Das Auslesesystem für den Ringabbildenden Čerenkovdetektor im HADES Spektrometer

Diplomarbeit
von
Michael Böhmer

Fakultät für Physik der Technischen Universität München
Physik-Department E12
Dezember 1999

Zusammenfassung

In der vorliegenden Arbeit wird die schnelle Ausleseelektronik für die 28 000 Kanäle im Photonenzähler des RICH Detektors vorgestellt, die im HADES-Experiment zum Einsatz kommt. Das Auslesesystem ist modular aufgebaut und besteht aus 450 Frontend-Modulen (FE) mit je 64 Verstärkerkanälen und einer A/D-Wandlungseinheit mit Speicher für 120 Ereignisse. Die Auslese der Frontends sowie eine weitere Datenzwischenspeicherung erfolgt über Readout-Controller (RC) mit VME-Busanschluß. Eine Detektortriggereinheit (DTU) steuert die Abläufe im Auslesesystem.

Die hohe Modularisierung und Konfigurierbarkeit des Systems ermöglicht den Einsatz auch an anderen bildgebenden Detektoren, ohne größere Modifikationen an der Hardware vornehmen zu müssen.

Im Rahmen der Diplomarbeit wurde die Entwicklung des RC-Moduls abgeschlossen und ein Prototyp fertiggestellt. Darüberhinaus wurde das FE-Modul in seinem digitalen Teil grundlegend überarbeitet und die DTU mit der noch fehlenden Steuerlogik vervollständigt.

Mit einem Prototypaufbau wurde das Auslesekonzept auf seine Funktionsfähigkeit hin überprüft. Die Zusammenarbeit des Systems mit den anderen Komponenten des HADES-Datenaufnahmesystems konnte im Labor erfolgreich getestet werden. Dabei wurde die Erfüllung der Designkriterien (10 μs Datenaufnahmezeit, 10^5 Hz Ereignisrate, 10 bit Digitalisierung und asynchrone Auslese über zwei Pufferstufen) bestätigt.

Nach Abschluß der Tests konnte die Kleinserienproduktion aller Module (FE und RC) begonnen und abgeschlossen werden. In einem ersten Testexperiment mit dem RICH am Schwerionenstrahl der GSI konnten erste Erfahrungen gewonnen werden, über die am Ende der Arbeit berichtet wird.

Zur Ansteuerung der Hardware wurde unter Beachtung der HADES Programmierrichtlinien ein umfangreiches Softwarepaket geschrieben, das eine einfache Schnittstelle zur Ansteuerung der komplexen Funktionen bereitstellt. Auf Grundlage dieser Bibliothek können nicht nur Programme zur Konfiguration des Auslesesystems geschrieben werden; eine einfache Integration der Steuerungsfunktionen in die slow control-Software EPICS wird dadurch ermöglicht.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung und Motivation

Ein aktuelles Gebiet in der Kern- und Teilchen-Physik ist die Untersuchung der Eigenschaften von Hadronen in Kernmaterie. Zu diesem Zweck wird momentan an der GSI in Darmstadt das HADES-Experiment aufgebaut, das einen experimentellen Zugang zu diesen Fragestellungen ermöglicht. HADES (**H**igh **A**cceptance **D**i-**E**lectron **S**pectrometer) ist speziell auf die Spektroskopie von Elektron-Positron-Paaren ausgelegt, die aus den leptonischen Zerfallskanälen von Vektormesonen stammen.

1.1 Physikalische Zielsetzung

Seit geraumer Zeit werden auf der Quantenchromodynamik basierende Modelle diskutiert, die eine Restauration der chiralen Symmetrie bei hoher Temperatur ($T > 160\,\mathrm{MeV}$) und Dichte vorhersagen [24], [3]. Als Konsequenz sind Änderungen der Eigenschaften von Hadronen zu erwarten, wenn sie in komprimierter und erhitzter Kernmaterie eingebettet sind. Erste experimentelle Hinweise auf Mediumsmodifikationen liegen bereits vor [1], [28], [23].

Neben Verkürzungen der Zerfallszeit (meßbar über die Vergrößerung der Resonanzbreite) werden Verschiebungen der Polmasse von leichten Vektormesonen (ρ, ω) erwartet. Bereits bei Dichte von Kernmaterie im Grundzustand (ρ_0) sollten meßbare Effekte zu beobachten sein. Beispielsweise können ω-Mesonen mit Hilfe eines π^--Strahls an gebundenen Protonen eines schweren Kerns produziert und die Dielektronen aus dem e^+-e^--Zerfallskanal gemessen werden [25], [26].

$$p + \pi^- \rightarrow n + \omega$$

Zur Untersuchung von hadronischer Materie bei höheren Dichten werden zentrale Schwerionenstöße genutzt. So erreicht man beispielsweise bei Au-Au-Stößen im GSI-SIS Energiebereich Dichten von bis zu $3\rho_0$ und Temperaturen von etwa 50 MeV. In einer zentralen Kollision entsteht dabei ein Feuerball aus dichter und aufgeheizter Kernmaterie, der nach einer charakteristischen Zeit von etwa $10\,\mathrm{fm}/c$ expandiert und abkühlt. Bei der vom SIS lieferbaren Strahl-Energie von 1 – 2 GeV/u entstehen die Vektormesonen (ρ, ω, ϕ) überwiegend durch Mehrteilchenstöße (Produktion unterhalb der Schwelle) innerhalb dieses

Feuerballs. Bei der Messung integriert man über Zerfälle zu unterschiedlichen Zeitpunkten und in unterschiedlich dichten Medien, wodurch die Interpretation des so erhaltenen Massenspektrums schwierig wird.

Meson	Reichweite $c\tau$ [fm]	Masse [MeV]	Zerfallskanäle	Wahrscheinlichkeit [%]
π^0	251	135	$\pi^0 \to \gamma\gamma$ $\pi^0 \to e^+e^-\gamma$	~ 99 ~ 1
η	30	547	$\eta \to \gamma\gamma$ $\eta \to \pi^0\pi^0\pi^0$	~ 39 ~ 32
ρ	1,3	770	$\rho \to \pi\pi$ $\rho \to e^+e^-$	~ 100 $\sim 4{,}5 \cdot 10^{-5}$
ω	23	782	$\omega \to \pi^+\pi^-\pi^0$ $\omega \to e^+e^-$	~ 89 $\sim 7{,}2 \cdot 10^{-5}$
ϕ	44	1 020	$\phi \to K^+K^-$ $\phi \to K_L^0K_S^0$	~ 50 ~ 34

Tabelle 1.1: Eigenschaften der leichten Vektormesonen nach [10].

In Tabelle 1.1 sind die Daten der leichtesten Mesonen aufgeführt; besonders das ρ-Meson ist wegen seiner kurzen Lebensdauer als Sonde geeignet: die Wahrscheinlichkeit, daß ein innerhalb des Feuerballs gebildetes ρ-Meson auch dort zerfällt, ist sehr hoch. Das ρ-Meson besitzt (ebenso wie das ω) einen stark unterdrückten leptonischen Zerfallskanal; das entstehende Leptonenpaar kann die hadronische Materie ohne starke Wechselwirkung verlassen und so zur Messung der invarianten Masse des zerfallenen Mesons genutzt werden.

Die invariante Masse des Mesons ist gleich der Norm des Viererimpulses:

$$m_{inv} = \left\|\overrightarrow{P_\rho}\right\| = \left\| \begin{pmatrix} \overrightarrow{p}_{e^+} + \overrightarrow{p}_{e^-} \\ E_{e^+} + E_{e^-} \end{pmatrix} \right\| \tag{1.1}$$

Wegen des hohen Impulses $p_{e^\pm} \approx 100\,\mathrm{MeV}/c$ kann die Ruhemasse des Elektrons bzw. Positrons von $m_{e^\pm} = 511\,\mathrm{keV}$ vernachlässigt werden. Es ergibt sich näherungsweise:

$$m_{inv} \approx 2 \cdot \sin\left(\frac{\alpha}{2}\right) \cdot \sqrt{p_{e^+} \cdot p_{e^-}} \tag{1.2}$$

Dabei bedeutet $p_{e^\pm}$ den Impulsbetrag des Elektrons bzw. Positrons und α den Winkel zwischen den beiden Impulsvektoren.

Durch Messung der Impulse und des Zwischenwinkels kann man die Massen der im Feuerball zerfallenden Mesonen spektroskopieren. Das neue HADES Spektrometer ist dafür ausgelegt, derartige Elektron-Positron-Paare mit großem Raumwinkel, hoher Massenauf"—lösung und sehr guter Untergrundunterdrückung zu messen.

1.2 Aufbau und Anforderungen des HADES Spektrometers

Als besondere Herausforderung an den Meßaufbau erweisen sich die geplanten Schwerionenstöße: so erwartet man bei zentralen Au-Au-Stößen im Mittel die Entstehung von bis zu 200 geladenen hadronischen Teilchen, die überwiegend in Vorwärtsrichtung, d.h. in die hinter dem festen Target gelegene Hemisphäre emittiert werden. Da aber nur in einem geringen Teil der zentralen Stöße auch wirklich ein ρ-Meson gebildet wird und außerdem der dileptonische Zerfallskanal stark unterdrückt ist, muß durch geeignete Maßnahmen die Nachweiswahrscheinlichkeit für Dileptonen maximal gehalten und eine effektive Unterdrückung des hadronischen Untergrundes (hauptsächlich p, π^+ und π^-) erreicht werden.

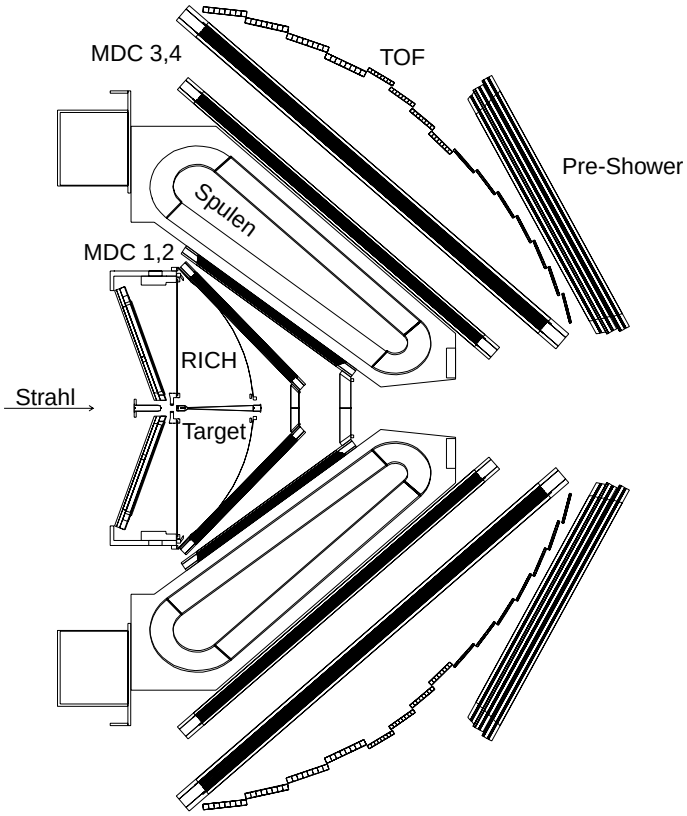


Abbildung 1.1: Schnitt durch das Dileptonenspektrometer HADES. Das Spektrometer ist rotationssymmetrisch zur Strahlachse aufgebaut, wobei jedes Detektorsystem aus sechs Teilstücken besteht. Die Magnetspulen erzeugen nur im Bereich zwischen inneren (MDC 1,2) und äußeren Driftkammern (MDC 3,4) ein Feld.

Im folgenden wird anhand von Abbildung 1.1 auf die grundlegende Struktur des Experimentaufbaus von HADES eingegangen, soweit dies zum Verständnis des Auslesekonzepts notwendig ist. Eine ausführliche Beschreibung des Spektrometers sowie der darin verwendeten Detektoren ist im HADES-Proposal [11] zu finden.

Rings um das segmentierte Target befindet sich ein Ringabbildender Čerenkov-Zähler (RICH), dessen Aufgabe die Leptonenidentifizierung von durchlaufenden Teilchen ist. Auf

die Funktionsweise des RICH wird in Abschnitt 1.3 näher eingegangen. Als nächstes Detektorsystem bilden vier Module von Vieldraht-Driftkammern (MDC) zusammen mit einem Magneten [2] aus sechs supraleitenden Spulen ein Magnetspektrometer, das Informationen über den Impuls der geladenen Teilchen liefert. Es werden die Spuren der Teilchen vor und nach dem Magnetfeld gemessen; aus der Ablenkung erhält man bei bekanntem Magnetfeld den Impuls des gemessenen Teilchens.

Den Abschluß des Spektrometers bildet ein Leptonen-Identifikationssystem, bestehend aus der Flugzeitwand sowie dem Preshower-Zähler. Geladene Teilchen, die unter Polarwinkeln größer 45° aus dem Target laufen, werden in der TOF-Wand über die Flugzeit identifiziert: Elektronen bewegen sich schneller als die schweren π-Mesonen und Protonen.

Teilchen, die unter Polarwinkeln kleiner 45° gestreut werden, können in der TOF-Wand nicht mehr eindeutig identifiziert werden, da die Flugzeitunterschiede zwischen Elektronen und den relativistischen Pionen zu klein werden. Zur Leptonenidentifikation benutzt man einen Preshower-Zähler, der aus drei Drahtkammern mit dazwischen liegenden Bleikonvertern besteht. Ausgenutzt wird hierbei, daß die in den Bleikonvertern von den Leptonen abgegebene Bremsstrahlung im Feld der Atomkerne in einen Schauer von Elektron-Positron-Paaren konvertiert und in den Drahtkammern nachgewiesen wird. Bei Hadronen findet diese Schauerbildung in wesentlich geringerem Umfang statt, eine Unterscheidung in Lepton und Hadron ist somit möglich.

Zusätzlich läßt sich aus der Gesamtanzahl der Treffer im TOF-Detektor auf die Pionen- und Protonen-Multiplizität des betrachteten Ereignisses schließen; dies wird zur Erzeugung eines Multiplizitätstriggers für das Detektorsystem genutzt. Bei Schwerionenstößen kann somit zwischen peripheren (kleine Multiplizität) und zentralen Stößen (hohe Multiplizität) unterschieden werden.

1.3 Der RICH-Detektor

In Čerenkov-Zählern nutzt man die Tatsache aus, daß Teilchen, die sich mit Geschwindigkeiten $v > v_P$ in einem Medium mit einem Brechungsindex $n > 1$ bewegen, eine kegelförmige elektromagnetische Welle erzeugen. Die Phasengeschwindigkeit v_P des Lichtes ist im Medium definiert als

$$v_P = \frac{c_0}{n} \tag{1.3}$$

wobei c_0 die Vakuumlichtgeschwindigkeit und n der Brechungsindex des betrachteten Mediums ist. Bewegt sich nun ein relativistisches Teilchen mit der Geschwindigkeit $v > v_P$ im Medium, so wird in Vorwärtsrichtung unter dem Winkel θ_C elektromagnetische Strahlung abgegeben:

$$\cos\theta_C = \frac{c}{v} = \frac{c_0}{n \cdot v} \leq 1 \tag{1.4}$$

Gibt man nun ein Material mit einem bestimmten Brechungsindex n vor, so können nur Teilchen mit Geschwindigkeiten $v \geq c_0/n$ in diesem Medium Čerenkov-Licht erzeugen. Der Öffnungswinkel θ_C ist dabei ein Maß für die Geschwindigkeit des Teilchens.

Bei HADES wird der Čerenkov-Effekt ausgenutzt, um die emittierten relativistischen Elektronen bzw. Positronen ($v/c \approx 1$) von den deutlich langsameren Hadronen ($v/c \approx 0{,}95$) zu unterscheiden. In der von HADES verwendeten Geometrie ist ein Aufbau der lichtempfindlichen Detektoren in Strahlrichtung nicht möglich. Abhilfe schafft das Konzept des Ringabbildenden Čerenkov-Zählers: das entstehende Licht wird über einen Spiegel in die Hemisphäre hinter dem Target geworfen und dort nachgewiesen. Die abbildenden Eigenschaften des Spiegels bewirken eine Fokussierung des Čerenkov-Lichts in die Detektorebene; der Lichtkegel wird als Čerenkov-Ring abgebildet.

Die Information über die Teilchengeschwindigkeit steckt im Durchmesser des Rings; die Geschwindigkeitsmessung wird damit auf eine Ortsmessung reduziert. Bei HADES erwartet man, daß sich fast alle zu detektierenden Leptonen mit annähernd gleicher Geschwindigkeit $v \approx c$ [9] bewegen. Daher variiert der Ringdurchmesser nur unwesentlich, was eine anschließende Ringerkennung (die wegen der höheren Verarbeitungsgeschwindigkeit in Hardware implementiert ist) wesentlich erleichtert.

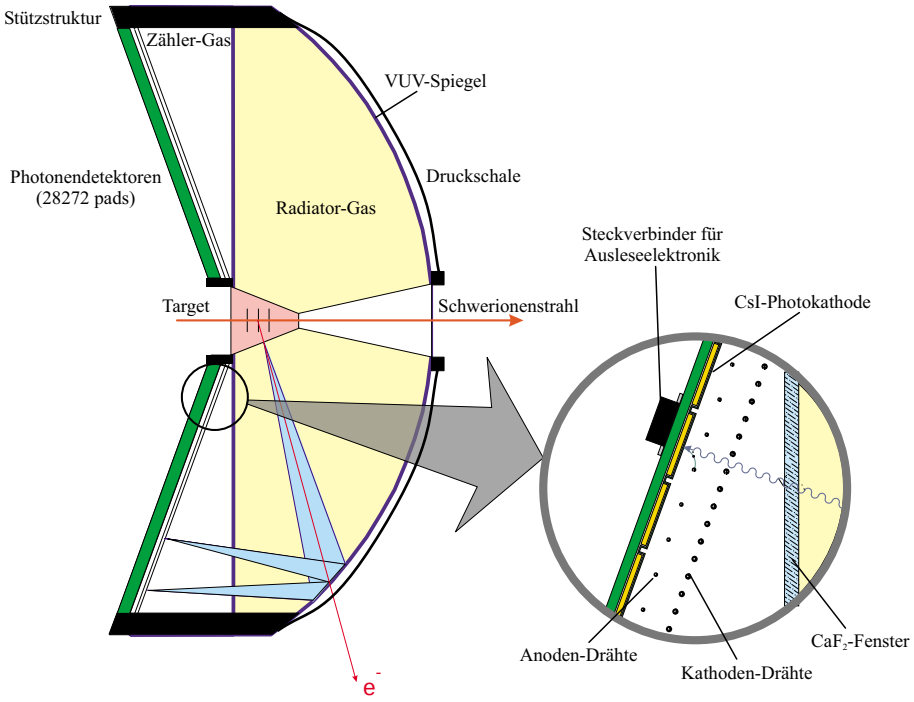


Abbildung 1.2: Schnitt durch den HADES RICH. Der Photonendetektor besteht aus einer Vieldrahtproportionalkammer mit CsI-Photokathode. Im Detailbild erkennt man die Segmentierung der Kathodenebene zum Erreichen der benötigten hohen Ortsauflösung. Die Ankoppelung der Ausleseelektronik erfolgt über die ebenfalls dargestellten Steckverbinder.

Der Aufbau des HADES RICH ist als Querschnitt in Abbildung 1.2 gezeigt. Der Detektor besteht aus zwei durch eine CaF_2-Scheibe getrennten Gasvolumen. Das in Strahlrichtung gelegene Volumen dient als Radiator. Es ist mit C_4F_{10} (Brechungsindex $n \approx 1{,}004$ [11]) gefüllt und enthält außerdem den aus mehreren Segmenten zusammengesetzten Spiegel. Aufgrund der verwendeten Geometrie (das Target liegt in Strahlrichtung hinter der CaF_2-

Trennscheibe) und der Stoßkinematik gelangen fast alle im Stoß entstehenden Teilchen direkt in das Radiatorgasvolumen; in seltenen Fällen können die geladenen Teilchen jedoch auch direkt in das Zählergas gelangen. Das Radiatorgas ist so gewählt, daß nur geladene Teilchen mit Geschwindigkeiten $v > 0{,}99\,c$ [11] Čerenkov-Licht aussenden können, nicht jedoch Teilchen mit kleineren Geschwindigkeiten. Das Radiatorgas ist somit hadronenblind und nur auf Elektronen und Positronen empfindlich.

Da die im Schwerionenstoß entstehenden Teilchen in die hinter dem RICH positionierten Detektorsysteme gelangen sollen, mußten für die Konstruktion des Spiegels und der Druckschale, die das Radiatorgasvolumen strahlabwärts abschließt, Lösungen mit großer Strahlungslänge gewählt werden. Details zu diesen Komponenten finden sich in [15]. Das entstehende Licht wird über den Spiegel in das rückwärtig gelegene Zählervolumen gelenkt. Dabei muß es die Trennscheibe passieren, für die wegen der benötigen hohen Transmission im VUV-Bereich ($\lambda \approx 140\,\mathrm{nm}$) Kalziumfluorid als Material gewählt wurde.

Das Zählervolumen bildet zusammen mit den eigentlichen Detektormodulen einen Vieldraht-Proportionalzähler zum Nachweis der im Radiatorvolumen erzeugten Čerenkov-Photonen. Da man pro Lepton den Simulationen nach nur mit einer sehr geringen Anzahl von etwa 12 – 20 Photonen [31] pro Ring rechnen kann, ist der Zähler auf den Nachweis von einzelnen Photonen ausgelegt. Die Čerenkov-Photonen durchlaufen das Zählergas (CH_4, hohe Transmission im VUV-Bereich) und treffen auf eine speziell beschichtete Kathodenebene, die die Photokathode des Zählers bildet. Sie ist pro Sektor aus 4 712 einzelnen, mit einer Schicht aus Graphit und Cäsiumiodid (CsI) versehenen Elementen (den Pads), aufgebaut. Beim Auftreffen auf ein Pad löst ein Čerenkov-Photon mit im Mittel 20% Wahrscheinlichkeit (Quanteneffizienz) ein Photoelektron aus, das nach der Gasverstärkung im CH_4 (ca. 10^5) als Influenzladung auf dem Pad nachgewiesen werden kann. Sollten geladene Teilchen in den Photonendetektor gelangen, so würde dies im allgemeinen zu einer deutlich höheren Ladungsmenge und damit Pulshöhe führen. Somit kann über die Pulshöhe in der Off"—line-Analyse eine Unterscheidung zwischen Photonen und geladenen Teilchen durchgeführt werden. Hierbei wird auch ausgenutzt, daß die Teilchen charakteristische Signaturen im Detektor hinterlassen: Leptonen bilden einen Ring mit mittleren Pulshöhen, geladene Hadronen je nach Eintrittswinkel einen Strich oder einen Fleck mit hohen Pulshöhen.

Im Rahmen der vorliegenden Arbeit wurde die Signalauslese und die digitale Weiterverarbeitung der Daten des RICH Detektors bis hin zur Datenaufnahme zur Einsatzreife gebracht. In den folgenden Kapiteln werden nach dem HADES Auslese- und Triggerkonzept die Details des RICH-Auslesesystems beschrieben und über erste Erfahrungen im Laborbetrieb berichtet.

Da sich bei der Beschreibung des Auslesesystems fachspezifische Ausdrücke und in der Kollaboration häufig benutzte Abkürzungen nicht vermeiden lassen, ist am Ende der Arbeit ein Glossar (ab Seite 91) angefügt, das diese Begriffe jeweils kurz erläutert. Aus Gründen der Übersichtlichkeit wurde auf die Erklärung der Begriffe direkt im Text verzichtet.

Kapitel 2

Das HADES Auslese- und Triggerkonzept

Das HADES Spektrometer ist nicht nur als Instrument höchster Auf"—lösung und Akzeptanz konzipiert, sondern auch für hohe Ereignisraten ausgelegt, die eine optimale Nutzung des vom SIS-Beschleuniger verfügbaren Strahlstroms (bis zu 10^8 Au-Ionen pro Sekunde) ermöglicht [11]. Im Mittel erwartet man für ein Target mit 1% Wechselwirkungsdicke 10^6 Kernreaktionen pro Sekunde, wovon erfahrungsgemäß etwa 10% ausreichend zentral ablaufen. Die Ausleseelektronik muß daher in der Lage sein, eine Rate von bis zu 10^5 Ereignissen pro Sekunde zu verarbeiten. Aufgrund der hohen Pionen- und γ-Multiplizität ($\pi^0 \to \gamma\gamma$) und trotz der großen Strahlungslänge von Target und Detektorsystem wird pro zentralem Stoß etwa ein Elektron aus externer Konversion erwartet. Dagegen sind die interessanten e^+-e^--Paare aus Mesonenzerfällen um einen Faktor 10^4 – 10^5 unterdrückt.

Die Rate von 10^5 beschränkt die maximal erlaubte Totzeit des Detektors auf 10μs. Betrachtet man beispielsweise den RICH, der pro Sektor 4712 auszulesende Pads besitzt, und vernachlässigt sämtliche zur Verwaltung der Daten notwendigen Zusatzinformationen, so ergibt sich bei einer Rate von 10^5 Hz eine auszulesende Roh-Datenmenge von

$$10^5 \,\frac{1}{\mathrm{s}} \cdot 6 \text{ Sektoren} \cdot 4\,712 \,\frac{\text{Pads}}{\text{Sektor}} \cdot 16 \,\frac{\text{Bits}}{\text{Pads}} \approx 5{,}27 \text{ Gigabyte/s} \tag{2.1}$$

Dies ist eine Datenmenge, die mit keinem heute kommerziell erhältlichen System zu verarbeiten ist. Auch die Speicherung der Daten wäre bei den Strahlzeiten, die jeweils mehrere Wochen dauern, nicht durchführbar.

Für HADES wurde deshalb ein mehrstufiges Auslesesystem entworfen, das folgenden Ansprüchen gerecht werden soll [17]:

1. Hohe Ratenfestigkeit bis zu 10^5 Hz
2. Erfassung und Speicherung nur von physikalisch interessanten Ereignissen

3. Eindeutige Zuordnung von Daten zu einem Ereignis
4. Leichte Anpaßbarkeit an die verwendeten verschiedenen Detektorsysteme

Dieses Trigger- und Auslesekonzept soll im folgenden kurz vorgestellt werden.

2.1 Das Triggersystem

Im Gegensatz zu früheren Experimenten wird bei HADES ein neues, asynchron arbeitendes mehrstufiges Auslesesystem verwendet. Bei einem synchronen System wird mit einem gemeinsamen Triggersignal die Datenaufnahme gestartet und anschließend das gesamte Detektorsystem ausgelesen; nach jedem Trigger erhält man sofort einen kompletten Datensatz. Das langsamste Detektor-Auslesesystem bestimmt die Totzeit des Gesamtsystems. Die oben beschriebenen Anforderungen erfordern jedoch ein neues Konzept: der Datenweg vom Detektor zum Speichersystem wird in mehrere asynchron arbeitende Pufferstufen zerlegt. So kann das Triggersignal zur schnellen Speicherung in der ersten Pufferstufe genutzt und damit eine kurze Totzeit erreicht werden; ein mehrstufiges Triggersystem koordiniert den Weitertransport der Daten zu weiteren Pufferstufen und ermöglicht eine Selektierung von interessanten Datensätzen. Den Daten hinzugefügte Verwaltungsinformationen ermöglichen anschließend das Zusammensetzen der zu speichernden Datenpakete zu einem kompletten Ereignis.

Eine ausführliche Erläuterung zum HADES-Triggerkonzept findet man in [21], hier wird nur eine kurze Übersicht über die einzelnen Triggerstufen gegeben.

Der LVL1-Trigger (Multiplizität)

Der LVL1-Trigger wird aus den Daten von TOF bei einer für einen zentralen Stoß hinreichend großen Multiplizität erzeugt. Er bewirkt die Erfassung der in den einzelnen Detektorsystemen anstehenden analogen Daten sowie deren digitale Speicherung in der ersten Pufferstufe (der LVL1-Pipe). Außerdem werden die so erfaßten Daten der zur Entscheidungsfindung verwendeten Detektoren an entsprechende Bildverarbeitungs- und Entscheidungseinheiten weitergeleitet.

Der LVL2-Trigger (Leptonenpaare)

Aus den Ergebnissen der Bildverarbeitungssysteme wird in einer Entscheidungseinheit der LVL2-Trigger gewonnen. Die wesentlichen Entscheidungskriterien für den Nachweis eines Leptons sind: Ringerkennung im RICH, Schauererkennung im PreShower oder geeignete Flugzeit im TOF. Bei ausreichenden Hinweisen auf ein für die betrachtete Physik interessantes Ereignis werden über den LVL2-Trigger die jeweiligen Datensätze in die zweite Pufferstufe (die LVL2-Pipe) übertragen, andernfalls werden die Datensätze direkt in der LVL1-Pipe gelöscht.

Die Kombination aus LVL1- und LVL2-Trigger sorgt also dafür, nur potentiell relevante Datensätze zu erzeugen und zu transportieren, die ein e^+-e^--Paar enthalten. Wegen der notwendigen Geschwindigkeit sind die Bewertungssysteme komplett in Hardware aufgebaut. Eine weitere Siebung der Datensätze ist notwendig, da stets ein gewisser Anteil von falschen Entscheidungen getroffen wird: einerseits erlaubt die Realisierung der Mustererkennung in Hardware nur relativ einfache Algorithmen, andererseits können bei unvollständigem Nachweis von Teilchen in den Detektorsystemen falsche Identifizierungen entstehen.

Der LVL3-Trigger (Teilchenspur-Analyse)

Die letzte Triggerstufe ist komplett in Software realisiert. Die Ergebnisse der Teilchenspur-Analyse der MDC-Daten werden mit den Ringmittelpunkten aus der Entscheidungseinheit verglichen und bei entsprechender Übereinstimmung der LVL3-Trigger erzeugt. Mit diesem Trigger werden die CPUs angewiesen, bestimmte Datensätze aus den LVL2-Pipes über ein ATM-Netzwerk zu einem zentralen Server zu transportieren, dort zu einem kompletten Datensatz zusammenzusetzen und zur späteren Analyse auf Magnetband zu speichern. Diese Triggerstufe dient im wesentlichen zur Unterdrückung von Untergrundereignissen und Fehlzuordnungen von e^+-e^--Paaren.

2.2 Die LVL1-Pipe

Die LVL1-Pipe bildet die erste Pufferstufe des Auslesesystems. In ihr werden auf einen LVL1-Trigger hin die Daten der einzelnen Detektoren bis zur Entscheidung über ihre weitere Verwendung zwischengespeichert. In der LVL1-Pipe befinden sich daher Detektordaten für alle Ereignisse mit und ohne Leptonengehalt. Da das Triggersystem eine bestimmte Zeit (bei HADES rechnet man mit 200 – 400 μs [20]) zur Erzeugung des LVL2-Triggers benötigt, ist die LVL1-Pipe auf die Speicherung mehrerer kompletter Datensätze ausgelegt. Die Pufferfunktion erfordert außerdem die Möglichkeit des vollkommen asynchronen Schreib- und Lesezugriffs auf die LVL1-Pipe.

Bei HADES ist das Augenmerk auf zentrale Schwerionenstöße gerichtet. Die resultierende hohe Multiplizität an geladenen Teilchen bedeutet für alle Detektoren, die nicht hadronenblind wie der RICH sind, eine hohe Belegungsmultiplizität der Detektorzellen. Die Speichergröße der LVL1-Pipe ist daher durch die Segmentierung der einzelnen Detektoren bedingt und beispielsweise beim RICH ausgelegt zur Speicherung von 120 Ereignissen (siehe Abschnitt 3.1).

Datensätze, die die Signatur eines Leptonenpaares nicht enthalten, können noch vor dem Transport verworfen werden. Das im Vergleich zum Transport der Daten erheblich schnellere Löschen in der LVL1-Pipe verhindert das Voll-Laufen der LVL1-Pipe und ermöglicht eine effektive Nutzung der Pufferstruktur in der Datenaufnahme.

2.3 Die LVL2-Pipe

Die zweite Pufferstufe des HADES Auslesesystems wird als LVL2-Pipe bezeichnet. Im Gegensatz zur LVL1-Pipe stehen in der LVL2-Pipe nur noch Datensätze, die Signaturen für ein Dilepton-Ereignis aufweisen. Eine weitere Besonderheit ist, daß alle Datensätze in der LVL2-Pipe bereits so formatiert sind, daß beim weiteren Transport keinerlei Softwareeingriffe in die Datensätze notwendig sind. Diese Formatierung ermöglicht erst den schnellen Transport von Daten aus der LVL2-Pipe zur Speicherung in der DAQ.

Die LVL2-Pipe wird über den LVL2-Trigger mit Datensätzen aus der LVL1-Pipe gefüllt. Man erwartet bei der vorgesehenen Erzeugung des LVL2-Triggers an dieser Stelle eine Reduktion der zu transportierenden Daten um den Faktor 100. Dies reduziert die Anforderungen an die Bandbreite des Auslesesystems.

Die LVL2-Pipe stellt außerdem die eigentliche Schnittstelle zur DAQ-Software dar.

2.4 Speicherung in der DAQ

Als letzte Stufe des Auslesesystems steht die Speicherung der Datensätze in der Datenaufnahme (data aquisition, DAQ). Hierbei wird nochmals über den LVL3-Trigger eine Selektierung der Datensätze vorgenommen. Durch die in den ersten zwei Stufen vorgenommene Datenreduktion ist dies zeitlich unkritisch, die maximal erlaubte Zeit zur Bearbeitung eines Datensatzes hat deutlich zugenommen.

Bei HADES werden kommerzielle Systeme zur Speicherung der Daten eingesetzt. Die in den LVL2-Pipes der einzelnen Detektoren gepufferten Datensätze werden auf Anweisung des LVL3-Triggers hin über ein schnelles ATM-Netzwerk auf einem zentralen Server gesammelt und dort vom Eventbuilder zu kompletten Ereignissen zusammengesetzt. Im Gegensatz zu synchronen Auslesesystemen entstehen diese Datensätze erst mit Verzögerung, die durch das mehrstufige Triggersystem bedingt sind. Beim Zusammensetzen der Ereignisse muß auch auf das asynchrone Verhalten des Auslesesystems Rücksicht genommen werden: Daten einzelner Detektoren können mit Verspätung relativ zueinander eintreffen; vor der Speicherung steht deshalb wiederum eine Pufferstufe, die diese Verzögerungen ausgleicht.

Kapitel 3

Das RICH Auslesesystem

3.1 Erwartete Datenmengen

Im gesamten RICH-Detektor befinden sich insgesamt 28 272 ladungsempfindliche Pads, deren Pulshöhen zusammen mit der Koordinate des entsprechenden Pads gespeichert werden müssen (siehe Formel 2.1). Um die notwendige Datenrate bei der Auslese der Analogdaten zu reduzieren, ist es sinnvoll, unter Beachtung der Randbedingungen die Auslese des Detektors zu parallelisieren:

1. Eine obere Grenze für die Auslese wird durch die maximale Totzeit von 10μs vorgegeben. Innerhalb dieser Zeit ist es nur möglich, eine bestimmte Anzahl an Analogdaten zu digitalisieren und zu speichern.
2. Zur Reduktion der Datenmenge sollten nur Pulshöhen gespeichert werden, die von Photonen stammen, nicht jedoch das stets vorhandene Rauschen des Systems.
3. Die verwendeten Komponenten des Systems geben eine maximale Taktfrequenz von ca. 32 MHz vor, die prinzipiell nicht überschritten werden kann.
4. Der Anteil an Verwaltungsinformationen sollte in vernünftigem Verhältnis (etwa 10%) zu den Nutzdaten stehen.

Diese Randbedingungen haben beim RICH zur Entscheidung geführt, jeweils Gruppen von 64 Pads (im folgenden auch Kanäle genannt), parallel auszulesen. Diese Aufteilung reduziert einerseits die Datenmenge, die pro Gruppe in der erlaubten Zeit erzeugt wird (und damit die benötigte Bandbreite). Andererseits müssen Verwaltungsinformationen hinzugefügt werden, die beim späteren Zusammenführen der Daten eine eindeutige Zuteilung von Pulshöhen zu den Kanälen erlauben.

Der Detektor ist in 6 Sektoren mit jeweils 4 712 Pads aufgeteilt. Bei einer Aufteilung jedes Sektors in Gruppen zu 64 Kanälen ergibt sich die Anzahl der für einen Sektor mindestens notwendigen Auslesegruppen zu

$$4\,712\,\text{Pads} \cdot \left(64\,\frac{\text{Pads}}{\text{Auslesegruppe}}\right)^{-1} = 74\,\text{Auslesegruppen} \tag{3.1}$$

Aus technischen Gründen bei der Produktion des Detektors werden pro Sektor allerdings 75 Auslesegruppen benötigt; für den gesamten Detektor sind also $6 \cdot 75 = 450$ Auslesegruppen notwendig.

Um innerhalb einer Auslesegruppe einen Kanal eindeutig zu identifizieren, wird der Pulshöhe (die mit 10bit Auf"—lösung gespeichert wird) eine in 6bit kodierte Kanalnummer zugeordnet. Jeder Datensatz wird außerdem mit einer Nummer (dem sog. Triggertag) abgespeichert, die für alle Datensätze eines Ereignisses gleich ist und somit später eine eindeutige Zusammenführung von einzelnen Datensätzen erlaubt. Pro Auslesegruppe ergibt sich somit bei der angepeilten LVL1-Triggerrate von 10^5 Hz eine Datenrate pro Auslesegruppe von:

$$10^5\,\frac{1}{\text{s}} \cdot \left(64\,\text{Pads} \cdot 16\,\frac{\text{bit}}{\text{Pad}} + 16\,\text{bit}\right) = 12{,}4\,\text{Megabyte/s} \tag{3.2}$$

In dieser Formel ist die maximale Anzahl von 64 Kanälen und die Verwaltungsnummer berücksichtigt. Die Datenrate ist mit den verwendeten Komponenten zu bewältigen. Jedoch hat sich — bedingt durch das Einfügen von Verwaltungsinformationen — das Datenaufkommen bezogen auf den gesamten Detektor auf 5,4 Gigabyte/s erhöht.

Eine weitere Reduktion der Datenmenge, die in der LVL1-Pipe zu speichern ist, erreicht man, wenn man das elektronische Rauschen der nicht getroffenen Kanäle effektiv unterdrückt. Somit werden nur die Daten von wirklich getroffenen Pads gespeichert. Aus Simulationen und ersten Experimenten mit dem Prototypzähler erwartet man, daß pro Ereignis maximal 10% aller Pads wirklich ansprechen. Im Schnitt reduziert das bei homogen angenommenem Ansprechen des Detektors die Datenrate pro Auslesegruppe auf:

$$10^5\,\frac{1}{\text{s}} \cdot \left(64\,\text{Pads} \cdot 10\% \cdot 16\,\frac{\text{bit}}{\text{Pad}} + 16\,\text{bit}\right) = 1{,}4\,\text{Megabyte/s} \tag{3.3}$$

Weitere Anforderungen entstehen durch das in Abschnitt 2.1 erläuterte mehrstufige Triggersystem. Es ist notwendig, an zwei Stellen Zwischenspeicher einzufügen, die jeweils mehrere Datensätze bis zur weiteren Verarbeitung speichern können. Dabei ergibt sich aus der Geschwindigkeit der Einheit, die über die weitere Verwendung der Daten entscheidet, die jeweils notwendige Tiefe des Zwischenspeichers:

- Bei der LVL1-Pipe bestimmt die Verzögerung durch die Bildverarbeitungs- und Entscheidungseinheit die notwendige Tiefe. Man rechnet mit einer Durchlauf"—latenz von etwa 40 Ereignissen [20] bis zur Erzeugung des LVL2-Triggers. Bezogen auf die durchschnittliche Ereignisgröße (siehe Formel 3.3) ergibt sich die Tiefe der LVL1-Pipe zu

$$40 \cdot 10\mu\text{s} \cdot 10^5\,\frac{1}{\text{s}} \cdot \left(64\,\text{Pads} \cdot 10\% \cdot 16\,\frac{\text{bit}}{\text{Pad}} + 16\,\text{bit}\right) \cdot \frac{1}{16}\,\frac{1}{\text{bit}} = 296\,\text{Worte} \tag{3.4}$$

Aus Sicherheitsgründen wurde die LVL1-Pipe des RICH deutlich größer ausgelegt: die Speichertiefe von 1 024 Worte (entsprechend 128 durchschnittlichen Ereignissen) liegt um den Faktor 3 über dieser Abschätzung und ermöglicht selbst im unwahrscheinlichsten Fall (Ansprechen des gesamten Detektors) das Zwischenspeichern von 15 Ereignissen.

- Für die Tiefe der LVL2-Pipe entscheidend ist die Verarbeitungsgeschwindigkeit des Spurerkennungsalgorithmus auf der CPU. Diese ist jedoch nur schwer abschätzbar, da sich diese Algorithmen noch in der Entwicklung befinden; außerdem lassen sich sowohl durch Softwareoptimierung als auch durch die stetig voranschreitende Entwicklung auf dem Gebiet der Rechnertechnik Geschwindigkeitsoptimierungen vornehmen. Für die verwendete Ausleseelektronik wurde ein Speicher von 512 kByte vorgesehen. Die hierfür notwendige Abschätzung setzt jedoch Kenntnisse über den internen Aufbau des Auslesesystems voraus und wird deshalb erst in Abschnitt 5.6 gebracht.

