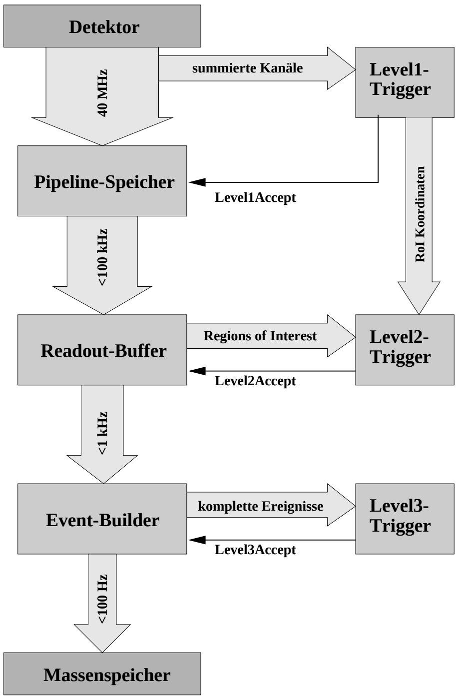


Abbildung 1.8: Detektor-Auslese und Trigger

alle Daten zu speichern. Deshalb ist es notwendig, die Rate, mit der die Daten gespeichert werden sollen, zu reduzieren. Diese Aufgabe erfüllt das *Trigger*-System. Es muß in Echtzeit die Datensätze auswählen, die relevante physikalische Informationen enthalten. Nimmt man an, daß es möglich ist, komplette Datensätze mit einer Rate von etwa 100 Hz zu speichern, liegt der Faktor, um den das Trigger-System die Datenrate reduzieren muß, in einer Größenordnung von 10^5.

Das Trigger-System besteht aus drei Stufen. Auf jeder Stufe findet eine Reduktion der Datenrate statt. Da die Entscheidung des Triggers eine gewisse Zeit in Anspruch nimmt, müssen die Daten zwischengespeichert werden, bis das Signal kommt, ob sie verworfen werden oder nicht. Die Zeitspanne, die der Trigger für die Berechnung braucht und für die die Daten deshalb zwischengespeichert werden müssen, ist die sogenannte *Latenzzeit*.

In Abbildung 1.8 ist der Datenstrom vom Detektor bis zur Speicherung auf einem Massenspeicher dargestellt. Auf dem direkten Weg werden die Datensätze in vollem Umfang, das heißt jeweils der komplette Datensatz eines Ereignisses, aber mit in jeder Triggerstufe verminderter Rate weitergeleitet. Die Trigger-Entscheidung der ersten beiden Stufen muß sehr schnell erfolgen. Deshalb wird für diese Trigger-Stufen nur ein Teil des kompletten Datensatzes ausgewertet.

So wird für die erste Stufe, den *Level-1-Trigger*, nicht die volle Granularität des Kalorimeters genutzt, sondern mehrere Kalorimeter-Kanäle werden zu Trigger-Zellen, sogenannten *Trigger-Towern* zusammengefaßt. Der Kalorimeter-Level-1-Trigger muß dann nur noch etwa 8000 Kanäle verarbeiten, die jedoch mit der Bunch-Crossing-Rate von 40 MHz ausgelesen werden müssen. Die Ereignisrate wird im Level-1-Trigger auf 10 kHz bis 100 kHz reduziert. Die maximale Eingangsrate für den Level-2-Trigger beträgt 100 kHz, die Latenzzeit, in der die Trigger-Entscheidung getroffen wird, 2 μs. Die immer noch hohe Zahl an zu verarbeitenden Kanälen sowie die hohe Geschwindigkeit, in der dies stattfinden muß, macht es erforderlich, den Level-1-Trigger aus speziell für diesen Anwendungszweck entworfenen integrierten Schaltkreisen aufzubauen, sogenannten ASICs (*Application Specific Integrated Circuit*). So ist es möglich, eine hohe Integratiosdichte der elektronischen Komponenten zu erreichen.

Für den Level-2-Trigger wird eine andere Vorgehensweise gewählt. Hier werden die Daten in voller Auflösung übertragen, jedoch nur für einige Bereiche des Kalorimeters, sogenannte *Regions of Interest* (RoI), in denen der Level-1-Trigger Energiedepositionen festgestellt hat, die auf physikalisch relevante Ereignisse hindeuten. Im Level-2-Trigger wird die Ereignisrate auf 100 Hz–1000 Hz reduziert. Die Latenzzeit beträgt etwa 10 ms. Der Level-2-Trigger soll auf der Basis von kommerziell erhältlichen Komponenten aufgebaut werden und ein hohes Maß an Flexibilität in den Algorithmen zur Trigger-Entscheidung bieten.

Nach der Stufe des Level-2-Triggers werden die Daten in vollem Umfang und Auflösung an den *Event-Builder* weitergegeben, der das im Detektor stattgefundene physikalische Ereignis rekonstruiert. Auf Basis dieser Daten führt der Level-3-Trigger dann nochmals eine Reduktion der Datenrate durch. Hierbei bedient er sich ähnlicher Algorithmen, wie sie auch bei der Offline-Analyse der Daten Verwendung finden. Technisch wird der Level-3-Trigger als Prozessor-Farm[5] realisiert. Durch die Ereignis-Struktur ist bereits eine Parallelisierung der Daten gegeben. Jedes Ereignis kann unabhängig von anderen bearbeitet werden. Letzlich werden die Daten mit einer Rate von maximal 100 Hz auf Massenspeichern abgelegt. Dies entspricht einer Datenrate von 100 MB/s.

1.3.3 Der Level-1-Trigger

Die Daten, auf deren Grundlage der Level-1-Trigger seine Entscheidung trifft, sind auf einem zweidimensionalen Raster angeordnet, das sich in Richtung von η und ϕ in Schritten von jeweils 0,1 erstreckt. Die Tiefensegmentierung des Kalorimeters bleibt nur insofern erhalten, als daß jeweils ein Raster von Daten für den elektromagnetischen und den hadronischen Teil des Kalorimeters existiert. Auf diesen Rastern versucht der Level-1-Trigger Energiedepositionen zu identifizieren, die von isolierten Elektronen, Photonen oder Jets stammen. Nach geeigneter Summation über alle Zellen, läßt sich auch eine Information über möglicherweise fehlende transversale Energie gewinnen. Aufgrund dieser Daten sowie unter Einbeziehung von Informationen, die das Myon-System bereit stellt, trifft der Level-1-Trigger die Entscheidung, ob das bearbeitete Ereignis an den Level-2-Trigger weitergeleitet wird.

Die wesentlichen Funktionsblöcke des Level-1-Triggers sind in Abbildung 1.9 dargestellt. Als Eingang dienen die 8000 Kanäle, über die das Kalorimeter den Betrag der im jeweiligen Trigger-Tower deponierten Energie bereitstellt. Die Zusammenfassung zu

---

[5] Als *Prozessor-Farm* wird eine Anordnung bezeichnet, in der eine große Zahl von Prozessoren parallel arbeitet. Dabei verarbeitet jeder einzelne Prozessor jeweils nur einen kleinen Teil der Gesamtdaten.

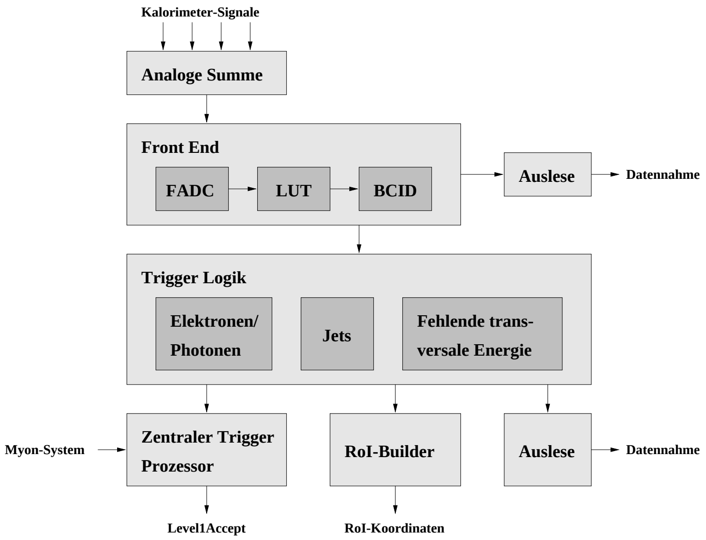


Abbildung 1.9: Blockdiagram des Level-1-Trigger-Systems

Trigger-Towern geschieht über analoge Summation der Signale der beteiligten Kalorimeterzellen.

Nach einer Vorverarbeitung im Front-End, die im nächsten Abschnitt genauer beschrieben werden soll, werden die Daten an die Trigger-Logik weitergegeben. Diese sucht nun nach physikalisch interessanten Signaturen. Für den Kalorimeter-Level-1-Trigger sind das im wesentlichen isolierte Elektronen bzw. Photonen, hadronische Jets und fehlende transversale Energie. Jeder dieser Bereiche wird von einem speziell angepaßten Triggerprozessor bearbeitet.

Der Algorithmus zur Identifizierung von isolierten Elektronen bzw. Photonen sucht nach Trigger-Zellen, deren Energiegehalt über einer gewissen Schwelle liegt. Der Fall, daß das Elektron bzw. Photon das Raster an einer Zellengrenze passiert und somit Energie gleichermaßen in zwei benachbarte Zellen deponiert, wird dadurch berücksichtigt, daß Paare von benachbarten Zellen betrachtet werden. Danach wird ein Isolationskriterium angelegt, indem geprüft wird, ob in dem Ring von Zellen, die um diejenige Zelle herum angeordnet sind, in der Energie oberhalb des Schwellwertes gefunden wurde, der Energiegehalt unter einer kleinen Grenze liegt. Ist dies der Fall, kann man davon ausgehen, daß es sich um eine räumlich isolierte elektromagnetische Energiedeposition handelt.

Um Elektronen bzw. Photonen von Jet-Spuren zu unterscheiden, nutzt man aus, daß Elektronen bzw. Photonen eine stärker lokalisierte Energiedeposition hinterlassen, und legt ein weiteres Isolationskriterium an. Es wird die Zelle des hadronischen Rasters betrachtet, die sich in Bezug auf den Wechselwirkungspunkt hinter der elektromagnetischen Zelle befindet. Findet sich hier nur wenig Energie, ist die Wahrscheinlichkeit klein, daß eine Jet-Spur fälschlicherweise als Elektron oder Photon mißdeutet wird.

Die Erzeugung von hadronischen Jets ist der dominierende Prozeß bei der Proton-Proton-Streuung. Er ist deshalb im wesentlichen untergrundfrei. Die Aufgabe des Jet-Triggers liegt darum nicht nur in der Identifizierung der Jets. Wichtig ist eine möglichst genaue Abschätzung der in ihnen enthaltenen Energie. Jets erstrecken sich über mehrere Trigger-Zellen. Deshalb wird ein quadratisches Fenster gewählt, das sich aus mehreren Zellen zusammensetzt. Dieses Fenster wird in kleinen Schritten über das Raster der Trigger-Zellen geschoben. Übertrifft die insgesamt im Fenster deponierte Energie eine gewisse Schwelle, wird ein Jet dieser Schwellenergie gezählt. Bei der Zählung muß der Fall berücksichtigt werden, daß Jets sich überlappen.

Die Größe des Fensters und die Schrittweite, in der es bewegt wird, beeinflussen die Effizienz des Jet-Algorithmus. Ist das Fenster zu klein, wird die Jet-Energie unterschätzt, ist es zu groß, wird sie überschätzt. Durch Simulation mit verschiedenen Parametern muß geklärt werden, welche Größe und Schrittweite benötigt wird [10].

Den Schwellwerten transversaler Energie, die in den Trigger-Algorithmen benutzt werden, kommt eine entscheidende Bedeutung zu. Da die Ereignisrate mit sinkender transversaler Energie der erzeugten Teilchen stark zunimmt, kann eine zu gering gewählte Schwelle oder ein Algorithmus, der die transversale Energie überschätzt eine zu hohe Trigger-Rate zur Folge haben. Bei zu hoch gewählter Schwelle oder einem ineffizienten Algorithmus gehen dagegen Ereignisse verloren.

Als Ergebnis stellen Elektron/Photon- und Jet-Trigger die Multiplizität der jeweiligen Ereignisse bei verschiedenen Energieschwellen sowie ihre Koordinaten zu Verfügung. Die Koordinaten werden im Falle einer positiven Trigger-Entscheidung über den *RoI-Builder* an den Level-2-Trigger gesendet.

Der $E_{\mathrm{T}}^{\mathrm{miss}}$-Trigger führt die vektorielle Summation der Teilchenimpulse durch, legt verschiedene Schwellen an die daraus ermittelte fehlende transversale Energie an und signalisiert, ob sie übertroffen werden. Ein effizienter $E_{\mathrm{T}}^{\mathrm{miss}}$-Trigger erfordert, daß ein möglichst großer η-Bereich des Detektors Berücksichtigung findet.

Im zentralen Trigger-Prozessor (CTP[6]), der auch noch Informationen aus dem Myon-System berücksichtigt, werden die Ergebnisse der Trigger-Logik zusammengeführt. Auf ihrer Grundlage wird die Entscheidung getroffen, ob das untersuchte Ereignis verworfen oder an die nächste Triggerstufe weitergeleitet wird. Der CTP generiert ein Signal *Level1Accept*, das diese Entscheidung repräsentiert und vom sogenannten *Timing, Trigger and Control system* (TTC) an die einzelnen Auslesesysteme verteilt wird, die die Trigger-Entscheidung benötigen.

Der Datenpfad vom Detektor zum Level-1-Trigger ist vom Pfad getrennt, der zum Level-2-Trigger und schließlich zur Speicherung führt. Um die Daten, auf deren Grundlage der Level-1-Trigger seine Entscheidung fällt, gegen die Daten voller Kalorimeter-Auflösung vergleichen zu können, ist es erforderlich, den Level-1-Trigger auszulesen. Diese Datenauslese ist zudem notwendig, um die korrekte Funktion des Level-1-Triggers überwachen und etwaige Fehler finden zu können.

1.3.4 Das Front-End des Kalorimeter-Level-1-Triggers

Das Front-End des Level-1-Triggers stellt die Schnittstelle zwischen Detektor und Trigger-Logik dar. Einer der Schwerpunkte der am Institut für Hochenergie-Physik in Heidelberg für ATLAS durchgeführten Arbeiten liegt bei diesem Front-End.

---

[6] *Central Trigger Processor*

Noch auf dem Detektor findet die analoge Summation der Daten statt, als deren Ergebnis 8000 Eingangskanäle für den Kalorimeter-Level-1-Trigger zu Verfügung stehen. Die Übertragung vom Detektor zum Trigger-System über eine Entfernung von bis zu 80 m findet auf elektrischem Weg mit analogen Signalen statt. Auch die Möglichkeit der Übertragung auf optischem Weg wurde untersucht [11].

Nach der Übertragung werden die analogen Daten im Front-End des Level-1-Triggers durch einen *Flash Analog Digital Converter* (FADC) digitalisiert. Die maximale Pulshöhe entspricht etwa 300 GeV. Dieser Bereich wird auf einen 8 Bit-Wert abgebildet.

Der transversale Anteil der in den einzelnen Kalorimeterzellen deponierten Energie ist abhängig vom Winkel, in dem die Zelle im Verhältnis zur Strahlrichtung vom Wechselwirkungspunkt aus liegt. Außerdem enthält das Signal einen konstanten Untergrund. Deshalb werden die digitalisierten Daten im Front-End auf transversale Energie kalibriert und die Nullinie wird eingestellt. Dies geschieht, indem jedem digitalisierten Wert über eine für jeden Kanal verschiedene Tabelle ein neuer Wert zugeordnet wird. Dieses Prinzip wird durch eine sogenannte *Look-Up Table* (LUT) technisch realisiert.

Die Kalorimeter-Pulse, die der Level-1-Trigger erhält, erstrecken sich über mehrere Bunch-Crossings. Es ist deshalb erforderlich, ihr Maximum zu bestimmen und es dem richtigen Bunch-Crossing zuzuordnen, bevor die Daten an die eigentliche Trigger-Logik weitergegeben werden. Diese Aufgabe erfüllt die *Bunch Crossing Identification* (BCID).

Die Aufgaben des digitalen Teils des Level-1-Trigger-Front-End lassen sich wie folgt zusammenfassen:

- Kalibration der Daten auf transversale Energie
- Subtraktion des konstanten Untergrundes
- Zuordnung des Pulsmaximums zum richtigen Bunch-Crossing (BCID).

Außerdem muß das digitale Front-End folgendes leisten:

- Serialisierung des Datenstroms für die Übertragung zum Triggerprozessor
- Schnelle Auslese der Trigger-Rohdaten zu Kalibrations- und Testzwecken.
- Bereitstellung von Testmöglichkeiten.

Wie in Abschnitt 1.3.3 erwähnt ist es erforderlich, eine Auslesemöglichkeit des Level-1-Triggers zu besitzen. Insbesondere die Auslese des Front-Ends ist wichtig, da hier die Rohdaten, auf deren Grundlage der Level-1-Trigger die seine Entscheidungen trifft, zu Verfügung stehen. Wesentliches Ziel dieser Diplomarbeit war es, für ein Demonstrations-Modul des digitalen Front-Ends diese Auslese zu implementieren.

Kapitel 2

Das Level-1-Trigger Demonstrations-System

Die Entwicklung der Elektronik von ATLAS erfolgt in mehreren Schritten. Bevor das finale System hergestellt werden kann, müssen Konzepte und Vorläufer der benötigten Komponenten getestet werden und ihre Funktionsfähigkeit muß gezeigt werden. Die Entwicklung der Elektronik des Level-1-Triggers befindet sich gegenwärtig in einer solchen Demonstrations-Phase.

Das existierende Demonstrations-System [12] für den Level-1-Trigger deckt den Bereich ab, der in Abbildung 1.9 als Front-End bezeichnet ist, sowie einen Teil der Trigger-Logik, der zur Identifizierung von Elektronen und Photonen dient. Das Hauptaugenmerk liegt auf den Komponenten, die zur Datenübertragung innerhalb des Systems dienen. In Abbildung 2.1 ist das Demonstrations-System dargestellt.

Es besteht aus drei Hauptkomponenten, dem FADC-Modul und dem Transmitter-Modul (TXM), die zum Front-End gehören, und dem Cluster-Processor-Modul (CPM), das zur Trigger-Logik gehört. Ein Satz von Modulen (FADC,TXM,CPM) bearbeitet vier Trigger-Kanäle. Über ein weiteres Modul, das *Timing Control Module* wird ein System-Takt und ein Trigger-Signal zu Verfügung gestellt. Alle Module sind als Standardmodule für ein VME-Bus-System ausgeführt. Das CPM verfügt zusätzlich über ein eigenes Leitungs-System, das zur Übertragung von Daten zwischen verschiedenen CPMs dient. In den folgenden Abschnitten sollen die einzelnen Module beschrieben werden, die im Level-1 Trigger Demonstrationssystem zum Einsatz kommen.

