Inaugural-Dissertation

zur
Erlangung der Doktorwürde
der
Naturwissenschaftlich-Mathematischen Gesamtfakultät
der
Ruprecht-Karls-Universität Heidelberg

vorgelegt von
Dipl.-Ing. Rolf Schneider
aus Heidelberg

Tag der mündlichen Prüfung: 19. Mai 2008

Entwicklung des Triggerkonzepts und die entsprechende Implementierung eines 200-GB/s-Auslesenetzwerks für den ALICE-Übergangsstrahlungsdetektor

Gutachter: Prof. Dr. Volker Lindenstruth

Prof. Dr. Ulrich Brüning

Entwicklung des Triggerkonzepts und die entsprechende Implementierung eines 200-GB/s-Auslesenetzwerks für den ALICE-Übergangsstrahlungsdetektor

Intelligente Triggersysteme erlangen durch den Einsatz moderner Informationstechnologie in der Hochenergiephysik immer größere Bedeutung. Gerade bei Schwerionenexperimenten wird eine große Anzahl an Teilchen erzeugt, durch die enorme Datenmengen entstehen. Mit einer frühzeitigen Filterung der Daten und dem Verwerfen uninteressanter Ereignisse kann die Effizienz des Gesamtsystems maßgeblich gesteigert werden. Im Experiment ALICE am CERN werden dabei neue Wege beschritten. Beim Übergangsstrahlungsdetektor werden die erfassten Signale noch auf dem Detektor mit über 65 000 Multi-Chip-Modulen parallel verarbeitet. Über ein Auslesenetzwerk gelangen die aufbereiteten Daten an eine globale Spurrekonstruktion, die zu der Entscheidung beiträgt, ob das Ereignis verworfen oder weiter bearbeitet werden soll. In dieser Arbeit werden ein Triggerkonzept für den Übergangsstrahlungsdetektor entwickelt und das Auslesenetzwerk implementiert. Eine besondere Herausforderung liegt dabei im effizienten Zusammenspiel der genannten Verarbeitungsstufen. Durch Simulationen und Analysen wird das Gesamtsystem daraufhin optimiert. Wie sich herausstellt, spielt die Auslese dabei eine wesentliche Rolle. In diesem Zusammenhang wird außerdem ein Design-Fluss für den eingesetzten ASIC erarbeitet. Die Untersuchungen zeigen, dass durch Optimierungen den extremen Anforderungen an dieses komplexe System Rechnung getragen werden kann. Während einer Strahlzeit wurden erste Prototypen erfolgreich getestet. Das Gesamtsystem befindet sich zur Zeit im Aufbau und wird 2008 in Betrieb gehen.

Design of a trigger layout and the corresponding implementation of a 200 GB/s readout network for the ALICE transition radiation detector

Through the use of modern information technology, intelligent trigger systems are gaining more and more importance in high-energy physics. Particularly in heavy ion experiments, the large number of generated particles results in an enormous amount of data. By filtering the data at an early stage and discarding irrelevant events, the efficiency of the entire system can be raised significantly. The ALICE experiment at CERN breaks new ground in this respect. With the Transition Radiation Detector, the acquired signals are processed parallel right on the detector using more than 65 000 multi-chip modules. Via a read-out network, the preprocessed data arrives at a global track reconstruction unit, which contributes to the decision whether an event is discarded or further processed. In this thesis, a trigger concept for the Transition Radiation Detector is developed and the read-out network is implemented. A special challenge is to achieve an efficient interaction of the above processing stages. By means of simulations and analyses, the entire system is optimized in this regard. It turns out that the read-out process plays a decisive role. In this context, a design flow for the used ASIC is developed. The analyses show that through optimizations the extremely high demands made on this complex system can be met. During a beam time, first prototypes have successfully been tested. The entire system is currently being assembled and will be brought on line in 2008.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Mehr über die Materie zu lernen, aus der wir alle gemacht sind, aus was sie besteht und wie sie zusammengehalten wird, ist das Ziel der Experimente und Untersuchungen am CERN[1]. Das CERN wurde 1954 gegründet und ist heute mit seinen über 6500 Mitarbeitern das größte Zentrum für Teilchenphysik und die größte Forschungseinrichtung Europas. Über die Hälfte aller Teilchenphysiker aus 80 Nationen kommen an das CERN um zu forschen. Die Universität Heidelberg ist mit dem Kirchhoff-Institut für Physik und dem Physikalischen Institut eine der 500 beteiligten Universitäten. Das CERN spielt eine wichtige Rolle bei der Entwicklung neuer Technologien, bei der Lehre und multinationalen Zusammenarbeiten.