3.2 Weitere Anforderungen

Das Auslesesystem muß in der Lage sein, die Daten aus der LVL1-Pipe mit hoher Geschwindigkeit und in der für das weiterverarbeitende System korrekten Form zu übertragen. Hierbei ist es wünschenswert, daß eine beliebige Transformation der jeweiligen Padkoordinaten erfolgen kann. Dies erleichtert sowohl die Entwicklung des Bildverarbeitungssystems und des Frontends als auch einen später eventuell notwendigen Wechsel zu einem System mit anderen Anforderungen.

Für das RICH Auslesesystem ergeben sich also folgende weitere Anforderungen:

5. Leichte Skalierbarkeit
6. Hohe Verarbeitungsgeschwindigkeit
7. Wartungsfreundlichkeit des Gesamtsystems

Um diese Anforderungen zu erfüllen, wurde das Auslesesystem in mehrere Module aufgeteilt: Die eigentliche Datenaufnahme und Zwischenspeicherung in der LVL1-Pipe findet direkt am Detektor im Frontend-Modul (entspricht einer Auslesegruppe) statt. Die Zusammenfassung der einzelnen Datensätze aus den LVL1-Pipes des RICH findet im Readout-Controller statt. Dieses Modul steuert über ein einfaches Interface die Auslese von Daten aus bis zu 64 Frontend-Modulen und deren Speicherung in der LVL2-Pipe. Als Schnittstelle zur zentralen Triggereinheit dient die Detektor-Triggereinheit (DTU), die die Umsetzung der einzelnen Triggerbefehle in Auslesezyklen vornimmt.

Diese Modularisierung ermöglicht eine leichte Erweiterung des Systems bei neu hinzukommenden Detektormodulen und erhöht die Verarbeitungsgeschwindigkeit des Auslesesystems durch die hohe Parallelisierung der Datenerfassung in den Frontend-Modulen. Der

Austausch defekter Module wird ebenfalls erleichtert.
Die Aufteilung des RICH Auslesesystems in drei Module stellt somit einen guten Kompromiß dar: der Verkabelungsaufwand am Detektor hält sich in Grenzen, die Übersichtlichkeit des Gesamtsystems bleibt erhalten.

3.3 Einbindung in das HADES-Triggerkonzept

Die Umsetzung der in Kapitel 2 vorgestellten Anforderungen an das Auslesesystem und dessen Einbindung in das Triggerkonzept werden hier schematisch anhand Abbildung 3.1 dargestellt.

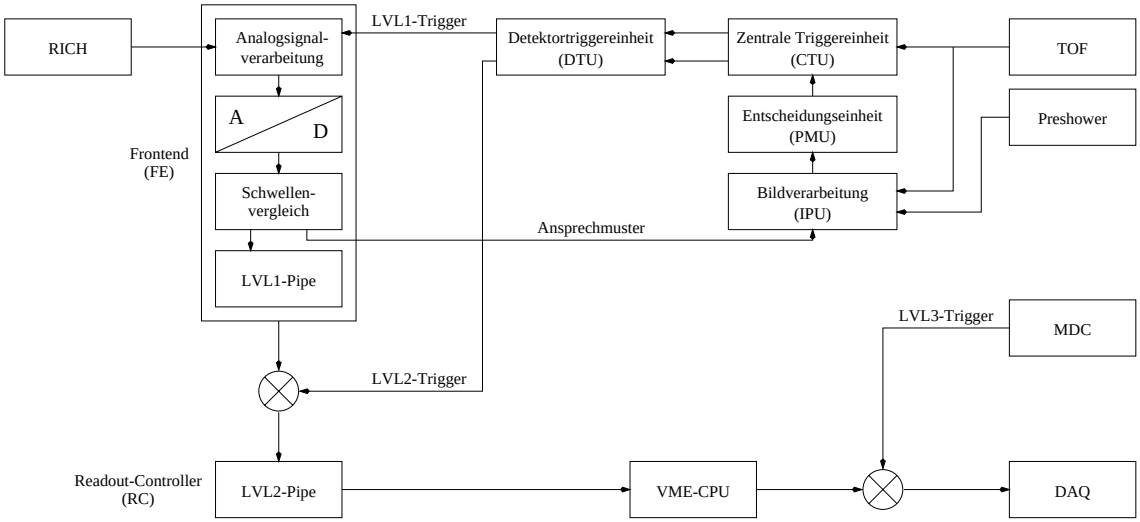


Abbildung 3.1: Auslese- und Triggersystem für den HADES RICH. Die üblicherweise verwendeten Abkürzungen für einige Komponenten sind ebenfalls angegeben (genaue Bedeutung siehe Glossar).

Die Erzeugung des Triggersignals läuft wie folgt ab: Aus der vom TOF-Detektor gelieferten Multiplizität wird ein Triggersignal erzeugt und in die Triggereinheit geleitet. Ist das System momentan aufnahmebereit, so wird dieser Multiplizitätstrigger in einen LVL1-Trigger umgesetzt, der an alle angeschlossenen Detektoreinheiten verteilt wird.

Beim RICH löst dieser LVL1-Trigger folgende Funktionen aus:

- Analoge Signalverarbeitung
- Analog-Digital-Wandlung der Pulshöhen
- Vergleich mit den vorgegebenen Schwellwerten
- Abspeichern in der LVL1-Pipe

Anschließend werden die Daten an die Bildverarbeitungseinheit weitergeleitet. Diese sucht in den Daten nach Ringsignaturen und leitet die Koordinaten der gefundenen potentiellen Ringmittelpunkte an die Entscheidungseinheit weiter. Dort werden diese Daten mit den aus TOF und Preshower gelieferten Koordinaten von möglichen Leptonenkandidaten verglichen. Bei genügend hoher Übereinstimmung der Teilchenkoordinaten wird über die Triggereinheit ein LVL2-Trigger mit positiver Entscheidung ausgegeben. Dies führt dazu, daß die Daten aus der LVL1-Pipe in die LVL2-Pipe übertragen werden. Sollte keine ausreichende Übereinstimmung zwischen RICH- und TOF- bzw. Preshower-Daten vorliegen, so werden die in der LVL1-Pipe zwischengespeicherten Daten per LVL2-Trigger mit negativer Entscheidung gelöscht.

Während der Zeit, in der die Bildverarbeitungs- und die Entscheidungseinheit die einlaufenden Daten untersuchen, können bereits neue LVL1-Trigger von der Triggereinheit erzeugt werden. Diese Daten werden zwischengespeichert und — sobald die weiterverarbeitenden Einheiten wieder empfangsbereit sind — an diese weitergeleitet.

Anschließend werden die Ansprechmuster (hit pattern) von MDC, TOF, Preshower und RICH durch einen Spurerkennungsalgorithmus verglichen. Bei Übereinstimmung der Ergebnisse wird ein LVL3-Trigger erzeugt und die ausgewählten Datensätze aus der LVL2-Pipe über ein ATM-Netzwerk an die Datenspeichereinheit übertragen.

Auch hier können während der Bearbeitung der in der LVL2-Pipe anstehenden Daten neue hinzukommen.

3.4 Technische Realisierung

In Abbildung 3.1 wurde bereits eine gewisse Aufteilung nach Funktionsblöcken vorgenommen. Für die technische Realisierung gibt es jedoch noch einige Randbedingungen, die eine weitere Unterteilung in einzelne Baugruppen erforderlich machen.

- Der Photonendetektor des RICH ist in sechs gleich aufgebaute Sektoren unterteilt. Jeder dieser Sektoren beinhaltet 75 Auslesegruppen, die in jeweils einem Frontend-Modul verwirklicht werden.
- Um eine möglichst hohe Auslesegeschwindigkeit bei geringem Verkabelungsaufwand zu erreichen, werden jeweils 5 Frontendmodule auf einer Treiberplatine (backplane) verkettet (daisy chaining). Pro Sektor erhält man damit 16 Ketten von Frontendmodulen.
- Diese Ketten müssen in sinnvollen Gruppen zusammengefaßt werden. Da der hierfür verwendete Readout-Controller auf dem VME-Bus basieren soll und somit die nutzbare Platinenfläche festgelegten Begrenzungen unterliegt, werden jeweils 8 Frontendketten an die Eingänge (ports) eines Readout-Controllers angeschlossen. Zusätzlich erzielt man auch in dieser Stufe durch die Parallelisierung des Ausleseprozesses Geschwindigkeitsgewinne.

- Die Auslese der in den Readout-Controllern implementierten LVL2-Pipe erfolgt durch VME-CPUs, die jeweils die Readout-Controller von zwei Sektoren in einem VME-Crate ansteuern. Diese Zusammenfassung kommt ebenfalls den Platzansprüchen der Ausleseelektronik entgegen, die wegen möglichst kurzer Kabellängen direkt hinter den Photonenzählermodulen im Experimentaufbau untergebracht ist.

Die Triggerverteilung ist aus praktischen Überlegungen ebenfalls in zwei Teile aufgespalten:

- Der zur Umsetzung des Multiplizitätstriggers notwendige Teil wird ebenso wie die Schnittstelle zur Bildverarbeitung (der IPU) auf einer zentralen Triggereinheit (der CTU) implementiert, die zentral für den gesamten Experimentaufbau Triggerbefehle erzeugt und über den Triggerbus an alle Detektorsysteme verteilt.
- Sämtliche zur Ansteuerung der jeweiligen Detektor-Ausleseelektronik notwendige Logik wird in sogenannte DTUs ausgelagert. Diese Baugruppe ist ebenfalls als VME-Karte ausgelegt und steuert beim RICH jeweils vier Readout-Controller an. Die DTU-Module sind für alle Detektorsysteme gleich ausgelegt und werden durch die Programmierung des steuernden FPGAs sowie eventuell notwendige Aufsteckmodule (z.B. zur Pegelwandlung) an die jeweiligen Detektoren angepaßt.

Der modulare Aufbau sowohl des Triggersystems als auch der Ausleseelektronik hat sich sowohl bei Labortests als auch bei den Elektroniktests im Rahmen von Teststrahlzeiten als sinnvoll erwiesen. So kann jede Gruppe mit einer Kombination aus CTU, DTU und der jeweiligen Ausleseelektronik getrennt ihre Systeme testen und anschließend das Auslesesystem durch Verbinden einer CTU mit den jeweiligen DTUs zu einem gemeinsamen Triggerbus erweitern.

In den Labortests der RICH-Ausleseelektronik wurde diese Skalierbarkeit ebenfalls ausgenutzt. Das System wurde Schritt für Schritt, ausgehend von einem Readout-Controller und einem Frontend, aufgebaut. Dabei auftretendes Fehlverhalten wurde in einem Minimalsystem reproduziert und somit einer systematischen Fehlersuche zugänglich gemacht. Als Ergebnis dieser Entwicklung konnte schon nach sechs Monaten der Schritt vom Prototypen zum endgültigen Serienmodell vorgenommen werden.

3.5 Auswahl der verwendeten Technologie

Zur Implementierung der benötigten Funktionen wurde anfangs eine Entscheidung über den Aufbau der Systemkomponenten getroffen. Dabei mußte ein Kompromiß zwischen den folgenden Anforderungen gefunden werden:

1. Flexibilität
2. Erweiterbarkeit

3. Geschwindigkeit

4. Kosten

Für das RICH-Auslesesystem wurde als Optimum eine Kombination aus Analog-ASICs, einigen Standardkomponenten und FPGAs ausgewählt. Zum besseren Verständnis dieser Wahl werden in den folgenden Abschnitten die zur Auswahl stehenden Technologien mit ihren Vor- und Nachteilen diskutiert.

3.5.1 Standardkomponenten

Als Standardkomponenten bezeichnet man in der Elektronik Bauelemente und integrierte Schaltungen, die auf Grund ihrer weiten Verbreitung in der Industrie in großen Stückzahlen und zu günstigen Preisen erhältlich sind. Bedingt dadurch gibt es bereits viele Geräte, in denen diese Komponenten verwendet werden, was sich sowohl in ihrer Zuverlässigkeit als auch einer breiten Anzahl an Anwendungsbeispielen positiv niederschlägt.

Ein typisches Beispiel für Logik-Standardbausteine sind die ICs der Baureihe 74xx, die in verschiedenen Familien seit etlichen Jahren von mehreren Herstellern gefertigt werden und an bestimmte Anforderungen angepaßt sind. Ebenso sind Standardspeicher in nahezu beliebigen Größen, Konfigurationen und Technologien erhältlich.

Jedoch zeigt bereits eine grobe Abschätzung, daß ein Aufbau der Ausleseelektronik nur aus solchen Standardkomponenten scheitern muß:

- Die elf auf einem Readout-Controller untergebrachten FPGAs enthalten insgesamt das Äquivalent von ca. 49 000 Logikgattern.
- Typische ICs der 74xx-Reihe enthalten in einem 16poligen Gehäuse vier Logikgatter.
- Der Platzbedarf eines in SMT montierten Gehäuses samt minimal benötigter Verdrahtung beträgt etwa $20 \cdot 10\,\mathrm{mm}^2$.

Somit würde z.B. im Readout-Controller alleine für die Steuerlogik ein Platzbedarf von

$$49\,000\ \text{Gatter} \cdot 0{,}25\ \frac{\text{IC}}{\text{Gatter}} \cdot 20 \cdot 10\ \frac{\text{mm}^2}{\text{IC}} = 2{,}45\ \text{m}^2 \tag{3.5}$$

entstehen, der aber auf einer VME-Buskarte nicht zur Verfügung steht.

Darüberhinaus wäre ein so aufgebautes System nicht flexibel genug; für jede Änderung in der Steuerlogik müßten Hardwareänderungen vorgenommen werden. Auch zeigen Überlegungen zum Stromverbrauch und der durch die vielen Verbindungen langen Signalwege, daß ein Aufbau des Systems aus Standardkomponenten nicht realisierbar ist.

Trotzdem wurden an einigen Stellen Standardkomponenten ins Auslesesystem integriert:

- Der VME-Bus-Chipsatz wurde als "off-the-shelf"[1]-Lösung gewählt. Dies hat den Vorteil, daß man das Spezialwissen der jeweiligen Firma nutzen und auf eine eigene, fehlerträchtige Umsetzung des teilweise komplexen VME-Bus-Protokolls verzichten kann.
- Die Speicher der LVL1- und LVL2-Pipe sowie die zum Mapping benötigten Speicherbausteine wurden ebenfalls aus gängigen Komponenten realisiert.
- An bestimmten Stellen im System wurden auf spezielle Aufgabenbereiche zugeschnittene Standardkomponenten der 74xx-Reihe verwendet, so zum Beispiel Treiberbausteine zum Aufbereiten von Datensignalen für die Flachbandkabel.

3.5.2 CPLDs

Wegen der mangelnden Flexibilität von Standard-Logikkomponenten wurden in der Industrie bereits vor einigen Jahren Anstrengungen unternommen, programmierbare Bausteine zu entwickeln, die es dem Anwender erlauben, eigene Logikfunktionen zu implementieren. Erste Ansätze dazu waren PALs und GALs, in denen bereits kleine Logikschaltungen untergebracht werden konnten.

Heute sind solche programmierbaren Bausteine, sogenannte CPLDs, mit über 10 000 Gattern erhältlich. Der Hauptvorteil dieser Bausteine ist, daß bedingt durch den inneren Aufbau nur sehr kurze und berechenbare Laufzeiten zwischen Ein- und Ausgängen des ICs entstehen. Allerdings bedingt der innere Aufbau von CPLDs auch, daß nur Schaltungen bestimmter Komplexität in einem Baustein untergebracht werden können. Diesen Nachteil hätte man auf Kosten größerer Laufzeiten durch eine höhere Anzahl an CPLDs ausgleichen können, jedoch waren zum Zeitpunkt der Überlegungen hauptsächlich Bausteine verfügbar, die zum Programmieren spezielle Programmiergeräte benötigen.

Dies jedoch steht im Gegensatz zu einer hohen Flexibilität: die Ausleseelektronik, die unter teilweise sehr beengten Bedingungen am Detektor untergebracht ist, kann unter Umständen nur unter hohem Zeitaufwand aus- und nach der Programmierung wieder eingebaut werden.

Aus diesen Gründen wurde von einem Einsatz von CPLDs in der RICH-Ausleseelektronik abgesehen.

3.5.3 Microcontroller

Der Einsatz der heute von vielen verschiedenen Firmen erhältlichen Microcontroller wurde ebenfalls in Betracht gezogen. Diese Systeme sind zu günstigen Kosten verfügbar, frei programmierbar und bieten eine hohe Anzahl an Ein-/Ausgabemöglichkeiten an.

Wie man bei der im Kapitel 5 folgenden Beschreibung des RICH Readout-Controllers sehen wird, müssen von der Steuerlogik viele Aufgaben parallel und unabhängig voneinander erledigt werden. Dies bedingt entweder den Einsatz mehrerer Microcontroller, was

[1] d.h. eine Lösung, die man fertig aus dem Regal des Herstellers zieht

sich in deutlich erhöhtem Platzbedarf der Komponenten niederschlägt, oder den Einsatz von Multitasking-fähigen Prozessoren. Die meisten heute erhältlichen Microcontroller sind jedoch nicht Multitasking-fähig oder müssen erst durch ein entsprechendes Betriebssystem diese Fähigkeit erhalten. Dies würde die Entwicklung eines gesamten Betriebssystems erfordern, was alleine den Rahmen einer Diplomarbeit deutlich übersteigen würde.

Ein weiteres Problem ist, daß die Steuerlogik z.B. auf dem Frontend-Modul wegen genau festgelegter zeitlicher Anforderungen mit minimaler Verzögerung auf Flanken von Steuersignalen reagieren muß. Betrachtet man einen hypothetischen Microcontroller mit einer Taktrate von 20 MHz (was einem heute erhältlichen typischen Microcontroller entspricht), so ergibt sich bei Abarbeitung eines Befehls pro Taktzyklus eine minimale Reaktionszeit von 50 ns, was jedoch beispielsweise für das T/H-Signal der Verstärker-Bausteine auf den Frontends zu langsam wäre: eine deutliche Verschlechterung des Meßergebnisses wäre die Folge.

Im RICH-Auslesesystem selbst wurde daher auf den Einsatz von Microcontrollern verzichtet. Jedoch finden diese Bausteine einen idealen Einsatzort in der Experiment-Steuerung und Überwachung (slow control), wo vielfältige Meß- und Regelaufgaben zu erledigen sind [29].

3.5.4 ASICs

Bereits in der frühen Phase der HADES-Planung wurde der Einsatz von ASICs sowohl im analogen Frontend als auch für den digitalen Teil des Datenaufnahme-Systems in Erwägung gezogen [11, DIGIplex]. Jedoch hat man mittlerweile von solchen Überlegungen Abstand genommen, da der Einsatz von ASICs (speziell mit gemischten Analog-/Digital-Systemen) deutliche Nachteile im Vergleich zur jetzt eingesetzten Lösung gehabt hätte:

- Verlust von Flexibilität: das Aufnahmesystem ist in Art, Umfang und Timing festgelegt;
- Hohe Entwicklungskosten: das im ASIC zu integrierende System muß vor der Serienfertigung ausreichend getestet werden;
- Hohe Stückkosten: die zu erwartenden Entwicklungskosten teilen sich auf die letztendlich benötige Stückzahl von ca. 500 Stück auf.

ASICs mit gemischten Analog-Digital-Systemen sind bei HADES nur in Form von TDCs im Einsatz, wo durch die weitaus größere Verbreitung der TDCs in anderen Experimenten die Entwicklungskosten pro Stück gering gehalten werden konnten. Die RICH Ausleseelektronik verwendet auf den Frontends zur Analogverarbeitung ASICs in Form der GASSIPLEX-Chips, die am CERN aus den AMPLEX-ASICs [16] entwickelt wurden. Auch beim Preshower-Detektor werden Analog-ASICs zur Datenerfassung benutzt.

3.5.5 FPGAs

Zeitgleich mit den ersten CPLDs wurde eine weitere Familie reprogrammierbarer Bausteine, die FPGAs, entwickelt, die ähnlich den CPLDs eine hohe Anzahl an Einzelgattern beinhalten, jedoch die Integration von Schaltungen höherer Komplexität erlauben. Erreicht wurde dies durch eine Änderung des inneren Aufbaus: das Innere eines FPGAs ist in eine Vielzahl von Logikblöcken aufgeteilt, die jeweils eine bestimmte Anzahl an logischen Verknüpfungen sowie Speicherelemente (Flipflops) enthalten. Zur Verbindung dieser von der Fa. Xilinx [30] CLB genannten Blöcke dient ein System von frei verschaltbaren Verbindungen. Die Verbindung zu den Anschlußpins erfolgt dabei über sogenannte IOBs, in denen ebenfalls Logikelemente untergebracht sind.

Als besonders nützlich hat sich die Möglichkeit erwiesen, FPGAs beliebig oft und ohne Ausbau aus der Schaltung neu konfigurieren zu können: auf diese Art und Weise konnten Korrekturen und neue Design-Ideen schnell verwirklicht und getestet werden. Die Programmierung der FPGAs erfolgte dabei über die VME-CPU.

Ebenfalls von Vorteil beim Übergang vom Prototyp zum Serienmodul des Readout-Controllers war die Möglichkeit, größere Bausteine (die über mehr interne Logikressourcen verfügen) anstelle der kleineren Modelle auf der Leiterplatte einzusetzen, ohne daß dabei Änderungen in der Leiterbahnführung notwendig waren. So konnten nach Abschluß der Fehlersuche (debugging) am Prototyp ohne Probleme zwei der bislang verwendeten FPGAs durch größere ersetzt werden, ohne daß die zu diesem Zeitpunkt bereits in Fertigung befindliche Leiterplatte geändert werden mußte (was zu größeren Kosten und Zeitverschub geführt hätte).

Allerdings ergeben sich durch den Einsatz von FPGAs auch einige Nachteile:

- Bedingt durch den inneren Aufbau muß beim Entwurf der Schaltungen verstärkt auf Laufzeiten zwischen einzelnen Funktionsblöcken geachtet werden und — wo immer möglich — auf den Einsatz von asynchronen Designs verzichtet werden. Dies allerdings führt generell zu deutlich stabileren Systemen und sollte auch beim Einsatz von CPLDs beachtet werden.

- Die verwendeten FPGAs der Firma Xilinx müssen nach jedem Einschalten neu programmiert werden. Dies führt zu einer Zeitspanne, in dem das Auslesesystem nicht verwendbar ist. Allerdings halten sich diese Zeiten in Grenzen, so daß hier der Vorteil, die Bausteine beliebig oft neu programmieren zu können, deutlich überwiegt.

- FPGAs der verwendeten Größenordnung sind im Vergleich zu CPLDs immer noch teurer. Durch die relativ kleine Stückzahl der zu bauenden Platinen fällt dieser Nachteil allerdings nicht allzu sehr ins Gewicht.

Die für die RICH-Ausleseelektronik getroffene Entscheidung, eine Mischung aus FPGAs und einigen Standardkomponenten zu verwenden, hat sich sowohl bei den Tests als auch ersten Einsätzen bei einer Teststrahlzeit als vorteilhaft erwiesen.

In den folgenden drei Kapiteln werden die einzelnen Module der RICH Ausleseelektronik vorgestellt sowie ihre Funktionsweise und das Zusammenspiel im Rahmen des Auslesesystems beschrieben. Dabei wird der Datenfluß “vom Detektor her” durch die Module erläutert.

Kapitel 4

Das RICH Frontend

4.1 Aufgaben

Das Frontend-Modul stellt das eigentliche Datenaufnahme-Modul im Auslesesystem des RICH dar. Die komplette Handhabung und Verarbeitung der Analogsignale, die aus den Photonendetektormodulen ausgelesen werden, sowie deren digitale Weiterverarbeitung in der LVL1-Pipe werden vom Frontendmodul übernommen. Im einzelnen müssen dabei folgende Aufgaben — zum Teil parallel und unabhängig voneinander — erledigt werden:

- analoge Signalverarbeitung: die Detektorsignale müssen verstärkt, differenziert und integriert (shaping), in eine T/H-Schaltung eingespeist und an den Eingangsbereich des ADC angepaßt werden.
- Analog/Digitalwandlung: die so vorbearbeiteten Signale müssen mit hoher Auf"—lösung und innerhalb einer festgelegten Zeit in Digitalwerte umgewandelt werden.
- Kanalzuordung: den so erhaltenen Digitalwerten müssen innerhalb eines Moduls eindeutige Kanalnummern zugeordnet werden.
- Schwellenvergleich: für jeden einzelnen Kanal muß ein Vergleich mit einer individuell einstellbaren Schwelle durchgeführt werden, um reale Signale vom Rauschen zu trennen und defekte Kanäle auszublenden.
- Mapping: die je nach Padplane andere Leiterbahnführung muß für jeden einzelnen Kanal getrennt durch Umordnen der Kanalnummern ausgeglichen werden.
- FIFO-Verwaltung: die Zugriffe auf die aus zwei FIFO-Bänken zusammengesetzte LVL1-Pipe müssen verwaltet werden.
- Auslesesteuerung: die Auslesezyklen zum Transportieren der Daten aus der LVL1-Pipe zur Bildverarbeitungseinheit und in die LVL2-Pipe (bzw. das Löschen der Daten) müssen vom Frontend-Modul gehandhabt werden.

- Busy-Handling: unterschiedliche Betriebs- und Ausnahmezustände im Frontend-Modul müssen mittels eines Busy-Signals dem Readout-Controller mitgeteilt werden.

Bei der Realisierung des Moduls sind jedoch einige Einschränkungen zu beachten: der verfügbare Platz, den ein einzelnes Frontend einnehmen kann, ist durch die Geometrie des Detektors festgelegt. Dies schränkt die Platinenfläche sowie die Anzahl und Größe der zu verwendenden Bauelemente stark ein; die Größe eines Frontend-Moduls ist auf $115 \cdot 64\text{mm}^2$ festgelegt.

Die analoge Signalverarbeitung stellt ebenfalls hohe Ansprüche an die Elektronik. Das unverstärkte Signal, das von den ladungsempfindlichen Vorverstärkern verarbeitet wird, entspricht Pulshöhen von ca. 10 mV. Um Verfälschungen durch Übersprechen von Signalen zu vermeiden, muß bis zum Einfrieren der Pulshöhe in den T/H-Stufen der GASSIPLEXe nach Möglichkeit jegliche Störquelle auf dem Frontend-Modul abgeschaltet sein. Da der Digitalteil des Frontends — bedingt durch die verwendeten Taktraten und schnellen Signalflanken — die größte Störquelle darstellt, muß dessen Aktivierung zeitlich genau auf die analoge Signalverarbeitung abgestimmt sein.

Das Frontend-Modul und seine Eigenschaften im analogen Teil (Auflösung, Linearität etc.) sind bereits mehrfach beschrieben worden [16], [17]. Im folgenden soll der Schwerpunkt auf den Digitalteil gelegt werden, der für den Einsatz am Experiment komplett neu gestaltet wurde. Diese Änderungen konnten dank der Flexibilität des FPGAs einfach und ohne Hardware-Modifikationen vorgenommen werden.

4.2 Blockschaltbild und Funktionsübersicht

Der prinzipielle Aufbau des Frontend-Moduls ist in Abbildung 4.1 dargestellt. Die einzelnen Funktionen sind dabei in Blöcken zusammengefaßt: der linke Block ist für die Analogverarbeitung zuständig. Je ein GASSIPLEX-Chip liefert über einen internen Multiplexer die in jeweils einer T/H-Stufe festgehaltenen Signalpegel von 16 Kanälen an einen Video-Multiplexer. Dieser schaltet eines der vier Eingangssignale auf einen einstellbaren Verstärker und führt das verstärkte Signal an den Eingang eines Analog-Digital-Wandlers. Dort endet der analoge Signalweg; alle weiteren Operationen werden mit den digitalisierten Daten durchgeführt.

Die beiden mittleren Blöcke beinhalten sämtliche Logikfunktionen zur Steuerung der Abläufe im Frontend. Dabei wurde eine strikte Trennung zwischen Datenaufnahme und Datenauslese vorgenommen; dies ermöglicht den asynchronen Ablauf beider Prozesse. Im oberen Funktionsblock enthalten sind Schwellenvergleicher, Mapping-Einheit und Triggertag-Speicher, die kontrolliert von einer Ablaufsteuerung vollautomatisch 64 Kanäle wandeln, den Vergleich mit den Schwellen und das Mapping durchführen und die so erhaltenen Daten zusammen mit dem Kontrollwort (dem Triggertag) in den LVL1-FIFOs ablegen. Die Steuerung dieser Ablaufsteuerung erfolgt nur mit Hilfe von LVL1-Triggerbefehlen.

Der untere Funktionsblock kontrolliert sämtliche Auslese- und Löschzyklen. Die Übertragung von Daten zur Ringerkennereinheit sowie in die LVL2-Pipe und die Löschung von Datensätzen werden hier durchgeführt.

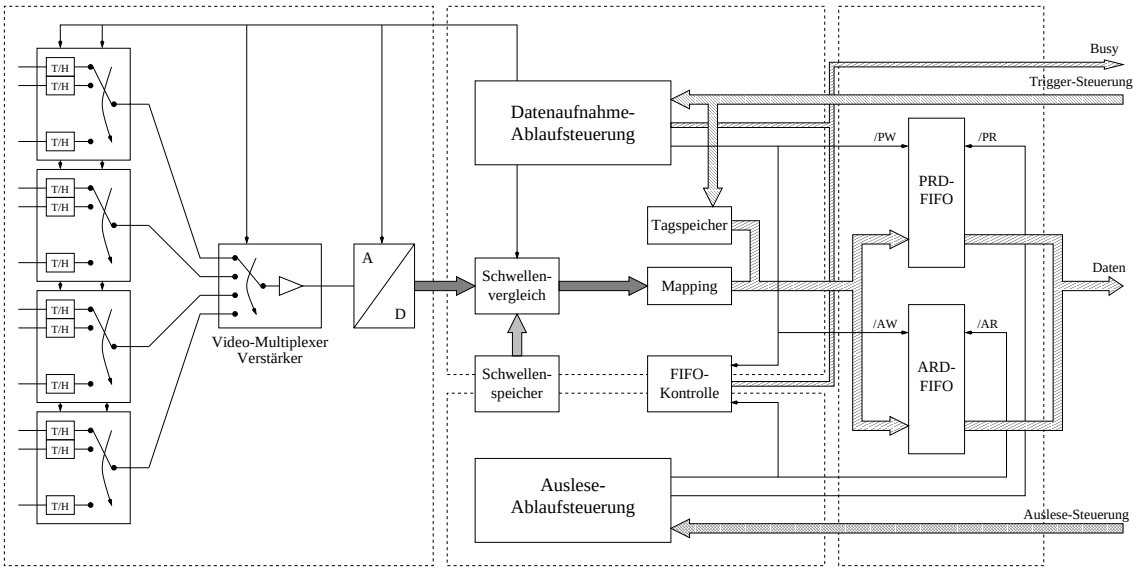


Abbildung 4.1: Blockschaltbild des Frontend-Moduls. Die strikte Trennung zwischen Datenaufnahme und Datenauslese ermöglicht voll asynchronen Betrieb des Moduls.

Eine Sonderstellung nehmen in der Steuerlogik der Schwellenspeicher und die FIFO-Kontroll-Logik ein: der Speicher für die einzelnen Kanalschwellen wird in einem speziellen Zyklus von der Auslese-Ablaufsteuerung beschrieben, steht aber im normalen Betrieb nur der Datenaufnahmesteuerung lesend zur Verfügung. Die FIFO-Kontrolle überwacht das korrekte Beschreiben und Auslesen der LVL1-Pipe; insbesondere werden Ausnahmesituationen wie ein fast voller FIFO rechtzeitig erkannt und durch Erzeugung eines Busy-Signals an den Readout-Controller weitergemeldet.

Der rechte Funktionsblock enthält die zwei FIFO-Bänke zur Implementierung der LVL1-Pipe sowie alle zur Anpassung an den Modulbus notwendigen Bauelemente wie Datenbustreiber und Pegelwandler. Die Steuerung dieser Bauelemente wird von beiden Ablaufsteuerungen übernommen. Der Anschluß des Moduls an den Frontend-Bus erfolgt über den Busstecker, dessen Pinbelegung in Tabelle 4.1 angegeben ist. Das Layout des Frontend-Moduls ist in Abbildung 4.2 zu sehen.

Im folgenden wird genauer auf die unterschiedlichen Betriebsmodi des Frontendmoduls eingegangen. Dabei werden sowohl die internen Abläufe im Modul selber wie auch die dafür notwendigen Steuer- und Statussignale eingeführt und erklärt.

4.3 Digitalisierung und Zwischenspeicherung

Die Datenaufnahme in den einzelnen Frontendmodulen wird über einen entsprechenden Befehl am LVL1-Triggerbus ausgelöst. In der DTU wird bei Eintreffen dieses Triggerkodes das Triggersignal `/TRG` erzeugt, das über den “Private Bus” genannten Teil des VME-Busses und den Readout-Controller an alle angeschlossenen Frontendmodule verteilt wird.

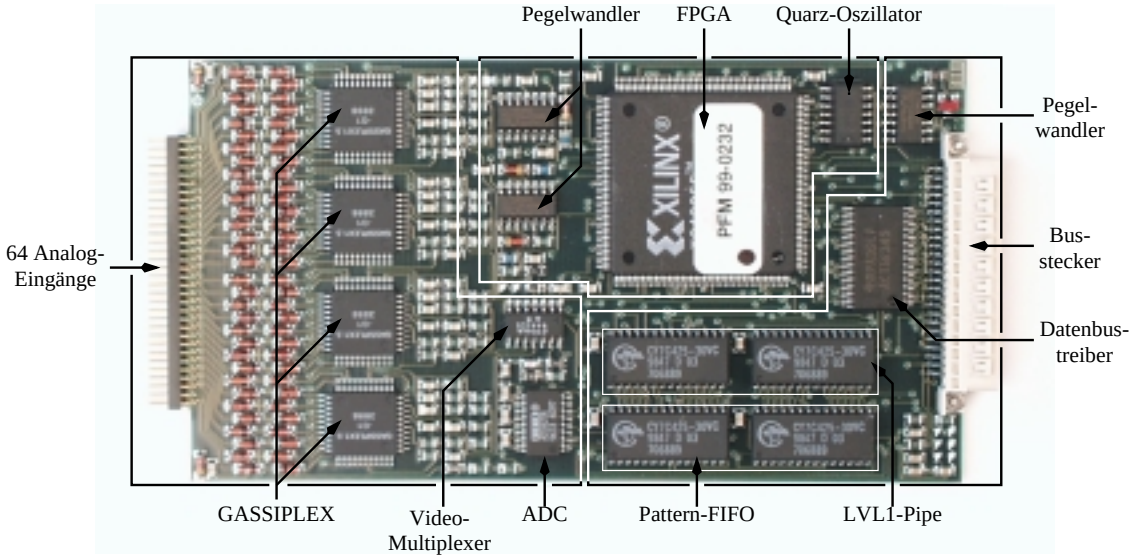


Abbildung 4.2: Layout des RICH Frontend-Moduls. Die Aufteilung in Funktionsblöcke entspricht der in Abbildung 4.1, lediglich die beiden Ablaufsteuerungen sind hier in einem Block zusammengefaßt.

Zum Zeitpunkt des Eintreffens von `/TRG` am Frontend ist der Digitalteil fast vollständig abgeschaltet, d.h. auch die später zur Datenaufnahme benötigte Taktquelle ist außer Betrieb. Das Frontend reagiert in diesem Betriebsmodus nur auf einlaufende Triggersignale und auf Auslese- und Löschzyklen. Dies reduziert den Anteil an elektronischem Rauschen und Übersprechen von Signalen auf das Analogsignal erheblich und ermöglicht erst die Aufnahme von korrekten Daten. Der Ablauf eines kompletten Datenaufnahmezyklus ist in Abbildung 4.3 zu sehen.

Die negative Flanke des `/TRG`-Signals setzt das T/H-Signal, das alle vier GASSIPLEX-Chips in den Hold-Modus schaltet. Damit ist das analoge Signal eingefroren; nun erst wird der für die weitere Ablaufsteuerung notwendige externe Quarz-Oszillator über die Steuerleitung `SYSCLKON` eingeschaltet. Davon ausgehende Störungen können das bereits gespeicherte und verstärkte Analogsignal nicht mehr verfälschen, jedoch muß wegen der nun stark erhöhten Störabstrahlung (die sich bei Digitallogik mit schnellen Signalflanken prinzipiell nicht vermeiden läßt) bei der weiteren Bearbeitung der Analogdaten durch geschicktes Timing die störenden Einflüsse möglichst klein gehalten werden. Zeitgleich mit dem T/H-Signal wird ein Busy-Signal (`BUSY`) erzeugt, das während der gesamten Wandlung gesetzt bleibt. Die Busy-Signale aller Frontends eines Readout-Controllers werden über ein Oder-Gatter logisch verknüpft und über die DTU an die CTU weitergeleitet. Somit bestimmt das “langsamste” Frontend die Länge des Busy-Signals und verhindert damit effektiv das Einlaufen weiterer Trigger während der Totzeit.

Alle zur jetzt ablaufenden Analog-Digitalwandlung notwendigen Signale werden durch die Aufnahme-Ablaufsteuerung erzeugt. Getaktet wird dieser Schaltungsteil mit 33,33 MHz über die Leitung `SYSCLK`. Bei dieser Taktfrequenz kann die geforderte Maximaltotzeit von 10 μs eingehalten werden, eine Verkürzung der Totzeit durch Erhöhung des Taktes

Signalname	Richtung	Pins		Richtung	Signalname
/SBUSY	out	B1	A1	in	/TRG
/TAGCLK	in	B2	A2	in	TAGDTA
FC1	in	B3	A3	in	FC0
FC3	in	B4	A4	in	FC2
/STRBO	out	B5	A5	in	/STRBI
n.c.	-	B6	A6	out	/SMACK
GND	-	B7	A7	-	GND
SD1	bi	B8	A8	bi	SD0
SD3	bi	B9	A9	bi	SD2
SD5	bi	B10	A10	bi	SD4
SD7	bi	B11	A11	bi	SD6
SD9	bi	B12	A12	bi	SD8
SD11	bi	B13	A13	bi	SD10
SD13	bi	B14	A14	bi	SD12
SD15	bi	B15	A15	bi	SD14
+5V	-	B16	A16	-	+5V
+3.3V	-	B17	A17	-	+3.3V
GND	-	B18	A18	-	GND
GND	-	B19	A19	-	GND
-3.3V	-	B20	A20	-	-3.3V
-5V	-	B21	A21	-	-5V
GND	-	B22	A22	in	SPULSER
XDIN	in	B23	A23	out	*XDOUT*
XDONE	out	B24	A24	in	*XCCLK*
/XINIT	bi	B25	A25	in	*/XPRG*

Tabelle 4.1: Pinbelegung des Frontend-Bussteckers. Die Signalrichtungen sind aus Sicht des Moduls angegeben, die Konfigurationssignale für die FPGAs in kursiver Schrift gesetzt.

ist möglich, erfordert aber den Neuentwurf des im FPGA implementierten synchronen Designs.

Die GASSIPLEX-Chips werden durch die Signale T/H_TTL, CLK_TTL und RES_TTL angesteuert:

- T/H_TTL steuert die T/H-Stufen an.
- CLK_TTL bedient die GASSIPLEX-internen Multiplexer. Mit jedem Puls schaltet der GASSIPLEX einen Kanal weiter, nach 16 Pulsen steht am Ausgang wieder Kanal #0 an.
- RES_TTL setzt die interne Logik (Multiplex-Zähler etc.) des GASSIPLEX zurück.

Der Videomultiplexer vom Typ EL4441CS [8] wird über die Signale MUXA1 und MUXA0

gesteuert: das binär kodierte Signal wählt den entsprechenden Eingang des Multiplexers aus.

Zur Analog-Digital-Wandlung wird ein ADC vom Typ ADS820 [5] eingesetzt. Dieser ursprünglich für Video-Anwendungen entwickelte Chip wird nur über das Signal `ADCCLK` angesteuert. Der ADC folgt dem Eingangssignal, solange `ADCCLK` high ist; bei der negativen Flanke wird das Signal in einen analogen Hold-Speicher übernommen und, während `ADCCLK` low ist, in ein 10 bit Digitalwort gewandelt. Aufgrund der internen Pipeline-Struktur erscheint das Ergebnis der Wandlung aber erst sieben Taktzyklen später an den Ausgängen; dies muß bei der Ansteuerung berücksichtigt werden.

Die 16 Kanalinhalte jedes einzelnen der 4 GASSIPLEXe werden durch ein Multiplex-Verfahren sequentiell an den Eingang des Videomultiplexers geführt. Der genaue Ablauf dieses Verfahrens ist in Abbildung 4.4 zu sehen: Über die Signale `MUXA1` und `MUXA0` wird der GASSIPLEX-Chip ausgewählt, mit `CLK_TTL` wird der Kanal des einzelnen GASSIPLEX an seinen Ausgang geschaltet. Dies hat den Vorteil, daß man die maximal mögliche Wandelfrequenz des ADC von 20 MHz nutzen kann, ohne die GASSIPLEXe zu übertakten. Diese Art des Multiplexing ermöglicht erst das Einhalten der Totzeit von 10 μs.