2.1 FADC-Modul und Transmitter-Modul

Das FADC-Modul [13] dient zur Digitalisierung der analog übertragenen Kalorimeter-Daten. Es wird auf ein 8 Bit-Wort ein dynamischer Bereich von 256 GeV abgebildet. Der kleinste darstellbare Energiewert, auch *Least Significant Bit* (LSB) genannt, ist demnach 1 GeV. Die Daten werden mit der Bunch-Crossing-Rate von 40 MHz digitalisiert und parallel ausgegeben, so ergibt sich eine Datenrate von 40 MByte/s auf jedem Trigger-Kanal. Das FADC-Modul verfügt über einen Playback-Memory, das über den VME-Bus beschrieben werden kann. Statt der digitalisierten Analog-Daten können dann Daten aus

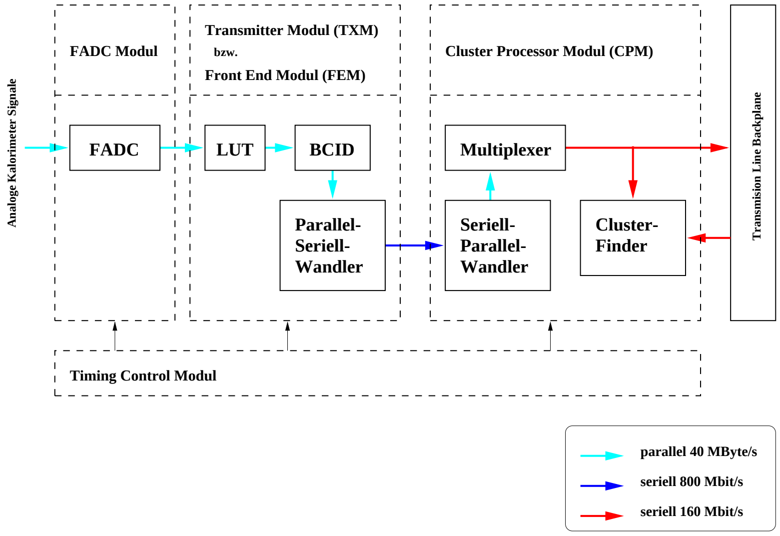


Abbildung 2.1: Demonstrations-System für den ATLAS Level1-Trigger

diesem Speicher ausgegeben werden.

Nach der Digitalisierung wird auf dem TXM die Energie-Kalibration durch eine LUT sowie das BCID durchgeführt. Danach haben die Daten die von der Trigger-Logik benötigte Struktur.

Digitalisierung und Vorverarbeitung der Daten für die Trigger-Logik werden räumlich sehr nah beieinander durchgeführt. Zur Trigger-Logik muß jedoch eine gewisse Strecke überwunden werden. Eine parallele Übertragung der Daten würde im finalen System 64000 Leitungen erfordern. Deshalb überträgt man die Daten seriell statt parallel. Ein kommerziell erhältlicher Chip-Satz von Hewlett-Packard leistet die nötige Umwandlung zwischen parallelem und seriellem Format. Er wird in Abschnitt 2.3.2 beschrieben. Durch Serialisierung und Übertragung von zwei Kanälen auf einer Leitung wird die Zahl benötigter Leitungen um einen Faktor 16 reduziert. Auf jeder seriellen Leitung werden dann 640 Mbit/s übertragen.

2.2 Cluster-Processor-Modul

Auf dem Cluster-Processor-Modul wird nach Energiedepositionen in wenigen angrenzenden Kalorimeter-Zellen gesucht, die von Elektronen oder Photonen stammen. Die Identifikation dieser *Cluster*, das *Cluster-Finding*, erfordert einen zweidimensionalen Algorithmus, bei dem Daten von Trigger-Kanälen, die zu aneinandergrenzenden Kalorimeter-Zellen gehören, untereinander ausgetauscht werden.

Diese Querkommunikation zwischen Komponenten, die verschiedene Trigger-Kanäle verarbeiten, stellt eine der Hauptschwierigkeiten beim Design des Level-1-Triggers dar. Sie muß sehr schnell erfolgen, da für die Verarbeitung eines Ereignisses nur die Zeit von 25 ns zwischen zwei Bunch-Crossings zu Verfügung steht. Das Volumen der erforderlichen Kommunikation wächst sehr schnell mit abnehmender Zahl von Kanälen, die auf

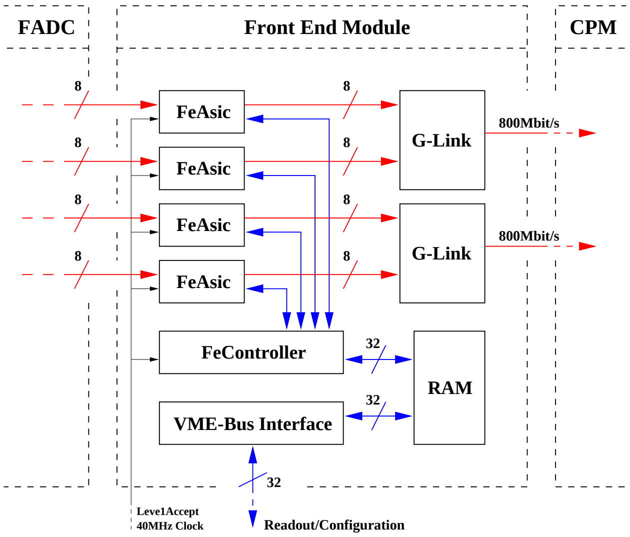


Abbildung 2.2: Das Front-End-Modul

einer verarbeitenden Einheit integriert werden. Deshalb ist eine hohe Integrationsdichte wünschenswert. Dieser sind jedoch technische Grenzen gesetzt, insbesondere durch die Zahl der Eingangskanäle.

Im vorhandenen Demonstrations-System sind die CPMs verschiedener Kanäle über eine spezielle Leiterplatte miteinander verbunden, die *Transmission Line Backplane*. Auf dieser Backplane befindet sich eine große Anzahl von Leitungen, auf denen Signale mit einer Datenrate von 160 Mbit/s zwischen einzelnen Cluster-verarbeitenden Elementen übertragen werden. Diese Rate wird durch Multiplexen aus der Rate von 40 MByte/s gewonnen, die nach der Seriell-Parallel-Wandlung auf dem CPM anliegt.

2.3 Das Front-End-Modul

Die Auslese des Front-Ends des Level-1-Triggers ist ein wichtiger Aspekt des Trigger-Systems. Um auch diese Funktion bereitzustellen, wurde ein neues Demonstrations-Modul entwickelt, das *Front-End-Modul* (FEM). Es verarbeitet vier Trigger-Kanäle und kann über VME-Bus konfiguriert und ausgelesen werden. Im Rahmen des Demonstrations-Systems ersetzt es das Transmitter-Modul und bietet zusätzlich die Möglichkeit der Auslese der Daten über den VME-Bus. Abbildung 2.2 zeigt ein Blockdiagramm des FEMs.

Auf dem Modul befindet sich für jeden der vier Kanäle ein ASIC, der *Front-End Asic* (FeAsic) [14], der LUT und BCID für einen Trigger-Kanal enthält. Zur Serialisierung der Daten für die Übertragung zum CPM wird ein kommerziell erhältlicher Chip von Hewlett-Packard eingesetzt, der HDMP-1012, in der Abbildung als G-Link bezeichnet. Über diese beiden Komponenten, den FeAsic und den G-Link verläuft der schnelle Datenpfad von der Digitalisierung zur Trigger-Logik.

Die Auslese des FEMs findet auf einem langsameren Datenpfad über den VME-Bus statt. Um die Verbindung zwischen FeAsics und VME-Bus herzustellen ist eine weitere

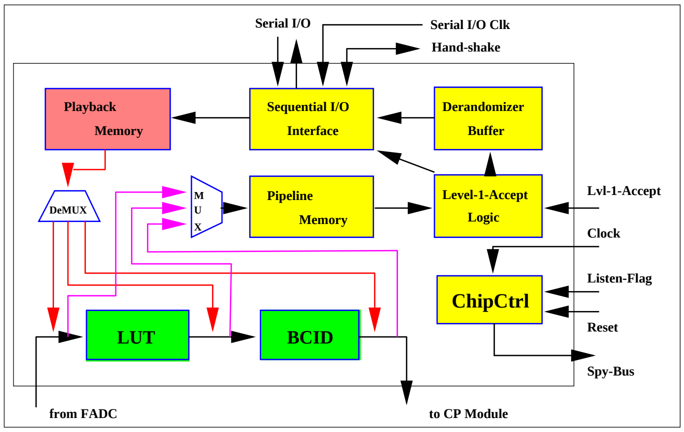


Abbildung 2.3: Blockdiagramm der wesentlichen Komponenten des FeAsic [14]

Komponente erforderlich, der *Front-End-Controller* (FEC). Zur Zwischenspeicherung der Daten ist ein Dual-Ported-RAM vorhanden, das sowohl vom VME-Bus, als auch vom FEC beschrieben und ausgelesen werden kann.

Auf dem FEM ist außerdem vor und hinter dem FeAsic eine Umsetzung der elektrischen Logik-Pegel erforderlich. Der Fe-Asic arbeitet mit TTL-Pegeln[1], vom FADC-Modul kommen ECL-Pegel[2]. Der G-Link benötigt PECL-Pegel[3].

In den folgenden Abschnitten sollen die einzelnen Komponenten des FEM beschrieben werden.

2.3.1 Front-End-ASIC

Der am Institut für Hochenergiephysik in Heidelberg entwickelte FeAsic [16] stellt LUT und BCID für einen Kanal zu Verfügung. Außerdem besitzt er eine serielle Schnittstelle über die er konfiguriert und ausgelesen wird. Abbildung 2.3 zeigt ein Diagramm des FeAsic. Er soll in einer endgültigen Version auch im finalen System zum Einsatz kommen.

Die LUT ist ein Speicher mit 8 Adressleitungen und einer Datenwortbreite von 10 Bit. Sie ersetzt die vom FADC kommenden 8 Bit-Worte durch ein neues 10 Bit-Wort. Dadurch daß das Ausgangswort breiter als das Eingangswort ist, besitzt man die Möglichkeit, Signale, die durch eine nichtlineare Zuordnung auf das 8 Bit-Wort komprimiert worden sind, auf ein 10 Bit-Wort abzubilden, dessen Zusammenhang mit dem Signal linear ist.

In der BCID-Einheit [17] wird der Energie-Puls zunächst durch einen FIR[4]-Filter mit konstanten Koeffizienten geschickt, der das Signal verschärft und die Flanken schmälert. Dann wird das Maximum gesucht. In dem zum Maximum gehörigen Datum wird der

---

[1] *Transistor-Transistor Logic.* Versorgungsspannung: 5 V, Signal High: 2,4 V bis 5 V, Signal Low: 0 V bis 0,4 V [15]

[2] *Emitter-Coupled Logic.* Versorgungsspannung: $-5,2$ V, Signal High: $-0,95$ V bis $-0,7V$, Signal Low: $-1,9$ V bis $-1,6$ V [15]

[3] *Positive ECL.* Um 5,2 V nach oben verschobene ECL-Pegel

[4] Finite Impulse Response

Wert des Maximums weitergegeben. Alle anderen Werte des Pulses, die zu anderen Bunch-Crossings gehören, werden auf Null gesetzt. Die Daten werden mit 8 Bit Breite ausgegeben und zur Trigger-Logik übertragen.

Der FeAsic besitzt einen internen Speicher, den *Pipeline-Memory.* Seine Größe beträgt 128 10 Bit-Worte. Er wird mit Daten aus dem schnellen Verarbeitungszweig zyklisch beschrieben. Ob die Daten dabei direkt nach dem Eingang, nach der LUT oder nach der BCID-Einheit entnommen werden, kann über einen Multiplexer gesteuert werden (siehe Abbildung 2.3). Der Pipeline-Memory kann im sogenannten *Stop-Modus* über die serielle Schnittstelle des FeAsic ausgelesen werden. Dabei wird mittels eines *Level1Accept*-Signals das Beschreiben des Speichers unterbrochen, und die zu diesem Zeitpunkt dort befindlichen Daten können ausgelesen werden.

In einem zweiten Auslesemodus, dem *kontinuierlichen Modus*, wird der Pipeline-Memory nicht angehalten. Bei jedem *Level1Accept*-Signal wird das zugehörige Datum in einen FIFO[5]-Speicher, den *Derandomizer Buffer* kopiert. Von dort kann es wiederum über die serielle Schnittstelle des FeAsic ausgelesen werden.

Der Derandomizer Buffer ist nötig, um statistische Fluktuationen des *Level1Accept*-Signals auszugleichen. Die Triggerfrequenz wird zwar im Mittel auf 100 kHz begrenzt, jedoch lediglich durch die Forderung, daß maximal 16 Triggersignale in 16 μs auftreten dürfen. Innerhalb dieser Grenze müssen Triggersignale untereinander nur einen Abstand von 75 ns besitzen. Treten die Triggersignale so dicht hintereinander auf, ist eine Auslese aus Geschwindigkeitsgründen nicht möglich. Deshalb müssen die auszulesenden Daten zwischengespeichert werden. Die nachfolgende Auslese muß dann nur die Forderung erfüllen, Daten mit der maximalen Triggerfrequenz von 100 kHz, das heißt innerhalb von 10 μs auslesen zu können.

Im kontinuierlichen Auslese-Modus ermöglicht der FeAsic es zusätzlich, die Daten von vier oder acht Bunch-Crossings aus der Umgebung des vom Trigger akzeptierten Bunch-Crossings auszulesen.

Zu Testzwecken ist auf dem FeAsic ein weiterer Speicher enthalten, das *Playback-Memory.* Er kann über die serielle Schnittstelle beschrieben werden und als Datenquelle für den schnellen Zweig des FeAsics dienen. Daten aus dem Playback-Memory können vor oder hinter der LUT oder hinter der BCID-Einheit eingekoppelt werden. So ist es möglich, einzelne Komponenten des FeAsics zu testen, oder definierte Daten an den Triggerprozessor zu geben.

Außerdem stellt der FeAsic noch einen acht Bit breiten Datenausgang zu Verfügung, über den interne Register beobachtet werden können. Dieser *SpyBus* ermöglicht es, die korrekte Funktionsweise des FeAsics auf einem direkten Weg zu beobachten. Durch eine drei Bit breite Adresse, kann gesteuert werden, welche Register über den SpyBus ausgegeben werden.

Der FeAsic besitzt zwei Reset-Leitungen, *StRegReset*, um den gesamten ASIC in einen definierten Zustand zu bringen und *SyncReset*, um die Adressen des internen Speicher auf voreingestellte Werte zurückzusetzen.

Die serielle Schnittstelle des FeAsics benötigt sechs Leitungen, einen Dateneingang und

---

[5] *First In First Out*

Name	Richtung	Beschreibung
Lvl1-Accept	Eingang	Level1Accept Trigger-Signal
Clock40MHz	Eingang	Systemtakt-Signal
FadcInput[0:7]	Eingang	Eingang für digitalisierte Daten
ToCPM[0:7]	Ausgang	Ausgang für verarbeitete Daten
ListenFlag	Eingang	Selektiert Richtung der seriellen Schnittstelle
SyncReset	Eingang	Reset der Adresszähler der internen Speicher
StRegReset	Eingang	Reset der internen Ablauflogik
SerialInp	Eingang	Serieller Eingang
SerialOut	Ausgang	Serieller Ausgang
SerialClk	Eingang	Takt für die serielle Schnittstelle
SerialAck	Eingang	Handshakesignal an FeAsic
SerialRdy	Ausgang	Handshakesignal vom FeAsic
SpyBusCtrl[0:2]	Eingang	Steuersignal für SpyBus
SpyBus[0:7]	Ausgang	SpyBus Ausgabe

Tabelle 2.1: Ein-und Ausgangssignale des FeAsic

einen Datenausgang, eine Taktleitung, zwei Handshake[6] -Leitungen und eine Leitung, die steuert, ob der FeAsic Daten sendet oder empfängt. Dadurch, daß sowohl eine Eingangs- als auch eine Ausgangsleitung vorhanden sind, lassen sich mehrere FeAsics hintereinander zu einer sogenannten *Daisy-Chain* zusammenschalten. Die separate Taktleitung ermöglicht es, die serielle Schnittstelle asynchron zum Betriebstakt des FeAsics zu benutzen. In Tests des FeAsics mit Hilfe des HP82000-Chip-Testers, der im ASIC-Labor Heidelberg zu Verfügung steht, konnte gezeigt werden, daß die serielle Schnittstelle mit Taktraten bis zu 140 MHz betrieben werden kann [18].

In Tabelle 2.1 werden die Ein-und Ausgangssignale des FeAsics zusammengefaßt. Der FeAsic wurde in einem 0,7 μm CMOS-Prozess gefertigt. Er enthält etwa 3000 Gatter und 1,2 kByte RAM. Die Chipfläche beträgt 16,4 mm^2.

2.3.2 Hewlett-Packard Gigabit-Link

Eine verbreitete Methode, Daten für ihre Übertragung zu bearbeiten, stellt die Parallel-Seriell-Wandelung dar. Die in einem parallelen Format vorliegenden Daten werden in ein serielles Datenformat umgewandelt, übertragen und anschließend wieder in das parallele Format zurückgewandelt. Dadurch kann die Anzahl der benötigten Leitungen reduziert werden. Aus Kosten- und räumlichen Gründen stellt dies oft die einzige Möglichkeit dar, große Datenmengen zu übertragen.

Für die Datenübertragung vom Front-End zum Level-1-Trigger wird ein kommerziell erhältlicher Chipsatz von Hewlett-Packard eingesetzt, der HDMP-1012 Sender, und der HDMP-1014 Empfänger [19], auch Gigabit-Link oder G-Link genannt. Der Sender wandelt

---

[6] Über sogenannte Handshake-Leitungen tauschen zwei kmmunizierende Komponenten Informationen aus, ob sie bereit sind, Daten zu empfangen oder zu senden. Dadurch ist eine asynchrone Kommunikation möglich, ohne daß Daten verloren gehen.

einen parallelen Datenstrom von 16 oder 20 Bit Breite in einen seriellen Strom um, der dann elektrisch oder optisch zum Empfänger übertragen werden kann, der die umgekehrte Wandlung vornimmt.

Bei der seriellen Übertragung kann der G-Link-Chipsatz in einem Bereich zwischen 150 Mbit/s und 1,6 Gbit/s Übertragungsrate arbeiten. Er benutzt ein synchrones Übertragungsprotokoll mit Fehlererkennung. Dazu werden für jedes parallele Wort vier zusätzliche Bits benötigt. 640 Mbit/s Nutzdatenrate entsprechen also 800 Mbit/s Übertragungsrate .

Der G-Link-Chipsatz wird in einem bipolaren Prozess gefertigt. Ein- und Ausgangssignale benutzen PECL-Pegel. Der Sender nimmt eine Leistung von 2,0 W auf, der Empfänger 2,6 W.

2.3.3 Front-End Controller

Der FEC befindet sich zwischen dem Dual-Ported-RAM, über das die Kommunikation des FEMs mit dem VME-Bus abläuft, und den vier FeAsics. Seine Aufgaben bei der Interaktion mit den FeAsics sind:

- Schnelle Auslese bei Triggerraten von bis zu 100 kHz
- Laden der LUT und des Playback-Memorys und Konfiguration der FeAsics
- Generierung von zusätzlichen Trigger-Signalen zu Testzwecken
- Auslese des SpyBus

Bei der Interaktion mit dem VME-Bus über das Dual-Ported RAM muß der FEC folgendes leisten:

- Cross-Check und Formatierung der ausgelesenen Daten
- Einlesen der Daten, die an den FeAsic gesendet werden sollen
- Steuerung über Kommandos
- Bereitstellung von Statusinformationen

Der FEC wurde auf einem FPGA der Firma Xilinx realisiert. FPGAs und der Prozeß der Schaltungsentwicklung werden in Kapitel 3 beschrieben. In Kapitel 4 wird ausführlich auf die Funktionsweise und Realisierung des FECs eingegangen.