Zur Zeit befindet sich am CERN ein neuer Beschleunigerring, der LHC[2], im Aufbau. Er wird der leistungsstärkste Teilchenbeschleuniger der Welt sein. Für die Experimente werden Teilchen in entgegengesetzter Richtung beschleunigt und können an vier Punkten zur Kollision gebracht werden. Der Beschleuniger kann dabei sowohl mit Protonen als auch mit Schwerionen arbeiten. Für drei der Experimente werden Protonen bis zu einer Schwerpunktsenergie von 14 TeV beschleunigt. Hierbei werden insbesondere neue Elementarteilchen gesucht und CP-Verletzungen im B-System untersucht. Diese Arbeit beschäftigt sich in erster Linie mit dem vierten Experiment, ALICE[3], das vor allem für den Schwerionenbetrieb vorgesehen ist. Dabei kollidieren Bleiionen mit einer Schwerpunktsenergie von 1150 TeV. Diese Energie ist so hoch, dass nach der Quantenchromodynamik dabei die Protonen und Neutronen ihre Identität verlieren und die Quarks freigeben. Zusammen mit den Gluonen bilden sie das Quark-Gluonen-Plasma, eine Form der Materie, wie sie kurz nach dem Urknall geherrscht hat. Beim Abkühlen dieses Plasmas entsteht die uns bekannte, aus Quarks zusammengesetzte Materie. Die Bildung dieser Materie konnte bisher experimentell nicht nachgewiesen werden, da bestehende Teilchenbeschleuniger die notwendigen Energien nicht erreichen.

Das Experiment ALICE besteht aus einer Vielzahl von Detektoren und datenverarbeitender Elektronik. Die Detektoren sind dabei so angeordnet, dass sie nach einer Kollision möglichst viele Zerfallsprodukte und deren observable Größen erfassen. Alle Detektoren zusammen werden dabei eine Länge von 25 m und eine Höhe von 15 m haben. Eine große Herausforderung bei Schwerionenexperimenten ist die hohe Anzahl und Dichte der zu detektierenden Teilchenspuren. Bei ALICE finden etwa 10 000 Kollisionen pro Sekunde statt, und es werden bei einer zentralen Kollision bis zu 20 000 primäre geladene Teilchen produziert. Um alle interessanten Spuren erfassen zu können, werden sowohl hochauflösende, jedoch langsame,

[1] Organisation Européenne pour la Recherche Nucléaire, http://www.cern.ch
[2] Large Hadron Collider
[3] A Large Ion Collider Experiment

Detektoren eingesetzt als auch sogenannte Triggerdetektoren. Mit ihnen können interessante Ereignisse früh erkannt werden und die Auslese der langsameren Detektoren wird nur in diesen Fällen gestartet.

Gerade in der Hochenergiephysik erlangen die Triggersysteme immer größere Bedeutung. Sie tragen maßgeblich zur Effizienz und Qualität der Gesamtsysteme bei. Physik, Elektronik und Informatik lassen sich heutzutage nicht mehr trennen, und insbesondere der Anteil der Informationstechnologie wird immer größer. Auch aus der theoretischen Physik lässt sich die Informatik nicht mehr wegdenken. Darüber hinaus erfordern physikalische Experimente häufig sehr spezielle und eigens dafür entwickelte Hard- und Software. Ein Beispiel, bei dem dieser Ansatz konsequent verfolgt wird, ist der Übergangsstrahlungsdetektor (TRD) des ALICE-Experiments. Er dient sowohl als Trigger- wie auch als Trackingdetektor. Als Trigger hat er die Aufgabe, bestimmte Spursignaturen innerhalb sehr kurzer Zeit zu erkennen und trägt damit zur Entscheidung bei, ob die hochauflösende Zeit-Projektions-Kammer ausgelesen wird. Das besondere hierbei ist, dass die erfassten Daten noch auf dem Detektor verarbeitet werden. Hierzu kommen über 65 000 Multi-Chip-Module zum Einsatz, die die Daten digital wandeln, vorverarbeiten und mit jeweils vier CPUs parallel verarbeiten. Damit ist es möglich, noch vor der Auslese der Daten eine Filterung vorzunehmen und uninteressante Ereignisse zu verwerfen. Die vorverarbeiteten Daten gelangen über ein Auslesenetzwerk an die globale Spurrekonstruktion, die die Datenströme zusammenfasst, weiterverarbeitet und eine Triggerentscheidung trifft.

Diese Arbeit beschäftigt sich mit der Entwicklung eines Triggerkonzepts und des Auslesenetzwerks für den TRD. Die besondere Herausforderung liegt hierbei in der Betrachtung des Gesamtsystems und dem Zusammenspiel der lokalen Vorverarbeitung der Daten, der Auslese und des globalen Trackings. Im Folgenden werden das ALICE-Experiment, die Anforderungen an das Triggersystem und der TRD vorgestellt. Das darauf folgende Kapitel gibt zunächst einen Überblick über die Simulationsumgebung für das Experiment. Es werden die Leistungsfähigkeit des TRDs analysiert und die der lokalen Vorverarbeitung optimiert. Zusammen mit einem Ansatz zum globalen Tracking lassen sich damit die Anforderungen an das Auslesenetzwerk für den Detektor definieren. In Kapitel 4 werden der Aufbau des Tracklet-Prozessors – eines der beiden Chips des Multi-Chip-Moduls – erläutert und ein Design-Fluss für diesen Chip erarbeitet. Darauf folgt die Entwicklung eines Auslesenetzwerks, das sich in das Triggersystem integriert und den extremen Ansprüchen genügt. Ein wesentliches Modul des Tracklet-Prozessors ist die Netzwerkschnittstelle, mit der das Netzwerk aufgebaut wird. Der Entwurf und die Implementierung dieses Moduls sind in Kapitel 6 beschrieben. Darauf werden die Simulationsergebnisse, Testaufbauten für die Auslese und der erfolgreiche Einsatz der Multi-Chip-Module in Prototypen für Teile des TRDs vorgestellt. Das anschließenden Kapitel fasst die Ergebnisse dieser Arbeit zusammen und gibt einen Ausblick auf die Integration des Detektors.