Die so erhaltenen digitalisierten Pulshöhen werden in den Xilinx FPGA geführt. Dort findet der Schwellenvergleich zeitgleich mit dem Mapping statt: der Schwellenspeicher ist als 16 bit breites SRAM ausgelegt; die unteren 10 bit werden als Schwellenwert zum Vergleich herangezogen, die oberen 6 bit ergeben die zuzuordnende Kanalnummer. Die Daten werden zusammen mit der Kanalnummer aus dem FPGA an die Dateneingänge beider FIFO-Bänke geführt; bei einem positiven Vergleich (d.h. der Analogwert ist höher als die Schwelle) wird ebenfalls ein Schreibsignal (`/PWR` und `/AWR`) erzeugt. Da zum Vergleich mit den einlaufenden Analogdaten ein echter Größenvergleich durchgeführt wird, ist es durch Setzen einer Kanalschwelle von 1023 möglich, den betreffenden Kanal komplett abzuschalten. Somit können defekte oder nicht benötigte Kanäle einfach und ohne Modifikation an der Hardware abgeschaltet werden.
Das Frontend kann zur besseren Abstimmung auf die IPU auch um einen eigenen Schwellebvergleicher für die Pattern-Daten erweitert werden; ob dies notwendig ist, kann allerdings erst nach intensiven Tests während einer Strahlzeit entschieden werden.

Während der im Frontend laufenden Wandlung wird von der DTU das mit dem vorhergegangenen LVL1-Befehl[1] eingetroffene Triggertag auf seriellem Weg über die Leitungen `/TAGCLK` und `TAGDTA` an die Frontends übertragen und dort in einem Register zwischengespeichert. Nach Wandlung aller 64 Frontendkanäle wird dieses Triggertag als letztes Wort in beide FIFO-Bänke geschrieben. Um die zu einem Subevent gehörenden Daten vom nächsten Datensatz abzugrenzen, wird das Triggertag auf besondere Art und Weise markiert: Jede FIFO-Bank ist aus zwei parallelgeschalteten FIFOs mit jeweils 9 bit Breite aufgebaut. Es ergeben sich also zwei FIFO-Bänke mit jeweils 18 bit Breite. Bei normalen Datenworten werden Bit 17 und 16 jeweils auf Null gesetzt, bei einem Triggertag hingegen wird Bit 17 (das ENDBIT) auf high gesetzt. Die Auslesesteuerung nutzt diese Zusatzinformation, um das Ende eines Datensatzes zu ermitteln.

[1] nähere Informationen zur RICH DTU siehe Kapitel 6

Nach Abschluß der Wandlung werden die GASSIPLEXe über einen Puls auf RES_TTL zurückgesetzt, das BUSY-Signal zurückgenommen und der Taktgenerator über SYSCLKON ausgeschaltet. Das Frontend ist nun wieder für einen neuen Datenaufnahme-Zyklus bereit.

4.4 Auslese der Frontend-Module

4.4.1 Der Frontend-Bus

Um die in der LVL1-Pipe des Frontends gespeicherten Daten auszulesen, werden von der RICH DTU sowohl LVL1- als auch LVL2-Triggerbefehle ausgewertet und in entsprechende Auslesezyklen am Readout-Controller umgesetzt. Bedingt durch die Verkettung mehrerer Frontend-Module zu einer Gruppe wurde die Einführung eines Bussystems notwendig, dessen Grundfunktionen im folgenden beschrieben werden.

Das Bussystem der Frontends orientiert sich an gängigen Bussystemen, ist jedoch in einigen Punkten an die speziellen Anforderungen der Auslese angepaßt: so fehlen Adreßleitungen zur Adressierung einzelner Module. Um das System einfach skalierbar zu halten, wurde statt einer Einzelmodul-Adressierung eine Verschaltung über daisy chaining gewählt.

Prinzipiell kann der Bus in zwei verschiedenen Arten betrieben werden:

daisy chain mode: jeweils ein Modul reagiert auf die Steuersignale. Nach Beendigung der notwendigen Aktionen in diesem Frontend werden die Steuersignale an das nächste Modul in der Kette weitergereicht.

bus mode: alle Module reagieren gleichzeitig auf die Steuersignale. Das langsamste Modul in der Kette bestimmt dabei die Dauer des jeweiligen Zyklus. Dies ermöglicht schnelle parallele Aktionen in der Modulkette.

Das Blockschaltbild des Frontendbus ist in Abbildung 4.5 wiedergegeben, die einzelnen Signale sind in Tabelle 4.2 aufgeführt.

Signalname	Signalart	Bedeutung	Richtung
FC[3:0]	Steuersignal	Function Code: zeigt aktuellen Buszyklus an	in
/STRBI	Steuersignal	Strobe-Signal: zeigt gültigen Function Code an	in
/STRBO	Steuersignal	schaltet /STRBI zum nächsten Modul weiter	out
/SMACK	Kontrollsignal	Acknowledge-Signal: bestätigt Zyklusende	out
SD[15:0]	Datensignal	bidirektionaler Datenbus	bi

Tabelle 4.2: Signale des Frontend-Busses. Die Richtungsangabe ist aus Sicht des Frontends angegeben, die Signalpegel sind TTL-kompatibel.

Die Art des Buszyklus wird den Modulen einer Kette über die Steuersignale FC[3:0] mitgeteilt. Der Funktionskode bestimmt die Reaktionsweise der Module. Der Ablauf des Zyklus

selbst wird über das Kontrollsignal `/SMACK` gesteuert, dessen Funktionsweise je nach Busmodus unterschiedlich ist. Die momentan definierten Funktionskodes sind in Tabelle 4.3 aufgelistet, eine Erweiterung ist in Zukunft aber denkbar.

FC	Zyklus	Kurzname	Busmodus
0	Pattern Readout	PRDOUT	bus
1	reserved for future use	—	—
2	Analog Readout	ARDOUT	bus
3	Analog Delete	ADELETE	block
4	Reset FIFOs	RFIFO	block
5	Reset Daisy Chain	RDAISY	block
6	Write Configuration	WCFG	bus
7	Read Configuration	RCFG	bus
8 ⋮ 14	reserved for future use	—	—
15	No Operation	NOP	—

Tabelle 4.3: Derzeit definierte Funktionskodes für das RICH Frontend.

Für die Auslese selbst sind dabei nur die Zyklen PRDOUT, ARDOUT, ADELETE und RDAISY relevant. Die Zyklen RFIFO, WCFG und RCFG dienen der Konfiguration der Frontend-Module und werden in Abschnitt 4.5 erklärt. Im PRDOUT-Zyklus werden die Daten zum Ringerkenner, im ARDOUT-Zyklus in die LVL2 transportiert. Der ADELETE-Zyklus dient zur schnellen Löschung nicht benötigter Datensätze in der LVL1-Pipe.

Realisiert wird der Frontend-Bus in Form von Platinen, die auf die Geometrie des Detektors abgestimmt sind. Diese sogenannten backplanes erfüllen trotz ihres einfachen Aufbaus einige wichtige Aufgaben:

- Spannungsversorgung der Frontends
- Signalaufbereitung
- mechanische Stabilisierung

Die Frontend-Module stellen einige Anforderungen an die Qualität der Versorgungsspannungen. So muß neben einer guten Masseanbindung gewährleistet sein, daß die benötigten Spannungen sauber und ohne große Toleranzen an den Modulen anliegen: bei zu niedriger Spannung verlieren die FPGAs ihre Programmierung, bei zu hoher Spannung kommt es unweigerlich zur Zerstörung der empfindlichen Bauteile. Auf den Backplanes wurden daher Sicherungsmaßnahmen in der Spannungsversorgung getroffen: Zenerdioden lösen bei Überspannung Schmelzsicherungen aus und verhindern somit Schäden an den Modulen.

Die Verbindung der Backplane zum Readout-Controller erfolgt über ein Flachbandkabel, was sich auf den Platzbedarf der verwendeten Steckverbinder positiv auswirkt, aber auch eine Signalaufbereitung und -anpassung erforderlich macht:

- Für die vom Readout-Controller gelieferten Signale wurden Abschlußwiderstände zur Minimierung von Signalreflexionen auf dem Kabel vorgesehen.

- Um Probleme wegen der geringen Ausgangstreiberleistung der Xilinx FPGAs zu vermeiden, wird das `/STRBO`-Signal des letzten Frontends aktiv getrieben. Dies führt wegen des verwendeten Treibers zur Invertierung des `/STRBO`-Signals (siehe auch Abschnitt 5.4.1).

- Die `/SMACK`-Leitungen erhalten auf den backplanes jeweils einen Pullup-Widerstand (durch dessen niedrige Dimensionierung von 390 Ω die Signalflanken erheblich verbessert werden konnten). Die Zusammenfassung erfolgt durch ein NAND-Gatter mit aktivem Ausgangstreiber (was die Signalqualität nach dem Durchgang durch das Flachbandkabel erheblich verbessert). Wie auch beim `/STRBO`-Signal muß die so stattfindende Invertierung im Readout-Controller berücksichtigt werden.

Bedingt durch die Geometrie der Photonendetektorkathode mußten für den RICH 16 jeweils unterschiedlich geformte Backplanes angefertigt werden (siehe Abbildung 4.6). Dabei stabilisiert die teilweise rechtwinklige Anordnung der Frontend-Module den mechanischen Aufbau erheblich; durch die Verbindung der Backplanes untereinander wird insgesamt die direkt am Detektormodul angebrachte Ausleseelektronik stabilisiert.

4.4.2 Zyklen zur Muster- und Analogdatenauslese

Im folgenden werden PRDOUT- und ARDOUT-Zyklus gemeinsam beschrieben; die Zyklen selbst laufen im Frontend identisch ab und unterscheiden sich nur im Ziel der Daten und der ausgelesenen FIFO-Bank. Der Ablauf eines Auslesezyklus ist in Abbildung 4.7 dargestellt. Die folgende Beschreibung bezieht sich auf ein einzelnes Modul in der Kette; die Richtung der Signale wird dabei aus Sicht des Frontends angegeben. Der Ablauf in den folgenden Modulen erfolgt analog dazu; auf Besonderheiten, die nur das letzte Modul in der Kette betreffen, wird gesondert eingegangen.

Der Auslesezyklus startet mit Anlegen des Funktionskodes RDAISY. Durch Anlegen eines 64 ns langen `/STRBI`-Impulses werden die daisy chaining-Kontrollen der Module in einen definierten Zustand gebracht: nur das erste Frontend in der Kette reagiert auf die Steuersignale, die folgenden Module verhalten sich passiv.

Zur Auswahl der auszulesenden Bank wird nun statisch der Funktionskode PRDOUT oder ARDOUT angelegt. Das Frontend generiert aus jedem `/STRBI`-Puls ein Lesesignal für die FIFO-Bank und legt über den Bustreiber die Daten auf die Leitungen `SD[15:0]`. Dabei wird der Wert des obersten Datenbits (ENDBIT) der FIFO-Bank überprüft, um das Ende eines Datensatzes festzustellen. Bei gesetztem Endbit zieht das Frontend die Leitung `/SMACK` während des `/STRBI`-Pulses auf low und meldet so das Ende des Datensatzes an die Auslesesteuerung. Alle nun folgenden `/STRBI`-Pulse werden ohne weitere Aktionen des Moduls auf den `/STRBO`-Ausgang — und damit an den `/STRBI`-Eingang des nächsten Moduls — weitergeleitet.

Der Auslesezyklus wiederholt den oben beschriebenen Vorgang für jedes weitere Modul in der Kette. Das letzte Modul verhält sich wie alle anderen vorher, nur wird hier der /STRBO-Ausgang an den Readout-Controller zurückgeführt. Anhand dieses Signals erkennt der Readout-Controller, daß die Auslese der Kette beendet ist. Als Reaktion hierauf wird der Funktionskode NOP angelegt und der Auslesezyklus beendet.

4.4.3 Der Löschzyklus

Beim ADELETE-Zyklus wird der Bus im Bus-Modus benutzt: das Löschen der Analogdaten erfolgt parallel (was zu einer deutlichen Zeitersparnis führt), das Modul mit den meisten Daten (das am längsten zu derem Löschen braucht) bestimmt dabei die Länge des Zyklus.

Der Löschzyklus wird durch statisches Anlegen des Funktionskodes ADELETE gestartet (siehe Abbildung 4.8). Anschließend wird /STRBI auf low gelegt; die Auslese-Ablaufsteuerung in den einzelnen Modulen setzt /SMACK auf low und beginnt selbständig, Lesepulse für die FIFO-Bank mit den Analogdaten zu erzeugen. Der Bustreiber wird nicht angeschaltet, die ausgelesenen Daten also effektiv gelöscht. Während dieses Vorgangs wird ebenfalls die oberste Datenleitung der FIFO-Bank überwacht. Zeigt ein high während eines Lesepulses das Ende des zu löschenden Datensatzes an, so unterbricht die Auslese-Ablaufsteuerung die Generierung der Lesepulse und setzt /SMACK auf high zurück.

Da bei diesem Vorgang nicht auf Lauf- und Verarbeitungszeiten der Daten im Readout-Controller geachtet werden muß, können die Lesepulse mit der für die FIFOs maximal erlaubten Geschwindigkeit erzeugt werden. Dies führt zu einem schnellen Ablauf des Löschzyklus.

Der Readout-Controller wartet, bis sämtliche /SMACK-Signale der Kette wieder auf high gehen und beendet dann den Löschzyklus durch Umschalten auf den Funktionskode NOP.

4.5 Konfiguration des Frontend-Moduls

Für die korrekte Inbetriebnahme der Frontend-Module ist es notwendig, den Schwellenspeicher für die Kanäle vor der Aufnahme von Meßdaten mit Schwellen zu beschreiben. Zu diesem Zweck wurden zwei Konfigurationszyklen (WCFG und RCFG) eingeführt. Im WCFG-Zyklus wird der Schwellenspeicher mit Werten beschrieben, im RCFG-Modus zu Kontrollzwecken ausgelesen. Der Ablauf beider Zyklen ist in Abbildung 4.9 dargestellt.

4.5.1 Schreiben der Schwellen

Der WCFG-Zyklus verläuft ähnlich wie der PRDOUT-Zyklus. Gestartet wird durch Anlegen des Funktionskodes RFIFO. Durch den folgenden Puls auf /STRBI werden alle Frontends der Kette komplett zurückgesetzt, d.h. die interne Logik des FPGAs wird in den Anfangszustand gebracht, der Adreßzähler für den Schwellenspeicher auf Null gesetzt und

die FIFOs durch einen Masterreset auf der /MR-Leitung gelöscht. Der Schwellenspeicher bleibt unbeeinflußt.

Anschließend wird der Funktionskode WCFG angelegt. Vor jedem jetzt folgenden /STRBI-Puls müssen die zu schreibenden Werte auf SD[15:0] anliegen; beim /STRBI-Puls werden die Werte vom FPGA übernommen, in den Schwellenspeicher geschrieben und anschließend der Adreßzähler um eins erhöht. Pro Frontend müssen also stets 64 Werte übermittelt werden, da keine direkte Adressierung des Schwellenspeichers möglich ist. Nach dem 64. /STRBI-Puls schaltet die Auslese-Ablaufsteuerung alle weiteren /STRBI-Pulse auf /STRBO durch und ermöglicht so die Konfiguration des nächsten Moduls in der Kette.

Die Daten enthalten — wie in Abschnitt 4.3 beschrieben — nicht nur die Schwellenwerte, sondern auch die Mapping-Informationen für die einzelnen Kanäle. Durch die festgelegte Reihenfolge bei der Datenaufnahme und dem Beschreiben des Schwellenspeichers ergibt sich also folgende Zuordnung (siehe Tabelle 4.4).

Hardware		Schwellenspeicher	
GASSIPLEX	Kanal	SD[15:10]	SD[9:0]
0	0	0	Schwelle 0
1	0	16	Schwelle 16
2	0	32	Schwelle 32
3	0	48	Schwelle 48
0	1	1	Schwelle 1
1	1	17	Schwelle 17
⋮	⋮	⋮	⋮
2	15	47	Schwelle 47
3	15	63	Schwelle 63

Tabelle 4.4: Organisation des Schwellenspeichers. Die Reihenfolge, in der die GASSIPLEX-Kanäle digitalisiert werden, liegt fest, jedem Kanal kann aber eine eigene Nummer und Schwelle zugeteilt werden.

4.5.2 Lesen der Schwellen

Die Auslese der Schwellenwerte geschieht im RCFG-Zyklus analog zu den PRDOUT- bzw. ARDOUT-Zyklen mit folgenden Unterschieden:

- Statt des Funktionskodes RDAISY wird RFIFO angelegt.
- Statt des Funktionskodes PRDOUT bzw. ARDOUT wird RCFG angelegt.
- Jedes Frontend in der Kette liefert einen festen Satz von 64 Datenwerten.

Am Ablauf des Zyklus selbst ändert sich nichts, deshalb wird hier nur auf die Beschreibung in Abschnitt 4.4.2 verwiesen.

4.6 Busy-Erzeugung

Neben den bereits erwähnten Kontroll- und Steuersignalen wird auch ein Busy-Signal (`/SBUSY`) vom Frontend generiert. Dieses Signal wird nur während eines gerade laufenden Datenaufnahmezyklus generiert und im Normalfall mit dem Beenden des Zyklus wieder zurückgenommen.

Eine Ausnahme ergibt sich aber bei fast vollem FIFO: bedingt durch das Auslesekonzept mit dem mehrstufigen Triggersystem werden die Daten im Pattern-FIFO stets sofort ausgelesen, die im Analog-FIFO stehenden Daten jedoch mit einer Verzögerung von etwa 200 μs, d. h. der Analog-FIFO wird im allgemeinen voller sein als der Pattern-FIFO. Die FIFO-Kontrolle überwacht daher kontinuierlich den Füllstand des Analog-FIFOs und setzt beim Erreichen einer bestimmten Schwelle mittels des FIFOFULL-Signals ebenfalls `/SBUSY`. Die Schwelle ist so gewählt, daß noch ein kompletter Datensatz (also 65 Datenworte) in die FIFOs paßt.

Das von allen Frontends zusammengefaßte `/SBUSY`-Signal wird als LVL1BSY an die CTU weitergeleitet und dient dort zum Ausmaskieren der einlaufenden Hardware-Trigger. So wird zuverlässig ein Überlauf der LVL1-Pipe verhindert; erst nach Ausführung der zu diesem Zeitpunkt noch in der PMU bearbeiteten LVL2-Trigger kann das Auslesesystem wieder neue Daten in die LVL1-Pipe übernehmen. Hierbei bestimmt das langsamste Glied in der Kette (also das Frontend mit dem größten Datensatz) die Verarbeitungsgeschwindigkeit des Gesamtsystems.

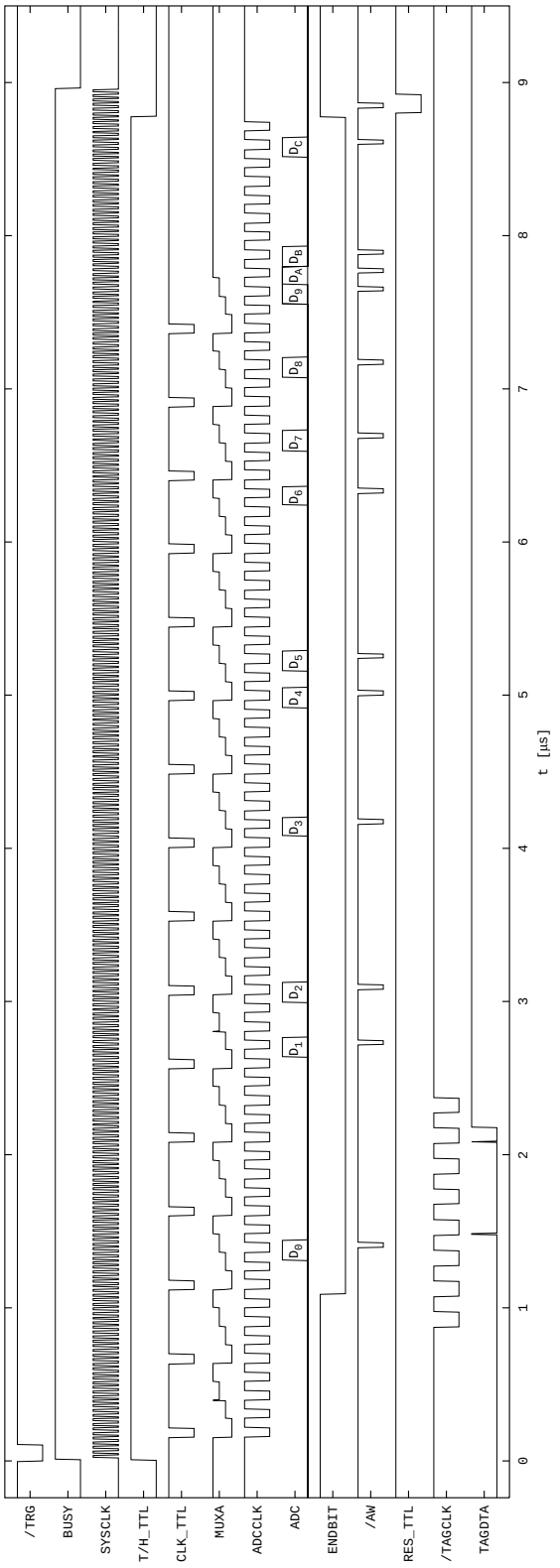


Abbildung 4.3: Datenaufnahmezyklus im Frontend-Modul. Die Steuersignale des Multiplexers sind als `MUXA` in dezimaler Form aufgetragen und zeigen den jeweils gewählten GASSIPLEX an.

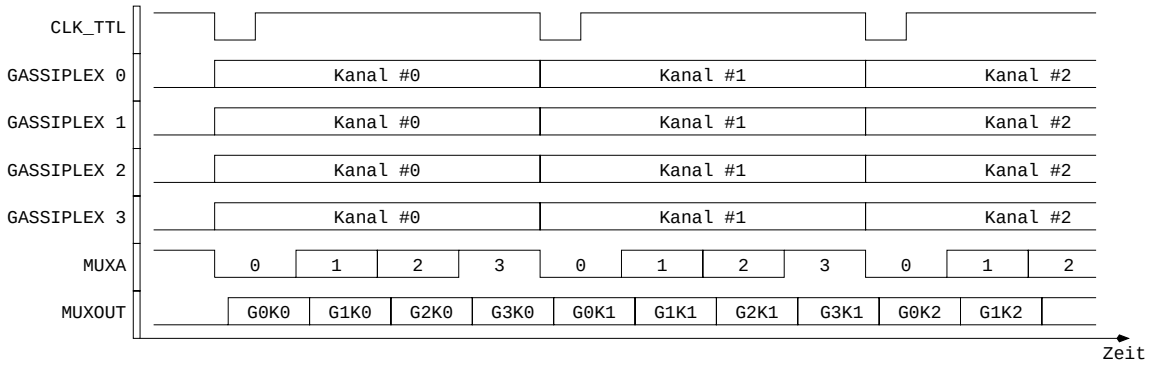


Abbildung 4.4: Multiplex-Verfahren des Frontends. Die Steuersignale sind als `MUXA` zusammengefaßt, `MUXOUT` zeigt das an den ADC geführte Ausgangssignal des Multiplexers. Die leichte zeitliche Verschiebung des Ausgangssignals entsteht durch die Durchlaufverzögerung des Multiplexers. Die Bezeichnung `GxKy` gibt GASSIPLEX- und Kanalnummer an.

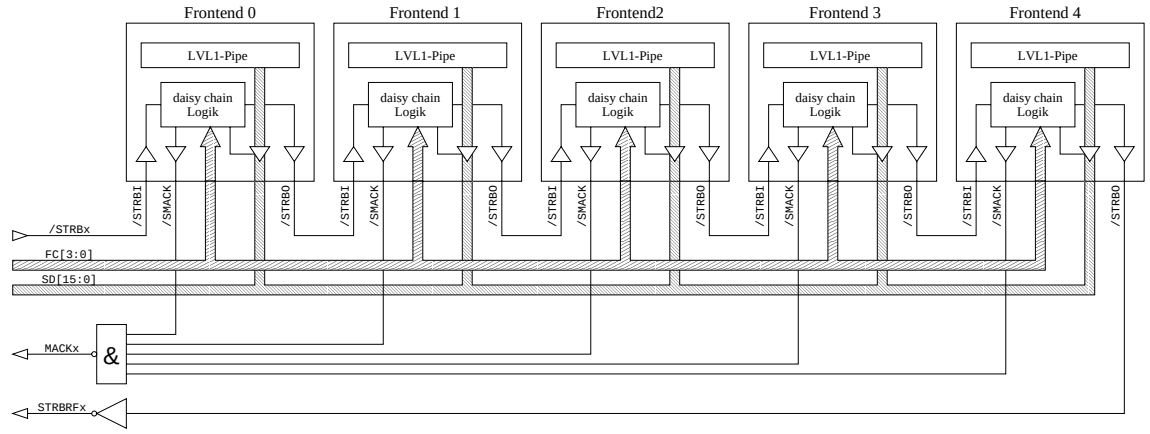


Abbildung 4.5: Blockschaltbild des Frontend-Busses. Die Verbindung zum Port des Readout-Controllers ist im linken Teil des Bildes zu sehen.

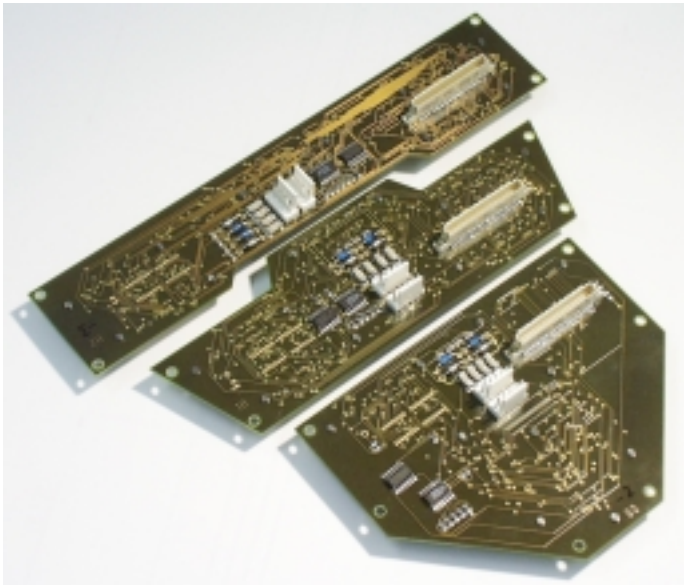


Abbildung 4.6: Layout einiger Frontend-Busplatinen ("backplanes"). Die Verbindung zum Readout-Controller erfolgt über die Stecker im rechten Teil der Platinen, die Frontend-Module werden auf der Rückseite der Platinen aufgesteckt. Die Signalaufbereitung erfolgt in den beiden ICs, die unbestückten Positionen sind für Abschlußwiderstände vorgesehen. Die Spannungsversorgung erfolgt über die weißen Steckverbinder in Platinenmitte.

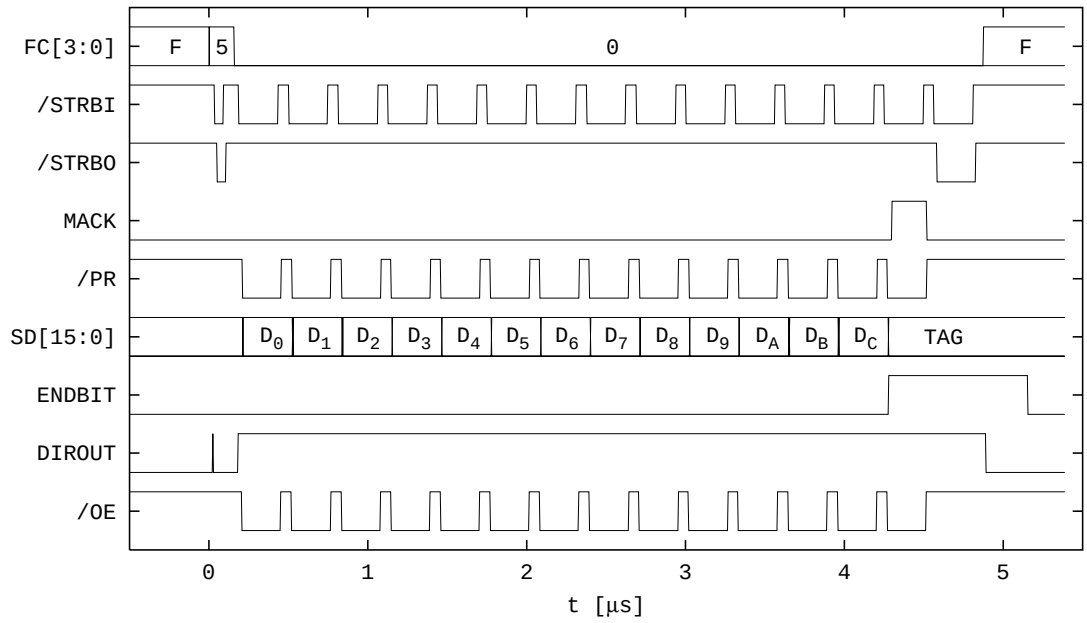


Abbildung 4.7: PRDOUT-Zyklus am Frontend (gilt analog für den ARDOUT-Zyklus). Es werden 13 Datenworte aus einem Modul ausgelesen und der Zyklus beendet. `DIROUT` und `/OE` sind die Steuersignale des Frontend-Bustreibers.

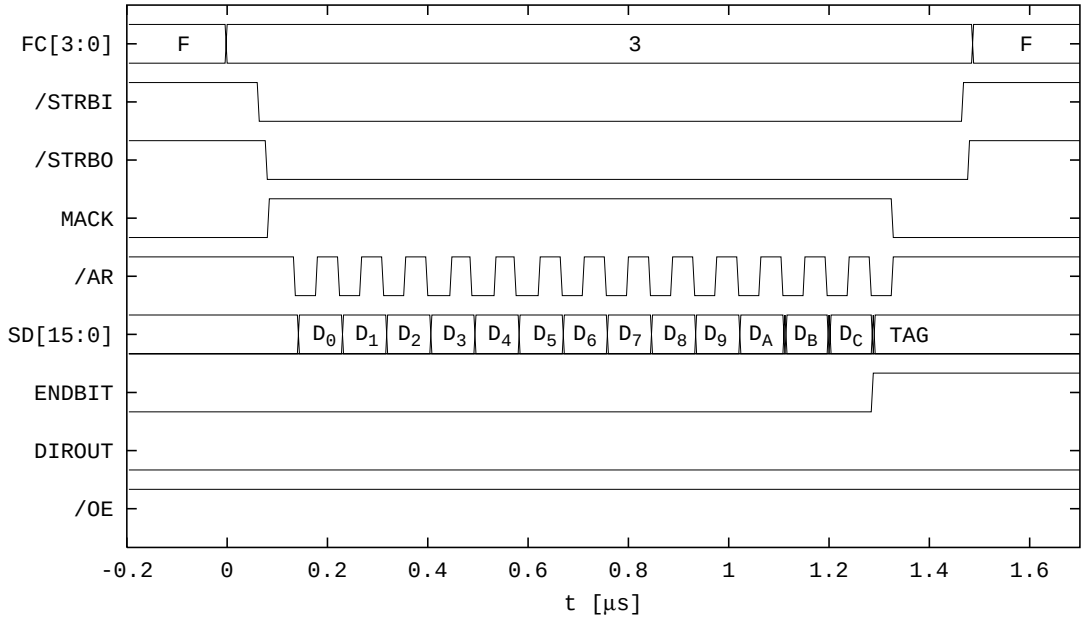


Abbildung 4.8: ADELETE-Zyklus am Frontend. Es wird ein gleich großer Datensatz wie in Abbildung 4.7 gelöscht, aber nur knapp ein Drittel der Auslesezeit benötigt. Das Kontrollsignal `/SMACK` ist hier als `MACK` vor dem invertierenden Pegelwandler aufgeführt.

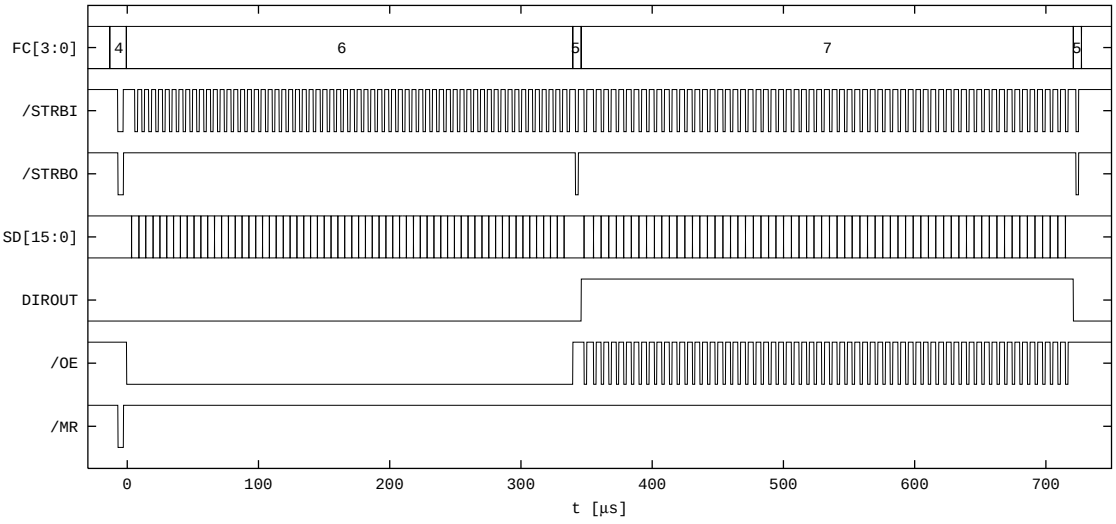


Abbildung 4.9: WCFG- bzw. RCFG-Zyklus am Frontend. Die Konfigurationssoftware beschreibt erst den Schwellenspeicher und liest zur Kontrolle die Werte zurück.

Kapitel 5

Der RICH Readout-Controller

5.1 Aufgaben

Die zentrale Komponente im RICH-Auslesesystem stellt der Readout-Controller dar. Wie in Abschnitt 3.1 erläutert, muß der Readout-Controller eine Vielzahl von Funktionen bieten, die ohne gegenseitige Beeinflussung störungsfrei und mit hoher Zuverlässigkeit ablaufen. Aus diesen Anforderungen und seiner zentralen Lage im Auslesesystem ergeben sich somit folgende Aufgaben:

- **Schnittstelle zur CPU:** Der Readout-Controller bietet als einzige Komponente des Auslesesystems die Möglichkeit des direkten CPU-Zugriffs. Somit müssen sämtliche Konfigurationsvorgänge (wie das Laden und Konfigurieren der FPGAs auf Readout-Controller und Frontend-Modul) sowie die Auslese der LVL2-Pipe über den Readout-Controller abgewickelt werden. Ebenso müssen leistungsfähige Diagnosemöglichkeiten vorhanden sein, die — bedingt durch den Einbau der Elektronik am Strahlrohr und die fehlende direkte Zugriffsmöglichkeit durch den Benutzer — es erlauben, nur durch Softwarezugriffe ferngesteuert Fehlerursachen bei Fehlverhalten festzustellen und das Auslesesystem wieder in einen benutzbaren Zustand zu bringen.

- **Verwaltung der LVL2-Pipe:** Ein Readout-Controller muß die Datensätze von bis zu 40 Frontend-Modulen vollautomatisch und in kurzer Zeit zusammenfassen, auf ihre Gültigkeit kontrollieren und anschließend mit dem korrekten Subevent-Header in der LVL2-Pipe ablegen. Bei diesem Vorgang sind die Daten durch Mapping in das für die Analysesoftware benötigte Format zu bringen. Ebenso müssen die Zugriffe der CPU auf die LVL2-Pipe koordiniert werden.

- **Schnittstelle zwischen DTU und Frontend-Modul:** Die Umsetzung der von der RICH DTU gelieferten Auslesebefehle in die von den Frontends benötigten Auslesezyklen erfolgt ebenfalls durch den Readout-Controller. Ebenfalls müssen dabei für das Triggersystem die entsprechenden Busy-Signale generiert werden.

- **Schnittstelle zur IPU:** Über den Readout-Controller werden die Daten nach der Zusammenfassung von den einzelnen Frontend-Modulen und dem für die Bildverarbeitung notwendigen Mapping an die IPU übertragen. Um die Verzögerung des LVL2-Triggers möglichst klein zu halten, ist hierfür eine schnelle Übertragung notwendig.

Im folgenden wird auf die Implementierung der einzelnen Funktionen im RICH Readout-Controller eingegangen.

5.2 Blockschaltbild

Bedingt durch die teilweise enge Verzahnung der einzelnen Funktionen mußte das Blockschaltbild des Readout-Controllers (siehe Abbildung 5.1) stark vereinfacht werden, um eine gewisse Übersichtlichkeit zu wahren. Die Einteilung der Hardware in Funktionsblöcke, die den einzelnen Aufgabenstellungen in Abschnitt 5.1 entsprechen, ist jedoch deutlich zu erkennen:

Unten rechts ist die Schnittstelle zur CPU untergebracht. Realisiert ist diese Funktion durch einen VME-Chipsatz, der die eigentliche Anbindung an den VME-Bus übernimmt. Zur Kontrolle des Readout-Controllers dient der Interface-FPGA, der neben der Umsetzung des externen Datenbusses auf den internen Datenbus auch vielfache Test- und Diagnosemöglichkeiten beinhaltet.

Die Verwaltung der LVL2-Pipe befindet sich im linken unteren Funktionsblock. Sie ist unterteilt in den eigentlichen Speicher der LVL2-Pipe, der aus zwei Bänken SRAM besteht, der LVL2-Mapping-Einheit sowie den Memory-FPGA, der sämtliche Kontrollfunktionen für das Beschreiben und Auslesen der LVL2-Pipe übernimmt.

Die Schnittstelle zwischen DTU und Frontend-Modulen ist im rechten oberen Funktionsblock untergebracht. Als zentrales Steuerelement dient dabei der Timing-FPGA, der die komplette Umsetzung der DTU-Auslesebefehle in Auslesezyklen sowie die Fehlerkontrolle übernimmt. Zur Ansteuerung der einzelnen Ports dient jeweils ein Port-FPGA, dessen Funktionen ebenfalls vom Timing-FPGA gesteuert werden.

Eng verbunden mit diesem Funktionsblock ist die Schnittstelle zur IPU mit der LVL1-Mapping-Einheit im oberen linken Funktionsblock.

Die Platine des RICH Readout-Controllers ist in Abbildung 5.2 zu sehen.

5.3 Grundlegendes zur Funktionsweise

Zum besseren Verständnis der in diesem Abschnitt genauer erläuterten Funktionen ist es unumgänglich, zuerst einige prinzipielle Eigenschaften des Readout-Controllers zu diskutieren. Um die Komplexität des Systems nicht unnötig zu erhöhen, wurde bei der Entwicklung des Controllers bewußt darauf geachtet, die Hardware getrennt in zwei Betriebsarten

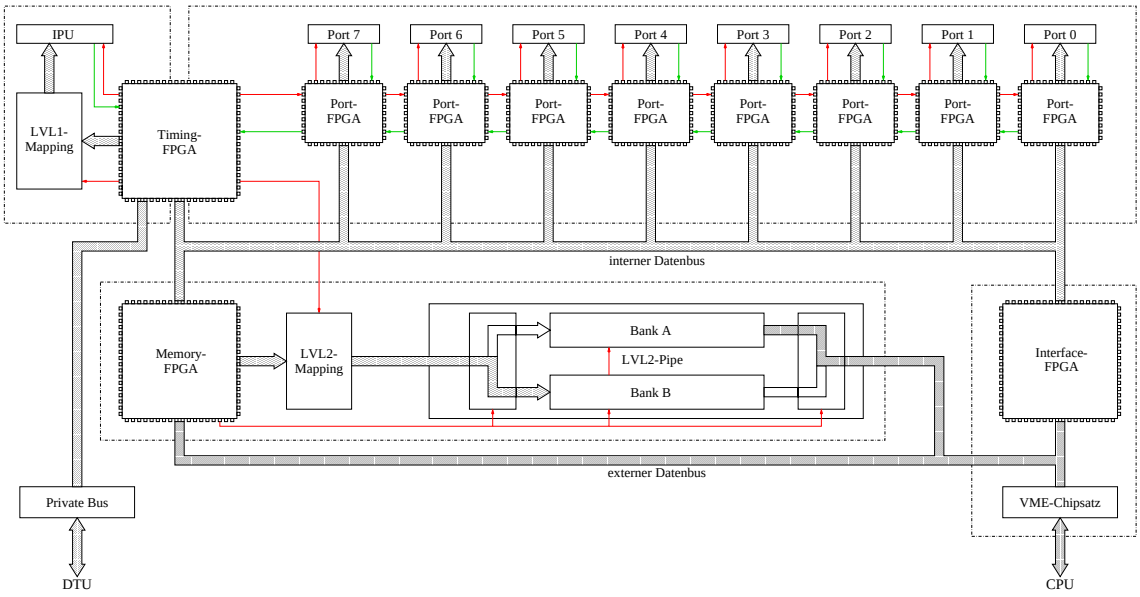


Abbildung 5.1: Blockschaltbild des RICH Readout-Controllers. Die Funktionsblöcke finden sich im Layout der Platine (Abbildung 5.2) wieder.