Kapitel 3

FPGA-Design

Die Logik für den FEC wurde in der Hardware-Beschreibungssprache Verilog entwickelt und auf einem FPGA der Firma Xilinx vom Typ XC4000 implementiert. FPGAs und der Design-Prozeß sollen in diesem Kapitel von einem allgemeinen Standpunkt aus beschrieben werden.

3.1 Der XC4000-FPGA von Xilinx

3.1.1 Das Prinzip des FPGA

Ein FPGA (*Field Programmable Gate Array*) ist ein Logikbaustein, dessen Funktion programmiert werden kann. Im Gegensatz zu beispielsweise ASICs[1] wird die Funktion nicht bei der Herstellung des Chips festgelegt, sondern kann vom Anwender programmiert und sogar während des Betriebs reprogrammiert werden. So wird es möglich, schnell Änderungen der Schaltung umzusetzen und Fehler zu beheben. Da neben den Bauelementen, die die eigentliche Funktion zu Verfügung stellen auch noch eine erhebliche Menge an Bauelementen nötig ist, um die Programmierbarkeit zu ermöglichen, verbrauchen FPGAs im Vergleich zu ASICs wesentlich mehr Chip-Fläche, sind langsamer und in hohen Stückzahlen teurer. Für Anwendungen, deren Anforderungen an Platzverbrauch und Geschwindigkeit durch FPGAs erfüllt werden können und die nur geringe Mengen an Chips benötigen, ist der Einsatz von FPGAs günstig. Für das vorliegende FEM war somit die Realisierung des FECs auf einem FPGA die ideale Lösung, da nur einzelne Boards benötigt werden, der FEC speziell auf diese Aufgabe ausgelegt ist und die Anforderungen an seine Geschwindigkeit nur mäßig sind.

Im Prinzip besteht ein FPGA aus drei Komponenten: Logikblöcken, in denen logische Funktionen und Flip-Flops[2] zu Verfügung gestellt werden, Ein-/Ausgabeblöcken, die die Pins des Chips mit der internen Logik verbinden, sowie einem Verbindungsnetzwerk, das die Blöcke untereinander verbindet. Die Komponenten sind in einer zweidimensionalen Matrix auf dem Chip angeordnet (Abbildung 3.1).

---

[1] Application specific integrated circuit

[2] *Flip-Flops* sind digitalelektronische Bauelemente, die die Speicherung von einem Bit Information ermöglichen.

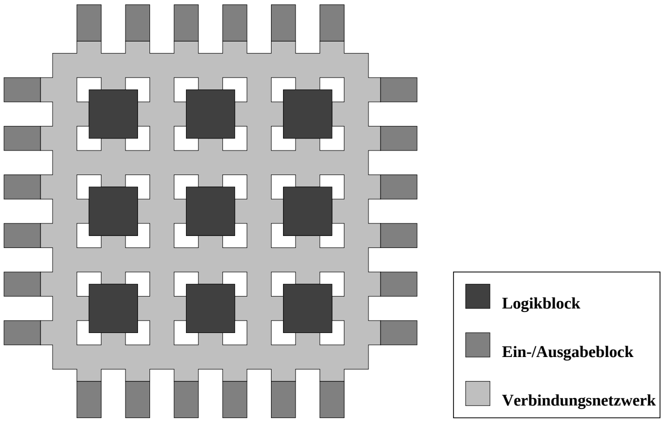


Abbildung 3.1: Prinzipieller Aufbau eines FPGAs

Ein Hauptproblem bei der Programmierung von FPGAs ist, neben der Abbildung der Schaltungslogik auf die Logikblöcke, die Vernetzung der Blöcke. Im Prinzip sind zwar alle Blöcke miteinander vernetzbar, jedoch ist die Anzahl der Verbindungen beschränkt und durch die Schaltbarkeit der Verbindungen entstehen Signalverzögerungen. Es ist somit erforderlich, eine möglichst geschickte Aufteilung der Schaltung auf die Logikblöcke zu finden, die die Verbindungen kurz hält, aber trotzdem mit Hilfe der vorhandenen Ressourcen alle Verbindungen herstellen kann.

Um dieses Problem zu vereinfachen, gibt es auf dem Xilinx-FPGA weitere Elemente, die zum Beispiel die Verteilung eines Takt- oder Resetsignals ermöglichen, ohne die Ressourcen des konfigurierbaren Verbindungsnetzwerks zu benutzen. Dennoch bleibt die Abbildung der Schaltung auf die Ressourcen des FPGAs eine anspruchsvolle Aufgabe, die von komplexer Software erledigt wird. Sie ist jedoch, gerade bei großen und komplexen Designs, auf Vorgaben des Designers angewiesen, um eine effiziente Umsetzung der Schaltung zu erreichen. Es kann auch geschehen, daß Designs, obwohl sie von den Ansprüchen an die Logik-Ressourcen implementierbar wären, nicht vernetzbar sind. Aus diesem Grund wird man die Ressourcen von FPGAs in der Regel nicht vollständig ausnutzen können.

3.1.2 Die FPGA-Familie XC4000 der Firma Xilinx

Die Firma Xilinx brachte 1985 den ersten FPGA auf den Markt. Sie gehört auch heute noch zu den Marktführern. Sie bietet mit der 4000-Serie einen komplexen FPGA an, der im folgenden näher beschrieben werden soll. Für den FEC wird ein XC4010E-3 eingesetzt.

Der Aufbau des XC4000 [20] folgt dem in Abschnitt 3.1.1 beschriebenen. Die Logikblöcke werden bei Xilinx CLB (*Configurable Logic Block*), die Ein-/Ausgabeblöcke IOB (*Input/Output Block*) genannt. Die Konfiguration des XC4000 geschieht über statische RAM-Zellen. Dies bedeutet, daß er jederzeit wieder gelöscht und reprogrammiert werden kann, ohne daß er aus der fertigen Schaltung entfernen werden müßte. Jedoch verliert er seine Konfigurations-Informationen mit Abschalten der Spannungsversorgung.

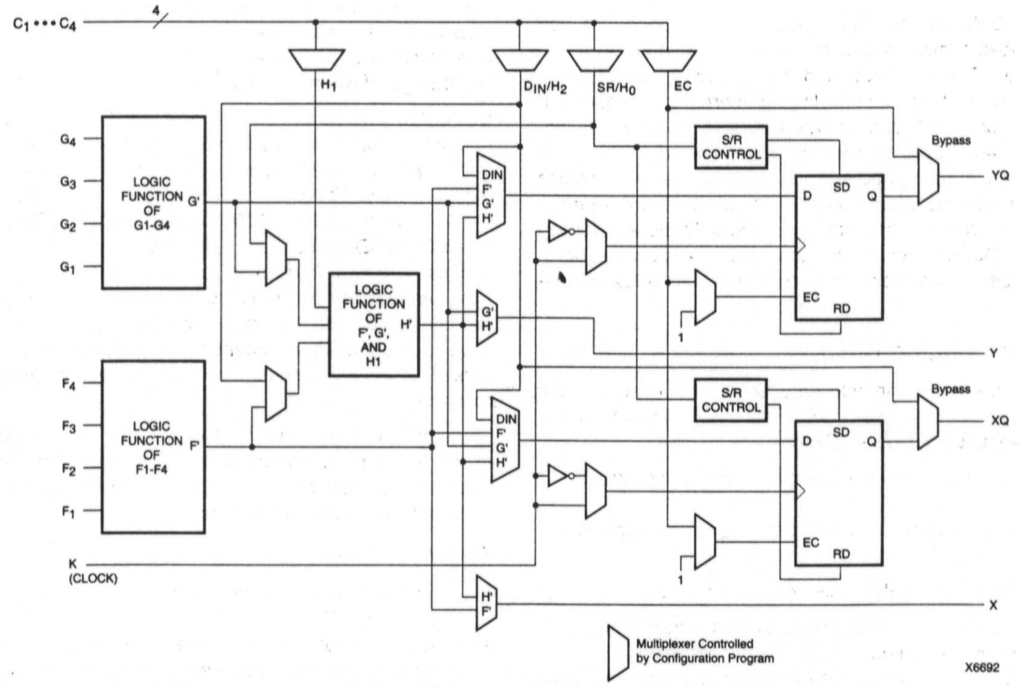


Abbildung 3.2: Vereinfachtes Blockdiagramm eines XC4000-CLB [20]

Der Logikblock des Xilinx XC4000

Der CLB (Abbildung 3.2 [20]) stellt die eigentliche Funktion des FPGAs zu Verfügung. Er enthält sogenannte *Funktionsgeneratoren* zur Implementierung von logischen Gattern sowie Flip-Flops zur Speicherung von Daten.

Die Funktionsgeneratoren sind als statisches RAM implementiert, in denen die Wahrheitstabelle der gewünschten logischen Funktion abgelegt wird. Die Argumente der Funktion werden als Adresse an das RAM angelegt, worauf das Ergebnis als Datenwert ausgegeben wird. Die beiden Funktionsgeneratoren F und G ermöglichen es, jeweils eine beliebige Funktion von vier Eingangssignalen zu erzeugen. Über den Funktionsgenerator H lassen sich diese beiden und ein weiteres Signal von außen verknüpfen. Die Ausgangssignale der Funktionsgeneratoren lassen sich entweder als Eingangssignale der beiden Flip-Flops im CLB verwenden oder nach außen führen. Es ist möglich, Funktionsgeneratoren und Flip-Flops in gewissen Grenzen unabhängig voneinander zu benutzen.

Die Flip-Flops, die im CLB enthalten sind, sind D-Flip-Flops, die mit positiver oder negativer Taktflanke betrieben werden können. Sie haben zudem eine Clock-Enable-Leitung, mit der geregelt wird, ob sie auf den Takt reagieren, ohne daß eine logische Verknüpfung mit dem Taktsignal nötig wäre.

Eine Besonderheit der Funktionsgeneratoren ist, daß das statische RAM, in dem die logische Funktion abgelegt wird, auch als Speicher verwendbar ist. So lassen sich in einem CLB bis zu 32 Bit speichern.

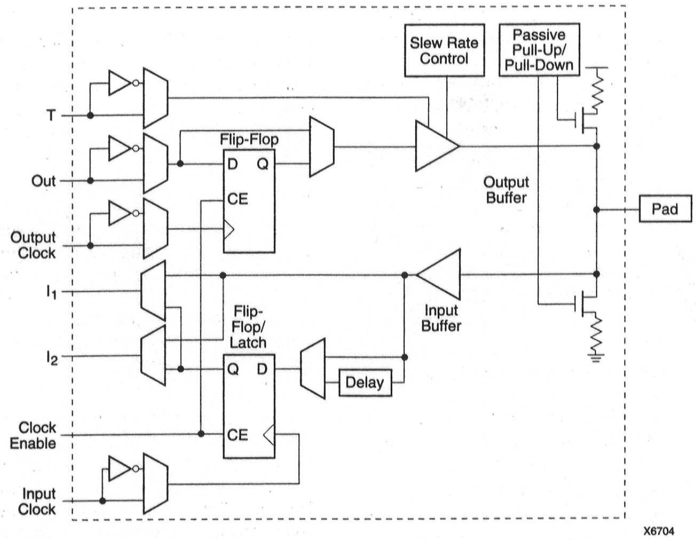


Abbildung 3.3: Vereinfachtes Block-Diagramm eines XC4000-IOB [20]

Der Ein-/Ausgabeblock des Xilinx XC4000

Die Verbindung der Pins des Chips zur internen Logik geschieht über jeweils einen IOB (Abbildung 3.3 [20]). Er ermöglicht es, die Pins als Eingang, Ausgang oder bidirektional zu betreiben. Zudem enthält der IOB zwei Flip-Flops, über das die Synchronisation von Ein- oder Ausgangssignalen mit einem internen Takt ermöglicht wird.

Das Verbindungsnetzwerk

Zwischen den CLBs liegen mehrere Leitungen verschiedener Länge, die vom CLB aus erreicht werden können. Es gibt Leitungen einfacher Länge, die in einer Matrix geschaltet werden, die dieselbe Auflösung hat wie die Matrix der CLBs, Leitungen doppelter Länge und Leitungen voller Länge, die von einer Seite des Chips bis zur anderen laufen. Zusätzlich gibt es noch ein gesondertes Leitungsnetz, das aus speziellen Logiktreibern, genannt *Global Clock Buffer*, versorgt wird, und über das alle CLBs mit nahezu gleicher Signalverzögerung erreichbar sind. Dieses Netz bietet sich daher zur Verteilung von Takt- oder anderen globalen Signalen an. Es erlaubt, alle Flip-Flops des FPGAs mit einem Taktsignal gleicher Phase zu versorgen. Dies ist eine wichtige Voraussetzung für ein sicheres synchrones Digitaldesign. Außerdem bietet das Verbindungsnetzwerk noch eine gesonderte globale Reset-Leitung an, die alle Flip-Flops auf einmal zurücksetzen kann.

Ein Problem bei der Implementierung von arithmetischen Funktionen wie Zählern oder Addierern ist die Übertragung eines möglichen Überlaufs. Durch die hierfür benötigte Zeit wird die Geschwindigkeit der Schaltung bestimmt. Da die Signale auf normalen Leitungen des Verbindungsnetzwerkes durch die vielen Schaltpunkte nicht besonders schnell sind, gibt es spezielle Leitungen, die senkrecht benachbarte CLBs verbinden und für die Übertragung eines Überlaufbits vorgesehen sind. Sie werden *Fast Carry Logic* genannt. Durch sie stei-

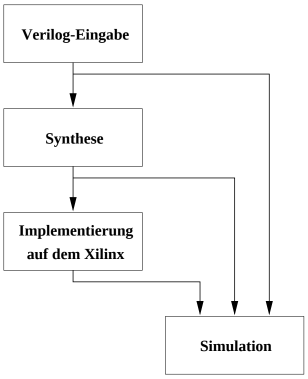


Abbildung 3.4: Der Design-Prozeß

gert sich die Leistung von manchen arithmetischen Funktionen wie zum Beispiel Zählern beträchtlich.

Die Konfiguration

Die statischen RAM-Zellen des XC4000 verlieren ihren Inhalt mit Abschalten der Versorgungsspannung. Es ist somit erforderlich, bei jedem Einschalten des XC4000 die Logikblöcke und das Verbindungsnetzwerk neu zu konfigurieren. Dies geschieht, indem die Konfigurationsdaten, die die Funktion des FPGAs programmieren, an den XC4000 gesendet werden. Dabei benötigt jeder CLB etwa 350 Bits an Daten, um die Logik und die dazugehörigen Verbindungen herzustellen.

Der XC4000 kann in verschiedenen Betriebsarten konfiguriert werden. Er kann die Konfigurationsdaten aktiv aus einem parallelen oder seriellen Speicher holen, oder passiv seriell oder parallel Daten empfangen. Es ist möglich, den XC4000 an den Bus eines Prozessors anzuschließen, um ihn über gewöhnliche Speicherzugriffe zu konfigurieren.

3.2 Der Design-Prozeß

Im folgenden soll die Schaltungsentwicklung mit Hilfe der Hardwarebeschreibungssprache Verilog, Xilinx XC4000-FPGAs als Zieltechnologie und der CAD[3]-Software Cadence als Software-Werkzeug beschrieben werden.

Der Design-Prozeß läuft in drei Stufen ab (Abbildung 3.4). Am Anfang steht die Design-Eingabe in Verilog. Hier wird zunächst das Verhalten einer Schaltung spezifiziert, die die gestellten Anforderungen erfüllt. Dies erfolgt auf einer abstrakten Ebene. Es werden keine konkreten Schaltungselemente wie Gatter oder Flip-Flops verwendet. In der zweiten Stufe, der Synthese, wird die abstrakte Verilog-Beschreibung weitgehend automatisch in einen Schaltplan umgewandelt, der aus den im FPGA vorhandenen Schaltungselementen zusammengesetzt ist. In der dritten Stufe, dem *Place and Route*[4], wird, wiederum

---

[3] *Computer Aided Design*

[4] Plazieren und Verbinden der Logik

weitgehend automatisch, der Schaltplan auf den ausgewählten Ziel-FPGA abgebildet, die Schaltungselemente werden auf die Logikblöcke verteilt und über das Verbindungsnetzwerk miteinander verbunden. Nach dieser Stufe sind alle Informationen vorhanden, um den FPGA zu konfigurieren, und die Datei, die in den FPGA geladen wird, kann erzeugt werden.

Ein wichtiger Teil des Design-Prozesses ist die Überprüfung der erzielten Ergebnisse, sowohl bei der Design-Eingabe als auch beim endgültigen Ergebnis. Dies wird durch Simulation der Schaltung bzw. Schaltungsbeschreibung erreicht. Sie findet in allen drei Stufen des Design-Prozesses statt.

3.2.1 Design-Eingabe in Verilog

Um eine Schaltung zu erstellen, hat man die Möglichkeiten aus vorhandenen Elementen wie Transistoren, logischen Gattern, Flip-Flops oder auch Speichern, Prozessoren, spezialisierten hochintegrierten Bauelementen die gewünschte Funktionalität aufzubauen. Die Aufgabe ist dann mit der Zeichnung eines Schaltplanes gelöst. Für kleinere Schaltungen ist diese Vorgehensweise durchaus angemessen und effektiv. Bei größeren Schaltungen, wie sie durch die Verfügbarkeit von Technologien, die eine sehr hohe Integration von Bauelementen zulassen, möglich werden, gerät man an die Grenzen dessen, was Menschen ohne technische Hilfe überschauen können.

Die Lösung dieses Problems besteht nun darin, daß man die Schaltung nicht mehr konkret in jedem einzelnen Bauelement, sondern ihr Verhalten in einer abstrakteren Sprache, einer sogenannten *Hardware Description Language* (HDL) beschreibt. Die Umsetzung der abstrakten Beschreibung in einen Schaltplan wird durch geeignete Computer-Programme vorgenommen. Je nach Abstraktionsgrad der Beschreibungssprache kann die Umsetzung, oder *Synthese*, einer Schaltung beträchtliche Komplexität erreichen. Während die Minimierung boolscher Gleichungen ein recht einfaches Problem ist, kann das Kodieren einer State-Machine[5] Synthese-Software vor unlösbare Probeleme stellen. Deshalb ist es erforderlich, bei der Design-Eingabe bereits auf die Umsetzbarkeit in Schaltungstechnik zu achten und der Synthese-Software anhand von weiteren Informationen über den Schaltungsaufbau Vorgaben zu machen, nach denen sie sich bei der Erstellung des Schaltplanes richten kann.

Die Hardware-Beschreibungssprache Verilog

Die Sprache *Verilog* [21][22] wurde entwickelt zu Entwurf, Beschreibung und Simulation digitaler Logik. Sie ermöglicht es auf einer von der tatsächlichen Implementation abstrahierten Ebene nahezu jede beliebige digitale Schaltung einschließlich Zeitverhalten zu beschreiben und zu simulieren.