2 Experiment

Der Large Hadron Collider (LHC) ist ein Beschleunigerring am CERN. Er befindet sich in 50–170 m Tiefe und hat einen Umfang von 27 km. Der Tunnel ist ursprünglich für den Large Electron-Positron Collider (LEP) gebaut worden, bei dem Elektronen und Positronen in entgegengesetzter Richtung beschleunigt wurden und kollidierten. Der LHC arbeitet in zwei Betriebsmodi. Mit ihm können sowohl Protonen mit einer Schwerpunktsenergie von 14 TeV als auch Schwerionen mit einer Schwerpunktsenergie von 1150 TeV zur Kollision gebracht werden. Damit wird der LHC der leistungsstärkste Teilchenbeschleuniger der Welt sein. Die beiden entgegengesetzt gerichteten Strahle werden von dem Super Proton Synchrotron (SPS) auf 450 GeV vorbeschleunigt und in den LHC eingespeist. Durch Magnete auf der Bahn gehalten, kreisen die Protonenwolken für mehrere Stunden und werden an vier Punkten, an denen die Experimente stattfinden, zur Kollision gebracht.

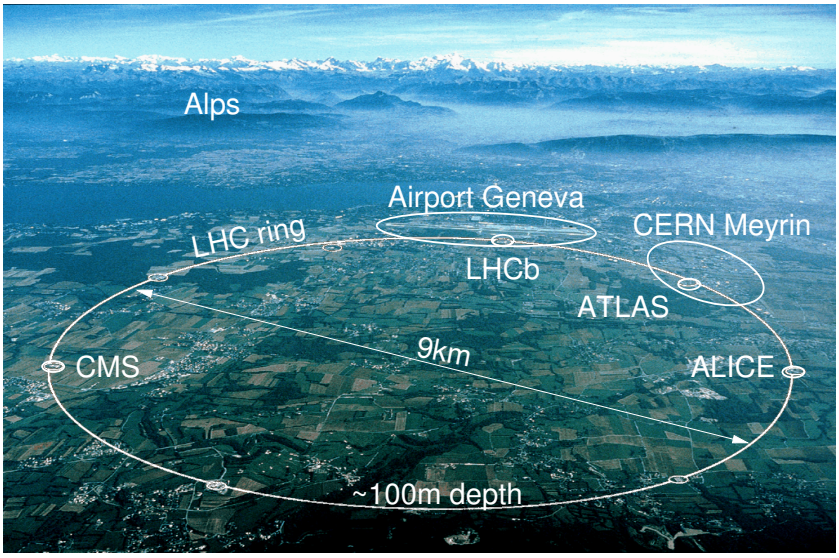

Abbildung 2.1: Luftaufnahme der Genfer Gegend mit dem Genfer See und den Alpen im Hintergrund. Die Dimension des LHC Beschleuniger-Rings kann mit der Größe eines internationalen Flughafens verglichen werden. Quelle: CERN

Bei den Experimenten ATLAS[1] und CMS[2] ist es das Ziel, neue Elementarteilchen, insbesondere das Higgs-Boson und supersymmetrische Teilchen, zu entdecken. Beim LHCb[3] werden

[1] A Toroidal LHC Apparatus
[2] Compact Muon Solenoid
[3] LHC Beauty Experiment

speziell CP-Verletzungen im B-System untersucht. Während diese drei Experimente mit Kollisionen von Protonen arbeiten, ist ALICE in erster Linie für den Schwerionenbetrieb vorgesehen.

2.1 ALICE

ALICE[4] ist eines der vier Experimente am LHC und untersucht die Entstehung der Materie nach dem Urknall. Bei ALICE werden schwere Bleiionen (^{208}Pb) mit so hoher Energie zur Kollision gebracht, dass nach der Quantenchromodynamik (QCD) – der Theorie der Quarks und Gluonen – die Protonen und Neutronen ihre Identität verlieren und die Quarks freigeben. Zusammen mit den Gluonen bilden sie eine neue Form der Materie, das Quark-Gluonen-Plasma (QGP), ein Zustand, wie er etwa 10 µs nach dem Urknall geherrscht hat. Beim Experiment hat sich nach etwa 10^{-23} s das Plasma ausgedehnt bzw. abgekühlt und emittiert die aus Quarks zusammengesetzten Hadronen (Protonen, Neutronen, Pionen, Lambdas, ...). Bisher konnte diese Bildung der uns bekannten Materie nicht experimentell nachgewiesen werden.