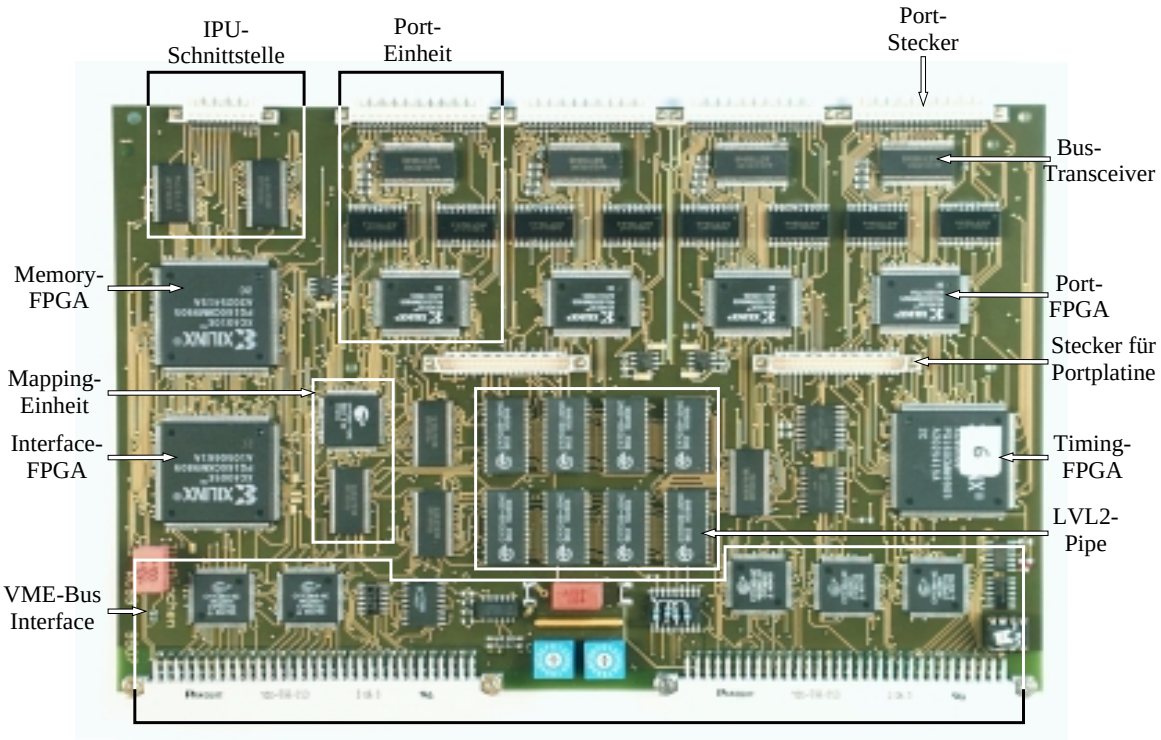


Abbildung 5.2: Layout des RICH Readout-Controllers. Die vier Port-Einheiten sind in identischer Form auf der (hier nicht aufgesteckten) Portplatine vorhanden. Die Basisadresse des Boards wird mit den beiden Hexadezimal-Schaltern eingestellt.

betreiben zu können: einem Modus, in dem das System komplett von der CPU durch Softwareroutinen gesteuert wird und einem automatischen Modus, in dem die volle Funktionalität und Geschwindigkeit des Systems zur Verfügung steht. Aus diesem Grund ist eine Doppelnutzung vieler Komponenten auf dem Readout-Controller notwendig, was allerdings im Blockschaltbild 5.1 nicht zum Ausdruck kommt. Insbesondere das Konzept der Datenbusse bedarf einer Erläuterung, ebenso wie im folgenden die Unterschiede zwischen den zwei unterschiedlichen Betriebsarten erklärt werden.

5.3.1 Der /STOP-Modus

Der Readout-Controller befindet sich nach der Initialisierung der FPGAs durch die VME-CPU automatisch im /STOP-Modus. In diesem Betriebsmodus verhält sich der Controller passiv: die Verbindung zur DTU ist getrennt, es erfolgen keinerlei Reaktionen auf von außen kommende Triggersignale. Als zentrales Steuerelement dient der Interface-FPGA, der Zugriffe auf alle im System befindlichen FPGAs zur Konfiguration ermöglicht. So ist es nur im /STOP-Modus möglich, die Frontend-Module zu konfigurieren, den Mapping-Speicher zu beschreiben und bestimmte Register mit für den RUN-Modus notwendigen Informationen zu laden. Der /STOP-Modus dient ebenfalls zur Fehlerverfolgung nach Fehlern, die im RUN-Modus bei der automatischen Datenaufnahme aufgetreten sind. Auch ist es im /STOP-Modus möglich, über Register im Interface-FPGA sämtliche Zyklen des RUN-Modus durch Softwaresteuerung von der CPU aus zu emulieren und so die gesamte Hardware kontrolliert testen zu können.

Durch Setzen eines Steuerbits im Interface-FPGA wird der Readout-Controller in den RUN-Modus geschaltet. Der Interface-FPGA gibt die gesamte Kontrolle über die Hardware an den Timing-FPGA ab, die Verbindung zur DTU wird hergestellt. Die weitere Steuerung des Readout-Controllers erfolgt nun über die DTU, die CPU steuert nur noch die zur Auslese der LVL2-Pipe notwendigen Funktionen. Die automatische Datenauslese erfolgt ausschließlich im RUN-Modus unter Steuerung des Timing- und des Memory-FPGAs, der Interface-FPGA hat nur noch stark beschränkte Kontrollmöglichkeiten. Eine Überwachung der laufenden Zyklen ist jedoch weiterhin möglich, auch läuft ein Teil der Kommunikation zwischen CPU und Readout-Controller weiterhin über den Interface-FPGA.

5.3.2 Die Struktur der Datenbusse

Wie man in Abbildung 5.1 sieht, wurde auf dem Readout-Controller eine Zweiteilung des Datenbusses in einen externen und einen internen Datenbus vorgenommen. Der externe Datenbus führt dabei vom VME-Chipsatz zum Interface- und Memory-FPGA, dem Mapping-Speicher sowie zur LVL2-Pipe. Der Datenbus ist zwischen Chipsatz und Speicher 32 bit breit ausgelegt, um beim Auslesen des Speichers hohe Datenraten zu ermöglichen. An Mapping-Speicher, Interface- und Memory-FPGA sind jedoch nur die unteren 16 bit des Datenbusses geführt, was aber für die notwendige Funktionalität bei weitem ausreicht. Somit sind von der CPU aus unabhängig vom Betriebsmodus des Readout-Controllers jederzeit Zugriffe auf diese Baugruppen möglich, wohingegen der Zugriff auf den internen

Datenbus nur über den Interface-FPGA erfolgen kann und gewissen Beschränkungen unterliegt.

Im /STOP-Modus steht der interne Datenbus exklusiv der CPU zur Verfügung. Über diese 16 bit breite Verbindung können alle acht Port- sowie der Timing-FPGA über Registerzugriffe konfiguriert werden. Ebenso ist bei entsprechender Ansteuerung eines Port-FPGAs der Zugriff auf die daran angeschlossene Frontend-Kette möglich, was beispielsweise beim Schreiben und Lesen der Schwellenwerte für die Frontends ausgenutzt wird.

Beim Umschalten in den RUN-Modus wird der CPU-Zugriff auf den internen Datenbus vom Interface-FPGA gesperrt. Unter Kontrolle des Timing-FPGAs wird der interne Datenbus jetzt zum Transport der von den Frontend-Modulen stammenden Analogdaten genutzt. In Abstimmung finden auch Zugriffe des Memory-FPGAs statt, die genaue Anbindung des Speichers über die Mapping-Einheit wird in Abschnitt 5.6 erläutert.

5.3.3 Datenformat

Entscheidend für die Auslegung der Datenbusse auf dem Readout-Controller war die Funktion als Datenkonzentrator. Beim Zusammenfassen der Datensätze von den einzelnen Frontend-Modulen ist streng darauf zu achten, daß die Analogdaten jedes einzelnen Kanals eindeutig identifizierbar abgespeichert werden. Um dies zu erreichen, werden den von den Frontends kommenden Daten in mehreren Stufen Informationen über ihre Herkunft hinzugefügt. Dabei wird aber darauf geachtet, daß nur die Informationen hinzugefügt werden, die zur einwandfreien Identifikation innerhalb der jeweiligen Stufe unbedingt notwendig sind. So werden die zum Transport benötigten Datenbusse so klein wie möglich gehalten, was sich positiv auf das Gesamtsystem auswirkt. Anhand des Datenpfades der Analogdaten vom Frontend bis in die LVL2-Pipe wird dieser Vorgang nun beschrieben (siehe Abbildung 5.3).

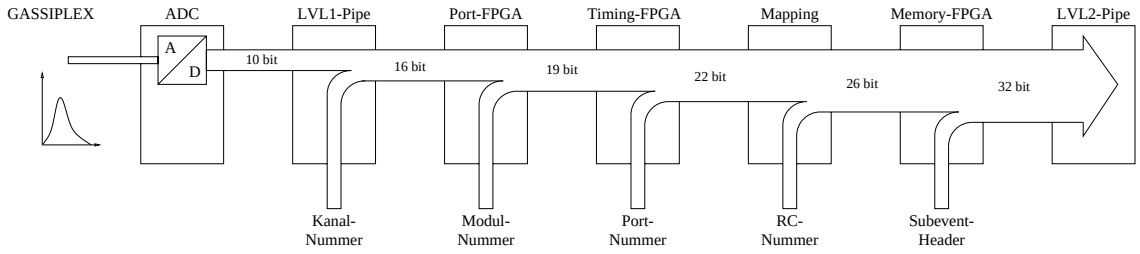


Abbildung 5.3: Datenfluß vom Frontend in die LVL2-Pipe

Im Frontend wird aus dem analogen Eingangssignal jedes Kanals ein 10 bit Digitalwert erzeugt. Da pro Frontend nur 64 Kanäle abgetastet werden, genügt an dieser Stelle das Hinzufügen von 6 bit zusätzlicher Information, die binär kodiert die Kanalnummer enthalten. In der LVL1-Pipe stehen die Daten also mit 16 bit Breite.

An einem Port können maximal 8 Frontend-Module angeschlossen werden; um wiederum eine eindeutige Zuordnung zu erhalten, werden beim Übertragen der Daten vom Frontend

auf den Readout-Controller 3bit angefügt, die die Nummer des jeweiligen Moduls anzeigen. Der Datenpfad zwischen Port- und Timing-FPGA ist somit 19 bit breit.

Durch die Auslesesteuerung werden 8 Ports bedient, also müssen beim Weitertransport in Richtung LVL2-Pipe wiederum 3 bit hinzugefügt werden, die die Nummer des Ports angeben, von dem das Datenwort stammt.

Da pro Sektor des RICH zwei Readout-Controller eingesetzt werden und somit 12 mögliche Datenquellen existieren, muß vor dem Speichern in der LVL2-Pipe die Nummer des Readout-Controllers den Daten hinzugefügt werden: dies geschieht per Konvention durch 4 bit in der Mapping-Einheit, die zuvor durch die CPU entsprechend konfiguriert werden muß. Hinter der Mapping-Einheit verläuft der Datenbus bereits mit 32 bit Breite; jedem Analogwert sind hier alle zu einer eindeutigen Zuordnung notwendigen Informationen beigelegt. Dies hat den Vorteil, daß bei einem fehlerhaften Wort im Datensatz nur ein einziger Kanal falsch identifiziert wird, die anderen jedoch durch die hohe Redundanz immer noch eindeutig sind.

32 bit Datenwort					
31 ... 26	25 24 23 22	21 20 19	18 17 16	15 14 13 12 11 10	9 ... 0
reserved	#RC	#P	#M	#CH	ADC

Tabelle 5.1: Datenformat des Readout-Controllers. Nach dem Hinzufügen der Verwaltungsinformationen (Kanal-, Modul-, Port- und RC-Nummer) ist jedes Datenwort 32 bit breit.

Um der DAQ-Software eine eindeutige Identifizierung der einzelnen Subevents zu ermöglichen, wird jedem Datensatz beim Speichern in der LVL2-Pipe ein sogenannter Subevent-Header mit Verwaltungsinformationen hinzugefügt. Die Struktur dieses Headers sowie sein Zustandekommen wird später in Abschnitt 5.6.1.3 beschrieben.

5.3.4 Die Mapping-Einheit

Das vom Readout-Controller intern verwendete Datenformat ist durch den Aufbau der Hardware festgelegt. Um bei der Verarbeitung der Daten in der IPU und der DAQ-Software eine entsprechende Flexibilität zu haben, wurde eine Möglichkeit vorgesehen, die Verwaltungsinformationen eines Datenwortes vor der Weitergabe an die IPU bzw. die Speicherung in der LVL2-Pipe in eine frei wählbare Form zu bringen. Dies geschieht in der Mapping-Einheit, die auf dem Readout-Controller in Form eines Dualported RAMs vom Typ CY7C025 [6] und eines 16 bit-Bustreibers vom Typ 74ACT16245 [13] implementiert wurde.

Das Mappen geschieht folgendermaßen: Während der Konfiguration des Readout-Controllers beschreibt die CPU über die Seite A den Mappingspeicher und legt damit das Mapping fest. Während der automatischen Auslese im RUN-Modus werden die Datenbits `DB[21:10]` an die Adreßeingänge der Seite B geführt; an den Datenausgängen von Seite B erscheinen dann auf dem sogenannten Memory Bus `MB[25:10]` die Informationen der jeweils angesprochenen Speicherzelle. Dieser Bus verbindet die Mapping-Einheit sowohl

mit der IPU-Schnittstelle als auch mit der LVL2-Pipe, wobei nur an die LVL2-Pipe auch die unteren Bits `DB[9:0]` des internen Datenbusses geführt werden.

Um für die IPU- und die LVL2-Daten jeweils getrennt ein Mapping durchführen zu können, wird das oberste Adreßbit der Seite B von der Auslesesteuerung entsprechend gesetzt: der untere Teil des Mapping-Speichers enthält die Map-Informationen für die LVL2-Pipe, der obere die für die IPU. Beide Mappings sind somit komplett entkoppelt; Änderungen im einen Teil wirken sich nicht auf den anderen Teil aus. Diese Lösung bietet bei nur geringer Datenverzögerung die größtmögliche Flexibilität. Die genaue Lage des Mapping-Speichers im Adreßraum ist in Abschnitt B.1 aufgeführt.

Da auf dem internen Datenbus auch das Triggertag übertragen wird, muß das Mapping abschaltbar sein. Dies wird durch den 16 bit-Bustreiber ermöglicht, der — ebenfalls vom Timing-FPGA gesteuert — das Vorbeischleusen des Triggertags am Mapping-Speicher ermöglicht.

Die Ansteuerung der Mapping-Einheit erfolgt dabei durch zwei Steuerleitungen:

- `/MAP` schaltet den Datenpfad durch den Map-Speicher, d.h. es findet ein Mappen der anstehenden Daten statt.
- `/TAG` schaltet den Map-Speicher in den passiven Modus und aktiviert den Bustreiber, der die Daten unverändert an den Memory Bus weiterleitet.

5.4 Die Steuerung der Frontend-Auslese

Als zentrale Funktionsgruppe des Readout-Controllers benötigt die Auslesesteuerung mehrere Schnittstellen zur Außenwelt:

1. Zur Entgegennahme von Auslese-Befehlen muß eine Schnittstelle zur DTU implementiert werden; über diese Schnittstelle werden auch bestimmte Betriebszustände (wie Busy und Error) gemeldet.
2. Für die Erzeugung des LVL2-Triggers müssen die Ansprechmuster der Frontend-Module an die Ringerkenner-Einheit transportiert werden. Dafür ist eine Schnittstelle vorgesehen, über die der komplette Transfer von Daten und zugehörigem Tag stattfindet.
3. Bei einem LVL2-Trigger mit positiver Decision müssen die Daten in der LVL2-Pipe gespeichert werden. Die LVL2-Pipe ist — wie im Blockschaltbild 5.1 dargestellt — eine eigene Funktionseinheit, die auch über eine eigene Schnittstelle angesprochen wird.

Diese Schnittstellen werden zum besseren Verständnis vor der Beschreibung der einzelnen Auslesezyklen kurz erläutert; ebenso wird auf den prinzipiellen Aufbau der Auslesesteuerung eingegangen.

5.4.1 Grundlegender Aufbau der Auslesesteuerung

Die Auslesesteuerung selbst wurde zentral im Timing-FPGA untergebracht. Zur Ansteuerung der einzelnen Modulketten wurden acht identische Porteinheiten vorgesehen, von denen vier direkt auf dem Readout-Controller und weitere vier auf der Aufsteckplatine (auch als Port-Platine bezeichnet) untergebracht sind. Jede dieser Porteinheiten besteht im wesentlichen aus vier Bauelementen:

- einem Bus-Transceiver vom Typ 74ACT16646, der als Schnittstelle zum Portstecker dient
- zwei Treiberbausteinen vom Typ 74ACT11244 für diverse Steuer- und Kontrollsignale
- sowie dem Port-FPGA, der sämtliche Funktionen des Bustransceivers steuert.

Durch diese Aufteilung werden die Vorteile von Standardbauteilen und FPGA genutzt: der Bus-Transceiver ist speziell für die Datenübertragung zwischen zwei unterschiedlichen Bussen (und damit auch für eine entsprechende Treiberleistung) ausgelegt; die Steuerlogik für dieses komplexe Bauelement befindet sich aber — jederzeit änderbar — in einem FPGA.

In der folgenden Funktionsbeschreibung werden die oben erwähnten Treiberbausteine bewußt übergangen, da sie keinen Einfluß auf die prinzipielle Funktionsweise haben und nur zur Aufbereitung der Steuer- und Kontroll-Signale dienen, die über das Flachbandkabel zur Backplane und zurück laufen.

Die Kommunikation zwischen Timing- und Port-FPGA läuft nur über `/STRBx`, `PSELx`, `MACKFx` sowie `STRBRFx` ab, wobei `x` die Nummer des jeweiligen Ports angibt:

- `STRBRFx` entspricht dem Signal `STRBRPx`, das auf der Backplane über einen invertierenden Treiber aus dem `/STRBO` des letzten Moduls der Kette erzeugt wird. Im Port-FPGA wird dieses Signal lediglich in einem Register gespeichert.
- `MACKFx` entspricht dem Signal `MACKx`, das über einen auf der Backplane befindlichen Logikbaustein aus den `/SMACK`-Signalen der Frontend-Module logisch oderverknüpft wird. Dieses Signal wird ebenfalls an ein Flipflop geführt; zusätzlich dient es als Zählimpuls für einen 3 bit-Zähler, der die Nummer des gerade angesprochenen Moduls mitzählt. Dies wird — wie in Abschnitt 5.3.3 erläutert — zur Erweiterung des Datenwortes genutzt; die Modulnummer erscheint dabei an den Ausgängen `DB[18:16]` des Port-FPGAs.
- `/STRBx` wird vom Timing-FPGA erzeugt und an den jeweiligen Port-FPGA geführt. Aus `/STRBx` wird ein Clock-Signal für den Bus-Transceiver zum Übernehmen anstehender Daten erzeugt; außerdem wird dieses Signal als `/STRBPx` an die Backplane geführt, wo es an den `/STRBI`-Eingang des ersten Moduls angeschlossen ist.

- `P-SELx` wählt jeweils eine Port-Einheit an. Mit diesem Signal werden die Datenausgänge des Bus-Transceivers und die Datenausgänge `DB[18:16]` des Port-FPGAs durchgeschalten. Um Kollisionen auf dem internen Datenbus zu vermeiden, darf jeweils nur eines der `PSEL[7:0]`-Signale aktiv sein.

In diesem Zusammenhang muß allerdings auf eine historisch bedingte Unstimmigkeit hingewiesen werden: im Prototyp des Readout-Controllers, der mit passiven Backplanes betrieben wurde, waren die Kontrollsignale `STRBRx` und `MACKPx` low-aktiv; mit der Einführung der neuen aktiven Backplanes hat sich aber die Polarität dieser beiden Signale verkehrt. Dies muß beim Anschluß von nur einem Frontend ohne Backplane beachtet werden; im Port-FPGA ist deshalb eine Möglichkeit vorgesehen, intern wieder auf die alte Polarität schalten zu können. Dies ermöglicht es, einfache Testaufbauten ohne Backplanes zu Debugging-Zwecken nutzen zu können.

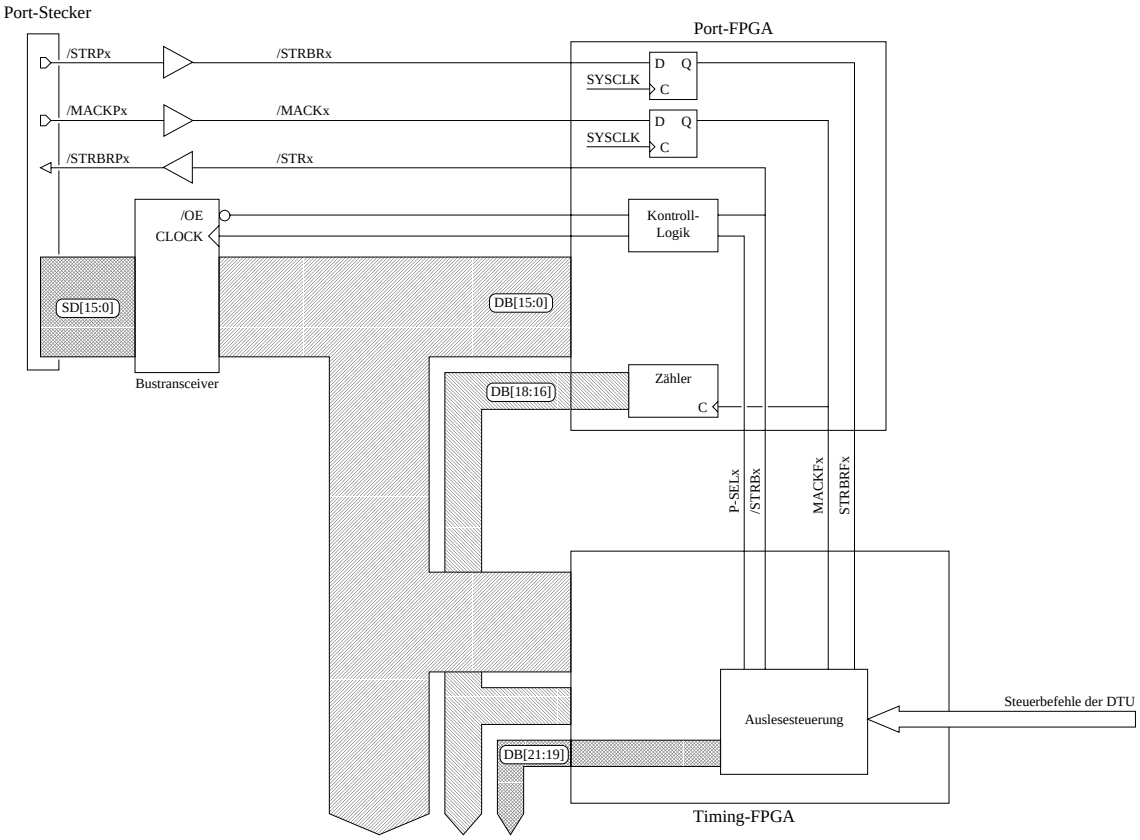


Abbildung 5.4: Blockschaltbild eines Ports. Die Bedeutung der Steuer- und Kontrollsignale ist im Text näher erläutert.

Die Ansteuerung einer Porteinheit zur Datenübertragung aus den Frontends verläuft folgendermaßen: Bei korrekt angelegtem Funktionskode bewirkt ein Puls auf `/STRBx`, daß im gerade angesprochenen Frontend ein Datenwort auf den Frontend-Bus ausgegeben wird. Die fallende Flanke von `/STRBx` schaltet dies auf den Bus, mit der steigenden Flanke

wird es in den Bus-transceiver übernommen. Die Länge des /STRBx-Pulses ist somit bestimmt durch die Zeit, die bis zum stabilen Anliegen der Datenbits am Bus-Transceiver benötigt wird. Anschließend kann der Timing-FPGA das so gespeicherte Datenwort samt der Information, aus welchem Modul es stammt, über einen P-SELx-Puls anfordern; die angesprochene Porteinheit schaltet das Datenwort dazu auf die Datenleitungen DB[18:0].

Im /STOP-Modus wird der Bus-Transceiver für das Konfigurieren der Frontend-Schwellenspeicher genutzt. Dabei werden über ein Register im Port-FPGA die notwendigen Steuersignale generiert; das Beschreiben und Lesen des Transceiver-Registers wird ebenfalls über eine Adresse des jeweiligen Port-FPGAs ermöglicht.

Eine detaillierte Registerbeschreibung der Port-FPGAs sowie des Timing-FPGAs befindet sich in Anhang B.5.

5.4.2 Die Schnittstelle zur DTU

Zur Anbindung an die RICH DTU werden von der Auslesesteuerung sechs Signale genutzt. Hier wird nur auf die grundlegende Bedeutung dieser Signale eingegangen, ihre Erzeugung und die genaue Nutzung zur Kommunikation zwischen DTU und Readout-Controller wird in Abschnitt 6 erklärt:

- /BEGRUN: Dieses Signal setzt den Readout-Controller und alle angeschlossenen Frontends zurück. Die LVL1-Pipe wird komplett gelöscht, sämtliche Adreß- und Datenzähler auf Null gesetzt. Die Ausleseelektronik ist anschließend wieder bereit, Daten aufzunehmen. /BEGRUN wird zum kontrollierten Start der gesamten Ausleseelektronik von HADES benutzt.
- /PRDOUT: Mit diesem Signal wird von der DTU die Übertragung der Daten zur IPU-Schnittstelle angefordert. Üblicherweise erfolgt dies sofort nach dem zugehörigen Triggersignal, bedingt durch die DTU-interne Arbitrierung kann es aber zu gewissen Verzögerungen kommen.
- /ARDOUT: Dieses Signal stellt den eigentlichen LVL2-Trigger dar. Es muß immer in Zusammenhang mit DECISION betrachtet werden: ist DECISION low, so werden Daten aus der LVL1-Pipe in die LVL2-Pipe übertragen. Ist DECISION high, werden die Daten des zugehörigen Events in der LVL1-Pipe gelöscht.
- DECISION: Dieses Entscheidungssignal ist nur in Zusammenhang mit /ARDOUT gültig. Zwischen einzelnen Pulsen von /ARDOUT kann es ungültige Werte annehmen, entscheidend ist jeweils der Wert bei der fallenden Flanke von /ARDOUT.
- /ERROR: Über diese Open-Collector-Leitung werden Fehler bei der Datenkonzentration an die DTU gemeldet. Durch Setzen dieser Leitung werden alle weiteren Trigger abgeblockt und das System zur Fehleranalyse gestoppt.
- /RCBSY: Dieses Open-Collector-Signal wird vom Timing- und vom Memory-FPGA benutzt, um der DTU einen gerade ablaufenden Zyklus bzw. eine volle LVL2-Pipe

zu melden. Währen /RCBSY low ist, können von der DTU keine weiteren LVL1- oder LVL2-Befehle an den Readout-Controller geschickt werden.

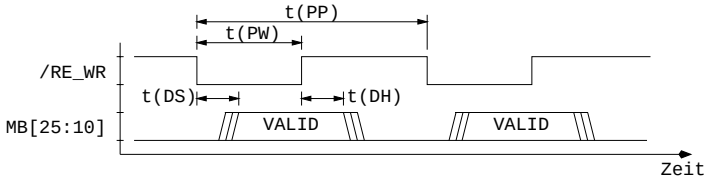
5.4.3 Die Schnittstelle zum Ringerkenner

Die Ansteuerung des Ringerkenners erfolgt im wesentlichen über zwei Signale:

- /RE_CYCL bereitet den Ringerkenner durch eine fallende Flanke auf eine beginnende Übertragung vor, bleibt während der Übertragung selbst low und zeigt das Ende der Übertragung durch eine steigende Flanke an. Diese Flanke wird ebenfalls zum Markieren des Triggertags genutzt.
- /RE_WR dient als Strobesignal für die einzelnen Datenworte, eine steigende Flanke zeigt gültige Daten an.

Zur Datenübertragung werden die Bits MB[25:10] genutzt, die — wie in Abschnitt 5.3.4 beschrieben — durch die Mapping-Einheit in das benötigte Format gebracht werden. Bedingt durch die Durchlaufzeit durch den Mapping-Speicher stehen die Daten allerdings erst kurz vor der steigenden Flanke von /RE_WR stabil an; dies muß beim Entwurf der Ringerkennereinheit berücksichtigt werden. Generell wird eingangsseitig der Einsatz eines schnellen Registers mit kurzen Setup-Zeiten für diese Datensignale empfohlen.

Die Schnittstelle zum Ringerkenner ist im unteren Teil der Platine des Readout-Controllers als 26-poliger Ministecker vom Typ SMC ausgeführt (siehe Tabelle 5.2). Die dort anstehenden Signale führen TTL-Pegel; um Störungen der Signale zu vermeiden, ist das Kabel zum Ringerkenner möglichst kurz zu halten. Das Timing der Schnittstelle selbst hängt stark von der Belegung der einzelnen Frontends ab. Es können daher nur Werte für den minimalen Abstand zwischen zwei /RE_WR-Pulsen sowie die generelle Setup-Zeit der Daten angegeben werden.



	min.	typ.	max.	
t(PW)	32	—	—	[ns]
t(PP)	64	—	—	[ns]
t(DS)	—	15	—	[ns]
t(DH)	—	15	—	[ns]

Abbildung 5.5: Timing-Parameter der IPU-Schnittstelle.

Die Schnittstelle besitzt zwei momentan ungenutzte Signale auf den Pins B11 und A12. Diese sind am Timing-FPGA angeschlossen und können bei Bedarf zur Erweiterung des Handshakes benutzt werden. Angedacht ist eine Rückmeldung der Ringerkennereinheit, wenn bei einem Ereignis zuviele Pads angesprochen haben. Da wegen der FIFO-Struktur der LVL1-Pipe eine laufende Übertragung nicht abgebrochen werden kann, ist nur eine Ausblendung der /RE_WR-Pulse möglich; die IPU kann den Rest der Übertragung zur Löschung ihrer FIFOs nutzen und mit Ende der Datenübertragung wieder aufnahmebereit sein.

Signalname	Richtung	Pins		Richtung	Signalname
MB10	out	B1	A1	out	MB11
MB12	out	B2	A2	out	MB13
MB14	out	B3	A3	out	MB15
MB16	out	B4	A4	out	MB17
MB18	out	B5	A5	out	MB19
MB20	out	B6	A6	out	MB21
MB22	out	B7	A7	out	MB23
MB24	out	B8	A8	out	MB25
GND	—	B9	A9	—	GND
n.c.	—	B10	A10	—	n.c.
/RE_FLCR	bi	B11	A11	out	/RE_CYCL
/RE_WR	out	B12	A12	bi	/RE_R
GND	—	B13	A13	—	GND

Tabelle 5.2: Pinbelegung der Ringerkenner-Schnittstelle. Die Signalrichtung ist aus Sicht des Readout-Controllers angegeben.

5.4.4 Die Schnittstelle zur LVL2-Pipe

Wie aus dem Blockschaltbild Abbildung 5.1 ersichtlich, wird die LVL2-Pipe vom Memory-FPGA verwaltet. Trotz der Komplexität dieses Bausteins genügen vier Steuerleitungen, um die korrekte Erstellung eines kompletten Subevents in der LVL2-Pipe sicherzustellen:

- `EVTBEG` zeigt über einen Puls den Start einer Datenübertragung zur LVL2-Pipe an.
- `EVTEND` beendet durch einen Puls die Übertragung; gleichzeitig wird während dieses Pulses das Triggertag zum Einbau in den Subevent-Header übergeben.
- `/MWR` dient als Strobe-Signal, die Daten stehen mit der steigende Flanke stabil an.
- `MSEL` dient zur Umschaltung der Speicherbänke und wird von der DAQ-Software generiert.

Im Abschnitt 5.5.3 wird genauer auf die Verwendung dieser Signale eingegangen.

5.5 Zyklen zur Frontend-Auslese

5.5.1 Zurücksetzen der Ausleseelektronik

Zum schnellen Zurücksetzen der Auslesehardware wurde beim Entwurf des Triggersystems der BeginRun-Trigger vorgesehen. Jeder Datenaufnahmezyklus (ein sog. Run) im HADES-System wird mit einem BeginRun-Trigger gestartet und von einem EndRun-Trigger beendet. Bei einem BeginRun-Trigger werden sämtliche Datenaufnahme-Systeme jedes Detektors initialisiert, d.h. die LVL1- und LVL2-Pipes gelöscht, Adreßzähler zurückgesetzt und

das System für die Aufnahme neuer Daten vorbereitet. Der RICH Readout-Controller unterstützt dieses Zurücksetzen über den Triggerbus, eine Software-gesteuerte Initialisierung ist aber ebenfalls möglich.

Ein BEGRUN-Zyklus läuft beim RICH wie in Abbildung 5.6 dargestellt ab:

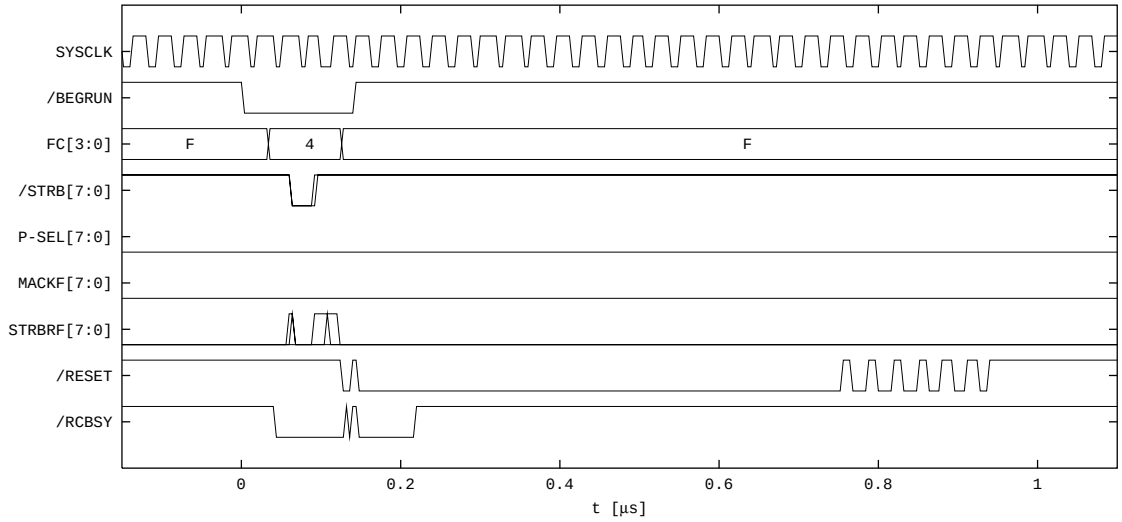


Abbildung 5.6: BEGRUN-Zyklus. Der Readout-Controller setzt erst die angeschlossenen Frontends, dann sich selbst zurück. Der Vorgang ist nach ca. 1 μs komplett abgeschlossen.

Von der DTU wird über den Private Bus ein kurzer Puls auf der `/BEGRUN`-Leitung geliefert. Der Timing-FPGA quittiert diesen Befehl, indem er die `/RCBSY`-Leitung auf low zieht. Anschließend wird auf den Leitungen `FC[3:0]` der Funktionskode RFIFO ausgegeben. Durch einen Puls von 64 ns Länge auf den Strobeleitungen `/STRB[7:0]` wird in allen angeschlossenen Frontend-Modulen ein kompletter Reset durchgeführt, d.h. alle internen Zähler- und Kontrollregister auf ihre Anfangswerte gesetzt sowie beide FIFO-Bänke der LVL1-Pipe durch `/MR` gelöscht.

Nach dem Zurücksetzen der Frontend-Module wird der Funktionskode auf NOP gesetzt. Der Readout-Controller selbst wird nun durch einen Puls von mindestens 200 ns auf der `/RESET`-Leitung in einen definierten Anfangszustand gebracht. Damit ist die Hardware bereit für neue Datenaufnahme-Zyklen; das Ende des BEGRUN-Zyklus wird der DTU durch Zurücknehmen von `/RCBSY` gemeldet.

Bei den etlichen Tests zeigte sich, daß die DAQ-Software Probleme mit dieser Art der Hardware-Initialisierung hatte. In späteren Versionen der Xilinx-Designs bzw. der DAQ-Software sollte dies behoben sein; bei Problemen mit dem BEGRUN-Zyklus kann das `/BEGRUN`-Signal (und damit die Initialisierung per Triggerbefehl) in der DTU über ein Register ausgeschaltet werden.

5.5.2 Musterübertragung zum Ringerkenner

Im PRDOUT-Zyklus werden vom Readout-Controller die Ansprechmuster aus den Frontend-Modulen zur Ringerkenner-Einheit übertragen. Im späteren Experimentaufbau wird ein Readout-Controller stets acht mit Frontend-Ketten bestückte Ports auslesen; aus Gründen der Übersichtlichkeit wird zur Beschreibung des PRDOUT-Zyklus nur ein vereinfachter Aufbau mit zwei Ketten benutzt. Der Ablauf ist in Abbildung 5.7 dargestellt; die Ansteuerung der beiden benutzen Ports erfolgt dabei wie in Abschnitt 5.4.1 beschrieben.

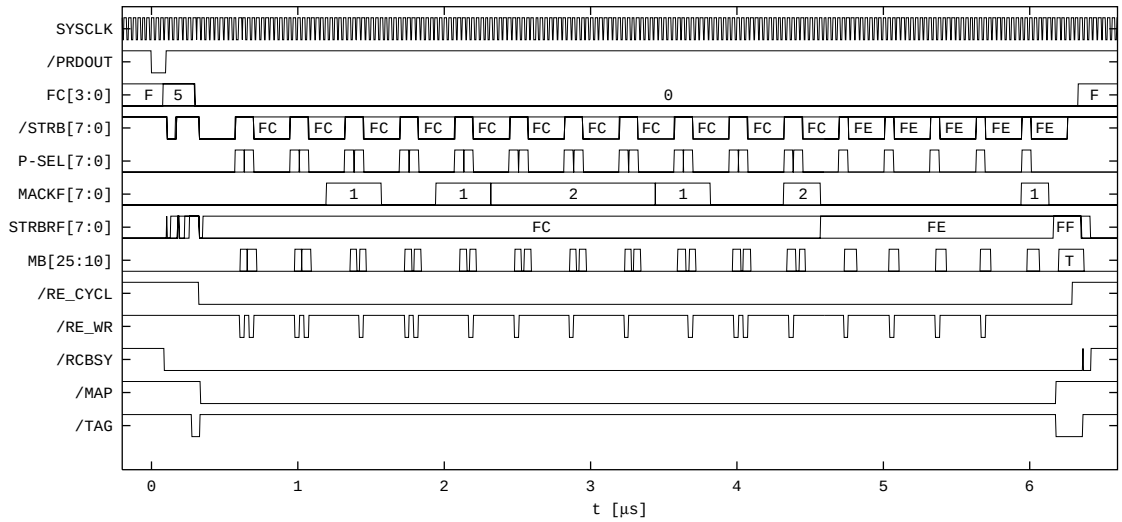


Abbildung 5.7: PRDOUT-Zyklus aus Sicht des Readout-Controllers. Es werden zwei Frontend-Ketten vollautomatisch ausgelesen und die Daten zur IPU übertragen.

Als Startsignal für den Zyklus dient das Private Bus-Signal `/PRDOUT`. Mit dessen fallenden Flanke setzt der Timing-FPGA das Signal `/RCBSY`, um weitere Triggerbefehle für die Dauer des Zyklus abzublocken. Gleichzeitig wird auf `FC[3:0]` der Funktionskode RDAISY angelegt und parallel auf allen Strobe-Leitungen `/STRB[7:0]` ein 64ns langer Puls ausgegeben. Dadurch wird in allen angeschlossenen Frontend-Modulen — wie in Abschnitt 4.4.2 beschrieben — die daisy chaining-Logik in einen definierten Zustand gebracht und die Module für den nun folgenden eigentlichen Auslesezyklus vorbereitet. Nach diesem Strobepuls zum Zurücksetzen wird der Funktionskode auf PRDOUT geändert. Zeitgleich mit dem Funktionskode RDAISY wird ebenfalls die `/RE_CYCL`-Leitung auf low gesetzt, um die IPU auf die nun folgende Übertragung vorzubereiten.

Bei jedem jetzt folgenden Puls auf einer der `/STRB[7:0]`-Leitungen wird der Zustand der zugehörigen `STRBRFx`- und `MACKFx`-Leitung überwacht. Dies dient zum Feststellen von zwei besonderen Fällen:

1. Geht während eines `/STRBx`-Pulses `MACKFx` auf high, so ist das von Port `x` stammende Datenwort ein Triggertag. Beim ersten derartigen Ereignis wird das Triggertag intern im Timing-FPGA als Referenz gespeichert, bei allen folgenden derartigen Ereignissen ein Vergleich von einlaufendem und Referenz-Triggertag durchgeführt.

2. Geht während eines /STRBx-Pulses STRBRFx auf high, so ist die Modulkette an Port x komplett abgearbeitet. Es werden keine weiteren /STRBx-Pulse mehr generiert.