Bei der Nachbildung von Schaltungen zu ihrer Simulation enwickelt Verilog seine volle Leistungsfähigkeit. Um es auch als Entwurfswerkzeug einsetzen zu können, muß

---

[5] Eine *State-Machine* ist ein Typ synchroner digitaler Schaltungen, der sich zu jedem Zeitpunkt in einem definierten Zustand befindet. Jeder Zustand entspricht einer bestimmten Konfiguration von Ausgangssignalen. Synchron zu einem vorgegebenen Takt wechselt die State-Machine ihren Zustand. Dabei ist die Abfolge der Zustände eindeutig als Funktion der Eingangssignale vorgegeben.

gewährleistet sein, daß aus dem Verilog-Code ein Schaltplan synthetisiert werden kann. Deshalb muß man sich an Konventionen und Einschränkungen halten, die von der Synthese-Software diktiert werden. Das hohe Abstraktionsniveau, das Verilog ermöglicht, bleibt jedoch zum Teil erhalten.

Die Philosophie von Verilog lehnt sich an die einer herkömmlichen prozeduralen, sequentiellen[6] Programmiersprache an. Der Unterschied besteht jedoch darin, daß nicht Variablen manipuliert, sondern Signale geschaltet werden, die Eigenschaften wie Laufzeiten haben und parallel verarbeitet werden müssen. Schaltelemente können unabhängig voneinander arbeiten, auch wenn sie Signale miteinander austauschen. Diese echte Parallelität muß in einer Hardware-Beschreibungssprache nachgebildet werden und das unterscheidet sie am stärksten von sequentiellen Programmiersprachen.

Verilog ist in seiner Semantik an prozedurale Programmiersprachen wie C angelehnt. Es gibt alle typischen Sprachelemente wie Variablen-Deklarationen und Zuweisungen, arithmetische und logische Operatoren, Schleifen-Konstrukte, Bedingungs-Abfragen, Unterprogramme, Ein- und Ausgabe-Routinen. Zusätzlich jedoch sind Sprachelemente erforderlich, die das parallele und ereignisgesteuerte Ausführen von Anweisungen ermöglichen, sowie Konstrukte zur Definition von Hardware-Eigenschaften wie Signalverzögerungszeiten.

Als Beispiel für eine Verilog-Beschreibung sei hier ein kurzes Programm vorgestellt, das ein 4-Bit-Schieberegister mit parallelem Ausgang und paralleler Lademöglichkeit, Clock-Enable[7] und asynchronem Reset beschreibt (Abbildung 3.5). Schieberegister solcher Art sind im Design des FEC mehrfach enthalten.

Das Beispiel beginnt mit der Deklaration des Moduls mit seinen Ein- und Ausgängen. Dann folgen die Deklaration des Schieberegisters und Zuweisungen an Ausgangssignale, die aus dem Register gespeist werden. Der erste `always`-Block definiert die Funktion des Schieberegisters, der zweite `always`-Block die Funktion des Reset-Signals. Die Form des Reset-Blockes ist genau festgelegt [23]. Nur in diesem Stil erkennt der Synthesizer, daß die Schaltung mit einem asynchronen Reset versehen werden soll. In einer Verilog-Simulation ohne die Notwendigkeit der Synthese eines Schaltplanes kann dieselbe Funktion auch anders beschrieben werden.

An diesem Beispiel erkennt man drei wesentliche Aspekte einer HDL-Beschreibung. Zum einen die Beschreibung einer synchronen Logik, die durch die positive Flanke des Taktsignals gesteuert wird (`@(posedge Clk)`) und auf Steuersignale wie *Load* oder *ClkEnable* reagiert. Dann die Beschreibung einer Operation durch ein Hochsprachensymbol <<, das den Shift-Operator darstellt, der den Registerinhalt verschiebt. Und schließlich die Beschreibung einer asynchronen Logik, der Reset-Logik, bei der über ein `if`-Konstrukt der Zustand des Schieberegisters in Abhängikkeit der Reset-Leitung definiert wird.

Dieses Code-Stück läßt sich simulieren. Die Simulation verläuft ereignisgesteuert und es gehen noch keine Informationen über das Zeitverhalten ein. Mit Eintreten des spezifizierten Ereignisses, beispielsweise der positiven Flanke des Taktsignals, werden gleichzeitig die davon abhängigen Signale geändert. Die logische Reihenfolge der Signale muß jedoch

---

[6] In einer sequentiellen Programmiersprache existiert eine eindeutige Reihenfolge, in der die Befehle eines Programms abgearbeitet werden.

[7] Als *Clock-Enable* wird ein Signal bezeichnet, das steuert, ob ein Flip-Flop oder eine Kombination aus Flip-Flops auf das Taktsignal reagiert.

```
module shiftreg (Out,ParOut,
                 Clk,ClkEnable,In,Load,ParIn,Reset);

  output Out;                  // Deklaration der Ausgaenge
  output [3:0] ParOut;

  input Clk,ClkEnable,In,Load; // Deklaration der Eingaenge
  input [3:0] ParIn;
  input Reset;

  reg [3:0] ShiftReg;   // Deklaration des Schieberegisters

  assign ParOut = ShiftReg;  // Signalzuweisungen
  assign Out = ShiftReg[3];


  always @(posedge Clk)   // synchroner Schaltungsteil
    if (ClkEnable)
      if (Load)
        ShiftReg = ParIn;
      else
        begin
          ShiftReg = ShiftReg << 1;  // schiebe Registerinhalt
          ShiftReg[0] = In;
        end


  always @(Reset)   // asynchroner Schaltungsteil
    if (Reset)
      assign ShiftReg = 0;
    else
      deassign ShiftReg;


endmodule
```

Abbildung 3.5: Verilog-Beschreibung eines 4-Bit-Schieberegisters berücksichtigt werden, das heißt Eingangssignale besitzen noch den Wert, den sie unmittelbar vor Eintreten des Ereignisses hatten. Dies entspricht der Realität in der späteren Schaltung, in der die Signale eine gewisse Zeit brauchen, um durch die Schaltungselemente zu laufen.

Man kann zum Beispiel zwei der 4 Bit-Schieberegister zu einem 8 Bit-Schieberegister aneinanderreihen, indem man den Ausgang des ersten Registers mit dem Eingang des zweiten verbindet und sie mit dem gleichen Taktsignal betreibt. Die Simulation ergibt nur dann das korrekte Ergebnis, wenn zum Zeitpunkt der Taktflanke der alte Wert des Ausgangs am Eingang des zweiten Registers anliegt und nicht der neue, da sonst das letzte Bit des ersten Schieberegisters in das erste Bit des zweiten Schieberegisters dupliziert und nicht verschoben würde.

Eine Simulation ohne Berücksichtigung von Laufzeiten reicht aus, um die logische Konsistenz und Funktion unter Optimalbedingungen zu überprüfen, jedoch nicht, um zum Beispiel eine maximale Taktfrequenz zu ermitteln. Dazu sind noch die Informationen aus der konkreten Hardware-Implementation nötig.

3.2.2 Synthese

Um von der Verilog-Beschreibung zu einem entsprechenden Schaltplan zu gelangen, muß ein Synthese-Schritt durchgeführt werden. Es existiert eine Bibliothek von Schaltungselementen, deren Funktionalität dem entspricht, was der Xilinx 4000 zu Verfügung stellt. Es sind zum Beispiel Gatter und Flip-Flops vorhanden. Mit dem Programm *Synergy* [24], das zu *Cadence* gehört, wird aus dem Verilog-Code ein Schaltplan generiert, der aus den Bibliotheks-Elementen aufgebaut ist, und dessen Verhalten äquivalent zum Verhalten der Verilog-Beschreibung ist.

Um zu zufriedenstellenden Ergebnissen zu gelangen, müssen neben der korrekten und synthesegerechten Beschreibung der Schaltung einige Vorgaben für die Synthese gemacht werden. Zum einen ist es zweckmäßig, die Hierarchie des Verilog-Codes zu erhalten, um die Überprüfung der Ergebnisse und die Fehlersuche zu erleichtern. Zum anderen ist es erforderlich, die Codierung der Zustände der State-Machines festzulegen.

Dies kann zum Beispiel erfolgen, in dem jedem Zustand eine binäre Zahl zugeordnet wird, deren Ziffern jeweils in einem Flip-Flop gespeichert werden. So wird die minimale Anzahl von Flip-Flops verwendet. Jedoch wird die Übergangslogik zwischen den Zuständen komplex, da jeder Zustand durch eine Kombination von mehreren Signalen repräsentiert wird, die alle in die Logik, die den nächsten Zustand erzeugt, eingehen müssen. Zudem kann es passieren, das beim Übergang zwischen zwei Zuständen, bei dem sich mehrere Zustandsbits ändern, ein oder mehrere unerwünschte Zwischenzustände auftreten, die möglicherweise kurzzeitige unerwünschte Ausgangssignale zur Folge haben. Beispielsweise kann ein Zustand, der durch 00 repräsentiert wird, beim Übergang zum Zustand 11, kurzzeitig den Wert 01 oder 10 annehmen, wenn sich nicht beide Bits exakt gleichzeitig ändern.

Eine alternative Codierung für State-Machines ist die sogenannte *One-Hot-Codierung.* Hier wird jedem Zustand ein eigenes Flip-Flop zugeordnet. Es gibt dann so viele Flip-Flops wie Zustände, und jederzeit ist nur eines aktiv. Dies erfordert mehr Flip-Flops als eine binäre Codierung, jedoch wird die Übergangslogik einfacher und damit die State-Machine

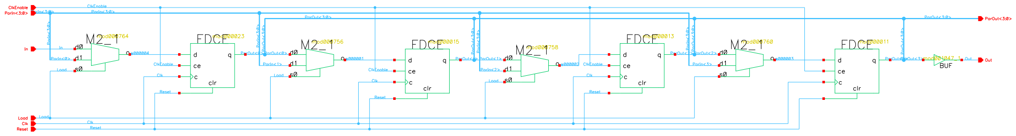


Abbildung 3.6: Schaltplan des synthetisierten 4-Bit-Schieberegisters

schneller. Da die Xilinx FPGAs sehr viele Flip-Flops enthalten, ist es am effizientesten, die *One-Hot-Codierung* zu verwenden,

Nach der Synthese, die bei größeren Designs durchaus Zeit in der Größenordnung von Stunden in Anspruch nehmen kann, erhält man einen Schaltplan. Im Beispiel des Schieberegisters aus Abschnitt 3.2.1 wird eine Schaltung (Abbildung 3.6) aus vier Flip-Flops generiert, deren Eingangsleitungen je nach Zustand der *Load*-Leitung über einen Multiplexer entweder mit dem parallelen Eingang oder dem vorhergehenden Flip-Flop verbunden sind. Um die Funktion des synthetisierten Designs zu überprüfen, kann der Schaltplan simuliert werden. Es sollten die gleichen Ergebnisse erzielt werden wie bei der Simulation des Verilog-Codes. Bei der Simulation des Schaltplanes sind jedoch schon Informationen über das Zeitverhalten enthalten. Gatter und Flip-Flops werden mit einer willkürlichen Durchlaufzeit versehen, die nicht der echten Durchlaufzeit entspricht. Diese wird erst im nächsten Schritt des Design-Prozesses ermittelt.

3.2.3 Implementierung der Schaltung auf dem FPGA

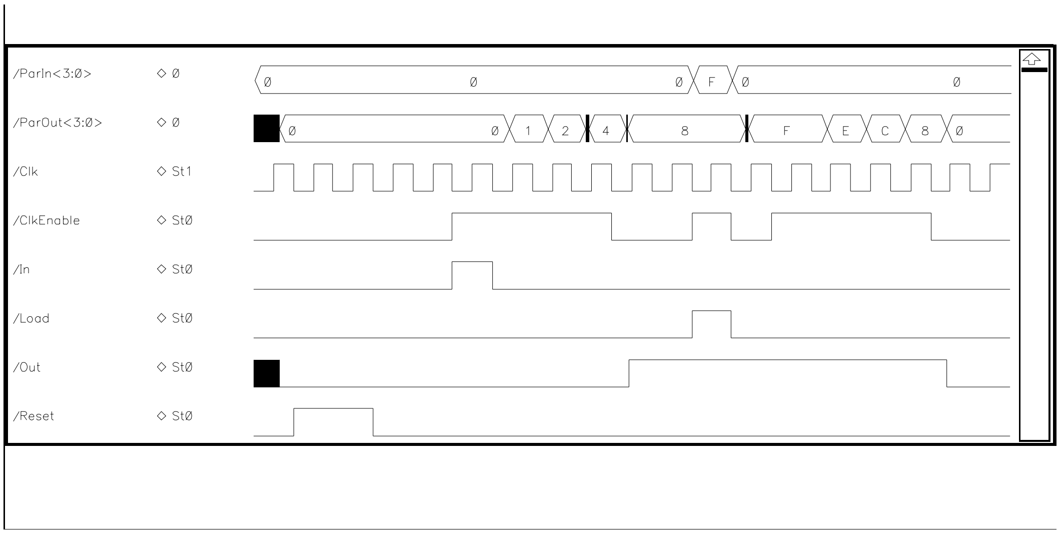
Um den Schaltplan in eine Konfigurations-Datei für den FPGA zu übersetzen, müssen die Elemente des Schaltplans auf die Logik-Blöcke des FPGAs verteilt werden, so daß sie sich über das Leitungsnetz des FPGAs korrekt und effektiv verbinden lassen. Dieser Schritt wird von einem Programm namens *ppr* (*Partitioning, placement and routing*[8]) durchgeführt. Außerdem werden die Signallaufzeiten analysiert und eine Datei erzeugt, die diese Informationen für die Simulation bereitstellt. Das Format dieser Datei, ist das *Standard Delay Format* (SDF). In diesem Schritt werden auch die genauen Einschränkungen für das fertige Design festgelegt und umgesetzt, wie beispielsweise die Zuordnung der Signale zu den Anschlüssen des FPGAs. Nach Durchlaufen dieser Stufe kann jetzt die Schaltung unter Berücksichtigung aller realen Laufzeiten simuliert werden. Es läßt sich eine maximale Taktfrequenz bestimmen, und das Verhalten der Simulation sollte dem Verhalten der tatsächlichen Schaltung entsprechen.

3.2.4 Simulation

Da bei dem gesamten Design-Prozeß keinerlei physikalische Hardware gebaut wird und sich die Schaltung lediglich als Modell im Computer befindet, kommt der Simulation eine besondere Bedeutung zu. Sie ist die einzige praktikable Möglichkeit, die Funktionsweise zu überprüfen und unvermeidliche Fehler bei der Entwicklung zu finden und zu beheben. Ein Werkzeug hierfür ist der Simulator Verilog-XL [21], der zu *Cadence* gehört.

---

[8] Aufteilen, Plazieren und Verbinden


/ParIn<3:0> ◇ 0
0
0
0
F
0
0
/ParOut<3:0> ◇ 0
0
0
1
2
4
8
F
E
C
8
0
/Clk ◇ St1
/ClkEnable ◇ St0
/In ◇ St0
/Load ◇ St0
/Out ◇ St0
/Reset ◇ St0

```
stlinit  ; initialisiere STL

defpin Clk              clk  ; Deklaration der Signale
defpin ClkEnable        in
defpin In               in
defpin Load             in
defpin ParIn<3:0>       in
defpin Reset            in
defpin Out              out
defpin ParOut<3:0>      out

deftiming .25ns 2.5ns 25ns  ; Definition Zeitskala
defclock ".....11111" Clk  ; Definition der Form des Taktsignals
defformat ClkEnable In Load ParIn Reset  ; Definition des Testvektors

deftest  ; Beginn des Testvektorabschnitts

xv(0 0 0 0 0)  ; Testvektoren
xv(0 0 0 0 1)
xv(0 0 0 0 1)
xv(0 0 0 0 0)
xv(0 0 0 0 0)
xv(1 1 0 0 0)
xv(1 0 0 0 0)
xv(1 0 0 0 0)
xv(1 0 0 0 0)
xv(0 0 0 0 0)
xv(0 0 0 0 0)
xv(1 0 1 0xf 0)
xv(0 0 0 0 0)
xv(1 0 0 0 0)
xv(1 0 0 0 0)
xv(1 0 0 0 0)
xv(1 0 0 0 0)
xv(0 0 0 0 0)
xv(0 0 0 0 0)

endtest  ; Ende des Testvektorabschnitts
```

Abbildung 3.8: STL-Programm zum Test des Schieberegister

Kapitel 4

Der Front-End-Controller

Der Front-End-Controller (FEC) stellt auf dem Front-End-Modul (siehe Abbildung 2.2) das Bindeglied zwischen FeAsics und VME-Bus dar. Seine Aufgabe liegt in der Interaktion mit den vier FeAsics auf der einen, und dem VME-Bus auf der anderen Seite. Die Kommunikation mit dem VME-Bus geschieht dabei über das Dual-Ported-RAM des FEMs.

Der FEC muß Daten aus den FeAsics auslesen, Eingabe- und Konfigurationsdaten an die FeAsics senden, sowie eine Möglichkeit zur Auslese des SpyBus der FeAsics und zur Generierung von zusätzlichen *Level1Accept*-Signalen bieten. Ein- und Ausgabedaten müssen bearbeitet werden. Gesteuert wird der FEC durch Befehle, die in den FEM-Speicher geschrieben werden. Rückmeldungen über den Betriebszustand des FEC erfolgen über ein Statusregister, das in den FEM-Speicher abgebildet wird. In Abbildung 4.1 sind die wesentlichen Funktionsblöcke des FEC dargestellt.

4.1 Kommunikation zwischen FeAsic und FEC

Datentransfers zwischen FeAsics und FEC finden über die serielle Schnittstelle der FeAsics statt. In einem Übertragungszyklus wird von jedem FeAsic ein 23 Bit-Wort gelesen oder geschrieben. Dabei erfolgt die Kommunikation mit allen vier FeAsics gleichzeitig. Einzelne oder mehrere FeAsics können jedoch ausmaskiert werden und werden dann nicht angesprochen.

Bei der Auslese eines FeAsics enthält das Auslesewort neben der Adresse und dem Inhalt der ausgelesenen Pipeline-Memory-Stelle Informationen über den Modus, in dem die Auslese stattfand, und einen möglichen Überlauf des Derandomizer Buffers. In Abbildung 4.2 ist das Format des Auslesewortes dargestellt. Die zwei Bits des *Loss Counters* (LC) geben an, wie viele Werte im kontinuierlichen Modus nicht vom Pipeline-Memory in den Derandomizer-Buffer geschrieben werden konnten und damit verlorengegangen sind. Dies kann geschehen, wenn der Derandomizer Buffer bereits voll ist, aber der FeAsic trotzdem noch weitere *Level1Accept*-Signale erhält. Die Bedeutung der zwei Modus-Bits ist in Tabelle 4.1 wiedergegeben[1].

---

[1] Zur Kennzeichnung von Binärzahlen wird hier die Konvention gewählt, die auch in der Hardwarebeschreibungssprache Verilog verwendet wird.