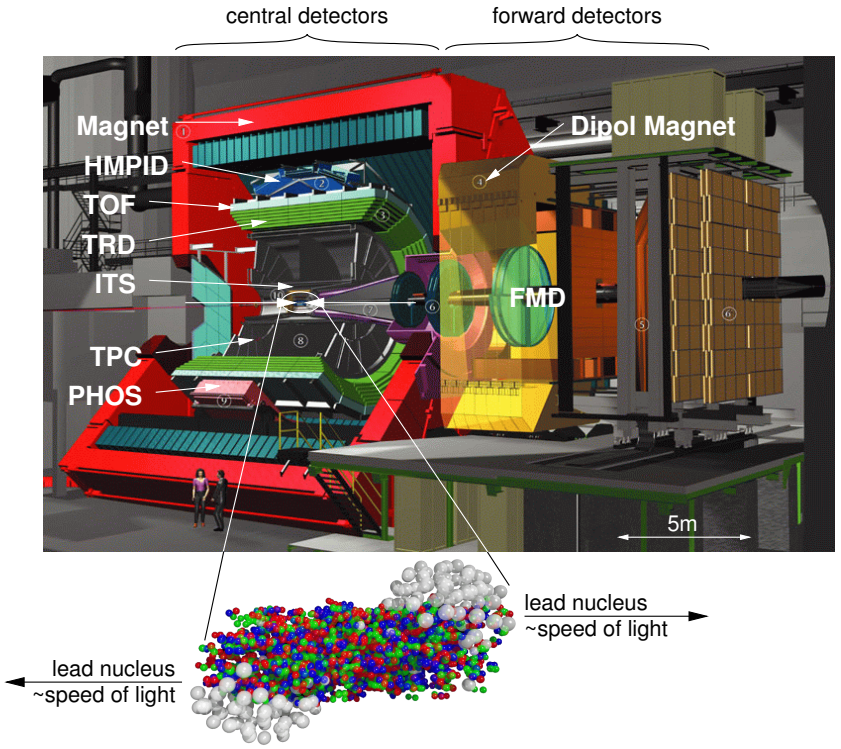

Abbildung 2.2: Darstellung des ALICE-Experimentes und seiner Detektoren. Gekennzeichnet ist der TRD und die TPC. Im Vertex werden Blei-Kerne bei annähernd Lichtgeschwindigkeit zur Kollision gebracht.

[4] A Large Ion Collider Experiment

Im Experiment unterscheidet man zwischen den Vorwärtsdetektoren (z. B. FMD, T0 und V0 [2]) und dem Zentralbereich. Alle Detektoren zusammen (Abbildung 2.2) werden etwa ein Gewicht von 10 000 t haben, 25 m lang und 15 m hoch sein. Im Folgenden werden die zentralen Detektoren des Experiments beschrieben. Der Detektor, der sich am nächsten am Kollisionspunkt (Vertex) befindet, ist das **Innere-Tracking-System** (ITS, [3]). Mit ihm sollen der genaue Vertex und die Sekundärvertices bestimmt und Teilchen mit niedrigem Impuls gefunden werden. Sein empfindlicher Bereich in radialer Richtung beträgt 3 cm bis 50 cm und grenzt direkt an den größten Detektor des Experiments, die **Zeit-Projektions-Kammer** (TPC, [4]). Sie hat die Form eines Hohlzylinders von 5,1 m Länge und deckt den Bereich von 0,57 m bis 2,78 m in radialer Richtung ab. Der Zylinder ist mit einem Ne-CO_2 (90:10) Gasgemisch gefüllt; zwischen einer Hochspannungselektrode in der Mitte und den Endkappen wird ein elektrisches Feld angelegt. Geladene Teilchen, die das Gas durchqueren, erzeugen durch Ionisation Elektronen, die zu den Endkappen wandern und dort detektiert werden. Die positiven Ionen werden durch die Elektrode abgesaugt. Durch seine Größe und die hohe Kanalzahl ist die TPC der zur Spurbestimmung genaueste Detektor und liefert die größten Datenmengen. Seine maximale Auslesefrequenz liegt bei etwa 200 Hz und ist durch die Driftgeschwindigkeit der Ladungsträger im Gas limitiert. Zur genauen Bestimmung von Impulsen und Vertices und zur Identifikation eines Teilchens werden die Daten des ITS, der TPC und des **Übergangsstrahlungsdetektors** (TRD, [5]) zusammen betrachtet. Durch die Architektur des TRDs (Kapitel 2.3), bei dem die Driftstrecken mit 3 cm relativ kurz sind, lässt sich eine deutlich geringere Auslesezeit gegenüber der TPC erreichen. Die Hauptaufgabe des TRDs besteht darin, interessante Ereignisse zu erkennen und die TPC zu triggern. Damit lässt sich die Effektivität der TPC (für interessante Ereignisse) um einen Faktor zehn steigern. Der TRD umgibt die TPC und liegt direkt innerhalb des **Flugzeit-Detektors** (TOF, [6]). Mit ihm lässt sich die Masse von hochenergetischen Teilchen bestimmen. Das **Photon-Spektrometer** (PHOS, [7]) dient speziell zum Nachweis hochenergetischer Photonen, mit denen Rückschlüsse auf die Temperatur der Kollision gezogen werden können. Hochenergetische Teilchen (Pionen, Kaonen und Protonen) mit Impulsen von 1 bis 5 GeV werden durch den **Hochenergie-Teilchen-Identifikations-Detektor** (HMPID, [8]) bestimmt. Alle Detektoren des Zentralbereichs umgibt ein Magnet, der ein homogenes Feld mit einer Stärke von 0,4 Tesla erzeugt.