Bedingt durch das verwendete Auslese-Protokoll kann nur eines der beiden Kontroll-Signale STRBRFx und MACKFx zu einem Zeitpunkt auf high gehen, nicht jedoch beide.

Der erste Puls auf STRB[7:0] beim Funktionskode PRDOUT geht stets an alle Ports. Wie man sieht, wird im Beispiel während dieses Pulses auf STRBRF[7:0] der Wert 0xFC zurückgemeldet. Das heißt, daß Port 2 bis Port 7 keine Modulketten angeschlossen haben, folglich werden beim nächsten Strobepuls diese Ports nicht mehr bedient. Nach Ende des Pulses werden über P-SEL0 und P-SEL1 die Datenworte aus den Portregistern abgerufen und mit je einem /RE_WR-Puls an die IPU übergeben.

Beim nächsten Puls auf /STRB0 und /STRB1 läuft der Vorgang analog ab, jedoch ändert sich das Verhalten im dritten Puls: hier wird MACKF0 gesetzt. Die Ablaufsteuerung regiert darauf, indem das Datenwort von Port 0 (das wie zuvor mittels P-SEL0 abgerufen wird) nicht an die IPU übergeben wird (es fehlt der /RE_WR-Puls), sondern intern als Referenz-Tag in einem Register gespeichert wird.

Beim fünften Puls wird ebenfalls MACKF0 gesetzt. Nun wird das von Port 0 kommende Datenwort gegen das Referenz-Tag verglichen; auch hier fehlt der /RE_WR-Puls. Im nächsten Puls meldet Port 1 das Ende eines Modul-Datensatzes über MACKF1.

Der weitere Verlauf des PRDOUT-Zyklus erfolgt nach den oben beschriebenen Regeln. Erkennbar ist das beispielsweise beim zwölften Puls: hier wird STRBRF1 gesetzt; als Reaktion darauf werden von der Ablaufsteuerung keine weiteren Pulse mehr auf /STRB1 ausgegeben.

Das Ende des Zyklus wird während des 17. Pulses erkannt: hier erscheint das letzte noch fehlende STRBRF-Signal von Port 0. Der Timing-FPGA legt daraufhin das gespeicherte Referenztag (als "T" gekennzeichnet) auf MB[25:10] und setzt als Markierung /RE_CYCL auf high zurück. Nach der so erfolgten Übergabe des Triggertags an die IPU wird der Funktionskode wieder auf NOP gesetzt und /RCBSY zurückgenommen; der Readout-Controller ist wieder bereit, neue DTU-Befehle entgegenzunehmen.

Die Ansteuerung der Mapping-Einheit kann anhand der Signale /MAP und /TAG verfolgt werden: /MAP ist während des gesamten Auslesezyklus aktiv und wird erst am Ende des Zyklus zurückgenommen, wenn das Triggertag an die IPU geschickt werden soll. Vor dem Anlegen dieses Triggertags wird /TAG aktiviert und eine direkte Übertragung zur IPU sichergestellt.

Sollte bei einer der Vergleichsoperationen zwischen Referenz- und einlaufenden Triggertag ein Fehler festgestellt werden, so setzt der Timing-FPGA die Leitung /ERROR auf low und protokolliert in einem Statusregister mit, welche Ports ein fehlerhaftes Triggertag geliefert haben. Der Zyklus wird trotz der Fehlerbedingung normal beendet, um die FIFOs des Frontends in einen definierten Zustand zu bringen. In der DTU werden durch /ERROR weitere Triggerbefehle unterbunden und der fehlerhafte Zustand zur Analyse eingefroren. Bedingt durch die FIFO-Struktur der LVL1-Pipe ist eine hardwaremäßige Fehlerbehandlung nicht möglich; an dieser Stelle greift die slow control in die Datenaufnahme ein und setzt das System wieder in Gang.

5.5.3 Übertragung von Analogdaten in die LVL2-Pipe

Der ARDOUT-Zyklus verläuft bis auf die fehlende Ansteuerung der IPU-Schnittstelle vom Ablauf her exakt wie der PRDOUT-Zyklus. Zur Verdeutlichung wurde als Beispiel die Auslese desselben Datensatzes wie in Abschnitt 5.5.2 gewählt; somit können die Unterschiede leicht verstanden werden.

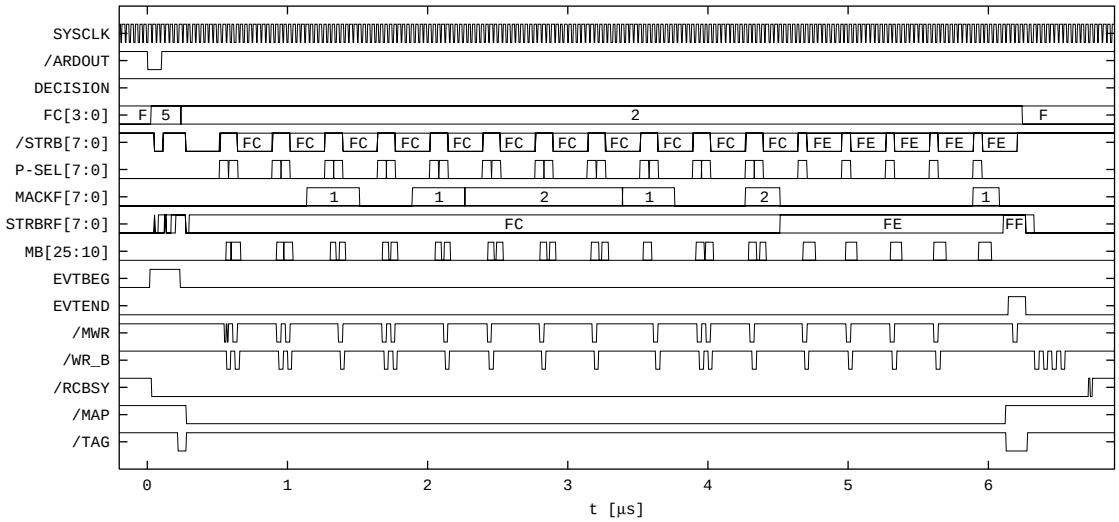


Abbildung 5.8: ARDOUT-Zyklus aus Sicht des Readout-Controllers. Es wird derselbe Datensatz wie in Abbildung 5.7 ausgelesen, jedoch in die LVL2-Pipe geschrieben. Die letzten vier Schreibpulse von `/WR_B` markieren die Erstellung des Subevent-Headers.

Als Start-Signal für den ARDOUT-Zyklus dienen hierbei `/ARDOUT` und `DECISION` (siehe Abbildung 5.8). Die fallende Flanke von `/ARDOUT` bei gesetzter `DECISION` leitet den Zyklus ein; gleichzeitig mit dem Funktionskode RDAISY wird jetzt `EVTBEG` auf high gesetzt. Der vom Timing-FPGA für jedes gültige Datenwort gelieferte Puls auf der `/MWR`-Leitung wird in ein Schreibsignal (hier `/WR_B` für die Speicherbank B) umgesetzt; wie im PRDOUT-Zyklus auch werden für die im Datenstrom enthaltenen Tags keine Strobe-Signale erzeugt.

Das Ende des Zyklus, das durch den Wert `0xFF` auf den Leitungen `STRBRF[7:0]` während des 17. Pulses erkannt wird, verläuft jedoch anders als beim PRDOUT-Zyklus:

Der Timing-FPGA setzt `/TAG` auf low und kurz darauf `EVTEND` auf high. Jetzt liegt das Referenztag zur Übertragung in die LVL2-Pipe auf den Datenleitungen an und wird mit einem Puls auf `/MWR` in ein Register im Memory-FPGA übertragen. Der Timing-FPGA beendet daraufhin den Zyklus, indem er den Funktionskode NOP ausgibt und sein `/RCBSY`-Signal deaktiviert.

Der Memory-FPGA jedoch hält sein `/RCBSY` (und damit auch die Leitung zur DTU) auf low, um in den nun folgenden 420 ns dem gerade erzeugten Datensatz in der LVL2-Pipe einen Subevent-Header voranzustellen und die LVL2-Pipe auf die Aufnahme des nächsten Subevents vorzubereiten. Anschließend setzt auch der Memory-FPGA sein `/RCBSY` zurück

und gibt den Readout-Controller für den nächsten Zyklus frei. Die Verwaltung der LVL2-Pipe wird in Abschnitt 5.6 eingehend erläutert.

5.5.4 Löschen von Analogdaten

Als letzter der von der Auslesesteuerung automatisch erzeugten Zyklen wird der ADELETE-Zyklus vorgestellt. Er dient zum Löschen von Datensätzen in der LVL1-Pipe, die keine physikalisch relevanten Daten enthalten. Im Gegensatz zu den Auslesezyklen PRDOUT und ARDOUT werden alle an einen Readout-Controller angeschlossenen Frontends im block-Modus angesprochen; das Löschen der Daten erfolgt parallel. Dieser Zyklus läuft also erheblich schneller als ein normaler Auslesezyklus ab. Zur Verdeutlichung wurde in den Abbildungen stets als Zeitreferenz das `SYSCLK`-Signal (32 ns) mit angezeigt.

Der typische Verlauf eines ADELETE-Zyklus ist in Abbildung 5.9 zu sehen. Um die Zeiten selbst vergleichen zu können, ist derselbe Datensatz wie in Abschnitt 5.5.2 gewählt.

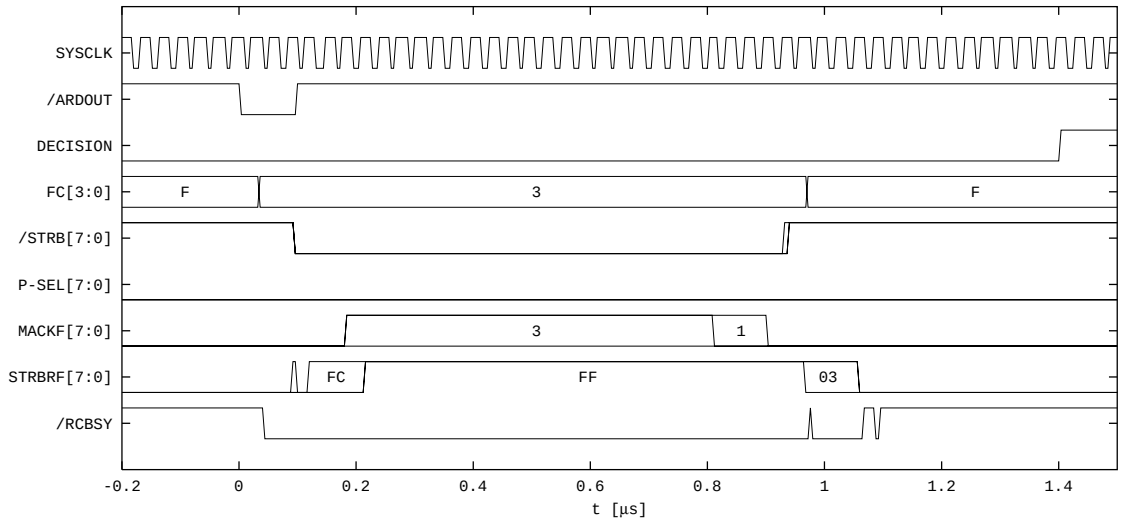


Abbildung 5.9: ADELETE-Zyklus aus Sicht des Readout-Controllers. Ein kompletter Datensatz wird in den angeschlossenen Frontends gelöscht. Die dafür benötigte Zeit ist deutlich kürzer als die Auslesezeit in Abbildung 5.8.

Der ADELETE-Zyklus wird wie der ARDOUT-Zyklus durch eine fallende Flanke auf `/ARDOUT` eingeleitet, jedoch muß `DECISION` auf low gesetzt sein. Die Auslesesteuerung gibt den Funktionskode ADEL auf den Leitungen `FC[3:0]` aus und setzt die Strobeleitungen `/STRB[7:0]` aller Ports auf low. Für den weiteren Verlauf des Zyklus sind nur die `MACKF[7:0]`-Leitungen entscheidend, die Signale `STRBRF[7:0]` werden ebenso wie `P-SEL[7:0]` im ADELETE-Zyklus nicht benutzt.

Der Timing-FPGA wartet nun solange, bis mindestens eines der `MACKFx` gesetzt ist und beendet den Zyklus erst, wenn alle `MACKF[7:0]` wieder zurückgesetzt sind. Somit bestimmt das Frontend mit den meisten zu löschenden Daten die Länge des Zyklus; in unserem Beispiel ist dies ein Modul an Port 0: `MACKF0` wird als letztes zurückgesetzt. Auch bedingt

der Ablauf des Zyklus, daß für Tests immer mindestens ein Frontend-Modul angeschlossen sein muß; fehlt dies, so geht der Readout-Controller beim ersten ADELETE-Zyklus in eine Endlosschleife, die nur durch einen Software-Reset beendet werden kann.

Wie üblich, wird mit Beginn des Zyklus `/RCBSY` auf low gesetzt und erst mit dessen Ende wieder zurückgesetzt.

Im Timingdiagramm ist auch deutlich zu sehen, daß durch das daisy chaining des Strobesignals in den Frontendketten an Port 0 und 1 eine nicht unerhebliche Verzögerung entsteht. Eine mögliche Lösung dieses Problems wird in Abschnitt 8.2.1 vorgestellt.

5.6 Die LVL2-Pipe

Die Struktur der LVL2-Pipe ist im Blockschaltbild 5.1 bereits zu erkennen: zwei Bänke von jeweils 128 kBx32 bilden den eigentlichen Speicher der LVL2-Pipe; jeweils eine Bank kann über Umschalter an den externen Datenbus (und damit über VME an die CPU) und den internen Datenbus (also an die Auslesesteuerung) geschalten werden. Gesteuert werden alle Vorgänge in der LVL2-Pipe über den Memory-FPGA, in dem alle Funktionen zu Verwaltung der LVL2-Pipe untergebracht sind.

5.6.1 Anforderungen

In der LVL2-Pipe werden — wie in Abschnitt 2.3 beschrieben — komplette Datensätze eines Readout-Controllers gespeichert. Diese Datensätze (im folgenden auch Subevents genannt) müssen von der CPU nach der Entscheidung des LVL3-Triggers entweder zur Speichereinheit transportiert oder verworfen werden.

Somit ergeben sich folgende Anforderungen an die LVL2-Pipe des Readout-Controllers, aus denen die komplette innere Struktur der LVL2-Pipe abgeleitet wird:

1. Es muß gleichzeitig mit der Auslese von bereits gespeicherten Subevents das Einschreiben neuer Subevents möglich sein. Auch muß eine Pufferfunktion gewährleistet werden, da der Auslese- und Bewertvorgang prinzipiell langsamer abläuft als das Einschreiben durch die in Hardware implementierte Auslesesteuerung.
2. Die Daten eines Subevents müssen schnell über den VME-Bus transportiert werden. Das wiederum heißt, daß das verwendete Speicherformat an die Eigenschaften des VME-Bus angepaßt sein sollte, um eine möglichst effektive Nutzung der Transferzyklen zu ermöglichen.
3. Die transportierten Subevents sollen ohne weitere Behandlung durch die CPU bereits alle Verwaltungsinformationen beinhalten, so daß kein Software-Overhead durch das Hinzufügen neuer Verwaltungsinformationen entsteht.

Im folgenden wird die Struktur der LVL2-Pipe genauer erläutert; insbesondere wird dabei auf die Anpassung der Hardware an die Anforderungen der DAQ-Software eingegangen.

5.6.1.1 Bank-Struktur

Die Forderung, gleichzeitig und völlig asynchron Lese- und Schreibzugriffe auf den Speicher der LVL2-Pipe zu ermöglichen, läßt sich prinzipiell durch zwei Designoptionen erfüllen: die Verwendung von speziellen RAM-Bausteinen oder die Nutzung von konventionellen Speicherbausteinen in einer Bankstruktur.

Für die Implementierung der RICH-LVL2-Pipe wurde aus praktischen Gründen die zweite Option genutzt: der Einsatz von Spezialspeicher ist nur bis zu einer bestimmten Speichergröße sinnvoll, bei der im RICH vorgesehenen Größe hätte dies eine Aufteilung in mehrere Adreßbereiche notwendig gemacht und damit die Ansteuerung verkompliziert. Außerdem ist eine voll asynchrone Benutzung nur bei wenigen Bausteinen wirklich möglich.

Die LVL2-Pipe wurde deshalb in einer Bankstruktur aufgebaut: eine Bank besteht aus vier Bausteinen des Typs CY7C109 [7] und beinhaltet 128 kB bei einer Breite von 32 bit. Zur Umschaltung der Adreß- und Datenleitungen werden spezielle Bausteine vom Typ 74CBT16212 [14] benutzt: diese "Cross Bar Switches" genannten Bausteine erlauben es, zwei Busse bidirektional in beliebiger Konfiguration auf zwei andere Busse durchzuschalten. Im Readout-Controller werden damit die Adreß- und Datenleitungen einer Bank entweder an den externen Daten- und Adreßbus (also die CPU) oder an das interne Bussystem (also den Timing- und Memory-FPGA) geschalten. Es ist somit immer eine Bank mit der VME-Seite, die andere mit der Readout-Controllerseite verbunden; über ein Steuersignal werden die Bänke komplett ausgetauscht.

Diese Bank-Struktur erlaubt es, daß einerseits der Readout-Controller vollkommen autonom Subevents in seiner Bank speichert, während andererseits die CPU die Daten in ihrer Bank bearbeiten kann. Bedingt durch die gewählte Größe der Speicherbänke ist außerdem gewährleistet, daß genügend Puffer für die Auslesesteuerung zur Verfügung steht: in der LVL2-Pipe des RICH Readout-Controllers können die Subevents eines durchschnittlichen Spills gespeichert werden, ohne daß eine Bankumschaltung notwendig wird.

Zur Umschaltung der Bänke steht der DAQ-Software das Signal `SWRQ` zur Verfügung: nach dem Abarbeiten einer Bank schickt die CPU über den Interface-FPGA einen kurzen Puls auf `SWRQ`, der vom Timing-FPGA registriert und in das statische Signal `MSEL` umgesetzt wird. Dieses Signal wird vom Memory-FPGA benutzt, um die Bänke umzuschalten. Einzelheiten zur Bankumschaltung findet man in Abschnitt 5.6.2.2.

5.6.1.2 Page-Struktur

Zur schnellen Auslese der LVL2-Pipe wird beim RICH Readout-Controller der sogenannte Blocktransfer-Modus benutzt. Bei normalen Speicherzugriffen legt die CPU für jedes zu lesende Datenwort eine Adresse an, beim Blocktransfer wird lediglich die Startadresse am Anfang der Übertragung vorgegeben. Die angesprochene VME-Bus-Karte zählt nun jeden Lesezugriff mit und erhöht dementsprechend einen internen Adreßzähler, der das jeweils nächste zu lesende Datenwort anwählt.

Eine Besonderheit des VME-Busses ist der leistungsfähige Blocktransfer-Modus (BLT32): hierbei werden jeweils bis zu 256Byte übertragen, zusätzlich können mehrere Startadressen vorgegeben werden.

Um diesen Modus auch mit CPUs älterer Bauart nutzen zu können, ist es notwendig, jedes Subevent auf einer in dieses Schema passenden Adresse abzulegen. Dazu wird der Speicher in Seiten (sogenannte Pages) unterteilt: jede Page beginnt auf einer durch 256 teilbaren Adresse. Sollte ein Subevent nicht ganz in eine Page passen, so wird die nächste Page ebenfalls benutzt, das nächste Subevent jedoch auf den Anfang der darauf"—folgenden Page gelegt. Die ermöglicht es der CPU, nach der Bewertung der einzelnen Subevents eine Tabelle mit den auszulesenden Pages zu erstellen und die zugeordnete Bank der LVL2-Pipe in einem einzigen Blocktransfer auszulesen.

Eine Besonderheit bei der Adressierung des Speichers ist jedoch zu beachten: Vom Memory-FPGA aus wird je ein 32 bit-Wort als Einheit adressiert; um auf 256 Byte-Grenzen zu kommen, werden intern auf dem Readout-Controller folglich Blöcke von 64 Langworten als Page bezeichnet. Die Adressierung auf dem VME-Bus ist davon unterschiedlich: hier werden Byte-Adressen benutzt; bei der Auslese ist es aber nicht möglich, ein einzelnes Byte innerhalb eines Langworts zu adressieren. Folglich werden hier die Adressen jeweils um vier erhöht, um auf das nächste Langwort zu kommen.

Mit dem gewählten Modus BLT32 sind mit dem auf dem Readout-Controller verwendeten VME-Chipsatz vom Typ CY7C960 Datenraten von bis zu 40 MB/s möglich, Tests mit der im Experiment verwendeten PowerPC-CPU ergaben Werte um 24 MB/s. Diese Datenraten sollten zur Auslese der LVL2-Pipe bei der vom Design her angestrebten Triggeruntersetzung von 1:100 ausreichend hoch sein; bei fehlender Untersetzung des LVL1-Triggers begrenzt nicht die VME-Busgeschwindigkeit, sondern das Speichermedium selbst die Übertragungsbandbreite.

5.6.1.3 Subevent-Header

In der Planungsphase der HADES-Datenaufnahme wurden intensive Überlegungen zur Struktur der zu speichernden Daten vorgenommen. Die Ergebnisse dieser Überlegungen sind in [19] festgehalten und für alle Auslesesysteme im HADES-Experiment verbindlich.

Für die LVL2-Pipe wurde festgelegt, daß jedes Subevent mit bestimmten Informationen über Inhalt und Aufbau versehen werden muß. Diese Informationen werden im sogenannten Subevent-Header gespeichert und sind jedem Subevent voranzustellen. Diese Aufgabe muß von der Ausleseelektronik ohne Eingriffe der CPU wahrgenommen werden.

Subevent							
Subevent-Header				Daten			
subEvtSize	subEvtDecoding	subEvtId	subEvtTrigNr	Data 0	Data 1	...	Data n

Tabelle 5.3: Struktur eines Subevents. Die vier Worte im Subevent-Header sind stets 32 bit breit, die Breite der Datenworte bestimmt sich aus dem subEvtDecoding.

Der Aufbau eines Subevents ist dabei wie in Tabelle 5.3 angegeben, der Subevent-Header selbst besteht aus vier 32 bit-Worten mit folgender Bedeutung:

- subEvtSize: gibt die Länge des gesamten Subevents inklusive Header in Bytes an
- subEvtDecoding: enthält Informationen darüber, in welchem Format das Subevent abgespeichert ist (Byte-Order, Datenwort-Länge)
- subEvtId: gibt den Entstehungsort des Subevents an
- subEvtTrigNr: hier wird das zu den Daten gehörige Triggertag abgespeichert

Aus diesem Header kann die DAQ-Software sämtliche Informationen gewinnen, die zur Dekodierung der enthaltenen Daten notwendig sind. Einzelheiten dazu findet man in [19].

5.6.2 Abläufe im Memory-FPGA

Die Hauptaufgabe des Memory-FPGAs ist die Verwaltung der LVL2-Pipe. Dazu gehören neben der Erstellung von Subevents auch Dinge wie die Handhabung der Bankumschaltung, Generierung von Schreib-/Lese-Signalen für die dem VME-Bus zugeteilte Bank und auch die Ansteuerung des Mapping-Speichers. Ebenso müssen für die auslesende CPU Statusinformationen über den Zustand der gerade beschriebenen Bank bereitgestellt werden. Im folgenden wird auf die dazu notwendigen Abläufe eingegangen.

5.6.2.1 Die Formatierung von Subevents

Die Erstellung von kompletten Subevents in der LVL2-Pipe erfolgt in enger Zusammenarbeit mit dem Timing-FPGA. Bei der Beschreibung des dazu notwendigen Ablaufes wird auf Abschnitt 5.5.3 verwiesen, in dem der Ablauf des ARDOUT-Zyklus beschrieben wurde.

Der Memory-FPGA verwaltet die dem Readout-Controller aktuell zugewiesene Speicherbank. Dabei wird jeweils ein 32 bit-Wort direkt über die Adreßleitungen `MA[16:0]` angesprochen. Eine der oben angesprochenen Pages umfaßt die unteren Adreßbits `MA[5:0]`, folglich können pro Bank 2 048 Pages angesprochen werden. Ähnlich wie bei der Verwaltung der LVL1-Pipe im Frontend-Modul muß darauf geachtet werden, einen Speicherüberlauf zu vermeiden.

Die Erstellung eines korrekt formatierten Subevents wird im Memory-FPGA von einer Ablaufsteuerung übernommen, die sowohl den Adreßzähler für die Speicherbank ansteuert als auch diverse Kontrollsignale generiert. Da vor jedes Subevent ein passender Subevent-Header eingefügt werden muß, die dafür benötigten Informationen jedoch erst am Ende der Übertragung aus der LVL1-Pipe feststehen, müssen am Anfang der ersten Page vier Langworte ausgespart werden.

Die Ablaufsteuerung wird mit der steigenden Flanke von `EVTBEG` gestartet; zu diesem Zeitpunkt steht der Adreßzähler bereits mit einem Offset von vier auf einer Page-Grenze.

Mit der steigenden Flanke von EVTBEG wird die Startadresse des Subevents in einem Register zwischengespeichert; diese Information wird zum späteren Hinzufügen des Subevent-Headers benötigt. Auch wird /RCBSY auf low gezogen, um einen laufenden Zyklus zu signalisieren.

Jedes aus der LVL1-Pipe übertragene Datenwort wird vom Timing-FPGA mit einem Puls auf /MWR an den Memory-FPGA gemeldet, der aus /MWR ein Schreibsignal für die Speicherbank (/WR_A bzw. /WR_B) erzeugt und nach dem Einschreibevorgang den Adreßzähler um eins erhöht. Gleichzeitig wird die Anzahl der bereits geschriebenen Worte mitgezählt, wobei auch hier die vier Langworte des Subevent-Headers berücksichtigt werden.

Am Ende des ARDOUT-Zyklus wird vom Timing-FPGA das Triggertag des Subevents übertragen. Dazu wird EVTEND auf high gelegt und das Triggertag durch einen /MWR-Puls in ein Register des Memory-FPGAs übernommen. Mit der fallenden Flanke von EVTEND startet der zweite Teil der Ablaufsteuerung, der die Erstellung des Subevent-Headers übernimmt.

Mit dem Ende der Übertragung stehen die beiden letzten für den Subevent-Header benötigten Daten fest:

- die Gesamtlänge des Subevents ergibt sich aus der Gesamtzahl der /MWR-Pulse
- das Triggertag wurde intern bereits gespeichert

Die Ablaufsteuerung blendet nun statt des Adreßzählers die zu Beginn gespeicherte Startadresse auf MA[16:2] ein, über MA[1:0] werden die vier Langworte des Subevent-Headers einzeln angewählt und mit den entsprechenden Werten beschrieben. Die Werte für subEvtDecoding bzw. subEvtId stammen dabei aus Registern im Memory-FPGA, die von der DAQ-Software während der Initialisierungsphase beschrieben wurden.

Nach dem Schreiben des Subevent-Headers setzt die Ablaufsteuerung den Adreßzähler auf die nächste Page-Grenze hoch; dabei wird ebenfalls anhand des Memreg (siehe Abschnitt B.3) geprüft, ob in der Bank noch genug Platz vorhanden ist. Ist dies nicht der Fall, so wird /RCBSY festgehalten und intern ein Signal (MEMFULL) für die Bankumschaltungs-Logik gesetzt. Während dieser Überprüfung wird ebenfalls der aktuelle Füllstand der Bank in einem Register zwischengespeichert; diese Information wird nach der erfolgten Bankumschaltung von der DAQ-Software benötigt.

Die Grenze, ab der eine Bank als "voll" angesehen wird, ist ebenfalls über ein Register konfigurierbar. Um Speicherüberläufe durch einen nicht korrekt initialisierten Readout-Controller und damit mögliche Datenverluste zu vermeiden, wird dieses Register nach dem Laden der FPGAs mit dem Wert Null initialisiert. Der Memory-FPGA überwacht den Wert dieses Registers; solange kein Wert größer Null geladen wurde, wird ebenfalls /RCBSY auf low gehalten und damit die Datenaufnahme verhindert.

Nach dem normalen Ablauf eines Zyklus wird /RCBSY wieder auf high gesetzt; der Memory-FPGA ist wieder bereit, ein neues Subevent in die LVL2-Pipe zu schreiben.

5.6.2.2 Bank-Umschaltung

Wie oben bereits angesprochen, erhält die CPU stets eine komplette Bank der LVL2-Pipe, um die darin enthaltenen Subevents gemäß des LVL3-Triggers weiterzutransportieren. Nach der Abarbeitung der in der Bank stehenden Subevents fordert die DAQ-Software über `SWRQ` im Interface-FPGA eine Bankumschaltung an. Das `SWRQ`-Signal wird im Timing-FPGA in ein statisches Signal namens `MSEL` umgewandelt, das anschließend an den Memory-FPGA weitergeführt wird.

Bankumschaltungen sind jedoch nur zu bestimmten Zeitpunkten und unter bestimmten Umständen erlaubt. Deshalb wurden zwei Möglichkeiten vorgesehen, der wartenden CPU die erfolgte Umschaltung mitzuteilen: die CPU fragt periodisch ein Register im Memory-FPGA ab, in dem jeweils in einem Bit die angeforderte und die aktuell geschaltete Bank gespiegelt werden. Nach der Umschaltung sind beide Bits gleich; die CPU kann also auf die Daten zugreifen. Im selben Register ist ebenfalls die Anzahl der benutzen Pages gespeichert, die CPU muß also nur einen VME-Zugriff machen, um diese beiden Informationen zu erhalten. Diese Betriebsart wird auch polling-Verfahren genannt.

Optional dazu wurde die Möglichkeit vorgesehen, eine erfolgte Bankumschaltung per Interrupt an die CPU zu melden. Dazu muß im Memory-FPGA die Interrupt-Erzeugung freigegeben und der Interrupt-Vektor geladen werden. Der Ablauf bei der Bankumschaltung ändert sich nun:

Die CPU fordert wie bisher per SWRQ eine Bankumschaltung an. Anschließend wird für kurze Zeit im oben genannten Register überprüft, ob die Umschaltung schon stattgefunden hat. Ist dies nicht der Fall, so schaltet die CPU die Interrupt-Erzeugung an und wartet auf die Benachrichtigung per Interrupt. In dieser Wartezeit können andere anstehende Aufgaben erledigt werden, ohne jedoch den Umschaltzeitpunkt zu verpassen.

Eine Umschaltung der Bänke ist nur erlaubt, wenn:

- momentan kein ARDOUT-Zyklus läuft
- **und** mindestens ein komplettes Subevent in der dem Readout-Controller zugewiesenen Bank steht
- **und** bereits eine Bankumschaltung angefordert, aber nicht durchgeführt wurde.

Dadurch wird verhindert, daß während einer Spillpause laufend leere Bänke umgeschaltet werden, was in der CPU zu unnötigem Overhead führen würde.

5.6.2.3 Ansteuerung des Mapping-Speichers

Neben den oben ausgeführten Aufgaben im RUN-Modus steuert der Memory-FPGA im /STOP-Modus auch den Mapping-Speicher an. Über `/RD_DPR` und `/WR_DPR` kann das zum Mappen benutzte Dualported Memory gelesen bzw. beschrieben werden. Im RUN-Modus

wird allerdings der CPU-Zugriff (obwohl technisch möglich) auf das Dualported Memory gesperrt; durch unbeabsichtigtes Beschreiben des Mapping-Speichers könnten unbemerkt falsche Datensätze erzeugt werden. Die Adreßlage der Mapping-Speichers ist in Abschnitt B.1 genau beschrieben.

Kapitel 6

Die RICH DTU

Wie in Abschnitt 3.3 beschrieben, erfolgt die Anbindung der Ausleseelektronik an das Triggersystem über die DTU. Im folgenden werden die in der RICH DTU bisher implementierten Funktionen beschrieben; Grundkenntnisse über den Triggerbus werden hierbei allerdings vorausgesetzt. Eine verständliche Beschreibung des Triggersystems findet man in [21], eine allgemeine Beschreibung der CTU bzw. DTU in [22].

Die Elektronik der DTU wurde an der Universität Gießen von Herrn E. Lins entwickelt, der auch die Anbindung an den Triggerbus als Ausgangsbasis für das RICH DTU-Design bereitstellte. Die eigentliche Ansteuerung der Readout-Controller wurde wegen der dazu benötigten speziellen Kenntnisse über die Ausleseelektronik in München implementiert. Im Zuge der Entwicklung wurden auch notwendige Änderungen am Basis-Design vorgenommen, die in Folge in das allgemeine Basis-Design für alle HADES DTUs übernommen wurden.

6.1 Aufgaben

Als Schnittstelle zwischen CTU und Ausleseelektronik kommen der RICH DTU hauptsächlich folgende Aufgaben zu:

- **Anbindung an den Triggerbus:** Die DTU muß sowohl die von der CTU einlaufenden Triggerbefehle entgegennehmen als auch die von den Readout-Controllern kommenden Statusmeldungen an die CTU weiterleiten. Die dazu notwendige bidirektionale Anbindung der DTU an die CTU erfolgt über den LVL1- und LVL2-Triggerbus mittels differentieller Signale.
- **Umsetzung der Triggerbusbefehle in Auslesezyklen:** Da nicht alle am Triggerbus gesendeten Kommandos für den RICH verbindlich sind, muß die DTU die einlaufenden Befehle nach ihrer Relevanz beurteilen und nur die tatsächlich benötigten Befehle in entsprechende Auslesezyklen umsetzen.

- **Busy-Handling:** Als Schnittstelle zur CTU muß die DTU die von der Ausleseelektronik erzeugten Busy-Signale in entspechende Triggerbussignale umsetzen und an die CTU weiterleiten, um das Einlaufen von Triggern in der Totzeit des Gesamtsystems zu verhindern.

Darüberhinaus sind in der RICH DTU etliche Status- und Überwachungsregister enthalten, die es erlauben, während des Betriebs bzw. nach einer Fehlersituation genaue Informationen über den Zustand der RICH Ausleseelektronik zu erhalten. Eine Aufstellung über die in der RICH DTU enthaltenen Register findet man in Anhang B.6.

6.2 Blockschaltbild

Die RICH DTU ist intern in mehrere, zeitlich von einander entkoppelte Stufen aufgeteilt (siehe Abbildung 6.1):

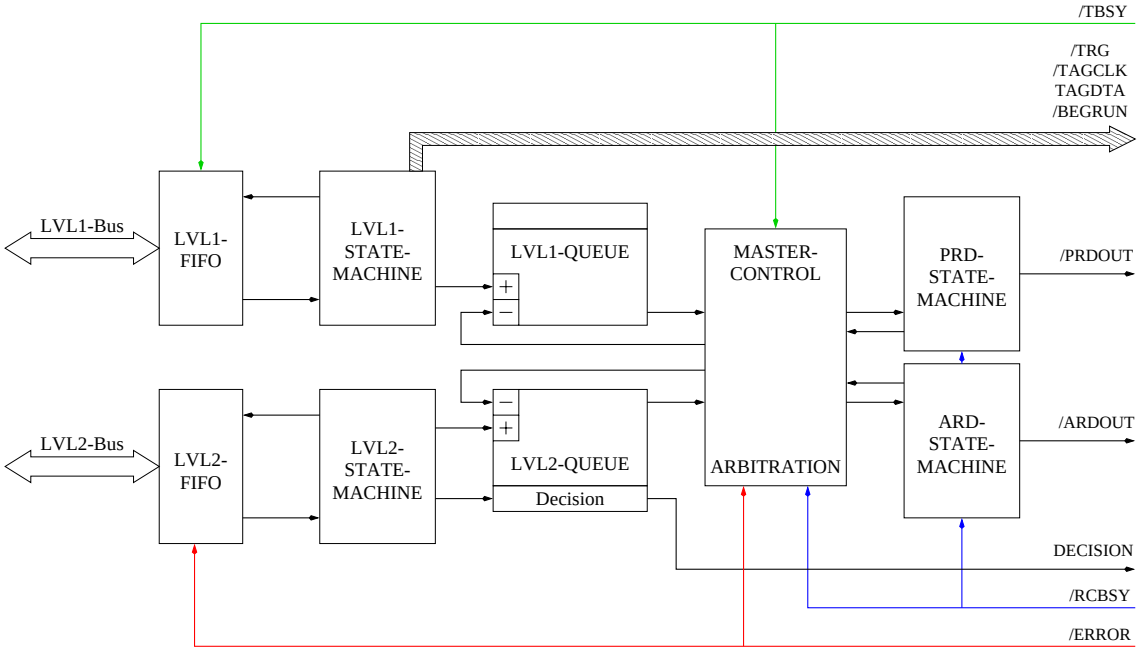


Abbildung 6.1: Blockschaltbild der RICH DTU. Die Steuerung der Ausleseelektronik ist in vier zeitlich entkoppelte Stufen unterteilt.

Die erste Stufe bilden dabei der LVL1- bzw. LVL2-Fifo, die das Triggerbusprotokoll handhaben und alle einlaufenden Befehle in einem 16-stufigen FIFO speichern. Diese für alle DTUs identische Anbindung stellt für die nächste Stufe Triggerbefehl und zugehöriges Triggertag bereit; außerdem können Statussignale (wie Busy und Error) über die Anbindung an die CTU geleitet werden.

Die zweite Stufe wird von zwei Ablaufsteuerungen (der LVL1- bzw. LVL2-Statemachine) sowie zwei Warteschlangen (LVL1- bzw. LVL2-Queue) gebildet. Hier ist der erste Unterschied zwischen LVL1- und LVL2-Triggerbus zu erkennen: Da die in den Frontends

implementierte Analog-Signalverarbeitung strenge Anforderungen an das Timing des T/H-Signal stellt (siehe Abschnitt 4.3), muß jeder einlaufende NormalTrigger-Befehl auf dem LVL1-Bus in einer genau definierten Zeit in ein Triggersignal für die Frontends umgesetzt werden. Die dazu gehörigen Auslesebefehle zur Übertragung der Ansprechmuster werden in der LVL1-Queue gespeichert. Alle für den RICH nicht interessanten Trigger-Befehle werden sofort nach ihrer Übertragung in den LVL1-Fifo wieder gelöscht.

Der LVL2-Fifo hingegen wird anders behandelt: für den RICH bestimmte Auslesebefehle werden samt dem DECISION-Bit in der LVL2-Queue in einem FIFO gespeichert. Es werden im Gegensatz zur LVL1-Statemachine keine Signale für den Readout-Controller direkt generiert.

Als dritte Stufe in der RICH DTU dient die MasterControl. Von diesem Funktionsblock wird anhand der in der LVL1- und LVL2-Queue anstehenden Befehlen arbitriert, welcher Auslesezyklus als nächster auszuführen ist.

Die vierte Stufe wird von den zur Ansteuerung des Readout-Controllern benötigen Ablaufsteuerungen gebildet, der PRD- bzw. ARD-Statemachine. Diese Ablaufsteuerungen übernehmen auch die Kommunikation mit der Ausleseelektronik, um den korrekten Ablauf der Zyklen zu überwachen.

Hinter der vierten Stufe steht als ausführende Instanz die eigentliche Ausleseelektronik, also die Kombination von Readout-Controller und Frontend-Modulen.

Alle vier Stufen werden untereinander über die DTU-interne 10 MHz-Clock aufeinander einsynchronisiert.

6.3 Die Umsetzung der Triggerbus-Befehle

Auf dem Triggerbus sind momentan sechs unterschiedliche Triggertypen definiert, die das von den bisher installierten Detektoren geforderte Mindestmaß abdecken; eine Erweiterung ist in Zukunft aber denkbar. Im folgenden wird erläutert, welche Triggerbefehle von der RICH DTU benötigt werden und wie die Umsetzung in die in Abschnitt 5.4 beschriebenen Auslesezyklen stattfindet. Auch wird auf die unterschiedliche Behandlung von LVL1- und LVL2-Triggerbefehlen eingegangen sowie die Arbitrierung zur Auslesesteuerung beschrieben.

6.3.1 Unterstützte Triggerbus-Befehle

Eine Übertragung auf dem Triggerbus besteht nach [21] stets aus einem 4 bit-Triggerkode, der den Typ des Triggers angibt, sowie dem Triggertag, das zentral von der CTU generiert wird. Das Triggertag ist eine frei wählbare 8 bit-Zahl; momentan wird von einigen Detektorsystemen noch eine aufsteigende Nummerierung des Tags erwartet, jedoch können in Zukunft auch bestimmte Tag-Nummern zur Kennzeichnung von Subevents mit bestimmten Eigenschaften benutzt werden.

Die CTU kann momentan die in Tabelle 6.1 aufgelisteten Triggertypen erzeugen; zu beachten ist, daß der Triggerbefehl in den unteren Bits `TC[2:0]` kodiert ist, wohingegen das oberste Bit `TC3` die von der PMU gelieferte Decision darstellt: ein gesetztes `TC3`-Bit bedeutet dabei eine negative Decision, ein nicht gesetztes hingegen eine positive. Dies gilt nur für LVL2-Triggerbefehle, bei LVL1-Triggerbefehlen ist das `TC3`-Bit stets nicht gesetzt.

Triggerkode		Triggername	RICH DTU	
`TC[2:0]`	dezimal		LVL1	LVL2
`000`	0	invalid	–	–
`001`	1	normal event	•	•
`010`	2	begin run	•	–
`011`	3	end run	•	–
`100`	4	calibration	–	–
`101`	5	spill on	–	–
`110`	6	spill off	–	–
`111`	7	invalid	–	–

Tabelle 6.1: Triggerkodes der CTU. Die vom RICH genutzten Triggerbefehle sind gekennzeichnet (•).