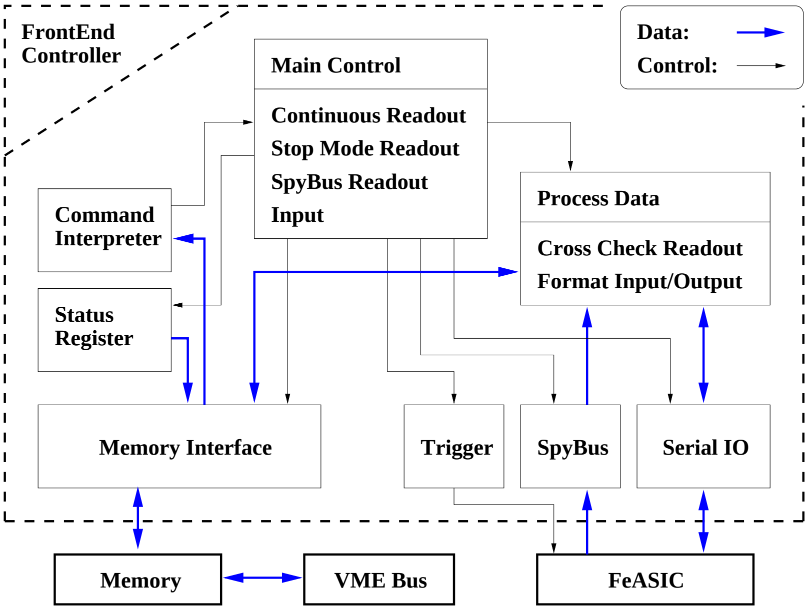


Abbildung 4.1: Blockdiagramm des FECs

Werden Daten zur Konfiguration an den FeAsic gesendet, besteht das Eingabewort aus einem 5 Bit-Token, einer 8 Bit-Adresse und einem 10 Bit-Datum (siehe Abbildung 4.2). Der FeAsic interpretiert das Token und setzt seine internen Register dem Ergebnis entsprechend. Für die meisten Tokens ist der Inhalt des restlichen Wortes nicht von Bedeutung. Das komplette Wort wird nur beim Laden des Playback-Memory und der LUT genutzt. Neben dem entsprechenden Token enthält es dann Adresse und Wert der zu beschreibenden Speicherzelle. Eine Auflistung der Tokens befindet sich in Anhang A.

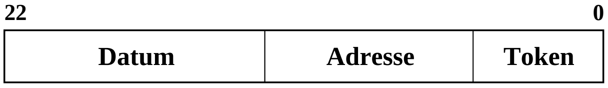
4.2 Kommunikation zwischen FEC und VME-Bus

Zwischen FEC und VME-Bus befindet sich das Dual-Ported-RAM, das auf dem FEM als Speicher dient. Es kann von beiden Seiten angesprochen werden. Von der Seite des VME-Bus werden Eingabedaten für den FeAsic und Befehle an den FEC in den Speicher geschrieben und Auslesedaten zurückgelesen. Der FEC liest Befehle und Eingabedaten und führt sie aus bzw. gibt sie an die FeAsics weiter. Außerdem schreibt er von den FeAsics gelesene Daten in den Speicher. Die Adressen der Speicherbereiche, die der FEC beschreibt oder liest, sind konfigurierbar.

In Abbildung 4.3 ist das Kommunikationskonzept zwischen FEC und VME-Bus schematisch dargestellt.

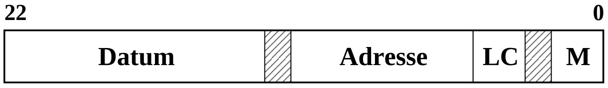
Der Adressbereich oberhalb des Dual-Ported RAMs, der dem FEC nicht zugänglich ist, wird genutzt um die Konfiguration des FPGAs und der Interruptfähigkeit des FEMs durchzuführen. Hier kann auch ein Reset-Signal ausgelöst werden, das den FEC in einen definierten Zustand versetzt.

Eingabe-Wort des FeAsics



Bit	Beschreibung
22-13	Datum
12-5	Adresse
4-0	Token

Ausgabe-Wort des FeAsics



Bit	Beschreibung
22-13	Datum
12	Nicht benutzt
11-5	Adresse
4-3	Loss Counter
2	Nicht benutzt
1-0	Auslese Modus

Abbildung 4.2: Format des FeAsic-Ein- und Ausleseworts

4.2.1 Das Befehls-Konzept zur Steuerung des FEC

Gesteuert wird der FEC über Befehle, die in den FEM-Speicher geschrieben werden. Diese Befehle können auch Eingabedaten an die FeAsics enthalten. Es gibt zwei Arten der Befehlsausführung, asynchrone Ausführung, bei der einzelne Befehle sofort ausgeführt werden und Ausführung von im Speicher abgelegten Befehlen als Programm.

Das Dual-Ported-RAM, das als FEM-Speicher eingesetzt wird, besitzt zwei besondere Speicherstellen, die einen sogenannten *Mailbox-Interrupt* zu Verfügung stellen. Wird eine solche Speicherstelle von der einen Seite des Dual-Ported-RAM beschrieben, löst das RAM einen Interrupt zur anderen Seite hin aus. Der Interrupt wird wieder gelöscht, wenn die Speicherstelle von der Seite aus gelesen wird, zu der das Interruptsignal gesendet wurde.

Beim FEM löst die oberste Adresse des RAMs, `0x1fff`[2], den Interrupt Richtung FEC aus. Diese Speicherstelle wird als Befehlsregister genutzt. So ist es möglich, daß auf einen Befehl, der in dieses Register geschrieben wird, sofort reagiert werden kann. Er wird asynchron ausgeführt. Ein eventuell laufendes FEC-Programm wird unterbrochen und nach der Ausführung des asynchronen Befehls wieder fortgesetzt.

Soll ein Programm ausgeführt werden, wird es zunächst in den FEM-Speicher geschrieben. Dann wird über asynchrone Befehle die Adresse übergeben, an der sich das Programm befindet, und der Programmablauf gestartet.

---

[2] Zahlen im hexadezimalen System werden hier, der C-Notation folgend, durch ein vorangestelltes *0x* gekennzeichnet

Modus-Bits	Bedeutung
'b00	Stop-Modus
'b01	Kontinuierlicher Modus, Auslese von einem Bunch-Crossing
'b10	Kontinuierlicher Modus, Auslese von vier Bunch-Crossings
'b11	Kontinuierlicher Modus, Auslese von acht Bunch-Crossings

Tabelle 4.1: Bedeutung der Modus-Bits des FeAsic-Auslesewortes

4.2.2 Auslese der FeAsics

Der FEC liest vier FeAsics gleichzeitig aus. Im fehlerfreien Betrieb laufen sie synchron, und es müssen Token und Pipeline-Adresse für alle vier FeAsics übereinstimmen. Der FEC überprüft dies (*Cross-Check*), entfernt redundante Informationen und formatiert die gelesenen Daten auf eine Breite von 32 Bit. So entstehen zwei Worte (Abbildung 4.4), die in den FEM-Speicher geschrieben werden. Die obersten zwei Bits der Auslese-Worte werden benutzt, um sie eindeutig zu kennzeichnen.

Der FeAsic bietet grundsätzlich zwei Arten der Auslese, zum einen eine kontinuierliche Auslese im laufenden Betrieb, bei der die Daten, die durch das Trigger-Signal akzeptiert worden sind, aus dem Derandomizer Buffer ausgelesen werden. Dies soll bis zu einer Trigger-Rate von 100 kHz möglich sein, das heißt in 10 μs müssen die vier Worte der vier FeAsics ausgelesen und in den FEM-Speicher geschrieben worden sein. Zu Testzwecken bietet der FeAsic auch die Möglichkeit, zu jedem positiven Trigger-Signal nicht nur eins, sondern vier oder acht Worte auszulesen. Hier stehen dann zur Auslese eines Wortes bei 100 kHz Triggerrate noch 2,5 μs bzw. 1,25 μs zu Verfügung.

Bei der anderen Art der Auslese, der *Stop-Modus-Auslese*, wird der FeAsic angehalten und der interne Pipeline-Speicher ausgelesen. Hier gibt es keine Zeitbeschränkung für die Auslese.

Im kontinuierlichen Modus werden Daten über den VME-Bus aus dem FEM-Speicher ausgelesen, während der FEC weitere Daten in den FEM-Speicher schreibt. Deshalb muß es einen Mechanismus geben, der gewährleistet, daß nur gültige Daten ausgelesen werden. Es dürfen nicht gleichzeitig Lese- und Schreibzugriffe auf dieselben Daten stattfinden. Um dies zu vermeiden, kann der FEC so konfiguriert werden, daß er abwechselnd zwei getrennte Speicherbereiche beschreibt und über den VME-Bus signalisiert, welcher Bereich gerade beschrieben wird und welcher gültige Daten enthält. So kann, während ein Bereich beschrieben wird, der andere ausgelesen werden. Damit keine Daten verloren gehen, muß von VME-Bus-Seite sichergestellt werden, daß die gültigen Daten ausgelesen werden, bevor sie wieder überschrieben werden.

4.2.3 SpyBus-Auslese

Durch den SpyBus ist es möglich, interne Register des FeAsics zu beobachten. Da der FeAsic mit einer Taktfrequenz von 40 MHz betrieben wird, ändern sich auch die Register mit dieser Frequenz. Bei jedem Taktzyklus müssen von jedem der vier FeAsics 8 Bit Daten ausgelesen werden, das heißt alle 25 ns 32 Bit. Dies entspricht einer Datenrate von 320 MByte/s. Diese Rate ist zu hoch, um sie in den FEM-Speicher zu schreiben. Aus

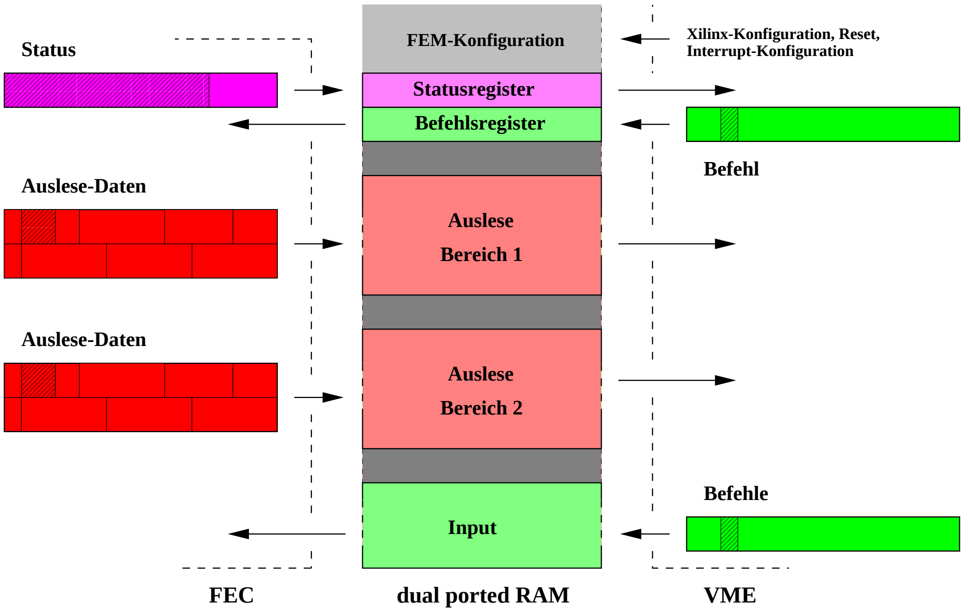


Abbildung 4.3: Kommunikation zwischen FEC und VME-Bus.

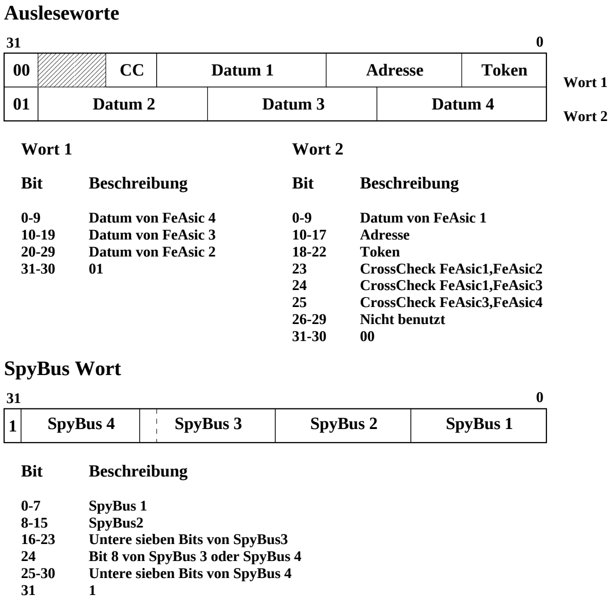
diesem Grund, bietet das FEM die Möglichkeit, die FeAsics asynchron mit einem Takt zu betreiben, den der FEC erzeugt. So kann der SpyBus mit herabgesetzter Taktfrequenz ausgelesen werden, während der FeAsic in seiner vollen Funktion, lediglich mit verminderter Geschwindigkeit, weiterarbeitet. Die Synchronisation mit dem FADC und dem G-Link geht jedoch verloren. Zum Testen der Funktion einzelner FeAsics ist diese Synchronität aber auch nicht erforderlich.

Im SpyBus-Auslese-Modus werden in den normalen Auslese-Datenstrom die Werte des SpyBus eingefügt. Abbildung 4.4 zeigt das SpyBus-Wort. Das höchstwertige Bit kennzeichnet das Wort als SpyBus-Wort. Da dadurch ein Bit eines SpyBus-Bytes überschrieben wird, gibt es die Möglichkeit, das verlorene Bit auf ein anderes Bit des SpyBus-Wortes abzubilden. So können alle 32 Bits des SpyBus ausgelesen werden, jedoch nur 31 Bit auf einmal.

4.3 Erzeugung von Trigger-Signalen

Um das FEM auch bei hohen Trigger-Raten testen zu können, ohne jedoch von außen ein besonderes Trigger-Signal erzeugen zu müssen, bietet der FEC die Möglichkeit, ein zusätzliches *Level1Accept*-Signal mit einstellbarer Frequenz zu erzeugen. Es wird allen vier FeAsics eingespeist.

Mit Hilfe des Playback-Memorys, der auf dem FeAsic die Einspeisung von Daten ermöglicht, und der Generierung von Trigger-Signalen kann zu Test- und Demonstrationszwecken eine kontinuierliche Auslese des FEM durchgeführt werden, ohne daß zusätzliche Module benötigt werden.



Wort 1

Bit	Beschreibung
0-9	Datum von FeAsic 4
10-19	Datum von FeAsic 3
20-29	Datum von FeAsic 2
31-30	01

Wort 2

Bit	Beschreibung
0-9	Datum von FeAsic 1
10-17	Adresse
18-22	Token
23	CrossCheck FeAsic1,FeAsic2
24	CrossCheck FeAsic1,FeAsic3
25	CrossCheck FeAsic3,FeAsic4
26-29	Nicht benutzt
31-30	00

Bit	Beschreibung
0-7	SpyBus 1
8-15	SpyBus2
16-23	Untere sieben Bits von SpyBus3
24	Bit 8 von SpyBus 3 oder SpyBus 4
25-30	Untere sieben Bits von SpyBus 4
31	1

Abbildung 4.4: Das Format der vom Front-End Controller in den Speicher geschriebenen Worte.

4.4 Beschreibung der FEC-Befehle

In diesem Abschnitt sollen die Befehle erläutert werden die der FEC versteht. Abbildung 4.5 enthält eine komplette Auflistung.

Die höchstwertigen vier Bits des Befehls-Wortes enthalten ein Token, das den Befehl identifiziert. Je nach Befehl belegen Parameter weitere Teile des Befehls-Wortes. Die Befehle und ihre Parameter werden in diesem Abschnitt beschrieben. Zu jedem Befehl existiert ein Mnemonic[3]. Es wird in Klammern angegeben. Die Befehle gliedern sich in fünf Bereiche. In Abbildung 6.3 ist ein kommentiertes Beispielprogramm abgebildet, das für eine Auslese der vier FeAsics im Stop-Modus verwendet wurde.

4.4.1 Setzen der Adressbereiche

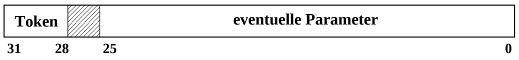
LoadAddressReadout1 (LAR1) Der Adressbereich, in den die ausgelesen Daten im Stop-Modus und im ersten Teil der kontinuierlichen Auslese geschrieben werden, wird gesetzt. Dieser Befehl besitzt zwei Parameter, die jeweils eine Adresse des FEM-Speichers im Bereich von `0` bis `0x1ffd` enthalten.

LoadAddressReadout2 (LAR2) Der Adressbereich in den die ausgelesenen Daten im zweiten Teil der kontinuierlichen Auslese geschrieben werden, wird gesetzt. Dieser Befehl besitzt zwei Parameter, die jeweils eine Adresse des FEM-Speichers im Bereich von `0` bis `0x1ffd` enthalten.

---

[3] Ursprünglich ist ein *Mnemonic* eine einprägsame Abkürzung für einen Assemblerbefehl.

Format der Befehle



Setzen der Adressbereiche

```
LoadAddressReadout1 stop_address,start_address
LoadAddressReadout2 stop_address,start_address
```

Auslese

```
StartReadoutCont
StartReadoutStop
Readout
StopReadout
```

Input und Programmablauf

```
StartInput
StopInput
Wait delay
Data data
```

FeAsic-Reset

```
StRegReset
SyncReset
```

Konfigurieren des FrontEndControllers

```
SetControlRegister control_register_value
SetMask mask_value
```

Abbildung 4.5: Die Befehle zur Steuerung des FrontEndControllers

4.4.2 Auslese

StartReadoutCont (STARC) Die kontinuierliche Auslese wird gestartet. Es werden die mit den beiden Befehlen *LoadAddressReadout1* und *LoadAddressReadout2* gesetzten Speicherbereiche abwechselnd vollständig beschrieben.

StartReadoutStop (STARS) Der mit dem Befehl *LoadAddressReadout1* gesetzte Speicherbereich wird mit ausgelesenen Daten beschrieben. Ist der Speicherbereich vollständig beschrieben, wird die Auslese beendet.

Readout (READ) Ein Datenwort wird ausgelesen und in den mit dem Befehl *LoadAddressReadout1* gesetzten Speicherbereich geschrieben.

StopReadout (STOR) Die Auslese von Daten wird abgebrochen.

4.4.3 Eingabedaten und Programablauf

StartInput (STAI) Die Abarbeitung von Befehlen ab der angegebenen Adresse wird begonnen. Dieser Befehl besitzt einen Parameter, der eine Adresse des FEM-Speichers im Bereich von `0` bis `0x1ffd` enthält.

StopInput (STOI) Die Abarbeitung von Befehlen wird beendet.

Bit	Name	Bedeutung
0	GLinkHalt	Die G-Links werden angehalten
1	DisableTrigger	Der externe Trigger wird nicht mehr verteilt
2	GenerateTrigger	Der interne Trigger wird generiert
3	SwapBits	Bit 31 und Bit 23 des SpyBus-Worts werden vertauscht
4-6	SpyBusCtrl	Auswahl der Register, die über den SpyBus ausgegeben werden
7	SpyBusEnable	Der SpyBus wird ausgelesen
8-11	TriggerDelay	Periode des erzeugten Trigger-Signals
12	Light	Eine Leuchtdiode wird geschaltet
13	EnableStatusWrite	Erlaubt das Schreiben des Status-Registers in den FEM-Speicher.