2.2 Triggerhierarchie

Während des Betriebs von ALICE finden etwa 10^4 Blei-Blei-Kollisionen pro Sekunde statt. Nur bei etwa einem hundertstel der Kollisionen erwartet man ein physikalisch interessantes Ereignis. Diese finden vor allem dann statt, wenn sich die beiden gegenläufig beschleunigten Kerne möglichst zentral treffen. Die Zentralität wirkt sich direkt auf die Anzahl (Multiplizität) und Verteilung der erzeugten Teilchen aus. Die Detektoren unterscheiden sich stark in ihrer Frequenz, in der sie Ereignisse verarbeiten können. Um die Auslese nur dann zu starten, wenn ein interessantes Ereignis stattgefunden hat, werden so genannte Triggerdetektoren eingesetzt. Aufgabe dieser Detektoren ist es, möglichst früh die uninteressanten Ereignisse herauszufiltern und die Auslese der langsameren Detektoren nur bei interessan-

ten Ereignissen zu starten. So hat zum Beispiel der für eine Spurrekonstruktion wichtigste Detektor, die TPC, eine relativ lange Auslesezeit von 5 ms. Der TRD, der als Triggerdetektor für die TPC fungiert, hat eine Auslese- und Verarbeitungszeit von 6,5 µs. Auf diese Weise wird nicht nur die Effektivität des Gesamtsystems maßgeblich gesteigert. Durch die Reduktion der riesigen Datenmengen können die weitere Verarbeitung erheblich beschleunigt und die Speicherung vereinfacht werden. Je früher eine Filterung und Vorverarbeitung des Datenstroms stattfindet, desto effektiver lässt sich das System gestalten. Durch eine hochgradig parallelisierte Verarbeitung der Daten direkt an der Stelle der Akquisition – noch auf dem TRD selbst – werden im Experiment ALICE neue Wege beschritten.

Das Triggersystem bei ALICE ist hierarchisch in vier Stufen unterteilt. Ein zentraler Trigger-Prozessor (CTP) sammelt und steuert alle Triggersignale. Diese Signale werden zusammen mit einem Taktsignal, dem LHC-Takt, über die TTC (Timing and Trigger Control, [1]) an alle Detektoren verteilt.

Noch vor der ersten Triggerstufe wird durch den T0- und V0-Detektor eine Kollision erkannt und ein Prätriggersignal erzeugt (Abbildung 2.6), das kurze Zeit später zur Verfügung steht. Die erste Triggerstufe, der **Level-0-Trigger** (L0), wird 1,2 µs nach einer Kollision durch den FMD ausgelost. Der FMD entscheidet anhand der Multiplizität und der Position der Wechselwirkung, ob eine Kollision stattgefunden hat. Noch während der Verarbeitungsphase des FMD startet die Auslese und Datenanalyse des TRD und sucht nach bestimmten Spurmustern (Kapitel 2.3.3). Wird der L0 nicht ausgelöst, bricht der TRD seine Verarbeitung ab. Wird der L0 akzeptiert, arbeitet der TRD weiter und die nächste Triggerstufe, der **Level-1-Trigger** (L1), startet. Hier werden mithilfe des Null-Grad-Kalorimeters und des Myonen-Spektrometers genauere Aussagen über die Zentralität bzw. Multiplizität gemacht. Sowohl der TRD als auch der L1 treffen ihre Entscheidung bis 6,5 µs nach der Kollision. Danach wird die TPC ausgelesen; der **Level-2-Trigger** (L2) nutzt diese Zeit, um mit den Daten weiterer Detektoren genauere Analysen vorzunehmen. Der **High-Level-Trigger** (HLT) ist die letzte Stufe im Triggersystem. Er nutzt die Daten des TRDs und der TPC um eine Spurerkennung vorzunehmen. Parallel hierzu findet bereits eine Datenkompression statt. Nach der letzten Stufe erfolgt die Datenakquisition.

2.3 Übergangsstrahlungsdetektor (TRD)

Der Übergangsstrahlungsdetektor wird als Trigger- und Trackingdetektor eingesetzt. Als Trigger hat er die Aufgabe, e^+e^- Paare innerhalb von 6,5 µs [13] zu erkennen (Kapitel 2.3.3) und trägt zur Entscheidung des Auslösens eines L1-Triggers bei.