LVL1-Befehle des RICH

Insgesamt werden drei LVL1-Befehle im aktuellen RICH DTU-Design benutzt:

- **Triggerkode 2 (BeginRun):** Dieser Triggerkode übermittelt der RICH DTU das für den nächsten Trigger gültige Triggertag. Bei entsprechend konfigurierter DTU wird ebenfalls ein Reset aller angeschlossenen Readout-Controller sowie der Frontends über einen BEGRUN-Zyklus durchgeführt (diese Aktion ist in Abschnitt 5.5.1 beschrieben). Darüberhinaus benötigt die RICH DTU diesen Triggerkode, um die in Abschnitt 6.3.4 erläuterte Arbitrierung korrekt durchführen zu können.
- **Triggerkode 3 (EndRun):** Dieser Triggerkode wird ebenfalls für die Auslese-Arbitrierung benötigt.
- **Triggerkode 1 (NormalEvent):** Aus diesem Triggerkode wird sowohl das Triggersignal für die Datenaufnahme in den Frontend-Modulen als auch der Auslesebefehl für den PRDOUT-Zyklus gewonnen. Es ist der einzige Triggerkode, der beim RICH Daten in den Frontend-Modulen erzeugt.

Eine Nutzung der Triggerkodes 5 (SpillOn) bzw. 6 (SpillOff) wurde ebenfalls für die Auslese-Arbitrierung in Erwägung gezogen; da jedoch nicht garantiert werden kann, daß diese Triggerkodes auch erzeugt werden [18], wurde auf die Verwendung dieser optionalen Triggerkodes verzichtet.

LVL2-Befehle des RICH

Am LVL2-Bus wird nur ein Triggerkode genutzt: der Triggerkode 1 (NormalEvent) wird zusammen mit dem obersten Bit `TC3` des Triggerkodes zur Erzeugung des ARDOUT- bzw. ADELETE-Zyklus herangezogen. Eine zusätzliche Benutzung des Triggertags ist nicht notwendig, da durch die Pipeline-Struktur in der LVL1-Pipe sowie der IPU bzw. PMU automatisch die richtige Reihenfolge an Auslesebefehlen für die in der LVL1-Pipe stehenden Datensätze erzeugt wird.

6.3.2 LVL1-Auslesesteuerung

Die Behandlung der am LVL1-Triggerbus einlaufenden Befehle erfolgt in der RICH DTU (siehe Abbildung 6.1) durch die LVL1-Statemachine. Wie oben bereits angedeutet, benötigen die RICH Frontends ein exaktes Timing für die Analog-Signalverarbeitung. Dieses Signal wird direkt aus den Steuersignalen des LVL1-Triggerbusses abgeleitet, um ein definiertes Timing zu gewährleisten. Die Nutzung der LVL1-Fifo-Signale ist nicht möglich, da durch eine nicht genau bekannte Anzahl von Logikstufen eine zu hohe und nicht berechenbare Verzögerung entstehen würde. Auch würde eine jederzeit mögliche Änderung am DTU-Basis-Design das Timing für die RICH Frontends empfindlich stören und Neuanpassungen notwendig machen. Für das jetzige RICH DTU-Design ergibt sich eine kaum noch minimierbare Durchlaufverzögerung zwischen Eingang des Triggerbefehls und Erzeugung des `/TRG`-Signals von etwa 50 ns.

Zur Erzeugung des `/TRG`-Signals werden die Signale `TD[3:0]` und `T` des LVL1-Triggerbusses überwacht; der Verlauf einer Übertragung am LVL1-Triggerbus ist zum besseren Verständnis in Abbildung 6.2 dargestellt. Die CTU legt als erstes den Triggerkode auf den Leitun-

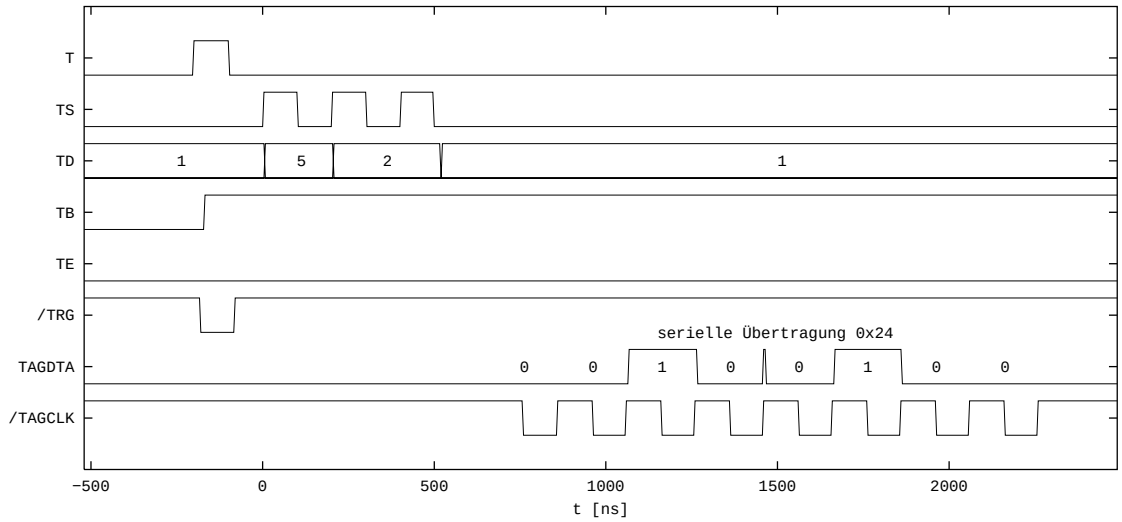


Abbildung 6.2: Timing auf dem Triggerbus. Das `/TRG`-Signal wird ohne Delay-Line erzeugt. Der Triggerbefehl liefert das Tag `0x25`, seriell wird jedoch das Tag `0x24` übertragen, das mit dem vorhergegangenen Triggerbefehl in die DTU übertragen wurde.

gen TD[3:0] an und erzeugt das Strobesignal T. Zur sicheren Erkennung des Triggerkodes muß der T-Puls um einige Nanosekunden verzögert werden; dies geschieht auf der CTU mit Hilfe einer Analog-Delayline. Die LVL1-Statemachine reagiert auf die oben beschriebene Bedingung am Triggerbus und leitet aus T das /TRG-Signal ab. Zur Feinabstimmung des Timings ist eine digital einstellbare Analog-Delayline für die RICH DTU vorgesehen, die das so erzeugte /TRG-Signal nochmals verzögern kann. Diese Feinabstimmung erfolgt über ein DTU-Register und kann ohne direkten Zugang zur DTU vorgenommen werden.

Die weiteren Strobesignale auf der TS-Leitung sowie das in zwei Nibbles übertragene Triggertag sind für die /TRG-Erzeugung ohne Bedeutung, jedoch wird die komplette Übertragung des Triggerbefehls abgewartet, ehe das bereits in der DTU stehende Triggertag über TAGDTA und /TAGCLK seriell an die Frontends übertragen wird.

Die Verwendung des Triggertags ist folgendermaßen geregelt: Jeder Triggerbefehl überträgt das Triggertag für den **nächsten** Triggerbefehl. Dies ist notwendig, um den langsameren Auslesesystemen die Möglichkeit zu geben, das bereits in der jeweiligen DTU stehende Triggertag als erstes Datenwort zu nutzen; beim RICH hingegen wird das Tag als letztes Wort eines Datensatzes in die LVL1-Pipe geschrieben (siehe Abschnitt 4.3). Die Übertragung des Tags erfolgt beim RICH in so kurzer Zeit, daß auch die Nutzung des mit dem jeweiligen Triggerbefehl übertragenen Triggertags möglich wäre.

Zur korrekten Nutzung der Triggertags ist es deshalb unbedingt erforderlich, beim Start der Datenaufnahme als ersten Triggerbefehl BeginRun in der CTU zu erzeugen. Mit diesem Triggerbefehl landet das erste Triggertag in der DTU; jeder weitere einlaufende Triggerbefehl wird im LVL1-Fifo gespeichert und bewirkt automatisch die Löschung des im FIFO vor ihm stehenden Triggertags. Somit steht immer das richtige Tag zur Übertragung an die Frontends bereit.

Jeder einlaufende NormalEvent-Triggerbefehl wird außerdem in der LVL1-Queue notiert, die Ausführung der Datenübertragung an die IPU muß wegen der Nutzung der gleichen Datenpfade wie bei der LVL1-Auslese jedoch über eine Arbitrierung in der MasterControl erledigt werden. Die eigentliche Datenaufnahme und die Musterübertragung sind zeitlich entkoppelt; die IPU kann nicht mit der sofortigen Übertragung der aktuell entstandenen Daten rechnen. Jede ausgeführte Datenübertragung zur IPU wird von der MasterControl an die LVL1-Queue gemeldet; somit ist immer die aktuell noch anstehende Zahl an Datenübertragungen in der LVL1-Queue zu finden.

6.3.3 LVL2-Auslesesteuerung

Die Bearbeitung des LVL2-Fifos erfolgt durch die LVL2-Statemachine. Nach jeder vollständigen Übertragung auf dem LVL2-Bus wird der Triggerkode überprüft: jeder Befehl, der nicht den Triggerkode 1 (NormalEvent) besitzt, wird ohne Einflußnahme der MasterControl gelöscht. NormalEvent-Trigger werden samt dem DECISION-Bit in der LVL2-Queue gespeichert. Die MasterControl arbitriert die in der LVL1- und LVL2-Queue anstehenden Auslesezyklen und arbeitet diese nach feststehenden Regeln nacheinander ab: stehen in beiden Queues Auslesezyklen an, so werden diese abwechselnd gestartet, wobei bevorzugt ein PRDOUT gestartet wird.

6.3.4 Auslese-Arbitrierung

In der RICH DTU müssen die über beide Triggerbusse angeforderten Auslesezyklen aufeinander abgestimmt werden, da beide Auslesezyklen (also die Übertragung von Daten zur IPU und die Verarbeitung der Daten in der LVL1-Pipe) im Frontend dieselben Busse benutzen. Als zusätzliche Schwierigkeit muß auch die in Abschnitt 4.3 bereits beschriebene Empfindlichkeit des Analogteils im Frontend auf Störsignale berücksichtigt werden: bei einem Auslesezyklus im Frontend wird im allgemeinen genug Störstrahlung erzeugt, um die Ladungsinformationen von den Pads zu verfälschen. Somit muß bei der Auslese auch auf die Datenaufnahme-Zyklen selbst geachtet werden.

Die dazu notwendige Arbitrierung findet in der MasterControl statt: dabei werden sowohl die Anzahl der in der LVL1-Queue wartenden Datenübertragungen als auch eventuell anstehende LVL1-Auslesebefehle im LVL2-Fifo beachtet. Zusätzlich erhält die MasterControl über `/TBSY` Informationen darüber, ob gerade im Frontend ein Datenaufnahme-Zyklus läuft. Zusätzlich muß das von der IPU erzeugt Busy-Signal `/IPUBUSY` beachtet werden, das das Starten von PRDOUT-Zyklen verhindert. In der RICH DTU können verschiedene Arbitrierungsmodelle über ein Register ausgewählt werden; hier wird nur das Standardverfahren (Modus: `NORMAL`, siehe Abschnitt B.6) beschrieben.

In die Bewertung dieser Informationen geht auch entscheidend ein, ob BeginRun- oder EndRun-Triggerbefehle auf dem LVL1-Triggerbus eingelaufen sind.

Insgesamt ergeben sich somit drei Szenarien, auf die die RICH DTU unterschiedlich reagiert:

Vor einem DAQ-Run:

Diese Situation entspricht einem frisch initialisierten Datenaufnahme-System. Von der CTU ist noch kein BeginRun-Befehl eingetroffen, die RICH DTU befindet sich im Wartezustand, in dem keinerlei Auslesezyklen gestartet werden. Die LVL1- und LVL2-Fifos sind jedoch bereits aufnahmebereit; jetzt einlaufende Triggerbefehle werden entsprechend behandelt: auf dem LVL1-Bus einlaufende Triggerbefehle lösen zwar Datenaufnahme-Zyklen in den Frontends aus, die zugehörigen Datenübertragungs-Anforderungen werden jedoch nicht bearbeitet und füllen die LVL1-Queue auf. Nach 15 NormalEvent-Befehlen wird die RICH DTU über den LVL1-Bus LVL1BSY melden und keine Befehle mehr entgegennehmen. Diese Situation wird beispielsweise bei einer fehlerhaften Konfiguration der CTU durch die DAQ-Software entstehen; durch das LVL1BSY werden keine Trigger mehr akzeptiert — es kommt zu einem Deadlock, der nur durch Neuinitialisieren des Systems aufgehoben werden kann.

Während eines DAQ-Runs:

Ein RUN wird mittels eines BeginRun-Befehls auf dem LVL1-Triggerbus eingeleitet. Die RICH DTU speichert das so übertragene Triggertag für den nächsten Triggerbefehl und

schaltet die MasterControl scharf. Jetzt einlaufende NormalEvent-Befehle auf beiden Triggerbussen werden wie gewohnt bearbeitet, wobei die Auslesearbitrierung unter Normalbedingungen nach folgenden Regeln arbeitet:

1. Es muß mindestens ein abgeschlossener Datenaufnahme-Zyklus in den Frontends stattgefunden haben.
2. Um einen neuen Auslesezyklus starten zu können, muß `/RCBSY` high sein.
3. Auslesezyklen dürfen nur gestartet werden, wenn `/TBSY` low ist.
4. Ein gesetztes `/IPUBSY` verhindert das Neustarten von IPU-Datenübertragungen.

Regel 1 verhindert, daß ein Auslesezyklus einen unvollständigen Datensatz aus den Frontends ausliest. Dies würde mit Sicherheit zu einem Fehler führen und die FIFO-Struktur der LVL1-Pipe stören.

Regel 2 verhindert neben einer Überlagerung von mehreren Zyklen auf dem Readout-Controller auch eine Überfüllung der LVL2-Pipe (siehe auch Abschnitt 5.6.2.1).

Regel 3 minimiert Störeinstrahlungen durch die Auslesezyklen. `/TBSY` geht erst nach der eigentlichen analogen Signalverarbeitung auf low; da zu diesem Zeitpunkt ohnehin im Frontend die Datenaufnahme-Ablaufsteuerung aktiv ist, kann ohne Gefahr für das Analogsignal auch die Auslese der FIFOs stattfinden.

Regel 4 ermöglicht das Übergehen von PRDOUT-Zyklen und bevorzugt ARDOUT- bzw. ADELETE-Zyklen, wenn die IPU noch beschäftigt ist, aber der Readout-Controller bereits den Zyklus beendet hat.

Beim Vollaufen einer der beiden Warteschlangen wird Regel 3 außer Kraft gesetzt, um die Triggerbusse nicht zu blockieren.

Nach einem DAQ-Run:

Das Ende eines DAQ-Runs wird durch einen EndRun-Trigger auf dem LVL1-Triggerbus angezeigt. In der RICH DTU wird dies intern gespeichert und zwei der Regeln außer Kraft gesetzt: Regel 1 wird unnötig, da bereits fertige Datensätze in den FIFOs stehen und keine neuen Trigger mehr einlaufen können. Regel 3 entfällt aus demselben Grund; alleine Regel 2 und Regel 4 bestimmen jetzt die Arbitrierung der Auslesezyklen.

Nach einem EndRun-Trigger werden folglich die noch in der LVL1-Pipe stehenden Datensätze abgearbeitet und an die DAQ weitergeleitet.

6.4 Busy-Handling

Im HADES-Triggerkonzept [21] wird die im Schnitt maximal erlaubte Totzeit nach der Ausgabe eines Triggerbefehls durch die CTU auf $10\,\mu$s verbindlich festgelegt. Systeme, die

eine längere Totzeit benötigen, müssen ein passendes Busy-Signal erzeugen und über die Triggerbusse an die CTU melden.

Da sich — bedingt durch Laufzeiten auf dem langen Triggerbus sowie die interne Verarbeitungszeiten in DTU und Ausleseelektronik — nach der Ausgabe jedes Befehls durch die CTU ein kurzer Zeitraum ergibt, in dem von den Auslesesystemen prinzipiell kein Busy gemeldet werden kann, erzeugen CTU und DTU einstellbar über Register jeweils eine Mindesttotzeit, die von den langsameren Systemen verlängert werden kann. Dazu werden die Busy-Signale der Ausleseelektronik in die jeweilige DTU eingeschleift.

Bei der RICH DTU können hierzu zwei Busy-Signale genutzt werden:

1. `/TBSY` gelangt von den Frontends über den Readout-Controller in die DTU und zeigt einen gerade laufenden Datenaufnahme-Zyklus in den Frontends an. Es wird in der RICH DTU in das LVL1BSY-Signal eingeblendet.

2. `/RCBSY` wird im Readout-Controller vom Timing- bzw. Memory-FPGA erzeugt und zeigt eine gerade laufende LVL1-Auslese an. `/RCBSY` wird **nicht** in das LVL2BSY-Signal der RICH DTU eingeschleift. Ein laufender LVL1-Auslesezyklus kann — bedingt durch die Steuerung der LVL2-Statemachine — nicht von einem neu einlaufenden LVL2-Busbefehl gestört werden; hier übernimmt die Busy-Generierung der LVL2-Queue in der RICH DTU die Busy-Erzeugung.

Im folgenden werden die bei der RICH DTU möglichen Busy-Bedingungen (getrennt nach LVL1- und LVL2BSY) beschrieben.

6.4.1 LVL1BSY

Die RICH DTU erzeugt in den folgenden Fällen ein LVL1BSY (TriggerBusy auf dem LVL1-Triggerbus):

- In den Frontends läuft gerade ein Datenaufnahmezyklus. LVL1BSY wird für ca. 10μs gesetzt und anschließend wieder zurückgenommen.
- In der LVL1-Pipe eines Frontends ist eine FIFOFULL-Bedingung aufgetreten. Diese Situation wird im allgemeinen durch die im LVL2-Fifo einlaufenden Auslesebefehle wieder behoben; ist dies nicht der Fall, so könnte ein Hänger in der PMU bzw. IPU das Eintreffen weiterer Auslesebefehle verhindern.
- Die LVL1-Queue in der RICH DTU ist vollgelaufen. Die RICH DTU erkennt diesen Zustand selbständig und arbitriert die Auslesezyklen entsprechend.
- Die FIFO-Verwaltung eines Frontend-Moduls arbeitet fehlerhaft und erzeugt trotz einer leeren LVL1-Pipe eine FIFOFULL-Bedingung. Diese Situation kann nur durch Neustart der DAQ-Software behoben werden und sollte im Normalbetrieb nicht auftreten.

Ein länger anstehendes LVL1BSY deutet auf eine Fehlersituation hin und kann durch Auslesen von Status-Registern in Readout-Controller bzw. RICH DTU genau zurückverfolgt werden.

6.4.2 LVL2BSY

Das LVL2BSY-Signal (ReadoutBusy auf dem LVL2-Triggerbus) wird alleine von der RICH DTU generiert:

- Die LVL2-Queue in der RICH DTU ist vollgelaufen. Die RICH DTU erkennt diesen Zustand selbständig und arbitriert die Auslesezyklen entsprechend.
- Die LVL2-Pipe eines Readout-Controllers meldet MEMFULL. In diesem Falle bleibt `/RCBSY` low, die ARD-Statemachine kann ihren internen Ablauf nicht beenden. In diesem Fall muß von der DAQ-Software eine Bankumschaltung angefordert werden, um die Busy-Bedingung zu beseitigen. Nach erfolgter Bankumschaltung läuft das System normal weiter.

Auch hier deutet ein lange anstehendes LVL2BSY auf eine Fehlerbedingung hin.

Kapitel 7

Inbetriebnahme und Tests

Das Hardware-Design des RICH Readout-Controllers wurde 1997 von Herrn D. Maier in Zusammenarbeit mit Herrn Ch. Theurer begonnen [27]. Die Designanforderungen standen zu diesem Zeitpunkt bereits fest, jedoch noch nicht die konkrete Umsetzung in ein funktionsfähiges Hardware-Design. Als Resultat entstand ein Prototyp (RC97), an dem erste Erfahrungen gewonnen wurden. Allerdings war nur die Auslese im /STOP-Modus über die CPU möglich, da die automatische Auslesesteuerung sowie ein FPGA-Design für den Memory-FPGA komplett fehlten. Die bei der Inbetriebnahme der Speichereinheit des RC97 gewonnenen Erfahrungen wurden in den zweiten Prototyp (RC98) eingearbeitet, der zu Anfang dieser Diplomarbeit als ungetestete Baugruppe zur Verfügung stand. Die verwendete Bezeichnung "Prototyp" bezieht sich im folgenden alleine auf den RC98. Das endgültige Serienmodul (RC99) entstand wiederum in enger Zusammenarbeit mit der Elektronikabteilung und konnte erstmals im Rahmen einer Teststrahlzeit im September 1999 an der GSI eingehenden Tests unterzogen werden.

7.1 Der Prototyp

Für die Weiterentwicklung des Prototypen bis zur Serienreife waren ausführliche Tests der Hardware notwendig. Um diese Tests durchführen zu können und möglichst viele Informationen über Fehler und Verbesserungsmöglichkeiten zu gewinnen, wurde ein kompletter Teststand aufgebaut. Das vorgesehenen VME-Crate samt zugehörigen Netzteil wurde ebenfalls in Betrieb genommen, um bereits bei der Entwicklung in der später im Experiment benutzten Umgebung arbeiten zu können.

Als Meßaufbau wurde ein Minimalsystem aus CTU, DTU, Readout-Controller und einigen Frontend-Modulen aufgebaut; die Triggererzeugung erfolgte dabei über einen Pulsgenerator mit Einzelpuls-Fähigkeit. Zur Analyse der im Readout-Controller ablaufenden Funktionen wurde eine mechanisch stabile Verbindungsmöglichkeit zum Logikanalyzer realisiert: Stiftleisten, die an der Frontplatte des Prototyps angebracht sind, erlauben jederzeit einfachen Zugang zu den relevanten internen Signalen des Readout-Controllers. Die Signale

selbst werden durch in die Durchkontaktierungen der Platine gelöteten Fädeldraht an die Stiftleisten geführt.

Zur Meßsignalerfassung kam ein 96-Kanal-Logikanalyzer vom Typ HP 1661 CS zum Einsatz, mit dem auch alle in dieser Arbeit abgebildeten Timing-Diagramme aufgenommen wurden. Die zeitliche Auf"—lösung von 4 ns ermöglichte dabei präzise Messungen vieler Signale; speziell bei den letzten Tests wurde zusätzlich ein Digital-Oszilloskop vom Typ Tektronics 2440 benutzt, da etliche Probleme analoger Natur mit dem Logikanalyzer nicht beobachtbar waren. Besonders die Signale an den Flachbandkabeln mußten einzeln überprüft werden; sowohl Pegel als auch Signalform wurden mit dem Oszilloskop gemessen und durch entsprechende Abschlußwiderstände an die Kabel angepaßt. Auch die Signalformen direkt hinter den CMOS-Treibern mußten auf Überschwinger geprüft werden, an vielen Stellen wurden Serienwiderstände zur Schwingungsdämpfung eingesetzt.

7.1.1 Durchgeführte Tests

Erwartungsgemäß wurde ein Großteil der Arbeit in die Inbetriebnahme und Abstimmung der automatischen Auslesesteuerung im Timing-FPGA investiert. Die Philosophie, erst die gesamte Umgebung dieses zentralen Bausteins zu testen und in Betrieb zu nehmen, erwies sich dabei als vorteilhaft. Bei der Fehlersuche mußten zwei Fehlerquellen beachtet werden: Schaltungsfehler (wie produktionsbedingte Lötbrücken) sowie fehlerhafte FPGA-Designs, die zu Fehlverhalten führen. Die Aufteilung in mehrere Testphasen erleichterte es in vielen Fällen, die Fehlerursache eindeutig festzustellen und zu beheben.

Zu Beginn der Tests wurde das VME-Businterface in Betrieb genommen. Hierbei konnte auf die Erfahrungen mit dem RC97 zurückgegriffen und viele mögliche Fehlerquellen bereits im Vorfeld vermieden werden. Wichtigstes Testkriterium war in dieser Phase, die FPGAs ohne Probleme initialisieren zu können.

Nächster logischer Schritt war die Überprüfung des Designs im Interface-FPGA. In Anbetracht der für die Debugging-Phase notwendigen Zugriffsmöglichkeiten wurde bei der Programmierung dieses FPGAs nicht auf eine konventionelle Registerstruktur zurückgegriffen. Stattdessen wurde ein spezielles Register entworfen, das neben den üblichen Schreib-/Lesevorgängen auch ein Zurücklesen der an den Pins wirklich anliegenden Signalpegel erlaubt. Eine detaillierte Beschreibung der Register des Interface-FPGAs ist in Anhang B.2 enthalten. Die Nutzung dieser Register erlaubte es, durch Setzen und Rücklesen von Signalpegeln etliche Lötbrücken aufzuspüren und zu beseitigen. In dieser Testphase wurde ein Großteil der Signalwege auf dem Readout-Controller auf ihre Funktionsfähigkeit getestet; gleichzeitig entstand in Zusammenarbeit mit dem Entwickler der DAQ-Software, Herrn M. Münch, eine Softwarebibliothek, die in transparenter Weise die Nutzung von Readout-Controller-Funktionen erlaubt, ohne jedoch umfangreiche Pflege bei notwendigen Hardware-Änderungen zu benötigen.

In der nächsten Testphase wurde die LVL2-Pipe in Betrieb genommen und getestet. Neben der eigentlichen Speicherverwaltung, die die zwei SRAM-Bänke ansteuert, wurde ebenfalls die Ansteuerung des Mapping-Speichers implementiert. Als zentrales Merkmal des Readout-Controllers wurde die LVL2-Subevent-Formatierung wie auch das Paging ausgiebig

getestet; ebenfalls wurde der zur schnellen Auslese unbedingt notwendige BLT32-Modus sowie die Interrupt-Generierung in Testreihen am RC98-Modul überprüft.

Als Ergebnis dieser Testphasen konnte nun auf die automatische Ablaufsteuerung gewechselt werden, da hierfür eine ausgiebig getestete Umgebung zur Verfügung stand. Die Inbetriebnahme und Tests der automatischen Ablaufsteuerung erfolgte in enger Zusammenarbeit mit Herrn D. Maier; gleichzeitig wurden in dieser Phase notwendige Erweiterungen am DTU-Design vorgenommen und das Frontend-FPGA-Design grundlegend neu gestaltet.

Als Folge der Tests wurden etliche Hardware-Modifikationen am RC98 wie auch an den FE-Bus-Platinen und den Frontends vorgenommen und auf die gewünschte Funktionsweise hin überprüft.

7.1.2 Erfolgreicher Betrieb im Labor

Am Ende der oben beschriebenen Arbeiten am Prototyp konnte ein Zustand erreicht werden, in dem das Minimalsystem bestehend aus einem Readout-Controller und acht angeschlossenen Frontend-Modulen auch bei hohen Triggerraten an der CTU stabil funktionierte. Da sich zu diesem Zeitpunkt die Detektormodule noch teilweise im Fertigungsstadium befanden, war der Anschluß von Modulgruppen zur Testauslese an den Detektor nicht möglich. Im Testbetrieb zeigte ein Aufbau mit den aktiven Busplatinen an zwei Ports ebenfalls stabiles Verhalten; infolgedessen wurde die Produktion der neuen Busplatinen parallel zur Serienfertigung des neuen Readout-Controllers aufgenommen.

Neben den Untersuchungen im Münchener Detektorlabor wurden ebenfalls Tests mit den anderen Auslesegruppen der HADES-Kollaboration durchgeführt. Diese sogenannten “full system”-Tests an der GSI Darmstadt wurden einerseits genutzt, um die Schnittstelle zur IPU zu testen, andererseits wurde das Zusammenspiel von mehreren DTUs verschiedener Detektorsysteme an einem gemeinsamen Triggerbus getestet. Auch diese Tests fanden noch ohne Detektoren statt. Im Rahmen eines solchen Testlaufs konnte ein gemeinsamer Triggerbus mit RICH und Preshower gebildet werden. Das Datenaufnahme-System wurde von der DAQ-Software kontrolliert und konnte erfolgreich Subevents von RICH und Preshower über das ATM-Netzwerk transportieren und in einem gemeinsamen “Event Builder” zu Ereignissen zusammenfassen. Das System lief dabei stabil, die Primärtriggerrate an der CTU konnte dank der voll implementierten Busy-Erzeugung auf über 200 kHz erhöht werden.

7.1.3 Durchgeführte Verbesserungen

Im Zuge der Hardwaretests sowie der FPGA-Design-Entwicklung wurden diverse Verbesserungen an der Hardware des Readout-Controllers vorgenommen. Diese Änderungen sind im RC99-Design bereits implementiert; eine genaue Kenntnis ist notwendig, wenn zu Testzwecken der Prototyp RC98 in Betrieb genommen werden soll.

- Für den Timing- und Memory-FPGA wurden auf dem RC99 FPGAs vom Typ XC4010E vorgesehen, auf dem RC98 hingegen XC4005E. Beide Typen unterscheiden sich nur in der internen Kapazität, nicht jedoch in Gehäusebauform oder Pinbelegung. Es ist deshalb prinzipiell möglich, FPGA-Designs für den RC99 auch für die RC98-Hardware zu übersetzen, solange die maximale Kapazität des XC4005E nicht überschritten wird.

- Der RC99 benutzt die in Abschnitt 5.4.1 beschriebenen invertierten Signale, der RC98 hingegen die alten nicht invertierten. Zusätzlich ist am RC98 an Port 0 und 1 nur der Betrieb der alten gepatchten passiven backplanes möglich, an Port 2 bis 7 nur der der passiven. Die neuen aktiven backplanes können nur am RC99 betrieben werden; für Testzwecke bedeutet dies jedoch keine starke Einschränkung.

- Die Readout-Controller-Seite des Mappingspeichers ist im neuen Design im /STOP-Modus deaktiviert, die normale Datenumleitung aktiviert. Somit können Speicherzyklen über die Steuerung des Interface-FPGA im Memory-FPGA nur ohne Mapping simuliert werden. Diese Änderung war notwendig, da andernfalls durch einen DPR-internen Konflikt das vollständige Beschreiben des Mapping-Speichers nicht möglich gewesen wäre.

- Am Interface-FPGA wurden Lötpads für fünf frei benutzbare Signale hinzugefügt. Die Steuerung dieser Leitungen erfolgt über das in Abschnitt B.2 beschriebene GIOP-Register, allerdings ist derzeit noch keine feste Anwendung für diese Signale definiert. Möglich wäre beispielsweise die Ansteuerung eines Anzeigemoduls mit Status-LEDs oder die Implementierung einer I^2C-Schnittstelle zu slow control-Aufgaben.

7.2 Die Kontrollsoftware

Bei der oben beschriebenen schrittweisen Implementierung des vollen Funktionsumfangs mußte gleichzeitig auch Software zur Ansteuerung der einzelnen Readout-Funktionen geschrieben werden. Als Ergebnis dieser Bemühungen entstand das Programm `racectrl`, das sämtliche zur Initialisierung und Konfiguration des RICH Auslesesystems notwendigen Vorgänge beinhaltet. Das Initialisieren der FPGAs, die Konfiguration von Readout-Controller und Frontends sowie die Abfrage von Statusmeldungen sind über einfache Kommandozeilen-Parameter möglich. `racectrl` entstand aus mehreren Testprogrammen mit jeweils klar abgegrenztem Funktionsumfang, die mit Fortschreiten der Arbeit jedoch zu unflexibel in der Benutzung waren. Deshalb wurden alle bislang benötigten Funktionsaufrufe in einem einzigen Programm `racectrl` zusammengefaßt, das dann um die später zusätzlich erforderliche Funktionalität erweitert wurde. Genauere Informationen zu `racectrl` enthält die Man-Page auf den VME-CPUs (`man racectrl`); ebenso ist es möglich, zu allen integrierten Befehlen eine kurze Beschreibung mittels `racectrl -h` abzurufen.

Die Entwicklung der eigentlichen Datenaufnahmesoftware wurde von Herrn M. Münch

durchgeführt. Für die dazu notwendigen Hardware-Zugriffe wurde eine einheitliche Schnittstelle in Form einer Library[1] geschaffen. Diese Art der Ansteuerung bietet mehrere Vorteile:

1. Dem Programmierer werden Funktionen mit genau bekanntem Funktionsumfang zur Anwendung bereitgestellt. Eine tiefere Kenntnis der zur Durchführung der Funktion selbst notwendigen Hardware-Zugriffe ist nicht notwendig; die Software-Entwicklung wird somit von der eigentlichen Hardware-Entwicklung entkoppelt.
2. Änderungen in den FPGA-Designs müssen nicht mehr dezentral in vielen Programmen beachtet werden, es genügt die Wartung eines Programmteils, der die Umsetzung von high-level- in low-level-Funktionen vornimmt. Für die RICH-Auslese bestimmte Programme lassen sich bei konformer Nutzung dieser Schnittstelle alleine durch Neukompilieren ohne Änderungen im Quelltext an neue Hardware-Gegebenheiten anpassen.
3. Die Wartung der Schnittstelle wird vom Hardware-Entwickler selbst übernommen, der die optimale Nutzung der Hardware durch die entsprechenden Routinen gewährleisten kann.

Beim RICH wird als Schnittstelle zur Hardware die Header-Datei `rc.h` benutzt: Diese Datei beinhaltet neben den Register-Offsets und Magic Words sämtliche für den Betrieb notwendigen Funktionen, die jeweils eine abstrakte Funktion in die entsprechenden Hardware-Zugriffe umsetzen.

Ein Beispiel hierfür ist das Schreiben der StatusId (genaue Beschreibung in Anhang B.3):

```
static void Rc_writeStatusid(Rc * my, unsigned char value)
{
    unsigned short savedStatusId;
    /* StatusID Register retten */
    savedStatusId = LVme_getW(my->lvme, statusid);
    /* Neue ID auf low byte einblenden */
    savedStatusId = ( savedStatusId & 0xFF00 ) | ( value & 0x00FF );
    /* neues Register zurueckschreiben */
    LVme_setW(my->lvme, statusid, savedStatusId);
}
```

Die DAQ-Software nutzt diese Funktion zum Konfigurieren der Hardware, die über die Basisadresse des Boards angesprochen wird; die Funktion selber kümmert sich um die zum Beschreiben des Registers notwendigen Aktionen. In diesem Falle werden nur die unteren acht Bits des Registers verändert, die oberen jedoch nicht. Die VME-Bus-Zugriffe werden über die bei HADES standardisierten Funktionen von `lvme.h` abgewickelt. Auch sieht man, daß sich die DAQ-Software nicht um die Basisadressen der einzelnen Register kümmern muß; diese werden zentral in `rc.h` vorgegeben und in den dort benutzten Funktionen als Symbol (hier: `statusid`) benutzt.

[1] Sammlung von thematisch zusammengehörigen Unterprogrammen

In Hinblick auf die durch die slow control zu realisierende Kontrolle der HADES-Ausleseelektronik wird `racectrl` nur über Kommandozeilen-Optionen gesteuert und kann in Skript-Dateien verwendet werden. Eine interaktive Benutzer-Oberfläche ist unnötig, da diese Funktionalität ebenfalls über die slow control-Steuersoftware EPICS realisiert wird. Nach einer erfolgreichen Testphase von EPICS kann durch Integration der Library in EPICS zugunsten einer höheren Geschwindigkeit auch von Skript-Dateien abgegangen werden.

Für den RICH existiert bereits eine Skript-Lösung, die es ermöglicht, durch Angabe von nur zwei Parametern komplette Detektorsektoren zu initialisieren und zu konfigurieren. Dies funktionierte während der Teststrahlzeit im September 1999 zuverlässig; die Inbetriebnahme des Systems konnte so auch von ungeschultem Personal vorgenommen werden.

Für die weitere Hardware-Entwicklung und Systemtests wurde neben dem oben erwähnten Kontrollprogramm auch noch ein Debug-Tool names `racedbg` geschrieben. Hier sind neben den in `racectrl` implementierten Funktionen hauptsächlich Routinen untergebracht, die spezielle Tests der Hardware ermöglichen. `racedbg` ist vorrangig für Hardware-Entwickler gedacht und sollte nur von entsprechenden Personen genutzt werden. Der Einsatz von `racedbg` in Skripten wird **nicht** empfohlen.

7.3 Das Serien-Modul

Als Ergebnis der weitgehend positiven Ergebnisse der Prototyp-Tests wurde beschlossen, mit den entsprechenden Verbesserungen am Hardware-Design eine Serie von 14 Readout-Controllern RC99 herzustellen. Diese Serie ist bereits für den Einsatz am HADES-Experiment bestimmt und beinhaltet alle in [12] geforderten Funktionen, die für die Ausleseelektronik verbindlich sind.

7.3.1 Qualitätskontrolle

Bedingt durch die hohe Komplexität und die große Anzahl an hochintegrierten Schaltkreisen der Readout-Controller konnte die Inbetriebnahme der einzelnen Exemplare nur nach eingehender Einzelprüfung vorgenommen werden. Dabei wurde jedes Exemplar mit einer Seriennummer versehen und jeweils ein Prüfprotokoll erstellt, auf dem die durchlaufenen Tests, dabei aufgetretene Fehlfunktionen sowie deren Beseitigung dokumentiert wurden.

Jeder Readout-Controller wurde einer intensiven Sichtprüfung unterzogen; dabei wurden in Einzelfällen herstellungsbedingte Lötfehler entdeckt und beseitigt. Ebenfalls wurde der Innenwiderstand der Baugruppe gemessen, um Schäden bei der anschließenden ersten Inbetriebnahme durch Kurzschlüsse zu vermeiden. Im Rahmen der dann folgenden Funktionstests wurden noch zwei Fehler auf den Platinen entdeckt und beseitigt; hierbei konnten die in der Serienprüfung gewonnenen Erfahrungen über mögliche Fehlerursachen positiv genutzt werden.

Dank dieser systematischen Tests konnten rechtzeitig zur Strahlzeit im September 1999 fünf funktional getestete Readout-Controller bereitgestellt werden; wie gefordert konnten so zwei komplette Sektoren mit Ausleseelektronik bestückt und getestet werden. Die grundlegenden Funktionalitätstests der restlichen neun Module stehen noch an.

7.3.2 Einsatz im Experiment

Als vorrangiges Ziel einer ersten Teststrahlzeit im September 1999 stand die Integration und Inbetriebnahme der Ausleseelektronik für zwei RICH-Sektoren sowie die Datenauslese mittels eines gemeinsamen Triggerbusses im Vordergrund. Vorbereitung und Integration der Elektronik litten unter engen zeitlichen Bedingungen. Der straffe Zeitplan zum Einbau des RICH in den Experimentaufbau erlaubte keine intensiven Tests der Elektronik unter normalen Umständen. Die Inbetriebnahme der Elektronik mußte deshalb während des Detektoreinbaus in den Strahlgang vorgenommen werden und konnte wegen der dabei unvermeidlichen Unterbrechungen nicht effektiv durchgeführt werden.

Dank des modularen Aufbaus der Elektronik konnte die Inbetriebnahme in Einzelschritten vorgenommen werden. Gestartet wurde mit dem Aufbau für einen halben Sektor: ein VME-Crate mit CPU, CTU, DTU und einem Readout-Controller sowie 38 Frontend-Modulen wurde von einem Pulsgenerator als Triggerquelle angesteuert. Bereits in dieser Phase auftretende Probleme mit den Frontend-Modulen und den backplanes verhinderten jedoch die Eingliederung der RICH Ausleseelektronik in den gemeinsamen Triggerbus.

Die bei der Strahlzeit auftretenden Fehler lieferten jedoch entscheidende Erkenntnisse über die Funktionalität der Hardware: es zeigten sich keine prinzipiellen Fehler im Auslesesystem. Als Hauptursache von Fehlern konnte die Kommunikation der Readout-Controller mit den Frontend-Modulen über das Flachbandkabel sowie die backplanes eingekreist werden; hier verursachten Effekte wie das Übersprechen von Signalen, die Fehlanpassung von Leitungsimpedanzen und Spike-Bildung durch Störeinstrahlung Fehlfunktionen, die jedoch stets vom Readout-Controller als solche erkannt und gemeldet wurden. Die Fehlerkontrolle auf dem Readout-Controller konnte so intensiv getestet werden und stellte ihre Funktionstüchtigkeit unter Beweis.

Gegen Ende der Strahlzeit konnte auch ein Test der analogen Signalverarbeitung in den Frontend-Modulen durchgeführt werden. Dabei wurde eine zu geringe Verstärkung im Videomultiplexer sowie eine Fehlanpassung an den dynamischen Bereich des ADCs festgestellt. Die Ursache hierfür konnte in vorher nicht bekannten Änderungen in der verwendeten Serie von GASSIPLEX-Chips lokalisiert werden. Diese Fehlanpassung wurde inzwischen beseitigt.

Im Rahmen der Strahlzeit konnte auch das von Herrn Dr. W. Schön, Herrn Th. Eberl und Fr. L. Fabietti erstellte Softwarepaket zur Datenanalyse zum ersten Mal unter Experimentbedingungen erfolgreich getestet werden. Die Off"—line-Analyse erwies sich dabei als ausgezeichnetes Hilfsmittel zur Fehlersuche.