Tabelle 4.2: Das Kontrollregister des FECs

Wait (WAIT) Vor der Abarbeitung des nächsten Befehls wird für eine Anzahl von internen Taktzyklen gewartet, die durch den Parameter *delay* angegeben werden. Die Zeitspanne, die gewartet wird, errechnet sich durch: (`0xffff`$-$delay)$\cdot$Taktperiode. Dieser Befehl besitzt einen Parameter, der einen Wert im Bereich von `0` bis `0xffff` enthält.

Data (DATA) Ein Datenwort wird an die FeAsics übertragen. Dieser Befehl besitzt einen Parameter, der ein 23 Bit breites Datenwort enthält.

4.4.4 Reset des FeAsic

StRegReset (STRES) Das Signal *StRegReset* wird für einen internen Taktzyklus auf High gesetzt.

SyncReset (SYRES) Das Signal *SyncReset* wird für einen internen Taktzyklus auf High gesetzt.

4.4.5 Konfiguration des FEC

SetControlRegister (SETCR) Das interne Kontroll-Register des FEC wird gesetzt. Tabelle 4.2 gibt die Bedeutung der einzelnen Bits des Kontroll-Wortes an. Dieser Befehl besitzt einen Parameter, der ein 13 Bit breites Wort enthält.

SetMask (MASK) Die Maske, die angibt, welche der vier FeAsics angesprochen werden, wird gesetzt. Dieser Befehl besitzt einen Parameter, der einen 4 Bit-Wert enthält. Jedes Bit ist einem FeAsic zugeordnet. Ist es gesetzt, wird der zugehörige FeAsic angesprochen. Das höchstwertige Bit gehört zu FeAsic 4, die anderen folgen in absteigender Reihenfolge.

Bit	Name	Bedeutung
0	InputMode	Ein Programm wird abgearbeitet
1	ReadoutActive	Die Auslese ist aktiv
2	Continuous	Die kontinuierliche Auslese ist aktiv
3	Frame	Gibt den Speicherbereich an, der gerade beschrieben wird
4	Frame1Done	Der erste Speicherbereich wurde vollständig beschrieben
5	Frame2Done	Der zweite Speicherbereich wurde vollständig beschrieben
6-9	Mask	Maske, die steuert, welche FeAsics angesprochen werden.
11-24	ControlRegister	Kontrollregister des FEC

Tabelle 4.3: Das Statusregister des FECs

4.5 Statusregister

Die Speicheradresse `0x1ffe` des FEM, die bei Beschreiben durch den FEC einen Interrupt in Richtung VME-Bus auslöst, wird als Statusregister des FECs benutzt. In ihr werden Informationen über den internen Zustand des FECs zu Verfügung gestellt, insbesondere die Gültigkeit der vom FEC aus beschriebenen FEM-Speicherbereiche, die für die kontinuierliche Auslese von entscheidender Bedeutung ist. Außerdem werden im Statusregister das Kontrollregister des FEC und das Maskenregister, das die Ansteuerung der FeAsics steuert, wiedergespiegelt. In Tabelle 4.3 ist die Bedeutung der einzelnen Bits des Status-Registers beschrieben.

4.6 Ergebnis des Designs

Das Design wurde als hierachisches Verilog-Design durchgeführt. Auf jeder Stufe des Design-Prozesses wurden Simulationen in einer Verilog-Testumgebung durchgeführt.

Als Ziel wurde ein FPGA vom Typ XC4010E-3 in einem 191-Pin-Gehäuse ausgewählt. Dieser FPGA bietet 400 CLBs und 160 nutzbare IOBs. Auf dem FEM wird er entweder parallel über den VME-Bus im sogenannten *asynchronous peripheral mode* oder über ein einsteckbares Kabel seriell konfiguriert. Die XC4000E-Serie zeichnet sich gegenüber der XC4000-Serie durch verbesserte Fast-Carry-Logic und erweiterte Möglichkeiten, die Konfigurations-RAMs als Speicher zu benutzen aus. Sie ist pin-kompatibel zu der XC4000-Serie und kann mit denselben Konfigurationsdaten geladen werden. Da die zu Verfügung stehende Software-Version den gewählten FPGA noch nicht unterstützte, wurde die Implementation für einen XC4010-4 durchgeführt. Wegen der Kompatibilität war dies keine Einschränkung, lediglich die Simulationsergebnisse gehen von einem zu langsamen Chip aus, geben jedoch eine zuverlässige untere Grenze für die Geschwindigkeit des tatsächlich eingesetzten Chips an.

Das Verilog-Design umfaßt etwa 2900 Zeilen Programmcode. Um die Fähigkeiten des Xilinx-FPGAs auszunutzen, wurden Zähler und Global Clock Buffer, die nur als Schaltpläne zu Verfügung stehen, als Verilog-Modul-Instanzen eingebunden. Das Design wurde in eine große Anzahl von Modulen aufgeteilt, da die Synthese-Ergebnisse mit einer aus kleineren Modulen bestehenden Hierarchie durchweg besser waren.

Die Synthese erfolgte mit *Synergy* von *Cadence*. Es wurden Vorgaben gemacht, die State-Machines One-Hot zu codieren und Hierarchie sowie die eingefügten Schaltpläne zu erhalten. Nach der Synthese wurde dem Design noch mit der Hand ein Schaltplan-Symbol hinzugefügt, das die Nutzung des globalen Resets des Xilinx-FPGAs ermöglicht.

Die Implementation auf dem FPGA erfolgte mit dem *Xilinx Design Flow* von *Cadence*, der die Xilinx-eigene Entwicklungsumgebung XACT integriert. Abbildung 4.6 zeigt einen Ausschnitt des Berichts, den XACT erzeugt. Der FPGA wurde zu ca. 70% ausgenutzt. Die längste Signallaufzeit zwischen zwei Flip-Flops (FROM:ffs:TO:ffs) beträgt weniger als 50 ns. Das Design läuft also mit mindestens 20 MHz. Die große Zahl verwendeter Flip-Flops von über 500 resultiert aus der großen Breite der zu verarbeitenden Daten[4], der Benutzung von Pipeline-Verarbeitung[5] und der One-Hot-Codierung der State-Machines.

In Abbildung 4.7 ist ein Auschnitt des Designs dargestellt, wie es von XACT angezeigt wird. Man erkennt IOBs, CLBs und zwischen ihnen verlaufende Verbindungen.

Für die Simulation des Designs wurde eine Testumgebung in Verilog erstellt. Sie enthält ein Modell des seriellen Interfaces des FeAsics und ein Modell des Dual-Ported-RAM. Beide Modelle enthalten Signallaufzeiten. So läßt sich das FEM simulieren, wie es von der VME-Bus-Seite erscheint. Die erzeugten Test-Vektoren entsprechen den Zugriffen über den VME-Bus. Die Test-Vektoren wurden mit Hilfe der *Simulation and Test Language* (STL) erzeugt.

Nach der Implementierung des Designs auf dem FPGA stehen die Signallaufzeiten auf dem FPGA zu Verfügung. Sie werden über eine Datei, die die Laufzeiten im *Standard Delay Format* (SDF) enthält, der Simulation zugänglich gemacht.

Die Simulation des fertigen Designs des FEC erfolgte bei einem externen Takt von 40 MHz. Intern wird der Takt für die meisten Funktionsblöcke auf 20 MHz halbiert. Die Taktfrequenz für die serielle Schnittstelle zum FeAsic beträgt ebenfalls 20 MHz. Für das Auslesen eines Datenwortes aus allen angeschlossenen FeAsics und das Schreiben in den FEM-Speicher benötigt der FEC 1,6 μs. Damit ist es möglich, bei einer Triggerrate von 100 kHz die Modi des FeAsics zu benutzen, bei denen ein bzw. vier Bunch-Crossings ausgelesen werden und den Modus, bei dem acht Bunch-Crossings gelesen werden, bei 75 kHz Triggerrate zu betreiben.

---

[4] Allein die zu einem Trigger-Signal einzulesenden Daten haben eine Breite von 4*23 Bit=92 Bit

[5] Die Verarbeitung der Daten erfolgt in mehreren Stufen, zwischen denen die Daten gespeichert werden Die verschiedenen Stufen können parallel arbeiten. Zum Beispiel können die FeAsics ausgelesen werden während gleichzeitig Daten ins Dual-Ported-RAM geschrieben werden. Somit ist eine schnellere Verarbeitung der Daten möglich als ohne Pipeline-Architektur.

```
(...)

    Partitioned Design Utilization Using Part 4010PG191-4


                                     No. Used   Max Available   % Used
      ----------------------------    -------   -------------   ------
      Occupied CLBs                       399        400           99%
      Packed CLBs                         283        400           70%
      ----------------------------    -------   -------------   ------
      Bonded I/O Pins:                    122        160           76%
      F and G Function Generators:        502        800           62%
      H  Function Generators:            83          400           20%
      CLB Flip Flops:                     567        800           70%

(...)

    CPU Times

      Partition:                        00:00:59
      Placement:                        00:24:14
      Routing:                          00:23:39
      Total:                            00:49:09

(...)

    Xact Performance Summary
    ----------------------------------------

        Deadline  Actual(*)  Specification
        --------  ---------  -------------
          <auto>   49.4ns    DEFAULT_FROM_FFS_TO_FFS=FROM:ffs:TO:ffs
          <auto>   22.1ns    DEFAULT_FROM_PADS_TO_FFS=FROM:pads:TO:ffs
          <auto>   35.6ns    DEFAULT_FROM_FFS_TO_PADS=FROM:ffs:TO:pads

(...)
```

Abbildung 4.6: Ausschnitt aus dem PPR-Bericht

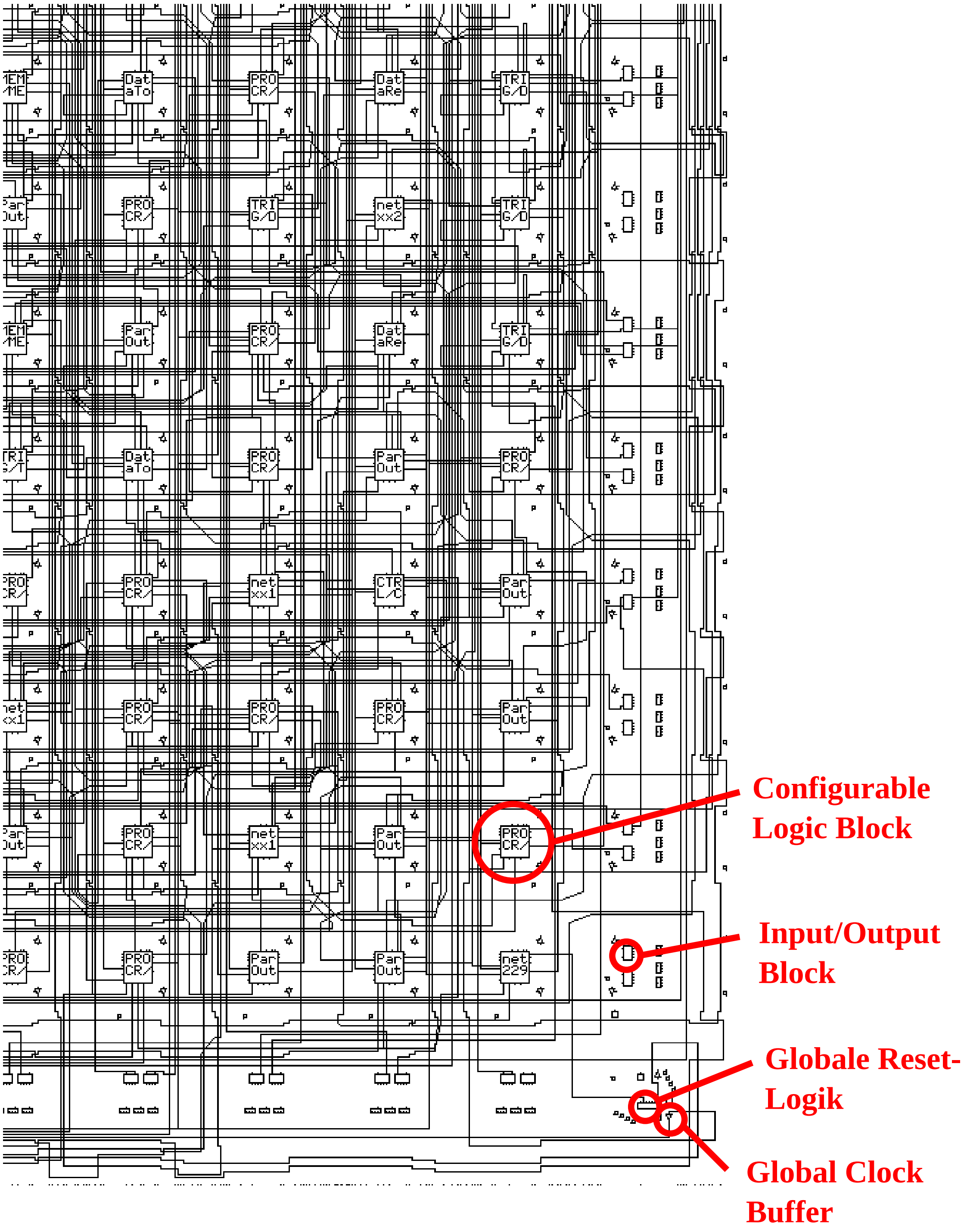


Abbildung 4.7: Auschnitt aus der Xilinx-Implementierung des FEC

Kapitel 5

Steuersoftware des FEMs

Zur Ansteuerung des FEMs ist Software erforderlich, die die Speicherzugriffe über den VME-Bus abwickelt, ein- und ausgehende Daten verarbeitet und eine Schnittstelle zum Benutzer bietet. Es wurden Routinen entwickelt, die den Zugriff auf das FEM erlauben. Ein Satz von Kommandozeilen-Befehlen und eine darauf aufsetzende grafische Oberfläche bilden die Schnittstelle zum Benutzer. Bei der Entwicklung der Software wurden neben der Bereitstellung der erforderlichen Funktionalität drei Ziele verfolgt:

- **Portabilität und Flexibilität.** Die Software sollte leicht auf andere Systemgegebenheiten angepaßt werden können und möglichst unabhängig von der konkreten Hardware- und Softwareumgebung sein.
- **Integrations- und Erweiterungsfähigkeit.** Es sollte leicht und transparent sein, die Software um zusätzliche Funktionalität zu erweitern oder sie als Komponente in größere Softwareumgebungen einzufügen.
- **Bedienungskomfort.** Es sollte eine komfortable Schnittstelle zum Benutzer vorhanden sein, die die Bedienung nicht durch technische Einzelheiten erschwert.

Die Entwicklung der Software erfolgte parallel zur Entwicklung der VME-Karte. Da deshalb die Hardware erst spät zu Verfügung stand, wurde das Konzept der Software von einem abstrakten Standpunkt aus erstellt.

5.1 Schichtenmodell der Steuersoftware

Um die angestrebten Ziele zu erreichen, war es zweckmäßig, die Software nach einem Schichtenmodell zu organisieren. Die einzelnen Schichten bauen aufeinander auf, zwischen den Schichten bestehen definierte Schnittstellen und die Kommunikation verläuft nur zwischen direkt benachbarten Schichten. Dadurch sind die einzelnen Schichten unabhängig von der Implementation der anderen Schichten.

In Abbildung 5.1 ist das Schichtenmodell angebildet, das der Steuersoftware des FEMs zu Grunde liegt. Es besteht aus vier Schichten, einer untersten Schicht, die Hardware und Daten in gespeicherter Form repräsentiert, einer Systemschicht, die Zugriffe auf Hardware und Daten sowie Verarbeitungsroutinen zu Verfügung stellt, einer Anwendungsschicht, die,

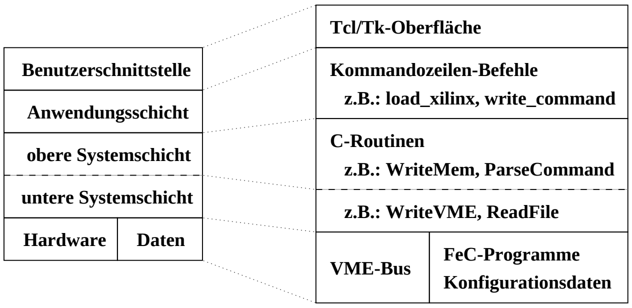


Abbildung 5.1: Schichtenmodell der Steuersoftware des FEMs

aufbauend auf die System-Routinen, die gewünschte Funktionalität bietet und als oberster Schicht einer Schnittstelle zum Benutzer in Form einer grafischen Oberfläche.

5.2 Hardware und Daten

Das FEM wird über den VME-Bus angesprochen. Das heißt, die Steuersoftware kommuniziert mit dem FEM über Lese- und Schreibzugriffe auf den VME-Bus. Die Hardwareschicht wird also durch den VME-Bus repräsentiert.

Die Steuersoftware arbeitet mit drei verschiedenen Typen von Daten, Auslesedaten, FEC-Programme, die auch Eingabedaten umfassen, und Konfigurationsdaten.

Die Auslesedaten werden als Text-Dateien erzeugt, deren Format zur Weiterverarbeitung oder Darstellung durch andere Programme geeignet ist.

FEC-Programme sind Folgen von FEC-Befehlen, die in den FEM-Speicher geschrieben und dort ausgeführt werden. Es sind Text-Dateien, die die Befehle in einer lesbaren Form enthalten. Abbildung 6.3 zeigt ein Beispiel-Programm. Daten, die an die FeAsics gesendet werden, können in FEC-Programmen in Form von `Data`-Befehlen enthalten sein.

Schließlich gibt es noch Konfigurations-Daten für das FEM. Dies sind die Konfigurationsdatei für den FPGA und eine Text-Datei, in der allgemeine Informationen, zum Beispiel die Basisadresse der VME-Karte, in einer lesbaren Form enthalten sind. Dadurch ist es möglich, manchen Änderungen der Hard- oder Softwareumgebung ohne Neucompilierung der Software durch Ändern der entsprechenden Konfigurationsdatei Rechnung zu tragen.

5.3 Systemschicht

Die Systemschicht stellt eine Schnittstelle zu Hardware und Daten zu Verfügung, die unabhängig von der konkreten Realisation der Hardware oder der Datenspeicherung ist. Sie kann in eine untere und eine obere Schicht unterteilt werden. Dabei enthält die untere Systemschicht die vom Rechner-System abhängigen Routinen zur Ansteuerung der Hardware und Verwaltung der Daten. Die obere Systemschicht setzt in einer hardware- und systemunabhängigen Weise darauf auf.

Speicherzugriff
`WORD ReadMem (FEM_ADDRESS Address)` `WORD WriteMem (FEM_ADDRESS Address, FEM_DATA Data)`
Übersetzung von FEC-Befehlen
`WORD ParseCommand (char *commandstring)` `BOOL Execute (char *program)`
Auswertung der Konfigurationsdatei
`BOOL InitGlobalConfig ()`
Konfiguration des Xilinx FPGA
`BOOL LoadXilinx (FILE *bitstream)`

Tabelle 5.1: C-Routinen der Schnittstelle zwischen Anwendungs- und Systemschicht

Die Systemschicht ist in ANSI-C unter Beschränkung auf die Funktionen der Standardbibliothek programmiert und damit auf nahezu jedem Rechnersystem compilierbar. Lediglich die Routinen der unteren Systemschicht müssen gegebenenfalls entsprechend angepasst werden.