Die analogen Signale der mehr als 1,2 Millionen Kanäle des TRDs werden mit einer Auflösung von 10 Bit und einer Rate von 10 MHz digital gewandelt und erzeugen damit einen maximalen Datenstrom von 14 Terabyte pro Sekunde. Durch 70 848 Prozessoren (TRAPs) werden diese Daten noch auf dem Detektor parallel vorverarbeitet und als Spursegmente (Kapitel 3.3) über ein Auslesenetzwerk zum globalen Tracking weitergeleitet. Das globale Tracking hat die Aufgabe, Spuren aus den Spursegmenten der verschiedenen Lagen des

TRDs zu rekonstruieren und genaue Aussagen über die Eigenschaften des gefundenen Teilchens zu machen. Der hierbei bestimmte Impuls und die Identifikation eines Teilchens tragen zu der Entscheidung des L1-Triggers bei. Außerdem werden in den TRAPs alle Rohdaten gespeichert und nach einem L1-Trigger ebenfalls ausgelesen. Diese Daten werden sowohl vom HLT als auch zur späteren Offline-Analyse der Daten herangezogen.

2.3.1 Funktionsweise

Durchquert ein geladenes Teilchen die Grenze zwischen zwei Medien mit unterschiedlichen Brechungsindizes, wird eine Übergangsstrahlung erzeugt. Die Strahlung ist dabei auf einen Vorwärtskegel beschränkt, der relativ zur Auflösung des Detektors klein ist. Die Energie ist direkt vom Lorentz-Faktor $\gamma = \sqrt{1-(\frac{v}{c})^2}$ abhängig [9] und lässt sich somit bei gegebenem Impuls eines Teilchens als Maß für die Masse verwenden. Hiermit lassen sich die viel schwereren Pionen von den Elektronen unterscheiden. Um die Wahrscheinlichkeit für Übergangsstrahlung zu erhöhen, verwendet man beim TRD einen Stapel von Folien auf einem mechanischem Träger aus ROHACELL[5] Schaum (HF71), der seinerseits auch als Radiator zur Übergangsstrahlung fungiert. Die emittierten Photonen haben die beiden Eigenschaften, dass sie in der Driftregion, nahe am Radiator, eine relativ große Ionisationswolke erzeugen. Diese Signatur (Abbildung 3.7) macht man sich zu Nutze, um bei der Auswertung Elektronen von Pionen unterscheiden zu können.

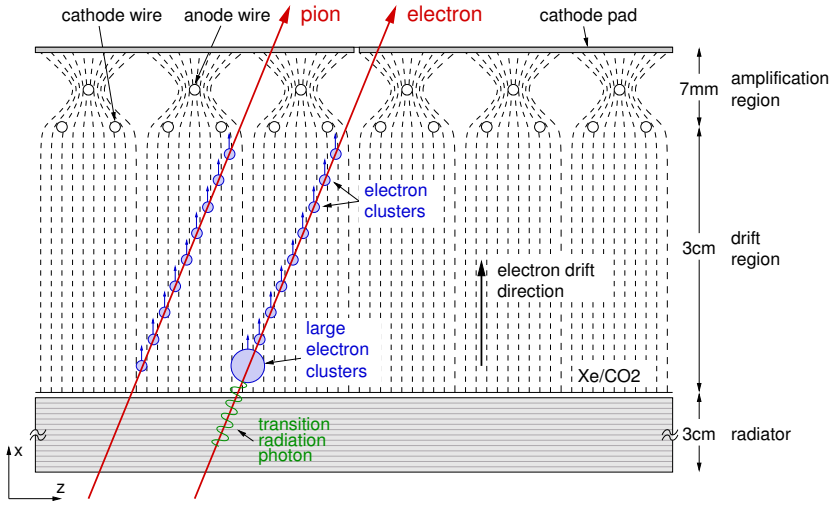

Abbildung 2.3: Driftkammer mit Verstärkungs-, Drift- und Übergangsstrahlungsregion. Die gesuchten Spuren stehen nahezu senkrecht zu den Auslese-Pads.

Ein Detektormodul (Abbildung 2.3) besteht aus dem Radiator und einer Driftkammer, die mit einem Gasgemisch (85 % Xenon, 15 % CO_2) gefüllt ist. Durchfliegt ein geladenes Teilchen die Kammer, wird das Gas auf dessen Bahn ionisiert. Die erzeugten Elektronen driften in Richtung der Verstärkungsregion. Durch eine am Anodendraht angelegte Hochspannung

[5] http://www.rohacell.com

kommt es zu einem Lawineneffekt, bei dem die Anzahl der Ladungsträger vervielfacht wird und sich die Gesamtladung um etwa einen Faktor 4000 [10][11] erhöht. Erst hierdurch kann eine Erfassung der Signale stattfinden. Die Elektronen fließen über die Anodendrähte ab, während die zurückbleibenden Gasionen zur Kathode driften. Die auf den Kathoden-Pads induzierte Ladung wird durch ladungsempfindliche Vorverstärker (PASA[6]) [18][19] in ein differentielles Spannungssignal gewandelt. Digital-Analog-Wandler (ADC) konvertieren das Signal und geben es an die auf dem Modul befindliche digitale Logik weiter. Die gesamte Driftzeit beträgt etwa 2 µs und die ADCs arbeiten mit einer Rate von 10 MHz. Damit ergibt sich eine Auflösung von 150 µm in Driftrichtung (x-Richtung).