Durch Verwendung eines speziellen FPGA-Designs für die Frontends wurden jedem Frontend definierte Werte für die einzelnen Kanäle zugewiesen und ausgelesen. Im Off"—line-Display konnte so das korrekte Mapping für die LVL2-Pipe überprüft werden; die von

einem einzelnen Frontend ausgelesene Padfläche erschien im Display wie erwartet als zusammenhängende Fläche.

Weitere Tests wurden zur Erstellung der Schwellendateien für die Frontend-Module durchgeführt. Die Frontends des RICHs sind auf der Analogseite so abgeglichen, daß jeder Kanal einen bestimmten Offset (den sog. "pedestal", der innerhalb jeder GASSIPLEX-Serie variiert) besitzt. Dieser Offset kann über eine einfache Messung bestimmt werden: man mißt bei auf Null gesetzten Schwellen das Signal von jedem Pad des Detektors und kann über die Off"—line-Analyse den Mittelwert $\overline{x}$ und die Standardabweichung σ jedes einzelnen Kanals berechnen. Es gilt

$$\text{Offset}: \qquad \overline{x}_{\text{Kanal}} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_{\text{Kanal, i}}$$

und

$$\text{Standardabweichung}: \qquad \sigma_{\text{Kanal}} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_{\text{Kanal, i}} - \overline{x}_{\text{Kanal}})^2}$$

wobei n die Anzahl der Messungen angibt. Die Sensitivität der Ausleseelektronik auf Rauschen wird durch Setzen einer individuellen Schwelle für jedes Pad festgelegt:

$$\text{Schwelle}: \qquad x_{\text{Schwelle, Kanal}} = \overline{x}_{\text{Kanal}} + s \cdot \sigma_{\text{Kanal}}$$

Mit dem oben beschriebenen Aufbau wurde bei ausgeschalteter Fehlerkontrolle das Rauschen eines halben Sektors aufgenommen und in die Analysesoftware eingelesen. Die dabei benutzten Datensätze waren ausreichend groß, um bei guter Statistik Rückschlüsse auf das Rauschen ziehen zu können. Ein typisches Beispiel für die so aufgenommenen Spektren ist in Abbildung 7.1 zu sehen.

Auf"—fällig sind hier vor allem zwei Besonderheiten:

1. Während über die gesamte ausgelesene Fläche der Mittelwert insgesamt kaum variiert, zeichnet sich links oben eine Fläche aus, in der die Kanäle deutlich überhöhte Werte liefern. Mit Hilfe der Off"—line-Analyse konnten diese Kanäle einem Frontend-Modul zugeordnet werden; eine genaue Betrachtung des betreffenden Moduls zeigte, daß der Analogteil wegen eines defekten Verstärkers falsche Werte lieferte. Beim späteren Aufbau können somit einfach defekte Frontend-Module erkannt und gezielt ausgetauscht werden.

2. Im unteren Bereich des Sektor liefern einige Kanäle sporadisch hohe Werte. Da dieser Bereich des Detektors nahe am Target liegt, könnte dies ein Anzeichen für leicht erhöhte Aktivität, bedingt durch den Strahlbetrieb in der vorhergehenden Nacht sein.

Das Rauschen der Kanäle wurde ebenfalls von der Analyse-Software ermittelt und graphisch dargestellt (siehe Abbildung 7.2). Man erkennt wieder deutlich die ausgelesene

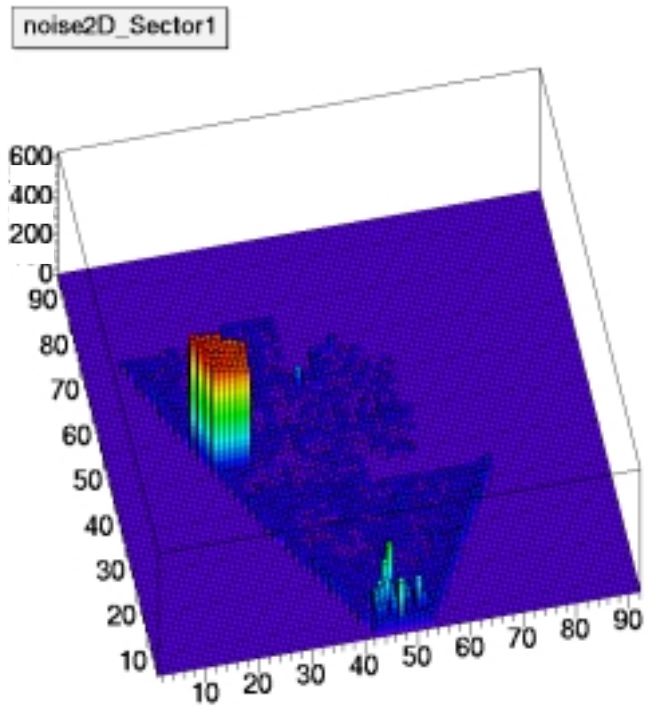


Abbildung 7.1: Off"—line-Analyse der Pedestals. In der x-y-Ebene sind die Padpositionen eingetragen, als z-Koordinate ist der Pedestalwert (Maximalwert: 1023, gemittelt über 100 Datensätze) gewählt. Die typische Form des Photonendetektor-Moduls zeichnet sich deutlich ab.

Fläche des Sektors. Auf"—fällig ist jedoch, daß das Rauschen in den Randbereichen leicht ansteigt. Hier erkennt man den Einfluß der elektrisch von den Pads isolierten Stahlstützstruktur des Detektormoduls; dieser Teil des Detektors ist leitend mit Erde verbunden und reagiert auf Spannungsschwankungen der ebenfalls mit Erde verbundenen Spannungsversorgungen. Bemerkenswert ist, daß sich die Struktur der Padplane im Rauschen abbildet. Die Padplane besteht aus drei Platinenflächen, die die ladungsempfindlichen Pads mit der CsI-Beschichtung tragen. Die elektrische Verbindung zu den kompakten Steckern der Frontends erfolgt dabei über Leiterbahnen auf der Rückseite der Padplane-Platinen. Bedingt durch die Größe der Pixel von $6{,}6 \cdot 6{,}6\,\mathrm{mm}^2$ sind dabei je nach Pad-Position unterschiedliche Leiterbahnlängen notwendig: direkt unter dem Stecker liegende Pads haben sehr kurze, weiter weg liegende Pads lange Leiterbahnen und damit höhere Kapazitäten, die sich in einem höheren Rauschen niederschlagen. Im linken und unteren Teil der Sektorfläche erkennt man die Lage der Stecker durch deutlich niedrigere Rausch-Werte im Vergleich zu den weiter außen liegenden Pads. Die ladungsempfindlichen Eingangskanäle der GASSIPLEX-Chips können das mit der Leiterbahnlänge korrelierte Rauschen noch deutlich trennen; die korrekte Wiedergabe der Steckerpositionen zeigt außerdem, daß das in den Frontends vorgenommene Mapping der Kanäle richtig vorgenommen wird.
Im mittleren Teil der Sektorfläche gehen die Steckerpositionen im Rauschen unter; eine Kontrolle der Schwellendateien für diese Frontends zeigte, daß irrtümlich ein falsches Mapping geladen wurde. Auch in diesem Falle konnte ein Fehler in der Konfiguration graphisch

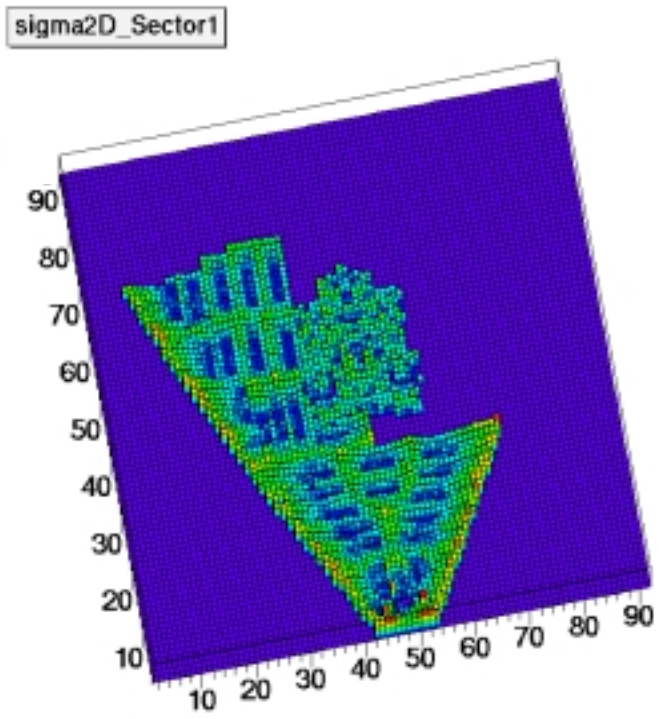


Abbildung 7.2: Rauschwerte für die in Abbildung 7.1 dargestellte Pedestal-Messung. Die Positionen der FE-Modulstecker sind deutlich erkennbar.

festgestellt werden.

7.3.3 Weitere Tests

Als Reaktion auf die bei der Strahlzeit aufgetretenen Fehlfunktionen wurden einige Komponenten des RICH Auslesesystems einer grundlegenden Überarbeitung unterzogen. Am Testaufbau im Labor wurde das Fehlverhalten der Hardware reproduziert und systematisch untersucht. Neben der Anpassung der Flachbandkabel (siehe Abschnitt 7.3.2) mußten die FPGA-Designs für Frontend und RICH DTU überprüft werden. Das Design der RICH DTU wurde von Grund auf neu programmiert und konnte bereits bei mehreren Testreihen seine Stabilität und Zuverlässigkeit unter Beweis stellen. Die Fehlersuche am Frontend ergab, daß oftmals auch nicht ansprechende Kanäle (mit Werten unterhalb der vorgegebenen Schwelle) in der LVL1-Pipe gespeichert wurden. Diese “Geisterkanäle” konnten durch sorgfältige Designoptimierungen beseitigt werden; zusammen mit den analogseitig überarbeiteten Frontends konnte so abschließend ein komplett mit Frontends ausgestatteter Readout-Controller im Labor in Betrieb genommen werden.

Mit diesem Setup wurde das mittlerweile fertiggestellte siebte Photonendetektormodul zur Hälfte ausgelesen. Im Gegensatz zur letzten Strahlzeit konnten Datensätze mit einer Primärtriggerrate von 50 kHz am Frontend aufgenommen werden; die Auslese erfolgte asynchron bei voller Nutzung der LVL1- und LVL2-Pipe. Als beschränkender Faktor erwies

sich hierbei die CTU, die keine höheren Triggerraten zuließ. Die Fehlerkontrolle konnte bei dieser Auslese ebenfalls wieder eingeschaltet werden. Die meisten der auftretenden Fehler erwiesen sich als Einzelbit-Fehler, nicht jedoch als Verletzung der FIFO-Struktur in den Frontends.

Die Analyse der Datenfiles bestätigte die erfolgreiche Anpassung der Analogelektronik. In Abbildung 7.3 sind die Offsets (die als Berechnungsgrundlage für die Schwellen dienen) dargestellt: wie erwartet, liegen die Werte in der Gegend von ca. 80 – 100 (bei einem dynamischen Bereich des ADCs von 1 023); einzelne Gebiete im Detektorsegment weisen aber niedrigere bzw. höhere Werte auf. Dies ist durch Fertigungstoleranzen innerhalb der GASSIPLEX-Serie bedingt: jeweils vier dieser Analog-ASICs werden gemeinsam über einen Widerstand angepasst, wodurch sich in ungünstigen Kombinationen höhere Abweichungen ergeben. Durch sorgfältige Prüfung der ASICs vor dem Bestücken wurde dies aber weitgehend vermieden.

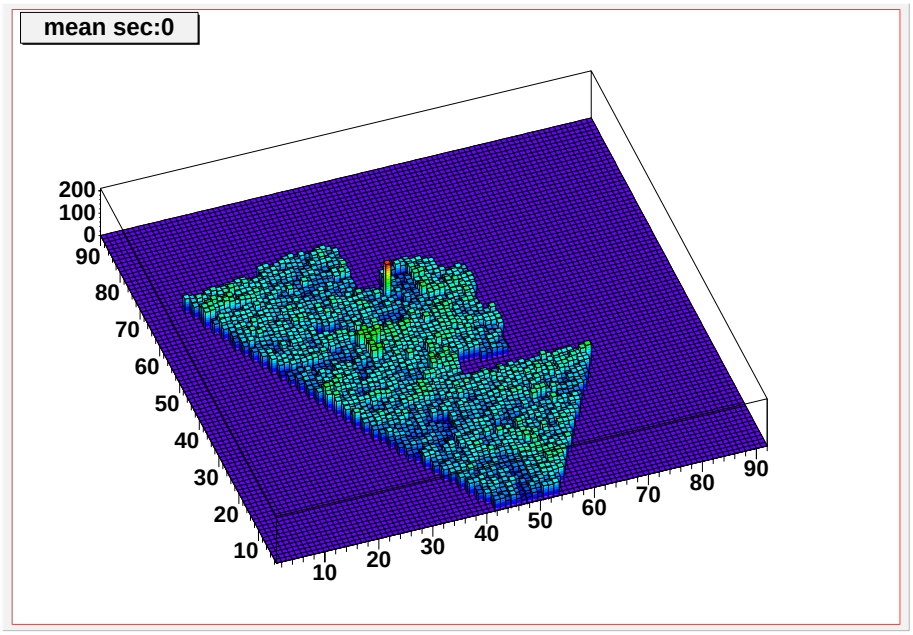


Abbildung 7.3: Offsetwerte der Frontends nach der Anpassung (Berechnungsgrundlage: 1600 Datensätze). Einzelne "Gebiete" (jeweils ein GASSIPLEX) zeigen leichte Abweichungen, im mittleren oberen Bereich des Sektors ist ein nicht funktionierender Kanal anhand seines deutlich überhöhten Wertes identifizierbar.

Die zugehörigen Rauschwerte sind in Abbildung 7.4 zu sehen: die bereits während der Strahlzeit zum ersten Mal beobachtete Struktur der Padplanes konne im Labor reproduziert werden. Allerdings sind die Rauschwerte deutlich niedriger geworden. Je nach Leiterbahnlänge (und damit der Kapazität am Eingang des GASSIPLEX) ergeben sich Werte von $\sigma \approx 1 - 4$, was von der Größenordnung her bereits innerhalb der Designziele bewegt.

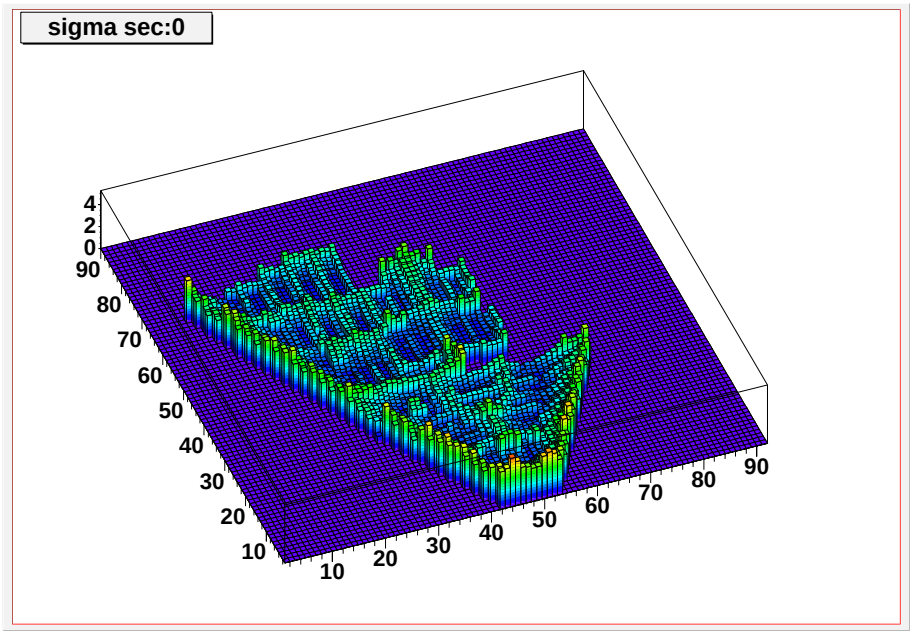


Abbildung 7.4: Rauschwerte der Frontends nach der Anpassung. Die Struktur der Padplane ist deutlich zu erkennen, ebenso das Ansteigen des Rauschens zu den Stützrahmen des Sektors. Die Werte des Rauschens liegen im Rahmen der Designerwartungen.

Kapitel 8

Ausblick

Der Experimentaufbau von HADES hat in den zurückliegenden Monaten deutlich sichtbare Fortschritte gemacht. Für den RICH wurden innerhalb eines Jahres der komplette Radiatortank, das CaF_2-Fenster und die Photonendetektor-Module für alle sechs Sektoren hergestellt. Die Elektronik konnte den Schritt vom Prototypen zur Serienproduktion aller benötigter Baugruppen vollziehen. Der erfolgreiche Test des Magneten, der Einbau von MDC-, TOF- und Preshower-Detektormodulen in zwei Sektoren lassen zusammen mit der Fertigstellung des RICHs mit zwei auslesebereiten Sektoren für das nächste Jahr erste Strahlzeiten mit physikalischen Fragestellungen erwarten.

Allerdings sind bis zu dieser ersten "richtigen" Strahlzeit im Frühjahr 2000 noch einige Aufgaben innerhalb der RICH Ausleseelektronik zu erledigen, die im Rahmen dieser Diplomarbeit nicht mehr abgeschlossen werden konnten. In den folgenden Abschnitten wird getrennt auf die notwendigen Schritte zur Inbetriebnahme und auf mögliche Verbesserungen eingegangen.

8.1 Inbetriebnahme des Gesamtsystems

Die durchgeführten Tests haben ergeben, daß sich das Funktionsprinzip des Auslesesystems in der Praxis bewährt. Fast alle der aufgetretenen Fehler entstehen nicht in der digitalen Steuerung der Auslese, sondern sind Folge der technischen Realisierung, bei der äußere Einflüsse auf die Elektronik zu zusätzlichen Komplikationen führen. Aufgrund der bisher gewonnenen Erkenntnisse sind vor der Nutzung als schnelles Auslesesystem im wesentlichen drei Schritte notwendig:

1. Die Beseitigung der entdeckten Hard- und Software-Fehler.
2. Erweiterung der Ansteuerung auf vier Readout-Controller pro DTU.
3. Integration der IPU in das Auslesesystem.

Das Hauptaugenmerk bei der Fehlerbeseitigung muß auf die Kommunikation zwischen Frontend und Readout-Controller gerichtet werden: diese verläuft über ein High-Density-Flachbandkabel, das in elektrischer Hinsicht die "Achillessehne" des Auslesesystems darstellt. Neben der gerade am Meßplatz am Strahl vorhandenen Störabstrahlung durch die unterschiedlichsten Elektronik-Baugruppen ergeben sich auch durch die elektrischen Eigenschaften des Flachbandkabels einige Probleme: von außen kommende Störsignale koppeln in das Kabel ein, es kommt zu Übersprechen zwischen benachbarten Signalleitungen, die Signalformen selbst werden durch die induktive und kapazitive Last des Kabels verfälscht. Der Einsatz eines Twisted-Pair-Kabels hätte zwar viele dieser Probleme verhindert, war aber wegen der beengten Geometrie alleine aus den Platzanforderungen der dazu benötigten Steckverbinder nicht möglich.

Die Beseitigung dieser Probleme wird neben dem Einsatz von Analogtechnik zur Anpassung von Abschlußwiderständen auch Änderungen in den FPGA-Designs notwendig machen; die Sensitivität der Steuerlogik auf Spikes und Glitches muß reduziert werden. Hierfür sind jedoch noch Messungen im Labor an den jeweiligen Signalleitungen und eine sorgsame Erprobung der daraufhin durchgeführten Änderungen notwendig, bevor das Auslesesystem in den gemeinsamen Triggerbus integriert werden kann.

Die Tests zur Fehlerbeseitigung werden gegenwärtig an einem Minimalsystem im Labor (bestehend aus CTU, DTU, einem Readout-Controller und einer Backplane mit fünf Frontend-Modulen) durchgeführt. Erste Fehler konnten bereits zurückverfolgt und beseitigt werden. Für die Auslese der beiden Detektormodule ist jedoch der Einsatz von vier Readout-Controllern in einem Crate notwendig, die von einer DTU angesteuert werden. Zwar zeigten erste Tests mit einer DTU und zwei Readout-Controllern keine prinzipiellen Hindernisse, jedoch ist mit kleineren Fehlern bei der Erweiterung auf vier Readout-Controller zu rechnen. Hier sind ebenfalls ausgiebige Tests zur Zuverlässigkeit des Systems notwendig, ehe an einen Einsatz unter Strahlzeitbedingungen gedacht werden kann.

Als letzter Schritt ist die Integration der bereits in Test befindlichen IPU in das RICH Auslesesystem zu nennen. Auf Grundlage der in den "full system"-Tests gewonnenen Erkenntnisse wurde das endgültige Timing der IPU-Schnittstelle festgelegt (siehe Abschnitt 5.4.3) und erste Tests zur Datenübertragung vom Readout-Controller zur IPU durchgeführt. Als neue, noch zu testende Komponente kommt jedoch die Einführung von zwei neuen Signalen auf dem Private Bus (`/IPUERROR` und `/IPUBUSY`) hinzu; diese Signale müssen in die Auslese-Arbitrierung der RICH DTU eingebaut werden. Die notwendigen Änderungen am FPGA-Design wurden mittlerweile durchgeführt, der Funktionstest mit der IPU steht dagegen noch aus. Auch dies wird gewisse Zeit in Anspruch nehmen.

Nach Durchführung dieser drei Schritte sollte die Inbetriebnahme des RICH Auslesesystems ohne größere Komplikationen möglich sein. Nicht erwähnt wurde bisher, daß natürlich auch die Auslese- und Steuersoftware weiterentwickelt und an die neuen Gegebenheiten der Hardware angepaßt werden muß. Nach Beseitigung der oben erwähnten Probleme und der Bestückung der restlichen vier Sektoren mit den gerade in Fertigung befindlichen Spiegeln kann der RICH als Ganzes in Betrieb genommen werden.

8.2 Mögliche Verbesserungen

Bei den Tests zur Inbetriebnahme des Readout-Controllers stand die Überprüfung der Funktionsweise im Vordergrund; dennoch wurden auch einige Tests zur Leistungsfähigkeit durchgeführt. Die Zahlen selbst sind nicht aussagekräftig, da die Tests am Prototyp mit nur acht Frontend-Modulen an den modifizierten Ports 0 und 1 durchgeführt wurden. Generell ist jedoch eine Erhöhung des Datendurchsatzes und eine Erweiterung in der Fehlerkontrolle zur weiteren Steigerung des Datendurchsatzes und der Datensicherheit möglich.

8.2.1 Modifikation der Frontend-Verkettung

Das Konzept der Kommunikation zwischen Readout-Controller und Frontend-Modul wurde zu Beginn der Elektronikentwicklung festgelegt, die Grundzüge sind in [16] zu finden. Das Prinzip der dabei verwendeten Verkettung ("daisy chaining") der Datenaufnahme-Module funktionierte damals problemlos, allerdings stellt der Übergang von der in Software implementierten Auslesesteuerung zu einer automatisch ablaufenden und dementsprechend erheblich schnelleren Steuerung durch die Readout-Controller-Hardware deutlich erhöhte Anforderungen an die Hardware. Speziell die Nutzung des Strobesignals zur Datenfreigabe in den einzelnen Modulen ist wegen der Durchschleifung des Signals durch alle Frontends einer Kette problematisch; bedingt durch die Durchlaufzeiten des Strobesignals ergibt sich für das letzte Modul der Kette eine nicht unerhebliche Verzögerung beim Aufschalten der Daten auf den Frontend-Bus. Diese Daten müssen zur korrekten Übernahme in die Port-Einheiten stabil anliegen, das Strobesignal muß deshalb ca. 200 ns lang sein. Die Länge eines Auslesezyklus hängt jedoch empfindlich von der Strobelänge ab, d.h. zur Steigerung des Datendurchsatzes ist eine Verkürzung des Strobepulses unumgänglich.

Dies kann ohne Änderungen in der bestehenden Hardware durch eine Modifikation der Verkettungs-Logik in den FPGAs erreicht werden: Das Strobesignal wird parallel an alle Frontends einer Backplane geführt, die Module erhalten nur noch ein Token, das nach der Abarbeitung eines Datensatzes an das nächste Modul weitergereicht wird. Die Auslesesteuerung benötigt zwischen zwei Strobepulsen ohnehin eine bestimmte Zeit zur Datenverarbeitung; in dieser Zeit kann das zeitunkritische Token weitergereicht werden. Zusätzlicher Vorteil dieser Methode ist, daß jedes Modul durch das Token stets weiß, wann der Frontend-Bus ihm exklusiv zugeteilt ist; der Datenbustreiber des Frontends kann so während der Ausgabe eines Datensatzes durchgehend angeschaltet bleiben. Dadurch entfallen die jetzt vorkommenden Tristate-Phasen auf dem Flachbandkabel, die eine Störeinstreuung begünstigen; der Datentransport sollte also nicht mehr so anfällig auf äußere Störsignale reagieren.

Die Umsetzung dieser Änderung macht jedoch folgende Modifikationen notwendig:

- Umdefinition der Funktionskodes: das oberste Bit des Funktionskodes wird zur Strobeleitung gemacht, der bislang als NOP definierte Funktionskode 15 (siehe Abschnitt 4.4) muß umdefiniert werden.

- Änderungen in der Verkettungs-Logik des Frontends: Der geänderte Auslesezyklus und die Weiterreichung des Tokens über die jetzige `/STRBI`- bzw. `/STRBO`-Leitungen muß implementiert, die Ansteuerung des Datenbustreibers geändert werden.
- Entwurf einer neuen Auslesesteuerung: Das neue daisy chaining-Konzept muß in eine Auslesesteuerung umgesetzt werden, die die neuen Steuer- und Kontrollsignale verarbeitet.

Nach den bisherigen Erfahrungen mit der Auslesesteuerung und dem Datentransport zwischen Frontend und Readout-Controller sollte sich eine Halbierung der Strobepuls-Breite erreichen lassen. Dementsprechend würde sich ein typischer Auslesezyklus etwa um den Faktor 1/3 verkürzen.

8.2.2 Erweiterungen in der Fehlerkontrolle

In seltenen Fällen konnten Umstände beobachtet werden, unter denen die in der Auslesesteuerung integrierte Fehlerkontrolle eine Fehlersituation nicht richtig erkannte. So wurden beispielsweise Datensätze ohne Triggertag fälschlicherweise als gültig anerkannt, der Fehler aber erst im nächsten Auslesezyklus erkannt. Es handelte sich ausnahmslos um Situationen, in denen die Fehlererkennung einen ungültigen Datensatz akzeptierte, dies jedoch erst mit einer Latenz von der Auslesezeit eines Datensatzes an die DTU meldete.

Zur Beseitigung dieser zwar selten auftretenden Fälle muß die Fehlerkontrolle dahingehend erweitert werden, daß der Ablauf des kompletten Protokolls zwischen Readout-Controller und Frontend überwacht wird. Dazu sind jedoch ebenfalls tiefgehende Änderungen an den FPGA-Designs notwendig, so daß die Änderung der Fehlerkontrolle am besten bei der Implementierung des neuen daisy chaining-Prinzips erfolgen sollte.

8.3 Einsatz für andere Systeme

Das hier vorgestellte System eignet sich nicht nur für die Auslese des HADES RICH Detektors. Die Verstärkerbausteine (GASSIPLEX) auf dem Frontend können durch einen integrierten "deconvoluter"-Schaltkreis und durch entsprechende externe Beschaltung auch für Silizium-Detektoren verwendet werden. Damit können neben Gasdetektoren wie Vieldrahtproportionalkammern auch Festkörperzähler wie Silizium μ-Streifenzähler und Pixeldetektoren mit hoher Rate ausgelesen werden. Dies eröffnet ein weites Feld für die Auslese von bildgebenden Detektoren in vielen Anwendungsbereichen, nicht nur in der Kern- und Teilchenphysik.

Berücksichtigt man noch die schnelle Schnittstelle zur Bilderkennungseinheit IPU, die ja außer Ringen auch andere einfache Muster erkennen könnte, so wird das Potential dieser Entwicklung deutlich. Die Skalierbarkeit des Systems erlaubt es, mit moderatem finanziellen Aufwand (ca. 75 DM/Kanal) Detektorsysteme von einigen 100 bis einige

100 000 Kanäle schnell und effizient auszulesen. Für die Off"—line-Bildverarbeitung (Kontrast etc.) ist es dabei sicher von Vorteil, daß die Pulshöhen mit vergleichsweise hoher Auflösung ausgelesen werden.

Anhang A

Glossar

Analogdaten Digitalisierte Pulshöhen von Pads des Photonendetektors.

Arbitrierung Entscheidungsfindung über die Zuteilung von Ressourcen an mehrere gleichberechtigte Partner.

ASIC **A**pplication **S**pecific **I**ntegrated **C**ircuit. Baustein, der speziell für einen bestimmten Anwendungszweck eines Kunden entworfen und in Kleinserie gebaut wurde.

asynchron Betriebsart bei Logiksystemen, bei dem in unterschiedlichen Stufen des Systems Zustandsänderungen zu beliebigen Zeitpunkten erlaubt sind. Laufzeiten zwischen den einzelnen Stufen bestimmen dabei das Verhalten des Systems entscheidend. Asynchrone Systeme sollten in FPGAs vermieden werden, da Laufzeiten schwer kontrollierbar sind und sich das System oftmals instabil verhält.

ATM **A**synchronous **T**ransfer **M**ode. Netzwerktechnologie, die im Gegensatz zu anderen Systemen eine Mindestbandbreite garantiert.

Backplane Leiterplatte, die Grundlage für ein System aus Steckkarten darstellt. Im Zusammenhang mit Ausleseelektronik eine Leiterplatte, die mehrere Frontends zu einer logischen Einheit zusammenfaßt und alle notwendigen Komponenten für das daisy chaining enthält.

Bit kleinste Einheit in der Digitaltechnik. Ein Bit speichert eine Informationseinheit.

Byte Einheit aus 8 Bit.

CLB **C**omplex **L**ogic **B**lock. Grundbaustein der FPGAs von Xilinx, in denen kombinatorische Logikblöcke sowie Speicherelemente (Flipflops) untergebracht sind.

CPLD **C**omplex **P**rogrammable **L**ogic **D**evice. Meist auf EEPROM-Technik basierender Logikbaustein mit anderem internen Aufbau als ein FPGA. Enthält an den Ausgängen Logikelemente, die eine kombinatorische Verknüpfung der Eingangssignale erlauben.

CPU **C**entral **P**rocessing **U**nit. Im Zusammenhang mit VME-Bus-Systemen ein vollständiges Computersystem auf einer Steckkarte.

Crate Gehäuse, das alle für ein bestimmtes Bussystem (z.B. VME, CAMAC) notwendigen Komponenten wie Stromversorgung und Backplane zusammenfaßt.

CTU **C**entral **T**rigger **U**nit. VME-Bus-Karte, die die von einem Startzähler kommenden Triggerpulse unter Beachtung von Totzeiten und Busy-Zuständen in Befehle auf dem Triggerbus umsetzt. Erzeugt die Triggerbefehle für das gesamte HADES Detektorsystem.

daisy chaining Bestimmte Art, mehrere Module an einem Bus zu verketten. Steuersignale werden dabei vom ersten Modul abgearbeitet und an den Rest der Kette weitergeleitet. Eine direkte Adressierung einzelner Module ist nicht möglich. Prinzipiell können beliebig viele Module verkettet werden.

Decision Aus den Bilderkennereinheiten gewonnene Entscheidung, ob die untersuchten Daten in die LVL2-Pipe übertragen werden sollen. Wird zusammen mit dem LVL2-Trigger am Triggerbus übertragen.

DPR **D**ual **P**orted **R**am. Spezieller Speicherbaustein, der von zwei Seiten her asynchron beschrieben und gelesen werden kann.

DRAM **D**ynamic **R**andom **A**ccess **M**emory. Flüchtiger Speicher, der externe Steuersignale zum Auf"—frischen des gespeicherten Inhalt benötigt. Eine Speicherzelle besteht aus jeweils einem Kondensator.

DTU **D**etector **T**rigger **U**nit. VME-Bus-Karte, die auf dem Triggerbus einlaufende Befehle in die für das jeweilige Auslesesystem notwendigen Ansteuersignale umwandelt. Für jedes Detektorsystem eigens angepaßt.

EEPROM **E**lectrically **E**raseble **P**rogrammable **R**ead **O**nly **M**emory. Nichtflüchtiger Speicher, der elektrisch beschrieben und gelöscht werden kann.

Event Bei HADES ein aus den Daten aller verwendeten Detektorsysteme zusammengesetztes Datenpaket. Wird zentral vom Common Event Builder aus den Subevents erstellt.

FIFO **F**irst **I**n **F**irst **O**ut. Speicher, der eingeschriebene Daten in der Reihenfolge des Einschreibens wieder ausgibt. Verwendet keinen externen Adreßbus und kann üblicherweise von zwei Seiten (eine schreibend, eine lesend) unabhängig benutzt werden.

FPGA **F**ield **P**rogrammable **G**ate **A**rray. Frei programmierbarer Logikbaustein, der meist beliebig oft reprogrammiert werden kann. Durch internen Aufbau aus Logikblöcken und variablen Verbindungswegen kann Logik hoher Komplexität in einem Baustein untergebracht werden.

Frontend (FE) Schnittstelle zur Analogelektronik des RICH Detektors. Beinhaltet die notwendigen Bauteile zur analogen Signalaufbereitung (Vorverstärker, T/H-Stufen, Multiplexer), den Analog-Digital-Wandler, Schwellenvergleicher, Logik zur Steuerung sowie die LVL1-Pipe.

GAL **G**eneric **A**rray **L**ogic. Von der Firma Lattice entwickelter reprogrammierbarer Logikbaustein auf EEPROM-Basis.

GASSIPLEX **GAS SI**licium multi**PLEX**er. Vom CERN entwickelter ASIC zur analogen Signalaufbereitung von Signalen an Halbleiter- und Gaszählern. Kann ebenfalls an Drahtkammern verwendet werden. Beinhaltet ladungsempfindliche Vorverstärker, T/H-Stufen sowie Multiplexer für jeweils 16 analoge Kanäle. Nachfolger des AMPLEX-ASICs.

GSI **G**esellschaft für **S**chwer-**I**onenforschung.

HADES **H**igh **A**cceptance **D**i **E**lectron **S**pectrometer. Hochauf"—lösendes Dileptonen-Spektrometer bestehend aus RICH, MDC, TOF und Preshower.

Hades ($\check{\alpha}\iota\delta\eta\varsigma$) griechisch für Unterwelt. Ort ohne Wiederkehr, an dem man alle Hoffnung fahren lassen kann.

I^2C **I**nter **I**C **C**ommunication. Von Philips entwickelter Standard für einen Zweidraht-Bus. Neben Bausteinen der Consumer-Elektronik sind auch Meß-Schaltungen mit dieser Schnittstelle erhältlich.

IOB **I**nput **O**utput **B**lock. Ein/Ausgabe-Block der Xilinx FPGAs, der Treiberstufen und Register für Signale enthält.

IPU **I**mage **P**rocessing **U**nit. Im Zusammenhang mit dem RICH Funktionsbaugruppe, die im Ansprechmuster des Detektor nach Ringen sucht und die Koordinaten potentieller Ringkandidaten an die PMU weiterleitet. Dient mit zur Erzeugung des LVL2-Triggers.

Longword (Langwort) Einheit aus 32 Bit.

LVL1-Pipe Speicher, der die gewonnenen Analogdaten bis zu einer Entscheidung über deren weitere Verwendung zwischenpuffert.

LVL1-Trigger Trigger, der die Erfassung von Analogdaten und deren Weiterleitung an die IPU startet.

LVL2-Pipe Speicher, der die aus der LVL1-Pipe ausgelesenen Daten sammelt und zur weiteren Verarbeitung bereithält. Enthält bereits fertig formatierte Subevents.

LVL2-Trigger Von der PMU generierter Trigger, der über die weitere Verwendung der in der LVL1-Pipe gespeicherten Daten entscheidet. Möglich ist entweder Löschen der Daten oder Weiterleitung in die LVL2-Pipe.

LVL3-Trigger Softwarebasiertes Signal, das über die endgültige Verwendung der in der LVL2-Pipe gespeicherten Daten entscheidet. Dabei werden die Daten entweder verworfen oder zur Off"—line-Analyse auf Band geschrieben.

Mapping Umformatierung von real vorhandenen Padkoordinaten (bestimmt durch Detektorgeometrie und Leiterbahnführung auf der Padplane) auf das vom weiterverarbeitenden System gewünschte Format. Beinflußt nur die Koordinaten, nicht jedoch die Analogdaten selbst.

MDC **M**ini **D**rift **C**hamber. Drahtkammer mit geringem Abstand zwischen den Drahtebenen und damit kleinen Driftzeiten.

Microcontroller Kleinstcomputer, der in einem Chip untergebracht ist. Wird hauptsächlich für Meß- und Regelaufgaben verwendet.

MU **M**atching **U**nit. Siehe **PMU**.

Multiplexer Schaltung, die aus mehreren Eingangssignalen eines auswählt und auf den Ausgang durchschaltet.

Nibble Einheit aus 4 Bit.

Off"—line-Analyse Software, die es ermöglicht, nach Strahlzeitende die gewonnenen Daten unter Berücksichtigung aller Experiment-Parameter zu analysieren.

Online-Analyse Softwarepaket, das während des Experiments Einblick in die gewonnenen Daten erlaubt. Dient hauptsächlich zur Kontrolle, um prinzipielles Fehlverhalten des Detektor zu erkennen.

Open-Collector bestimmte Technologie, den Ausgang von Logik-ICs zu fertigen. Ermöglicht durch einfaches Zusammenschalten solcher Ausgänge und Anlegen eines Pullup-Widerstandes eine logische AND-Verknüpfung (positive Logik) bzw. OR-Verknüpfung (negative Logik).

Pad Beim RICH einzelnes ladungsempfindliches Bildelement, das jeweils an einen Kanal eines GASSIPLEX geführt wird.

Padplane Leiterplatte, die alle Pads eines RICH-Sektor trägt.

PAL **P**rogrammable **A**rray **L**ogic. Einmal programmierbarer Logikbaustein der Firma AMD (jetzt: Vantis).

Pedestal Grundoffset der Kanäle beim RICH. Jeder Kanal liegt mit seinem Offset bereits ein Stück über dem Nullpunkt des ADC.

PMU **P**attern **M**atching **U**nit. Baugruppe, die die Koordinaten von Ringkandidaten aus dem RICH mit Leptonenidentifizierungen aus anderen Detektorsystemen vergleicht und daraus eine LVL2-Trigger-Entscheidung ableitet.

Port Schnittstelle des RICH Readout-Controllers zu einer Gruppe Frontends auf einer Backplane. Wird jeweils von einem Port-FPGA verwaltet, der alle notwendigen Steuersignale bereitstellt.

Readout-Controller (RC) Zentrale Funktionseinheit der RICH Ausleseelektronik. Dient als Schnittstelle zwischen den Frontends und der CPU. Beinhaltet Logik zur Datenkonzentration, -formatierung und -speicherung. Verantwortlich für Fehlerkontrolle, Erzeugung von Busy-Signalen und Steuersignalen für Schnittstellen zur CPU und IPU.

RICH **R**ing **I**maging **ČH**erenkov. Ringabbildender Čerenkov-Zähler.

SIS **S**chwer-**I**onen-**S**ynchrotron. Teilchenbeschleuniger an der GSI, an dem HADES aufgebaut wird.

SMT **S**urface **M**ounted **T**echnology. Bauelemente werden auf der Oberfläche der Leiterplatte befestigt, nicht in Bohrungen. Ermöglicht Miniaturisierung der Bauelemente, einfachere automatische Bestückung, beidseitige Verwendung der Leiterplatte. Erhöht ebenfalls Zuverlässigkeit durch sichere Lötstellen.

SRAM **S**tatic **R**andom **A**ccess **M**emory. Flüchtiger Speicher, der ohne externe Steuersignale gespeicherte Daten hält. Eine Speicherzelle besteht aus jeweils einem Flipflop.

Strahlrohrzwerg Menschenscheues und nachtaktives mystisches Lebewesen an der GSI. Hauptlebensraum ist der HADES Cave, in dem der Strahlrohrzwerg während des Strahlbetriebs Schabernack mit der Ausleseelektronik treibt.

Strobe-Signal Signal, das die Gültigkeit von bestimmten Signalleitungen auf einem Bussystem anzeigt. Wird ebenfalls zum Starten bestimmter Aktionen im Zielsystem verwendet.

Subevent Datenpaket, das die Daten eines Detektorsystems (bzw. Teilsystems) enthält. Wird beispielsweise beim RICH vom Readout-Controller aus den Datenpaketen aller angeschlossenen Frontends erzeugt.

synchron Logiksystem, bei dem Zustandsänderungen in verschiedenen Stufen des Systems nur zu festgelegten Zeitpunkten registriert werden. Laufzeiten haben keinen Einfluß mehr auf das Verhalten des Systems. Synchrone Systeme sind asynchronen — wo immer möglich — vorzuziehen.