Die Schnittstelle der Systemschicht zur Anwenderschicht wird durch einen Satz von Routinen definiert, die mit entsprechenden Parametern aufgerufen werden können. In Tabelle 5.1 sind sie aufgelistet.

Durch die Kapselung der Hardwarezugriffe spielt es keine Rolle mehr, wie der VME-Bus in das System integriert ist. Er kann auch durch ein Software-Modul ersetzt werden, das willkürliche oder Simulations-Daten erzeugt. Diese Eigenschaft war hilfreich für die Entwicklung der Software als die Hardware noch nicht zu Verfügung stand.

Ein wichtiger Teil der Systemschicht sind die Routinen zur Interpretation der FEC-Befehle. Sie erzeugen aus einem in Klartext lesbaren Programm die entsprechenden binären Codes, die letzlich an den FEC übergeben werden.

5.4 Anwendungsschicht

In der Anwendungsschicht werden Kommandozeilenbefehle zu Verfügung gestellt, über die das FEM gesteuert werden kann. Es sind Befehle vorhanden, um das FEM zu konfigurieren, Daten einzulesen und Ausleseprozesse durchzuführen. Tabelle 5.2 zeigt die vorhandenen Kommandozeilenbefehle. Sie sind ebenfalls in ANSI-C programmiert.

Die Befehle der Anwendungsschicht stellen einerseits die Schnittstelle zur grafischen Benutzeroberfläche dar. Andererseits kann der Anwender über sie auch direkt in einer textorientierten Umgebung auf das FEMzugreifen.

5.5 Grafische Benutzerschnittstelle

Aufsetzend auf die Kommandozeilen-Befehle der Anwenderschicht wurde eine grafische Benutzeroberfläche in *Tcl/Tk* [26] entwickelt. *Tcl* ist eine Skriptsprache, die mit ihrer

Auslese
`fast_readout number_frames`
FEC-Befehle und Programme

`write_command command`
`show_commands`
`execute_program program_file`
`list_programs`

Speicherzugriffe
`read_mem start_address end_address` `read_status`
Konfiguration
`dump_config` `load_xilinx bitstream_file`

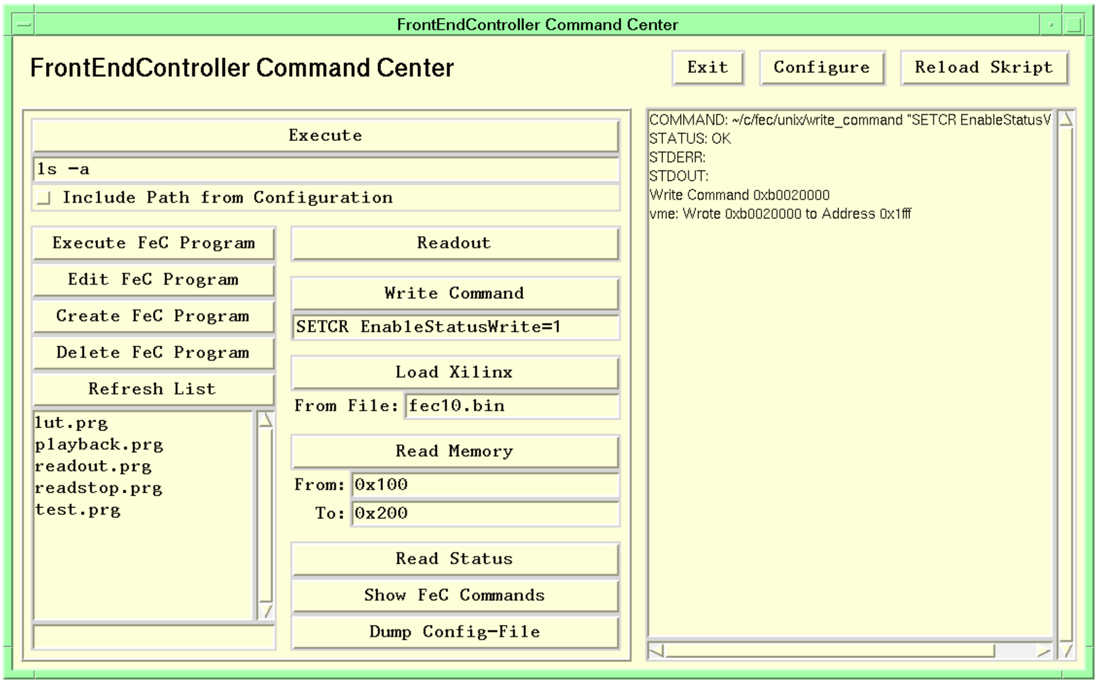
Tabelle 5.2: Kommandozeilenbefehle zur Steuerung des FEMs

Erweiterung *Tk* einen sehr einfachen aber dennoch leistungsfähigen Zugriff auf Elemente grafischer Benutzeroberflächen bietet. Das Tcl/Tk-Paket ist kostenlos für verschiedene Plattformen erhältlich [27].

Die Tcl/Tk-Schnittstelle bietet die volle Funktionalität der Kommandozeile und den Komfort einer grafischen Oberfläche. In Abbildung 5.2 ist das Hauptfenster des Tcl/Tk-Programmes dargestellt. Man kann erkennen, wie die Befehle der Anwenderschicht in der grafischen Oberfläche abgebildet werden.

Neben der Ausführung der Kommandozeilen-Befehle und der komfortablen Eingabe ihrer Argumente, bietet die grafische Benutzeroberfläche die Möglichkeit, FEC-Programme zu editieren und UNIX-Kommandos auszuführen.

Die klare Trennung zwischen grafischer Benutzerschnittstelle und Anwenderschicht ermöglicht es, über den unter UNIX zu Verfügung stehenden *Remote-Shell*-Mechanismus die Oberfläche auf einem anderen Rechner als den Rest der Software laufen zu lassen und über die Netzwerkverbindung zu kommunizieren. So kann die Tcl/Tk-Oberfläche auf einer Workstation laufen und die eigentliche Steuerungssoftware mit den zeitkritischen Routinen auf einem VME-Prozessor-Modul, das kein Tcl/Tk benötigt.



ls -a

☐ Include Path from Configuration

lut.prg
playback.prg
readout.prg
readstop.prg
test.prg

SETCR EnableStatusWrite=1

From File: fec10.bin

From: 0x100
To: 0x200

```
COMMAND: ~/c/fec/unix/write_command "SETCR EnableStatusV
STATUS: OK
STDERR:
STDOUT:
Write Command 0xb0020000
vme: Wrote 0xb0020000 to Address 0x1fff
```

Abbildung 5.2: Hauptfenster der Ansteuerungssoftware des FEM

Kapitel 6

Erster Test des Front-End-Moduls

Parallel zum Front-End Controller und der Ansteuerungssoftware für das Front-End Modul wurde von der Elektronikwerkstatt des Instituts für Hochenergiephysik Heidelberg die VME-Karte entwickelt und das Layout der Platine erstellt. Nachdem eine vollständig bestückte Karte zu Verfügung stand (siehe Abbildung 6.1), konnten erste Funktionstests durchgeführt werden. Ihr Ziel lag darin, die prinzipielle Funktionsfähigkeit der einzelnen Komponenten zu zeigen. Insbesondere die Auslese von Daten über den VME-Bus wurde getestet.

6.1 Testaufbau

Für die ersten Tests wurde ein VME-Crate[1] mit einer Motorola 68030-Prozessorkarte benutzt. Es wurden ein FEM und eins der FADC-Module eingesteckt, das auch im Demonstrationssystem zum Einsatz kommt.

Als Betriebssystem für die Prozessorkarte stand OS9 zu Verfügung, das sich aufgrund seiner Echtzeit-Eigenschaften und seiner vergleichsweise geringen Komplexität für Testaufbauten wie den beschriebenen eignet. Die komplette Ansteuerungssoftware des FEMs wurde unter OS9 compiliert. Da auf dem OS9-System eine Portierung des Tcl/Tk-Pakets zu Verfügung stand, konnte die Bedienung der Software über die grafische Benutzeroberfläche erfolgen. Zusätzlich wurde noch eine Reihe von C-Programmen entwickelt, die Eingabe-Daten für das FEM erzeugen, ausgelesene Daten analysieren und das Sichern und Vergleichen von Speicherbereichen ermöglichen.

Eingabedaten für die FeAsics wurden hauptsächlich über VME-Bus und FEC in das Playback-Memory des FeAsics geschrieben. Zu Vergleichszwecken wurde jedoch auch die Möglichkeit genutzt, Daten über den Playback-Speicher des FADC-Moduls direkt an das FEM zu senden. Die Auslese des FEMs erfolgte in beiden Fällen über den VME-Bus.

Da für die G-Link-Verbindung kein Empfänger zu Verfügung stand, wurde das schnelle Ausgangssignal lediglich mit einem Oszilloskop hoher Bandbreite beobachtet.

---

[1] Ein *Crate* ist der mechanische Aufbau, der dazu dient, Steckkarten aufzunehmen. Er stellt die Spannungsversorgung und eine Backplane zu Verfügung, die alle Karten miteinander verbindet. Auf dieser Backplane liegt auch der VME-Bus.

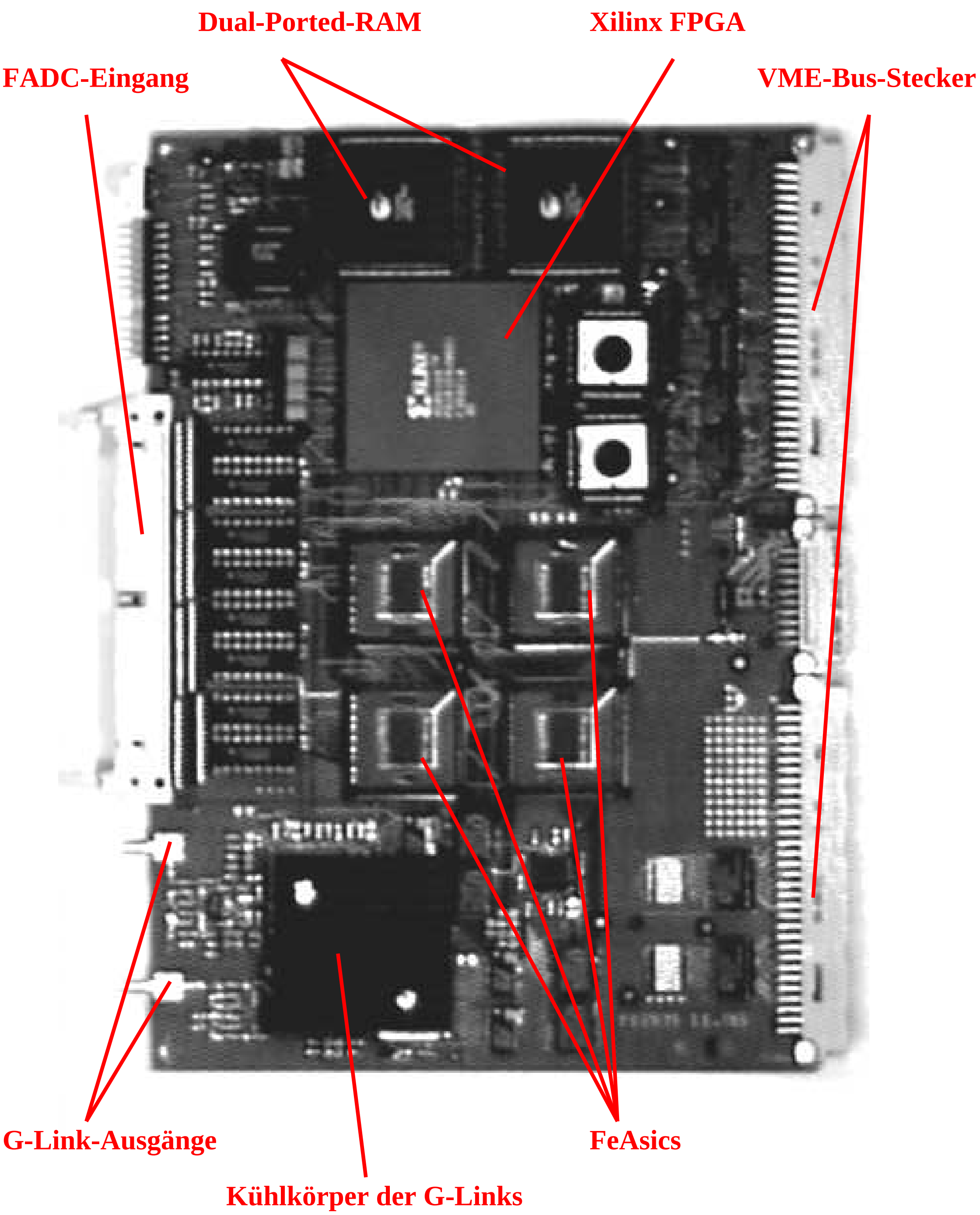


Abbildung 6.1: Foto des Front-End-Moduls

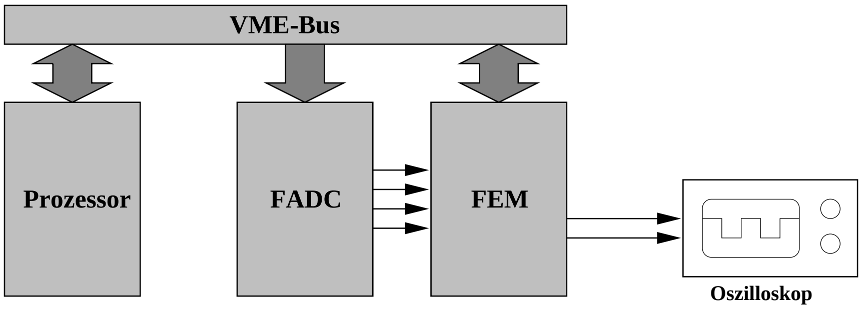


Abbildung 6.2: Der Aufbau zum Test des Front-End Moduls

Um das FEM zunächt unabhängig von anderen Komponenten betrieben zu können, wurde als Systemtakt für den FEC und die vier FeAsics ein 16 MHz-Takt-Signal verwendet, das auf dem VME-Bus vorhanden ist.

Der Testaufbau ist in Abbildung 6.2 schematisch dargestellt.

6.2 Auslese im STOP-Modus

Die wichtigste Funktion des FEM ist die Vorverarbeitung der digitalisierten Kalorimeterdaten für die Level-1 Trigger-Logik. Die dafür nötigen Verarbeitungseinheiten sind auf dem FeAsic implementiert. Über die Auslesefähigkeit des FEMs kann der Verlauf der Daten in der Verarbeitungskette direkt beobachtet werden.

Um die Funktionsfähigkeit der Auslese zu zeigen, wurden Daten erzeugt, die willkürliche Pulse verschiedener Höhe über einem konstanten Untergrund enthalten. Diese Daten wurden über das VME-Interface und den FEC in die Playback-Memorys der FeAsics und von dort in den schnellen Verarbeitungszweig eingespeist. Die LUT wurde mit Werten geladen, die so gewählt waren, daß sie den konstanten Untergrund der Pulse auf null reduzierten und das Signal in der Amplitude um einen Faktor zwei streckten. Dann wurden die FeAsics an drei verschiedenen Stellen wiederum über FEC und VME-Bus ausgelesen, direkt nach der Einspeisung, nach der LUT, und nach der BCID-Einheit. Die Auslese erfolgte im Stop-Modus, das heißt bei angehaltener Verarbeitungskette. So war es möglich das Pipeline-Memory der FeAsics auszulesen, der einen Ausschnitt von 128 aufeinanderfolgenden Worten der verarbeiteten Daten enthält. Das FEC-Programm, das zur Auslese der Daten benutzt wurde, zeigt Abbildung 6.3.

Die Ergebnisse sind in Abbildung 6.4 dargestellt. Die ausgelesenen Daten wurden über der Adresse aufgetragen, an der sie sich im Dual-Ported-RAM des FEM nach der Auslese befanden. Im ersten Bild sieht man die Originalpulse, ausgelesen direkt nach der Einspeisung. Im zweiten Bild sind die Pulse nach der Verarbeitung durch die LUT dargestellt. Man sieht wie sie gestreckt werden und die Nulllinie eingestellt wird. Im dritten Bild nach dem BCID verbleiben nur die Maxima der Pulse im Datenstrom.

Die Eingangsdaten wurden unter synchronen Bedingungen eingespeist, das heißt, die an der Auslese beteiligten Komponenten wurden bei allen drei Auslesevorgängen im selben zeitlichen Verhältnis zueinander gestartet und angehalten. In der Abbildung erkennt man

```
SETCR EnableFeClk=1,EnableStatusWrite=1,Light=0
MASK 0xf  ; alle FeAsics ansprechen
STRES     ; Reset an FeAsic
SYRES
DATA 0x6  ; FeAsic konfigurieren
DATA 0x9
DATA 0xc
LAR1 0x1ff0,0x100  ; Auslesebereich konfigurieren
LAR2 0x1ff2,0x1ff2
SETCR Light=1  ; LED an -> Auslese konfiguriert
STARS  ; Auslese Starten
WAIT 0xff00
SETCR Light=0  ; LED aus -> Stoppe Pipeline
SETCR GenerateTrigger=1  ; Generiere ein Level1Accept
SETCR GenerateTrigger=0
```

Abbildung 6.3: FEC-Programm zur Auslese von Daten im Stop-Modus

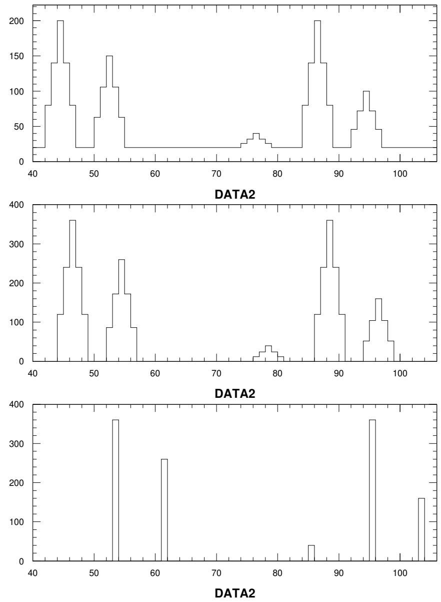


Abbildung 6.4: Aus der Verarbeitungskette des FeAsics ausgelesene Daten, aufgetragen über der Position im Datensatz, direkt nach Einspeisung, nach LUT und nach BCID

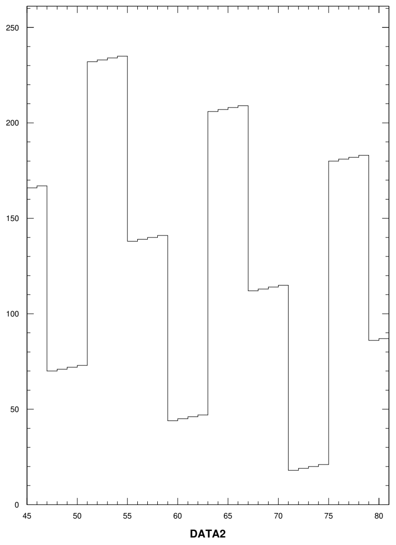


Abbildung 6.5: Im kontinuierlichen Modus ausgelesene Daten, aufgetragen über der Position im Datensatz

wie die Pulse sich in Richtung zunehmender Speicher-Adresse verschieben. Dies liegt daran, daß die Verarbeitung durch LUT und BCID eine gewisse Zeit in Anspruch nimmt, sich die Daten deshalb länger in der Verarbeitungskette aufhalten und die Auslese zu einem späteren Zeitpunkt stattfindet. Die Verarbeitung durch die LUT verschiebt das ausgelesene Signal um zwei, die BCID um weitere sieben Taktzyklen. Ein Taktzyklus entspricht zwei Speicher-Adressen, da die zu einem Zyklus gehörenden FeAsic-Worte in zwei 32 Bit-Worte gespeichert werden.