Die Größe der Kathoden ist so gewählt, dass die Elektronenwolken etwa 10 % ihrer Ladung auf den Nachbarkathoden deponieren. Obwohl eine Kathode im Mittel 7 mm breit (y-Richtung) ist, kann man durch Ausnutzung dieser Ladungsteilung eine Auflösung von etwa 400 µm erreichen.

Die Auflösung in Richtung des Strahls (z-Richtung) ist auf Modulebene begrenzt durch die Tiefe der Kathoden. Sie liegt in der Größenordnung von 10 cm.

Es muss berücksichtigt werden, dass sich durch das Magnetfeld von 0,4 Tesla die Steigung der Spuren ändert. Der Lorentz-Winkel (Kapitel 3.3.4) beträgt konstante 7°.

2.3.2 Mechanische Struktur

Der TRD ist zylindersymmetrisch um die Strahlachse aus 90 Stacks aufgebaut (Abbildung 2.4), 5 in z-Richtung und 18 in ϕ-Richtung. Betrachtet man eine einzelne Kammer bzw. ein einzelnes Modul, wird die Richtung entlang ϕ mit y-Richtung bezeichnet. Ein Stack besteht aus sechs Modulen, die in x-Richtung gestapelt sind. Der innere Radius des TRDs beträgt etwa 3 m, der äußere 3,7 m und in z-Richtung ist der TRD etwa 5 m lang. Alle Module haben in y-Richtung 144 Kathodenspalten, in z-Richtung sind die Module in entweder 12 oder 16 Zeilen unterteilt.

Alle signalverarbeitende Elektronik sitzt auf einer kleinen Platine, die zwei Chips trägt, den analogen Vorverstärker und die digitale Elektronik. Dieses Multi-Chip-Modul (MCM), dargestellt in Abbildung 2.5, verarbeitet die Signale von jeweils 18 Kanälen.

Die auf den Kathoden induzierte Ladung wird durch den PASA verstärkt und als differentielles Spannungssignal an die ADCs gegeben. Diese wandeln mit einer Rate von 10 MHz das analoge Signal in eine 10-Bit-Zahl. Der Präprozessor (Kapitel 3.3) filtert die Werte und bereitet sie zur Weiterverarbeitung auf. Mithilfe eines Prozessors (TRAP, Kapitel 4.1) werden damit die Parameter von Spursegmenten berechnet, die dann über das Netzwerk weitergeleitet werden.

[6] Pre-Amplifier – Shaper-Amplifier

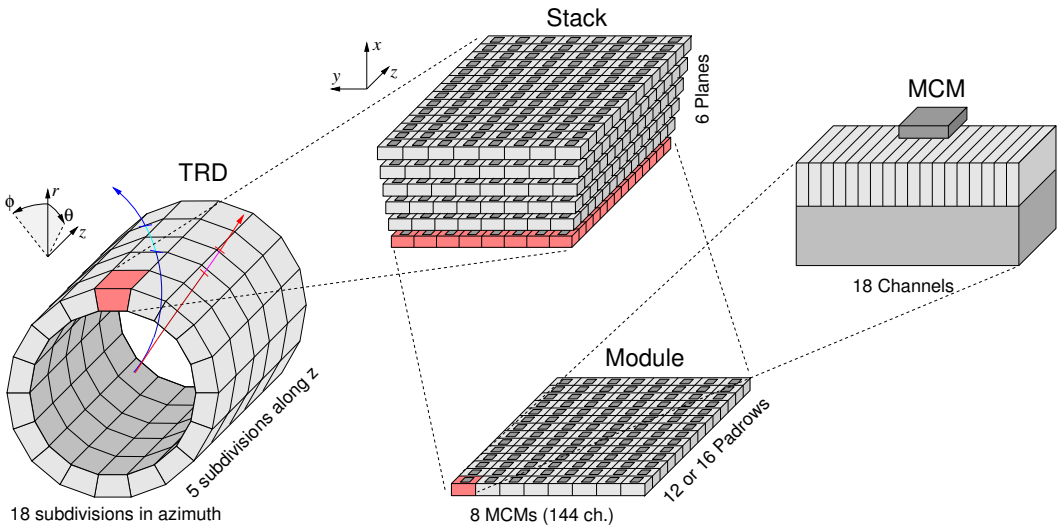

Abbildung 2.4: Aufbau des Übergangsstrahlungsdetektors (TRD), der aus 540 Modulen besteht. Auf jedem Modul sitzen bis zu 128 Multi-Chip-Module (MCMs). Jedes dieser MCMs verarbeitet 18 Kanäle.

Durch diese Vorverarbeitung erreicht man eine Reduktion der Datenmenge um einen Faktor 1000. Die Daten der 65 664 signalverarbeitenden MCMs werden über ein Netzwerk (Kapitel 5) ausgelesen und durch 1080 optische Übertrager zur Weiterverarbeitung an die globale Tracking-Einheit (GTU) gesendet.