T/H Track and Hold. Analogsystem, das dem Pegel eines Eingangssignals folgt (track) und kontrolliert durch ein externes Signal zur weiteren Verarbeitung analog in einem Kondensator speichert (hold).

Threshold Schwellenwert. Wird für jeden einzelnen Kanal aus dem elektronischen Rauschen der Komponenten ermittelt. Dient zur Separation von wirklichen Signalen an den Pads vom Rauschuntergrund.

TOF **T**ime **O**f **F**light. Detektor, der über Flugzeitunterschiede Rückschlüsse auf die Teilchensorte erlaubt.

Totzeit Zeit, in der ein Datenerfassungssystem blind für neue Trigger ist.

Tracking Spuranalyse. Aus mehreren Information über den Teilchenort wird versucht, die Trajektorie eines Teilchens beim Durchflug durch das Detektorsystem zu rekonstruieren.

TDC **T**ime **D**igital **C**onverter. Baustein, der die Dauer einer Zeitspanne in einen digitalen Wert umsetzt. Wird entweder über einen Zähler oder die ADC-Wandlung eines linear ansteigenden Signals realisiert.

Trigger Allgemein ein Signal, das Datenaufnahme und Auslese startet.

Triggerbefehl Anweisung der CTU an die DTU. Wird stets mit einem Triggertag verschickt. Dient zur Einleitung von bestimmten Aktionen an den DTUs sowie den Auslesesystemen.

Triggerbus Bei HADES verwendetes System, um einen zentralen Trigger zu allen Detektorauslesesystemen weiterzuleiten.

Triggertag (Subeventtag) Frei wählbares Kontrollwort, das zusammen mit einem Triggerbefehl auf dem Triggerbus übertragen wird. Dient als Kontrollwort beim Zusammenfassen von Analogdaten zu Subevents und Subevents zu Events. Wird durch die gesamte Kette bis in die Frontends übertragen und läuft bis zur Zusammenfassung zu Events mit den Analogdaten mit.

VME **V**ersa **M**odule **E**urocard. Bussystem, das unter Federführung von Motorola entwickelt wurde. Viele Baugruppen sind für dieses System erhältlich, da auf Lizenzgebühren verzichtet wurde.

Word (Wort) Einheit aus 16 Bit.

Zeitlplan verschärfter Zeitplan. Muß mit M$ Project erstellt werden.

Anhang B

Registerbeschreibung

Der RICH Readout-Controller wurde als A32-VME-Slave entworfen; die Anbindung an den VME-Bus erfolgt über den Chipsatz CY7C960/964 von Cypress [4]. Die Basisadresse des Boards wird mittels zweier Hexadezimal-Schalter festgelegt; für jedes Board muß dabei eine eigene Basisadresse eingestellt werden. Der Adreßbereich eines Readout-Controllers wird über den Chipsatz in Regions von jeweils 1 MB Größe unterteilt; jede dieser Regions ist dabei einer bestimmten Funktionseinheit des Controllers zugeordnet.

Die 32 Adreßbits des VME-Bus werden wie folgt genutzt:

Adreßbits			
`31 ... 28`	`27 ... 24`	`23 ... 20`	`19 ... 0`
`SW2`	`SW1`	`Region`	`nutzbarer Adreßbereich`

Vom Chipsatz werden insgesamt 16 Regions unterstützt, genutzt werden vom Readout-Controller jedoch nur fünf. Der Offset in der Tabelle bezieht sich dabei auf die gewählte Basisadresse:

Region	Offset	Funktionseinheit	Auswahlsignal
0	`0x000000`	—	—
1	`0x100000`	Speicher der LVL2-Pipe	`CS_MEM`
2	`0x200000`	interner Datenbus `DB[15:0]`	`CS_BUS`
3	`0x300000`	Xilinx-interne Register	`CS_REG`
4	`0x400000`	Xilinx-Konfigurationsleitungen	`CS_XC`
5	`0x500000`	Mapping-Speicher	`CS_DPR`
6	`0x600000`	—	—
⋮	⋮		
15	`0xF00000`	—	—

Tabelle B.1: Region-Übersicht. Die Auswahlsignale für die jeweiligen Schaltungsteile sind ebenfalls angegeben.

B.1 Memory Map

Eine genaue Speicherbelegung für den Readout-Controller kann nur für die Regions 1, 4 und 5 angegeben werden, da hier sowohl Größe als auch Offset des benutzen Speichers festliegen. Die in den Regions 2 und 3 liegenden Register sollten ausschließlich über die in `rc.h` bzw. `rc.c` definierten Funktionen angeprochen werden, da sich die Offsets einzelner Register sowie deren Inhalte im Zuge der Weiterentwicklung noch ändern können.

Für den RICH Readout-Controller geschriebene Software läßt sich bei korrekter Verwendung von `rc.h` und `rc.c` ohne Änderungen im Quelltext neu kompilieren und sich so problemlos an neue Xilinx-Designs anpassen. Die Verwendung fester Offsets im Quellcode oder eigener Ausdekodierung von Registerinhalten wird zwangsläufig zu Fehlverhalten der Software führen.

B.1.1 Region 1

In dieser Region ist die dem VME-Bus zugeteilte Bank der LVL2-Pipe zu finden. Zugriffe in diesen Bereich müssen 32 bit-aligned sein; die einzig erlaubte Zugriffsart ist D32. Der Speicher selbst liegt auf den Adressen `0x00000` bis `0x7FFFC`; die Verwendung des im Bereich `0x80000` bis `0xFFFFC` gespiegelten Speichers ist untersagt.

B.1.2 Region 4

In Region 4 sind die Konfigurationsleitungen zum Initialisieren der auf dem Readout-Controller (samt Portplatine) untergebrachten Xilinx-FPGAs sowie die entsprechenden Leitungen für die Xilinx-FPGAs auf den Frontend-Modulen implementiert. Zum Initialisieren der Frontend-FPGAs muß zuerst der Readout-Controller korrekt konfiguriert werden und im entsprechenden Port-FPGA die Durchschaltung der Konfigurationsleitungen aktiviert werden.

Das unter Offset `0x00003` liegende Konfigurationsregister ist das einzige Register des Readout-Controllers, daß ohne korrekt initialisierte FPGAs ansprechbar ist; es bietet damit der Software die Möglichkeit, das Vorhandensein eines Readout-Controllers auf einer bestimmten Adresse zu testen. Im Normalbetrieb nach der Initialisierung sollte das Konfigurationsregister nicht benutzt werden.

Xilinx-Konfigurationsregister								
Bit	7	6	5	4	3	2	1	0
read	XDONE	/XINIT	DONE	/INIT	CCLK	DIN0	/XPRG	/PRG
write	-	-	-	-	CCLK	DIN0	/XPRG	/PRG

Tabelle B.2: Belegung des Xilinx-Konfigurationsregisters

Die Leitungen `CCLK` und `DIN0` werden bei beiden Initialisierungsvorgängen benutzt. Für den Readout-Controller werden `/PRG`, `/INIT` und `DONE` verwendet, beim Initialisieren einer

Frontend-Kette hingegen `/XPRG`, `/XINIT` und `XDONE`. Genaue Informationen zum Konfigurationsablauf sind in [30] zu finden. Ein universelles Ladeprogramm für Xilinx-FPGAs wurde von Mathias Münch entwickelt, es sollte auf jeder zur DAQ benutzten CPU als `xild` vorhanden sein. Hinweise zur Benutzung sind der Man-Page zu entnehmen.

B.1.3 Region 5

In dieser Region ist der Speicher der Mapping-Einheit untergebracht. Zugriffe in diesem Bereich müssen 32 bit-aligned sein; die einzig erlaubte Zugriffsart ist D32. Von den 32 Datenbits werden allerdings nur die unteren 16 benutzt. Das Dualported RAM ist folgendermaßen aufgeteilt:

Dualported RAM	
Adresse	Bedeutung
`0x00000` ⋮ `0x03FFC`	LVL2-Mapping
`0x04000` ⋮ `0x07FFC`	LVL1-Mapping

Tabelle B.3: Aufteilung des Mapping-Speichers.

Die Verwendung der im Bereich `0x08000` bis `0xFFFFC` liegenden Spiegelbereiche ist untersagt.

Zugriffe auf den Mapping-Speicher sind nur im /STOP-Modus möglich; bei Zugriffen im RUN-Modus werden die jeweiligen Buszyklen zwar korrekt gehandhabt, jedoch keine Schreib-/Lese-Operationen am Speicher selbst durchgeführt.

B.2 Interface-FPGA

Im Interface-FPGA sind zwei Register zur Steuerung der Hardware und zur Signalüberwachung untergebracht. Weitere sechs Register beinhalten Informationen, die zu Performance-Tests und zur Fehlersuche benutzt werden können. Das Ansprechen der in Abschnitt 7.1.3 erwähnten I/O-Leitungen ist ebenfalls über ein eigenes Register möglich. Die Registerbelegung ist nur zu Informationszwecken angegeben; zum Programmieren sollten **nur** die in `rc.h` definierten Funktionen benutzt werden. Nur so ist die in Abschnitt 7.2 beschriebene effektive Trennung zwischen Hardwareprogrammierung und Anwendungsprogramm gewährleistet.

CTRL-Register

Das CTRL-Register bietet auf zwei Adressen Zugriff auf die Hardware des Readout-Controllers. Auf dem Offset `ctrlReg` ist das Setzen der Signale möglich, auf dem Offset `ctrlPad` das Zurücklesen der Signalpegel, die am Baustein anliegen. Das Setzen der Signale hat nur im /STOP-Modus Auswirkungen auf die Hardware, das Lesen der Signalpegel ist hingegen auch im RUN-Modus möglich.
In CTRL-Register sind sieben für den Normal-Betrieb wichtige Signale untergebracht:

`/STOP`	schaltet den Readout-Controller in den /STOP-Modus (low) bzw. in den RUN-Modus (high).
`SWRQ`	fordert im RUN-Modus eine Bankumschaltung an.
`/RESET`	setzt die Hardware des Readout-Controllers zurück; ist nur im /STOP-Modus aktiv.
`RCTR`	löscht die Status-Zähler TrgCtr, PrdCtr, ArdCtr sowie AdelCtr.
`/RCBSY`	ermöglicht der slow control das Zurückverfolgen von Busy-Bedingungen.
`/ERROR`	ermöglicht der slow control das Zurückverfolgen von Fehler-Bedingungen.
`FC[3:0]`	wird zur Konfiguration der Frontend-Module benötigt.

Name	Control															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	RCTR	-	/STOP	SWRQ	SYSCLK	EVTBEG	EVTEND	MSEL	/MWR	/RESET	/RCBSY	/ERROR	FC3	FC2	FC1	FC0
	Powerup	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
ctrlReg	read	register															
	write	register															
ctrlPad	read	0	0	pads													
	wite	reserved for future use															

Die übrigen Signale sind für die Ansteuerung der Hardware im /STOP-Modus vorgesehen und sollten im Normalbetrieb nicht benutzt werden.

TRG-Register

Das TRG-Register bietet auf zwei Adressen Zugriff auf die Hardware des Readout-Controllers. Auf dem Offset `trgReg` ist das Setzen der Signale möglich, auf dem Offset `trgPad` das Zurücklesen der Signalpegel, die am Baustein anliegen. Das Setzen der Signale hat nur im /STOP-Modus Auswirkungen auf die Hardware, das Lesen der Signalpegel ist hingegen auch im RUN-Modus möglich.
Für den normalen Betrieb sind nur die untersten acht Bits des Registers interessant: `/STRB[7:0]` werden zur Konfiguration der Frontend-Module benötigt.

Name	Trigger															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	/TRG	/TBSY	TAGDTA	/TAGCLK	/BEGRUN	/PRDOUT	/ARDOUT	DECISION	/STRB7	/STRB6	/STRB5	/STRB4	/STRB3	/STRB2	/STRB1	/STRB0
	Powerup	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1
trgReg	read	register															
	write	register															
trgPad	read	pads															
	wite	reserved for future use															

TRGCTR-Register

Dieses Register zählt alle einlaufenden Triggerbefehle für die Frontends mit. Das Zurücksetzen erfolgt entweder über RCTR im CTRL-Register oder beim Zurücksetzen über /RESET.

Name	TrgCtr															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	TRGCTR[15:0]															
	Powerup	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
trgCtr	read	TRGCTR[15:0]															
	write	reserved for future use															

PRDCTR-Register

Dieses Register zählt alle einlaufenden Befehle zur Datenübertragung zur IPU mit. Das Zurücksetzen erfolgt entweder über RCTR im CTRL-Register oder beim Zurücksetzen der Hardware über /RESET.

Name	PrdoutCtr															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	PRDOUTCTR[15:0]															
	Powerup	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
prdoutCtr	read	PRDOUTCTR[15:0]															
	write	reserved for future use															

ARDCTR-Register

Dieses Register zählt alle einlaufenden Befehle zur LVL1-Auslese mit. Das Zurücksetzen erfolgt entweder über RCTR im CTRL-Register oder beim Zurücksetzen über /RESET.

Name	ArdoutCtr															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	ARDOUTCTR[15:0]															
	Powerup	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ardoutCtr	read	ARDOUTCTR[15:0]															
	write	reserved for future use															

ADELCTR-Register

Dieses Register zählt alle einlaufenden Befehle zur Löschung von Datensätzen in der LVL1-Pipe mit. Das Zurücksetzen erfolgt entweder über RCTR im CTRL-Register oder beim Zurücksetzen über /RESET.

Name	AdeleteCtr															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	ADELCTR[15:0]															
	Powerup	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
adelCtr	read	ADELCTR[15:0]															
	write	reserved for future use															

TAG-Register

In diesem Register ist das jeweils letzte nach einem /TRG-Befehl von der DTU übertragene Triggertag gespeichert. Dieses Tag wurde mit dem letzten Datenaufnahme-Zyklus in die LVL1-Pipe geschrieben; das Tag, das beim letzten Auslesezyklus ausgelesen wurde, kann im Timing-FPGA unter Offset evtgReg im /STOP-Modus ausgelesen werden.

Name	Tag															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	-	-	-	-	-	-	-	-	TAG[7:0]							
	Powerup	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0
tagReg	read	reserved								register							
	write	reserved for future use															

LC-Register

Zu Debugging-Zwecken wird in diesem Register die Reaktion des Readout-Controller auf Befehle der DTU mitprotokolliert. Die untersten drei Bits geben dabei den DTU-Befehl an (PRDOUT, ARDOUT oder ADELETE); in Bit 3 wird der Zustand der /RCBSY-Leitung vor dem Puls auf der jeweiligen Befehlsleitung, in Bit 4 der Zustand am Ende des Pulses gespeichert. Bei korrekter Reaktion muß Bit 4 low, Bit 3 high sein; sind beide Bits low

oder high, wurde der Befehl vom Readout-Controller nicht akzeptiert. Die RICH DTU befindet sich dann in einer Endlos-Schleife und muß zurückgesetzt werden.

Name	LastCommand																
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	-	-	-	-	-	-	-	-	-	-	-	RCBA	RCBB	ADEL	ARDOUT	PRDOUT
	Powerup	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0
lcReg	read	reserved											register				
	write	reserved for future use															

GIOP-Register

Im den obersten acht Bits dieses Registers ist die Design-Version des Interface-FPGA abrufbar.
Die unteren fünf Bits werden zur Ansteuerung der fünf frei verfügbaren I/O-Leitungen benutzt; unter Offset `giopReg` ist das Setzen der Leitungen möglich, das Zurücklesen der Pegel hingegen unter Offset `giopPad`.

Name	GIOP															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	VERSION[7:0]								-	-	-	GIOP[4:0]				
	Powerup	0	0	0	0	0	0	0	1	X	X	X	0	0	0	0	0
giopReg	read	register								res.			register				
	write	register								res.			register				
giopPad	read	VERSION[7:0]								0	0	0	pads				
	write	reserved for future use															

B.3 Memory-FPGA

Im Memory-FPGA sind sechs Register zur Konfiguration der LVL2-Pipe untergebracht. Ein Register ist für die Bankumschaltung durch die DAQ-Software vorgesehen, zwei weitere enthalten Debug-Informationen.

StatusId-Register

In diesem Register sind sämtliche zur Interruptbehandlung notwendigen Statusbits zusammengefaßt.

`IRQREN` schaltet die Interrupt-Generierung im RUN-Modus an.

IRQSEN schaltet die Interrupt-Generierung im /STOP-Modus an.
STATUSID[7:0]speichert den Interruptvektor für den IACK-Cycle am VME-Bus.

Vor dem Anschalten des Interrupts muß in STATUSID[7:0] ein gültiger Vektor geschrieben werden und auf der CPU ein Interrupt-Handler gestartet werden.

Name	StatusId															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	IRQREN	IRQSEN	-	-	-	-	-	-	STATUSID7	STATUSID6	STATUSID5	STATUSID4	STATUSID3	STATUSID2	STATUSID1	STATUSID0
	Powerup	0	0	X	X	X	X	X	X	0	0	0	0	0	0	0	0
statusid	read	reg		reserved						STATUSID[7:0]							
	write	reg		reserved						STATUSID[7:0]							

Bug-Register

Die in diesem Register enthaltenen Statusbits bilden Xilinx-interne Signale zu Debug-Zwecken ab. Der Inhalt dieses Registers sollte nicht zur Programmierung von Anwender-Software benutzt werden.

Name	Bug																
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	MEMFULL	RUN1	RUN2	SM2START	SM2END	BMA16	BMA15	BMA14	BMA13	BMA12	BMA11	BMA10	BMA9	BMA8	BMA7	BMA6
	Powerup	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bugreg	read	reserved															
	write	reserved for future use															

Memreg-Register

Dieses Register dient zur Konfiguration der LVL2-Pipe. In MEMFULL[10:0] wird die maximal nutzbare Anzahl an 256 Byte-Pages gespeichert, die in einer LVL2-Bank gespeichert werden kann. Bei der Konfiguration muß dieses Register mit einem Wert größer Null beschrieben werden, um die LVL2-Pipe zu aktivieren. Für MEMFULL[10:0] wird als Maximalwert 0x7D5 empfohlen, da sonst Datenverluste nicht ausgeschlossen werden können.

Pages-Register

Dieses Register faßt alle Bits zur Bankumschaltung der LVL2-Pipe zusammen. Im einzelnen haben die Bits folgende Bedeutung:

Name	Memreg															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	-	-	-	-	-	MEMFULL[10:0]										
	Powerup	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0
memreg	read	reserved					MEMFULL[10:0]										
	write	reserved					MEMFULL[10:0]										

MSEL2Q zeigt die aktuell dem VME-Bus zugewiesene Bank an.
MSEL1Q zeigt die über SWRQ angeforderte Bank an.
MEMFULL zeigt eine vollgelaufene LVL2-Pipe an. Dieser Zustand kann nur über die Anforderung einer Bankumschaltung aufgehoben werden.

Name	Pages															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	MSEL2Q	MSEL1Q	MEMFULL	-	-	PAGES10	PAGES9	PAGES8	PAGES7	PAGES6	PAGES5	PAGES4	PAGES3	PAGES2	PAGES1	PAGES0
	Powerup	X	X	1	X	X	0	0	0	0	0	0	0	0	0	0	0
Reg	read	register			X	X	PAGES[10:0]										
	write	reserved for future use															

SedecHigh-Register

Dieses Register enthält die beiden oberen Bytes des SubEventDecoding-Langworts, das in den Subevent-Header eingebaut wird. Wegen des nur 16 bit breiten Datenbusses mußte eine Aufteilung in oberes und unteres Wort vorgenommen werden.

Name	SedecHigh															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	SEDEC_HIGH[15:0]															
	Powerup	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
sedec_high	read	SEDEC_HIGH[15:0]															
	write	SEDEC_HIGH[15:0]															

SedecLow-Register

Dieses Register enthält die beiden unteren Bytes des SubEventDecoding-Langworts, das in den Subevent-Header eingebaut wird.

Name	SedecLow															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	SEDEC_LOW[15:0]															
	Powerup	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
sedec_low	read	SEDEC_LOW[15:0]															
	write	SEDEC_LOW[15:0]															

SeidHigh-Register

Dieses Register enthält die beiden oberen Bytes des SubEventId-Langworts, das in den Subevent-Header eingebaut wird.

Name	SeidHigh															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	SEID_HIGH[15:0]															
	Powerup	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
seid_high	read	SEID_HIGH[15:0]															
	write	SEID_HIGH[15:0]															

SeidLow-Register

Dieses Register enthält die beiden unteren Bytes des SubEventId-Langworts, das in den Subevent-Header eingebaut wird.

Name	SeidLow															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	SEID_LOW[15:0]															
	Powerup	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
seid_low	read	SEID_LOW[15:0]															
	write	SEID_LOW[15:0]															

B.4 Timing-FPGA

Im Timing-FPGA sind zwei Register zur Konfiguration der automatischen Auslesesteuerung untergebracht. Zwei weitere Register dienen der Fehlerverfolgung durch die slow control.

Evtg-Register

In diesem Register steht unter `EVTTAG[7:0]` das letzte im Auslesezyklus benutzte Referenztag. In Zusammenhang mit dem LastCommand-Register des Interface-FPGA kann so

der letzte ausgeführte Zyklus samt Tag festgestellt werden.

Name	Evtg															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	-	-	-	-	-	-	-	-	EVTTAG[7:0]							
	Powerup	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0
evtgReg	read	reserved								EVTTAG[7:0]							
	write	reserved								EVTTAG[7:0]							

Err-Register

Dieses Register liefert im Fehlerfall Informationen über den Port, der ein fehlerhaftes Tag geliefert hat. Ein gesetztes Bit `ERRORx` gibt als fehlerhaften Port `x` an. Typische Fehlerbilder sind ein gesetztes oder ein ungesetztes Bit; dies deutet auf eine fehlerhafte Triggertag-Übermittlung an ein einzelnes Frontend hin. Das Zurücksetzen des Registers erfolgt durch einen beliebigen Schreibzugriff. Aus Kompatibilitätsgründen sollte aber stets der Wert `0x00` benutzt werden.

Name	Err															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	-	-	-	-	-	-	-	-	ERROR[7:0]							
	Powerup	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0
errReg	read	reserved								ERROR[7:0]							
	write	reset register to zero															

Msk-Register

Über die Bits `MASK[7:0]` kann bei wiederholten Fehlern auf bestimmten Ports die Tagüberprüfung gezielt für diese Ports deaktiviert werden. Das Setzen eines Bits schaltet den entsprechenden Port jedoch nicht komplett ab; dazu müssen zusätzliche Bits im Port-FPGA gesetzt werden.

Name	Msk															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	-	-	-	-	-	-	-	-	MASK[7:0]							
	Powerup	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0
mskReg	read	reserved								MASK[7:0]							
	write	reserved								MASK[7:0]							

Stw-Register

Über dieses Register wird die Weite der Strobepulse für die Auslesezyklen eingestellt. Die Breite des Pulses selbst wird über `STW[3:0]` in Einheiten der SYSCLK-Periode (momentan 32 ns) eingestellt. Das Bit `EN` muß zur korrekten Funktion der Auslesesteuerung auf high gesetzt werden. Für das momentane FPGA-Design wird ein Wert von `0x7` empfohlen.
Ebenso kann in diesem Register die aktuelle Designversion des Timing-FPGAs abgefragt werden.

Name	Stw															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	VERSION[7:0]								-	-	-	EN	STW[3:0]			
	Powerup	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0
stwReg	read	VERSION[7:0]								res.			EN	STW[3:0]			
	write	reserved											EN	STW[3:0]			

B.5 Port-FPGA

Jeder Port-FPGA verfügt über drei Register: ein Status-, ein Kontroll- sowie ein Zugangsregister. Die Adressierung der einzelnen Ports erfolgt in `rc.h` nur über die Portnummer, die Umsetzung auf den entsprechenden Adreßoffset geschieht automatisch.

Pstatus-Register

Im diesem Register sind die Kontroll-Leitungen der Frontend-Kette abfragbar. Die einzelnen Signale werden in jeweils einem Register gespeichert, zur Absicherung gegen Spikes werden die Signale über `SYSCLK` gesampelt. Dies muß bei der Ansteuerung im /STOP-Modus beachtet werden.
Die Version des Port-FPGAs ist ebenfalls in diesem Register abfragbar.

`STRBRF` signalisiert einen aus der Modulkette zurückkommenden Strobepuls.
`MACKF` zeigt das Ende eines Modul-Datensatzes an.
`BSY` ermöglicht das Rücklesen der Busyleitung des Ports. Wird von der slow control bei der Fehlerverfolgung benötigt.
`PBSY` sollte nicht mehr benutzt werden.

Feaccess

Über diesen Offset erfolgt der Zugriff auf die Register des Bustransceivers. Schreib- und Lesesignale werden automatisch in die entsprechenden Steuersignale umgesetzt.

Name	Pstatus															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	VERSION[7:0]								STRBRF	MACKF	PBSY	BSY	-	-	-	-
	Powerup	X	X	X	X	X	X	X	X	0	0	0	0	X	X	X	X
Pstatus	read	VERSION[7:0]								register				reserved			
	write	reserved															

Name	Feaccess															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	SD[15:0]															
	Powerup	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
???	read	SD[15:0]															
	write	SD[15:0]															

Pcontrol

Dieses Register ermöglicht die Ansteuerung verschiedener Portoptionen:

Name	Pcontrol															
16bit	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Name	-	-	-	-	-	-	-	-	-	-	PORTOFF	BSYOFF	INVERT	TSPPR	PRT-O	ENXCFG
	Powerup	reserved										0	0	0	0	0	0
Pcontrol	read	reserved										register					
	write	reserved										register					

`PORTOFF` schaltet den Port komplett ab[1].

`BSYOFF` schaltet die Generierung von `/TBSY` ab.

`INVERT` schaltet die Invertierung für die Signale `/STRBR` und `/MACK` ein. Zum Betrieb eines Frontend-Moduls ohne Backplane muß dieses Bit gesetzt werden.

`TSPPR` überbrückt die Register des Transceivers und verbindet internen Datenbus mit Frontend-Bus. **Nur für Testzwecke !**

`PRT-O` schaltet die Treiber des Transceivers in Richtung Frontend. **Nur für Testzwecke !**

`ENXCFG` schaltet die Konfigurationsleitungen für die Frontend-FPGAs durch. Wird zum Initialisieren der FPGAs benutzt. Nur an jeweils einem Port darf dieses Bit gesetzt sein.

[1] not implemented yet.

B.6 DTU-FPGA

In der RICH DTU wurden neben den in [22] beschriebenen Standardregistern etliche Zusatzregister mit RICH-spezifischen Inhalt integriert. Ein Großteil dieser Register dient der leichten Fehlersuche. Da sich die RICH DTU immer noch in der Weiterentwicklung befindet und sich die Inhalte der Register noch ändern können, werden hier nur die bislang definierten Register aufgeführt. Die Belegung der Register kann sich noch ändern.

Kill-Register

In diesem Register sind die zum Zurücksetzen der RICH DTU benötigten Signale enthalten. Ebenso ist die Versionsnummer des DTU-Designs in diesem Register auslesbar.

Name	Kill								
8bit	Bit	7	6	5	4	3	2	1	0
	Name	RD_HOLD	RD_RESET	VER[5:0]					
	Powerup	1	0	X	X	X	X	X	X
killReg	read	reg.		VER[5:0]					
	write	reg.		reserved					

`RD_HOLD` hält alle Statemachines in der DTU an. Sollte vor einem Reset auf low und zum normalen Betrieb auf high gesetzt werden.

`RD_RESET` eigentliches Reset-Signal.

Der empfohlene Ablauf eines Resetvorgangs in der RICH DTU läuft wie folgt ab:

1. `RD_HOLD` wird auf low gesetzt.
2. `RD_RESET` wird auf high gesetzt.
3. `RD_RESET` wird auf low gesetzt.
4. `RD_HOLD` wird auf high gesetzt.

L1QUEUE-Register

Dieses Register enthält die Statusbits der LVL1-Queue.

`PF[3:0]` gibt den Füllstand der LVL1-Queue an.

`PEMPTY` zeigt eine leere LVL1-Queue an.

`PLAST` wird beim Füllstand `0xE` gesetzt und dient zur erweiterten Arbitrierung.

`PFULL` wird beim Füllstand `0xF` gesetzt. **Darf nicht auftreten !**

`PP` zeigt an, daß PRDOUT-Zyklen auszuführen sind.

Name	L1Queue								
8bit	Bit	7	6	5	4	3	2	1	0
	Name	PP	PFULL	PLAST	PEMPTY	PF3	PF2	PF1	PF0
	Powerup	0	0	0	1	0	0	0	0
l1qReg	read	status bits							
	write	reserved							

L2QUEUE-Register

Dieses Register enthält die Statusbits der LVL2-Queue.

Name	L2Queue								
8bit	Bit	7	6	5	4	3	2	1	0
	Name	AP	AFULL	ALAST	AEMPTY	AF3	AF2	AF1	AF0
	Powerup	0	0	0	1	0	0	0	0
l2qReg	read	status bits							
	write	reserved							

`AF[3:0]` gibt den Füllstand der LVL2-Queue an.
`AEMPTY` zeigt eine leere LVL2-Queue an.
`ALAST` wird beim Füllstand `0xE` gesetzt und dient zur erweiterten Arbitrierung.
`AFULL` wird beim Füllstand `0xF` gesetzt. **Darf nicht auftreten !**
`AP` zeigt an, daß ARDDOUT- bzw. ADELETE-Zyklen auszuführen sind.

MASTER-Register

In diesem Register sind Informationen zur MasterControl enthalten.

Name	L2Queue								
8bit	Bit	7	6	5	4	3	2	1	0
	Name	RCBQ	MQ[2:0]			TBQ	MD[2:0]		
	Powerup	0	0	0	0	0	0	0	0
masterReg	read	status bits							
	write	reserved							

`RCBQ` invertiertes `/RCBSY`-Signal der Readout-Controller.
`MQ[2:0]` zeigt den derzeitigen Zustand der Statemachine an.
`TBQ` invertiertes `/TBSY`-Signal der Frontends.
`MD[2:0]` zeigt den nächsten Zustand der Statemachine an.

BUSY-Register

Dieses Register bietet Informationen zum Stand der Busy-Signale der Queues.

Name	L2Queue								
8bit	Bit	7	6	5	4	3	2	1	0
	Name	L1xB	L13B	L12B	L11B	LVL1BSY	LVL2BSY	-	START
	Powerup	0	0	0	0	0	0	0	0
busyReg	read	status bits						res.	status bit
	write	reserved							

L1xB Busy der LVL1-Queue (nicht benutzte Triggerbefehle).
L13B Busy der LVL1-Queue (Triggerbefehl EndRun).
L12B Busy der LVL1-Queue (Triggerbefehl BegRun).
L11B Busy der LVL1-Queue (Triggerbefehl NormalEvent).
LVL1BSY Busy der LVL1-Queue.
LVL2BSY Busy der LVL2-Queue.
START Ergebnis der Arbitrierung.

MODE-Register

Über dieses Register wird die Arbeitsweise der Arbitrierung eingestellt. Außerdem können bestimmte Meßsignale an die Lemo-Ausgänge der RICH DTU geschalten werden.

Name	L2Queue								
8bit	Bit	7	6	5	4	3	2	1	0
	Name	L2M[1:0]		L1M[1:0]		LI	STOP	MODE[1:0]	
	Powerup	0	0	0	0	0	0	0	0
modeReg	read	register							
	write	register							

L2M[1:0] wählt das Signal der unteren Lemobuchse aus.
L1M[1:0] wählt das Signal der oberen Lemobuchse aus.
LI schaltet beide Lemobuchsen auf statisch low.
STOP verhindert jegliche Auslesezyklen der Readout-Controller.
MODE[1:0] wählt den Arbitrierungs-Modus der RICH DTU.

obere Lemobuchse			untere Lemobuchse		
L1M[1:0]	Signal	Bedeutung	L2M[1:0]	Signal	Bedeutung
00	TBQ	invertiertes /TBSY	00	RCBQ	invertiertes /RCBSY
01	RCERR	invertiertes /ERROR	01	START	Arbitrierungssignal
10	_T	LVL1-Strobesignal	10	NITRG	/TRG-Signal der FE
11	IBQ	invertiertes /IPUBSY	11	NIPRDOUT	/PRDOUT-Signal des RC

Arbitrierungs-Modus		
`MODE[1:0]`	Modus	Beschreibung
`00`	`NORMAL`	Auslese nur, wenn `/TBSY` low ist (benutzt BeginRun, EndRun)
`01`	`BEER`	Auslese nur, wenn `/TBSY` high ist (benutzt BeginRun, EndRun)
`10`	`MICKEYMOUSE`	Auslese immer möglich (benutzt BeginRun, EndRun)
`11`	`EXTENDED`	Auslese nur, wenn `/TBSY` low ist (ohne BeginRun, EndRun)

Literaturverzeichnis

[1] G. Agakishiev et al. *Physical Review Letters*, 75, 1995.

[2] Thomas Bretz. Magnetfeldeigenschaften des Spektrometers HADES. Diplomarbeit, Technische Universität München, Fakultät für Physik, Institut E12, 1999.

[3] W. Cassing und E.L. Bratkovskaya. Hadronic and electromagentic probes of hot and dense nuclear matter. *Physics Reports*, 308:65 – 233, 1999.

[4] CYPRESS Semiconductor Cooperation. *CY7C960 Low Cost VMEbus Interface Controller Family*, 1997.

[5] Burr-Brown Corporation. *ADS820, 10-Bit, 20MHz Sampling ANALOG-TO-DIGITAL CONVERTER*, 1996.

[6] CYPRESS Semiconductor Corporation. *CY7C024/0241 CY7C025/0251 4Kx16/18 and 8Kx16/18 Dual-Port Static RAM with Sem, Int, Busy*, 1997. Revised January 8.

[7] CYPRESS Semiconductor Corporation. *CY7C109/CY7C1009 128Kx8 Static RAM*, 1998.

[8] élantec. *EL4421C/22C/41C/42C/43C/44C Multiplexed-Input Video Amplifiers*, 1996. Rev. C.

[9] Roman Gernhäuser. *Ein ringabbildender Čherenkovdetektor zur Untersuchung schwerer Projektilfragmente*. Doktorarbeit, Technische Universität München, 1998.

[10] Particle Data Group. Particle Physics Booklet. American Institut Of Physics, July 1994.

[11] HADES Collaboration. *Proposal for a High-Acceptance Di-Electron Spectrometer*, 1993.

[12] Josef Homolka und Anton Kastenmüller. *DAQ basic principles*, 1.6 Auflage, 10 1999.

[13] Texas Instruments. *SN54ACT16245, 74ACT16245 16-BIT BUS TRANSCEIVERS WITH 3-STATE OUTPUTS*, 1996.

[14] Texas Instruments. *SN74CBT16212 24-BIT BUS-EXCHANGE SWITCH*, 1996.

[15] Anton Kastenmüller. vorauss.: *Der HADES RICH Detektor*. Doktorarbeit. Technische Universität München, vorauss. 2000.

[16] Anton Kastenmüller. Eine schnelle Signalauslese für einen ortsempfindlichen Photonendetektor. Diplomarbeit, Technische Universität München, Fakultät für Physik, Institut E12, 1994.

[17] Anton Kastenmüller, Michael Böhmer, Jürgen Friese, et al. Fast detector readout for the HADES RICH. *Nuclear Instruments & Methods in Physics Research, Section A*, (433):438 – 443, 1999.

[18] Burkhard Kolb. DAQ-Meeting GSI Darmstadt, September 1999. Mündliche Mitteilung.

[19] Burkhard Kolb und Mathias Münch. *HADES raw data format*, 1.2 Auflage, 1998.

[20] Jörg Lehnert. DAQ-Meeting GSI Darmstadt, September 1999. Mündliche Mitteilung.

[21] Erik Lins. *The Trigger Distribution System for the HADES Detector*, 1.5 Auflage, Mai 1998.

[22] Erik Lins. *Detector Trigger Unit Technical Manual*, Juli 1999.

[23] M. Masera. *Nuclear Physics*, A590, 1995.

[24] R. Rapp und J. Wambach. Chiral Symmetry Restauration and Dileptons in Relativistic Heavy-Ion Collisions. preprint hep-ph/9909229, LANL e-print archive, erscheint in Adv. Nuc. Phys. E, September 1999.

[25] Walter Schön, Helmut Bokemeyer, Wolfgang Koenig, und Volker Metag. Simulation of recoilless production of ω mesons. *Acta Physica Polonica B*, 27(11):2959 – 2963, 1996.

[26] Walter Schön, Jürgen Friese, und Wolfgang Koenig. ω meson production in π^- + Nucleus reactions. In Ileana Iori, editor, *XXXVII International Winter Meeting On Nuclear Physics*. Università degli Studi di Milano, Januar 1999.

[27] Christoph Theurer. Eine schnelle Auslesesteuerung für den Photonendetektor im HADES-Rich. Diplomarbeit, Technische Universität München, Fakultät für Physik, Institut E12, 1998.

[28] Thomas Ullrich et al. *Nuclear Physics*, A610, 1996.

[29] Christian Wallner. Ein rechnergestütztes Steuer- und Kontrollsystem für den HADES RICH. Diplomarbeit, Technische Universität München, Fakultät für Physik, Institut E12, 1999.

[30] Xilinx, 2100 Logic Drive, San Jose, California 95124, USA. *The Programmable Logic Data Book*, 1999.

[31] Karl Zeitelhack et al. The HADES RICH detector. *Nuclear Instruments & Methods in Physics Research, Section A*, (433):201 – 206, 1999.

Danksagung

An dieser Stelle möchte ich mich bei all jenen bedanken, die mich während meiner Arbeit unterstützt haben.

Herrn Prof. H. J. Körner danke ich für die freundliche Aufnahme an seinem Institut und für das Interesse, das er dieser technologielastigen Arbeit entgegengebracht hat.

Jürgen Friese und Josef Homolka danke ich für die engagierte Betreuung, die es mir ermöglichte, stets zielgerichtet und eigenverantwortlich mein Thema zu bearbeiten und zu einem erfolgreichen Ende zu bringen.

Anton Kastenmüller danke ich für die vielen Antworten auf meine Elektronikfragen, für die nächtelange geduldige Einweisung in Eagle und nicht zuletzt für die Kekse zur Teepause, die mehr als einmal als Abendessen herhalten mußten.

Der Elektronikabteilung des Physik-Departments danke ich für die stets gute Zusammenarbeit und den nie enden wollenden Einsatzwillen, der selbst vor dem Wochenende nicht halt machte. Namentlich danke ich Dieter Maier, der entscheidende Teile des Readout-Controllers entwarf und die zentrale Ablaufsteuerung zum Leben erweckte. Mein Dank gilt ebenfalls Heinz Springer, dessen Einsatz an der Bestückungsmaschine uns erst die Teilnahme an der Strahlzeit ermöglichte. Herrn Norbert Franz danke ich für die Übernahme der Organisation, die stets die benötigten Bauteile zur richtigen Zeit bereitstellte und die Entscheidung für die Dampfphasenlötanlage, die uns die Fertigung der Frontend-Busplatinen stark erleichterte.

Den Kollegen von der Universität Giessen, namentlich Jörg Lehnert und Erik Lins, danke ich für die gute Zusammenarbeit, die Bereitschaft, auch spätabends noch ein Xilinx-Design zu testen und die Geduld beim Testen der Elektronik, wenn desöfteren ein rotes Lämpchen zu Zwangspausen führte.

Matthias “The code is obvious” Münch gilt mein Dank für den Versuch, mich in die Geheimnisse der DAQ-Programmierung einzuweihen. Die Erklärungen zu meinen unterschiedlichsten Fragen, von der Datenaufnahme bis zur Großwetterlage, waren stets hilfreich.

Karl Zeitelhack danke ich für das Verständnis, das er den schwierigen Aufgabenstellungen der Elektronik entgegenbrachte. Mehr als einmal mußte er deshalb seine mühsam und liebevoll erstellten Zeitlpläne ändern.

Walter Schön samt Heike und Hanna danke ich für die Unterstützung im Laufe meiner Arbeit — auf physikalischer, astronomischer, kulinarischer und persönlicher Ebene. Die

kurzen Unterbrechungen meiner Arbeit waren stets willkommene Ablenkung und gaben mir die Möglichkeit, auch in stressigen Zeiten wieder zur Ruhe zu kommen.

Mein Leidensgenosse Christian Wallner war trotz seines engen Terminplans stets zur Stelle, wenn Not am Mann war, sei es eine durchgearbeitete Nacht am Institut, ein Kneipenbesuch im Spanien oder eine Erklärung zur Analogtechnik. Die harten Squashpartien waren stets eine gute Abwechslung von der Arbeit.

Für die Unterstützung der Diplomanden in stimmungsmäßiger Hinsicht danke ich Angelika Elhardt, die auch sonst den Diplomanden hilfreich zur Seite stand.

Bei der Hilfe beim Kampf gegen die Tücken von PostScript und PDF gilt mein Dank Andreas Stolz. Mit seiner Hilfe konnte so manches Datenblatt doch noch zum richtigen Ausdruck auf dem Drucker bewegt werden. Auch für spätabendliche Diskussionen um satztechnische und gestalterische Streitpunkte war Andreas stets zu haben.

Allen “E12er” danke ich für die kollegiale Zusammenarbeit und die angenehme Arbeitsatmosphäre.

Ein ganz besonderer Dank gilt meinen Eltern, deren Voraussicht mir das Studium der Physik an der Technischen Universität München erst ermöglicht hat.