6.3 Kontinuierliche Auslese

Für die kontinuierliche Auslese wurde eine Rampe, bestehend aus aufeinanderfolgenden Werten von 0 bis 255, in das Playback-Memory der FeAsics geschrieben und in den Verarbeitungszweig eingespeist. Ohne die Verarbeitung anzuhalten, wurden die Daten über den Derandomizer Buffer der FeAsics vor der LUT wieder ausgelesen. Ein Level1Accept-Signal konstanter Frequenz wurde erzeugt.

Die Auslese der Daten erfolgte in dem Modus, in dem zu jedem Level1Accept-Signal Daten von vier aufeinanderfolgenden Taktzyklen ausgelesen werden. In Abbildung 6.5 sind diese Daten dargestellt. Man erkennt die jeweils vier aufeinanderfolgenden Werte der Rampe, die zu jedem Level1Accept-Signal ausgelesen werden. Die Rampe ist nicht kontinuierlich, da die Daten zwar fortlaufend in das Pipeline-Memory des FeAsics geschrieben, aber in der Zeit, in der kein Level1Accept-Signal auftritt, nicht ausgelesen werden.

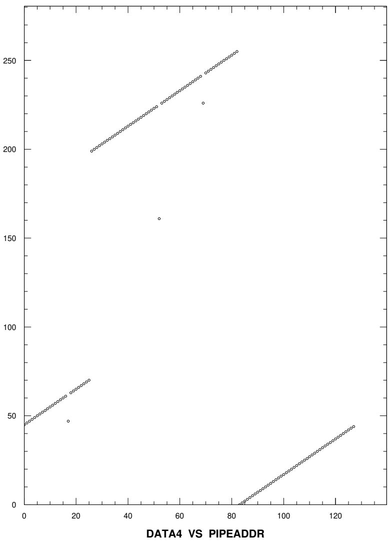


Abbildung 6.6: Ausgelesene Daten, aufgetragen über der Adresse des FeAsic-Pipeline-Memory. Eingespeist wurde eine Rampe. Man erkennt einzelne Fehler

6.4 Speicher-Fehler

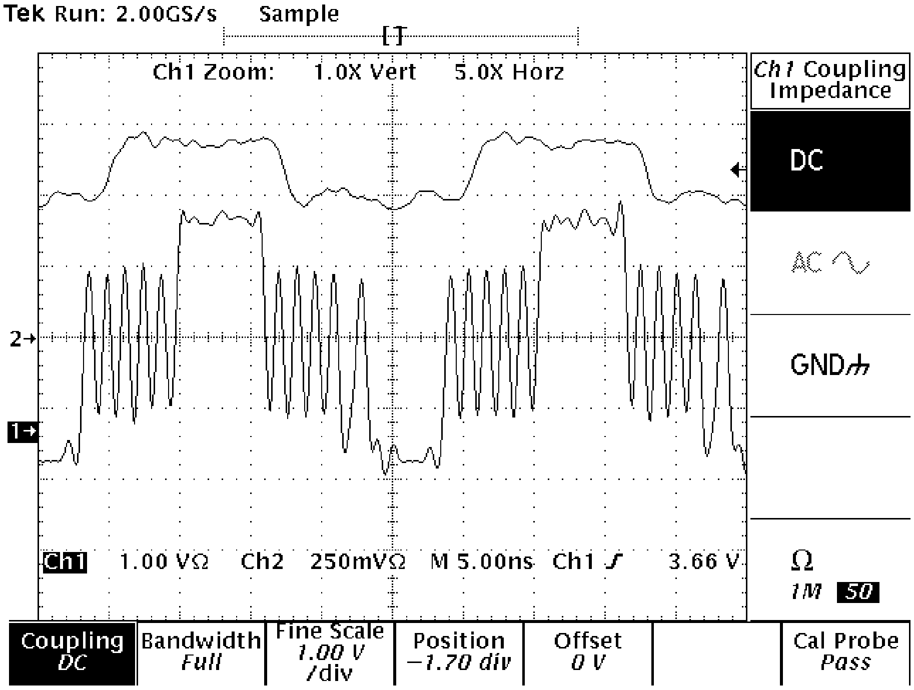
Bei den Tests wurden auf dem FEM verschiedene FeAsics benutzt. Bei einigen von ihnen kam es bei der Auslese zu Fehlern. Einzelne Datenworte wiesen einen falschen Wert auf.

In Abbildung 6.6 sind die aus einem fehlerhaften FeAsic ausgelesenen Daten gegen die Adresse des Pipeline-Memory aufgetragen. Eingespeist wurde über das Playback-Memory des FeAsics eine Rampe von 0 bis 256. Das Pipeline-Memory ist nur halb so gross wie das Playback-Memory. Deshalb wird nur die halbe Rampe ausgelesen und es tritt ein Sprung in den Daten auf, der in der Abbildung bei Adresse 26 liegt. Der zweite Sprung bei Adresse 82 kommt dadurch zustande, daß das Playback-Memory von Adresse 256 auf Adresse 0 umschaltet.

Um eine genauere Analyse der Fehler zu ermöglichen, wurden verschiedene Werte sowohl über das Playback-Memory des FeAsics als auch über den FADC in die FeAsics eingespeist. Dadurch konnte festgestellt werden, daß die Fehler nur bei bestimmten Adressen des Pipeline-Memory auftraten. Jeweils ein einzelnes Bit wies einen konstanten Wert auf, unabhängig davon, wie es beschrieben wurde.

Eine mögliche Erklärung für dieses Phänomen ist, daß beim Herstellungsprozeß der FeAsics einzelne schadhafte Speicherzellen erzeugt wurden. Die Speicher werden in der für den gewählten Prozeß minimal möglichen Strukturgröße gefertigt. Schon sehr kleine Störeinflüsse können so Schäden auf dem Chip zur Folge haben.

Da die Ausbeute eines Chip-Herstellungsprozesses in den seltensten Fällen vollständig ist, sind solche Fehler zu erwarten. Durch systematische Tests müssen die betroffenen



Coupling DC
Bandwidth Full
Fine Scale 1.00 V /div
Position −1.70 div
Offset 0 V

Abbildung 6.7: 40 MHz-Takt und G-Link-Ausgangsignal für einen Eingangswert von 0x5500 [28]

Chips identifiziert werden.

6.5 G-Link Sender

Um die Funktionsfähigkeit der G-Link-Sender zu testen, wurden über FADC und FeAsic verschiedene Eingangssignale angelegt. Zudem wurde extern ein Taktsignal mit einer Frequenz von 40 MHz eingespeist. Die 800 MHz-Ausgänge der G-Link-Sender wurden mittels eines Oszilloskops beobachtet.

Die erhaltenen Datenmuster entsprechen für einen G-Link dem von Hewlett-Packard spezifizierten Protokoll. In Abbildung 6.7 ist das Signal für einen G-Link-Eingangswert von 0x5500 zusammen mit dem Taktsignal dargestellt. Man erkennt, daß das Ausgangssignal mit dem Taktsignal synchronisiert ist und erkennt die Bitwechsel des Eingangssignal, die durch eine Flanke im Ausgangssignal repräsentiert werden. Um keinen Gleichspannungsanteil zu erzeugen[2], werden die Daten im zweiten Teil des Taktzyklus noch einmal gespiegelt übertragen.

Das Verhalten des zweiten G-Links wich von dem des ersten ab. Die Synchronisation mit dem Taktsignal war nicht immer gegeben und die ausgegebenen Datenmuster standen in manchen Fällen nicht in der richtigen Beziehung zu den Eingangsmustern. Die Ursache

---

[2] Ein Gleichspannungsanteil würde eine kapazitätsgekoppelte Übertragung des Signals unmöglich machen.

für dieses Verhalten ist noch nicht bekannt und Gegenstand weiterer Untersuchungen. Wahrscheinlich steht sie in Zusammenhang mit den sehr hohen Frequenzen, die der G-Link erzeugt. Bei diesen Frequenzen können schon kleine Unterschiede im Layout der Karte ein unterschiedliches Verhalten von zwei ansonsten identisch beschalteten Elementen hervorrufen.

6.6 Testprogramm

Der Test des FEM ist noch nicht abgeschlossen. Die grundsätzliche Funktionsfähigkeit konnte gezeigt werden, jedoch verbleiben einige Aspekte, die noch nicht vollständig untersucht wurden. Dies sind im einzelnen:

- Betrieb mit der vollen Taktfrequenz von 40 MHz.
- Test der kontinuierlichen Auslese mit realistischen FADC-Signalen, insbesondere bei hohen Triggerraten.
- Aufnahme der Daten durch einen G-Link-Empfänger.

Abschließend muß das FEM in das vorhandene ATLAS Level-1 Trigger- Demonstrations-System integriert werden. Es soll dann am CERN bei Teststrahl-Läufen zum Einsatz kommen, damit die Funktionsfähigkeit auch unter Bedingungen untersuchen werden kann, die dem Einsatz im finalen System möglichst nahe kommen.

Zusammenfassung und Ausblick

Im Rahmen des Demonstrations-Systems für den ATLAS Level-1-Trigger soll ein Modul zum Einsatz kommen, das die Funktionalität des digitalen Front-Ends enthält. Dies umfaßt Bunch-Crossing-Identifikation, Kalibration der Kalorimeterdaten auf transversale Energie und Einstellung der Nullinie sowie Übertragung der Daten zur Trigger-Logik. Zusätzlich muß es eine Möglichkeit geben, die vom Front-End bearbeiteten Daten auszulesen. Das Modul, das dieses leistet, das Front-End-Modul (FEM), wird am Institut für Hochenergiephysik Heidelberg entwickelt. Ziel dieser Diplom-Arbeit war es, die Auslese des Moduls zu implementieren.

Die Vorverarbeitung der vom Kalorimeter kommenden Daten für die eigentliche Logik des Level-1-Triggers wird von einem ASIC, dem Front-End-Asic (FeAsic), übernommen. Er besitzt eine serielle Schnittstelle, über die er konfiguriert und ausgelesen wird. Das FEM verfügt über eine VME-Bus-Schnittstelle, über die Konfigurations- und Auslesedaten gesendet werden. Die Bearbeitung der Daten zwischen FeAsic und VME-Bus wird von einem weiteren Baustein übernommen, dem Front-End-Controller (FEC). Im Rahmen dieser Diplomarbeit wurde der FEC auf einem Xilinx-FPGA implementiert.

Das FPGA-Design erfolgte mit Hilfe der Hardware-Beschreibungs-Sprache Verilog. Aus der Verilog-Beschreibung wurde ein Schaltplan synthetisiert und auf dem FPGA implementiert. Mittels Simulation der Hardware-Umgebung des FEC wurde die korrekte Funktion des Designs überprüft.

Zur Ansteuerung des FEMs wurde Software entwickelt, die das Konfigurieren und Auslesen der FeAsics ermöglicht. Um die Bedienung der Software zu vereinfachen, wurde sie mit einer in Tcl/Tk programmierten grafischen Benutzeroberfläche versehen.

In einem ersten Test des FEMs konnte die prinzipielle Funktionsfähigkeit aller Komponenten, insbesondere des FECs und der Software gezeigt werden. Dabei wurden einzelne Speicher-Fehler auf manchen FeAsics identifiziert, die vermutlich auf Fehler beim Herstellungsprozeß zurückzuführen sind.

Es bleibt noch, das FEM innerhalb des Demonstrations-Systems in Betrieb zu nehmen und seine korrekte Funktion am CERN unter Teststrahlbedingungen zu untersuchen.

Des weiteren wird eine neue Version des FEMs entwickelt werden, die eine höhere Integrationsdichte aufweist und mehr Trigger-Kanäle verarbeitet. So soll dem Ziel näher gekommen werden, alle 8000 Kanäle des finalen Systems in einem kompakten System verarbeiten zu können.

Anhang A

Tokens des FeAsics

Token	Name	Bedeutung
0x04	SrcFadc	Daten vom FADC
0x05	SrcPbBeforeLut	Daten vom Playback-Memory vor LUT
0x06	SrcPbBeforeBcid	Daten vom Playback-Memory vor BCID
0x07	SrcPbAfterBcid	Daten vom Playback-Memory nach BCID
0x1C	LutByPassOn	LUT deaktivieren
0x1D	LutByPassOff	LUT aktivieren
0x14	BcidOperation	BCID aktivieren
0x15	BcidByPass	BCDID deaktivieren
0x16	BcidFirMax	Maximum des FIR-Filters ausgeben
0x17	BcidFirFilter	FIR-Filter-Werte ausgeben
0x18	BcidSynNoEqual	Pulsmaximum Fir(t+1)<Fir(t)>Fir(t-1)
0x19	BcidAsymGEG	Pulsmaximum Fir(t+1)<=Fir(t)>Fir(t-1)
0x1A	BcidAsymGGE	Pulsmaximum Fir(t+1)<Fir(t)=>Fir(t-1)
0x1B	BcidSymEqual	Pulsmaximum Fir(t+1)<=Fir(t)=>Fir(t-1)
0x0C	ReadoutStop	Auslese im Stop Modus
0x0D	ReadoutDera1	kontinuierliche Auslese von 1 Bunch-Crossing
0x0E	ReadoutDera4	kontinuierliche Auslese von 4 Bunch-Crossings
0x0F	ReadoutDera8	kontinuierliche Auslese von 8 Bunch-Crossings
0x11	LoadPipelineDelay	Setzen der Pipeline-Verzögerung, Argument im Adress-Wert des FeAsic-Worts
0x08	CaptBeforeLut	Nehme Pipeline Eingangsdaten vor LUT
0x09	CaptBeforeBcid	Nehme Pipeline Eingangsdaten vor BCID
0x0A	CaptAfterBcid	Nehme Pipeline Eingangsdaten hinter BCID
0x12	LoadLUT	Laden der LUT, benötigt Adress- und Datenwert im Argument
0x13	LoadPlayback	Laden des Playback Memory, benötigt Adress- und Datenwert im Argument

Tabelle A.1: Vom FeAsic verarbeitete Tokens und ihre Bedeutung

Literaturverzeichnis

[1] Christoph Berger, *Teilchenphysik*, Springer-Verlag, 1992

[2] The LHC Study Group, *The Large Hadron Collider, Conceptual Design Report*, 1995, CERN/AC/95-05(LHC), http://www.cern.ch/CERN/LHC/YellowBook95/LHC95/LHC95.html

[3] B.W. Lee, C. Quigg and H. Thacher, Phys. Rev. **D16**, 1519 (1977)

[4] R.M. Barnett et al., Physical Review D54 1, (1996)

[5] Povh, Rith, Scholz, Zetsche, *Teilchen und Kerne*, Springer-Verlag, 1994

[6] European Committee for Future Accelerators, *Large Hadron Collider Workshop, Proceedings, Volume I*, 1990, CERN 90-10

[7] European Committee for Future Accelerators, *Large Hadron Collider Workshop, Proceedings, Volume II*, 1990, CERN 90-10

[8] ATLAS, *Technical Proposal*, CERN, 1994, CERN/LHCC/94-43

[9] ATLAS Homepage, *Atlas Figures*, http://atlasinfo.cern.ch/Atlas/ATLASFIGS/atlasfigures.html

[10] Gregor Wagner, *Entwicklung einer Testumgebung für einen ASIC im Rahmen des ATLAS Level-1-Triggers*, Diplomarbeit, Universität Heidelberg, 1996

[11] Ullrich Pfeiffer, *Analoge optische Signalübertragung vom Kalorimeter zum ATLAS Level-1-Triggersystem*, Diplomarbeit, Universität Heidelberg, 1996, HD-IHEP96-16

[12] Viraj Perera et al., *A First Level Calorimeter Trigger for the ATLAS experiment*, CERN/LHCC/95-56

[13] P.Hanke et al., *A Flash Analog-to-Digital Converter for Trigger Tests with the ATLAS Calorimetry*, RD27 Note 41, 1995

[14] Alexander Mass, *Front End and Fast Readout ASIC: FeAsic Design Report*, ASIC-Note HD-ASIC-17-0796

[15] M. Morris Mano, *Digital Design*, Prentice-Hall, 1984

[16] Alexander Mass, *Front End and Fast Readout ASIC: FeAsic. User Manual*, ASIC-Note HD-ASIC-19-0896

[17] I. Brawn et al., *Bunch Crossing Identification for the ATLAS Level-1 Trigger*, CERN/LHCC/95-56

[18] Alexander Mass, *Proposal of a Front End Digitization and Readout System for the ATLAS Level-1 Calorimeter Trigger*, CERN-LHC96-39

[19] Hewlett Packard, *Low Cost Gigabit Rate Transmit/Receive Chipset, Technical Data*

[20] Xilinx Inc., *The programmable Logic Data Book*, 1996

[21] Cadence Design Systems, Inc., *Verilog-XL Reference*, Online-Dokumentation, 1995

[22] Ulrich Golze, *VLSI-Entwurf eines RISC-Prozessors*, vieweg, 1995

[23] Cadence Design Systems, Inc., *Synergy HDL Synthesizer and Optimizer Modeling Style Guide*, Online-Dokumentation, 1995

[24] Cadence Design Systems, Inc., *Synergy HDL Synthesizer and Optimizer Reference*, Online-Dokumentation, 1995

[25] Cadence Design Systems, Inc., *Simulation and Test Language User Guide*, Online-Dokumentation, 1990

[26] John K. Ousterhout, *Tcl und Tk*, Addison-Wesley, 1995

[27] Sun Microsystems, Inc., *Tcl/Tk Project At Sun Microsystems Laboratories*, http://www.sunlabs.com:80/research/tcl/

[28] Klaus Schmitt, private Mitteilung

Danksagung

Ich möchte allen Menschen danken, die zum Gelingen meiner Diplom-Arbeit beigetragen haben und im weitesten Sinne daran beteiligt waren.

Prof. Meier gab mir die Gelegenheit, diese Arbeit überhaupt durchzuführen.

Prof. Kluge übernahm freundlicherweise die Zweitkorrektur der Arbeit.

Von Alexander Mass erfuhr ich eine ausgezeichnete Betreuung. Er ließ mir auch die nötigen Freiheiten, meine eigenen Ideen zu verwirklichen.

Klaus Schmitt und Peter Stock leisteten hervorragende Arbeit bei der Entwicklung der VME-Karte. Sie waren mir auch bei den Tests des FEMs eine große Hilfe.

Michael Keller sorgte dafür, daß ich mich nicht in den Widrigkeiten von UNIX und Cadence verlief.

Ohne Joachim Boelsems wäre die Zeit weitaus trister gewesen. Er stellte auch immer, wenn benötigt, seine Hilfe zu Verfügung.

Johannes Schemmel las meine Arbeit gründlich Korrektur und fand noch jede Menge Ungereimtheiten.

Allen Mitgliedern des Instituts und ASIC-Labors sei für die außergewöhnlich angenehme Arbeitsatmosphäre und stetige Hilfsbereitschaft gedankt.