2.3.3 Anforderungen an das Triggersystem

Bei der Erzeugung eines Quark-Gluonen-Plasmas wird eine Produktion von J/ψ und Υ Teilchen erwartet, die beide eine hohe Ruheenergie haben. Sie zerfallen in hochenergetische Elektron-Positron-Paare mit einem Transversalimpuls größer 3 GeV. Die e^+e^--Paare und eine hohe Multiplizität in bestimmten Raumbereichen deuten indirekt auf die Produktion eines QGPs hin. Bei einer zentralen Kollision entstehen bis zu 20 000 primäre geladene Teilchen, die im Akzeptanzbereich des TRDs liegen. Die Aufgabe des TRDs ist es, aus dem Datenstrom der 1,2 Millionen Kanäle Spuren zu rekonstruieren, und solche hochenergetischen e^+e^- Paare zu erkennen.

In Abbildung 2.4 ist ein Beispiel für ein Teilchen mit hohem Transversalimpuls (rot) dargestellt. Man bezeichnet eine solche Spur als steif. In einem Detektormodul (Abbildung 2.3) hinterlässt ein solches Teilchen eine gerades Spursegment, das nach Abzug des Lorentz-Winkels nahezu parallel zur Driftrichtung steht. Das parallel zur Strahlachse verlaufende Magnetfeld lässt Teilchen mit niederem Transversalimpuls eine Kreisbahn (blau) beschreiben. Solche Teilchen erscheinen in einem Modul schräger oder gar gekrümmt. Durch diese Eigenschaften ist es möglich, schon auf Detektorebene hoch- und niederenergetische geladene Teilchen voneinander zu unterscheiden.

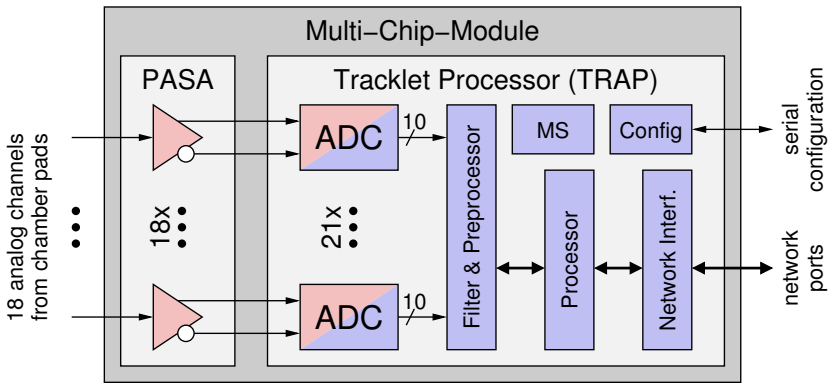

Abbildung 2.5: Ein MCM trägt zwei Chips, PASA und TRAP, mit denen er 18 Kanäle verarbeiten kann. Die analogen (rot) Signale werden durch den PASA verstärkt und digital (blau) gewandelt. Der Filter, Präprozessor und Prozessor verarbeiten die Daten. Die berechneten Spursegmente werden über die Netzwerkschnittstelle ausgelesen.

Die größte Zahl der produzierten geladenen Teilchen bei Blei-Blei-Kollisionen sind Pionen. Sie haben etwa den gleichen Transversalimpuls wie die gesuchten e^+e^- Paare. Pionen sind jedoch etwa 270-mal schwerer und damit langsamer als Elektronen und haben einen kleineren Lorentz-Faktor. Sie erzeugen also keine Übergangsstrahlung beim Eintritt in ein Modul. Anhand der Ladungsverteilung entlang einer Spur können somit die Elektronen von den Pionen unterschieden werden.

2.3.4 Zeitlicher Ablauf

Die TRD Elektronik verarbeitet während der Trigger einen Datenstrom von insgesamt etwa 17 Terabyte pro Sekunde und muss innerhalb von 6,5 µs eine Entscheidung treffen, ob ein interessantes Ereignis stattgefunden hat. Bis zur Übertragung der Daten zur globalen Tracking-Einheit arbeiten die einzelnen MCMs unabhängig voneinander und der zeitliche Ablauf kann anhand eines MCMs beschrieben werden.

Die Vorverstärker und die ADCs arbeiten kontinuierlich und ein Teil der Eingangswerte wird gespeichert, so dass beim Auslösen eines Prätriggers (kurz nach Zeitpunkt 0) Daten zur Verfügung stehen, die vor dem Prätrigger genommen wurden (Abbildung 2.6). Das bedeutet, dass die Daten ohne Verlust ab dem Ereignis (Interaktion) berücksichtigt werden können. Der Präprozessor arbeitet während der Driftzeit und filtert die Eingangswerte. Außerdem werden die Daten für die Bestimmung von Spurparametern so weit wie möglich vorverarbeitet, um die Rechenzeit nach der Driftzeit zu minimieren. Nach 2 µs wird der Präprozessor abgeschaltet und der Prozessor berechnet die endgültigen Spursegmente.

Ein MCM kann maximal bis zu vier Spursegmente verarbeiten. Die Prozessoren geben die berechneten Segmente an die Netzwerkschnittstelle (NI, Kapitel 6) weiter. Die NI ist in TRAP integriert. Mit ihr wird das Auslesenetzwerk des TRDs gebildet. Die Daten der verschiedenen MCMs werden über das Auslesenetzwerk gesammelt und an die globale Tracking-Einheit (GTU) geschickt. Durch die Reihenfolge der Auslese (Kapitel 5) und die

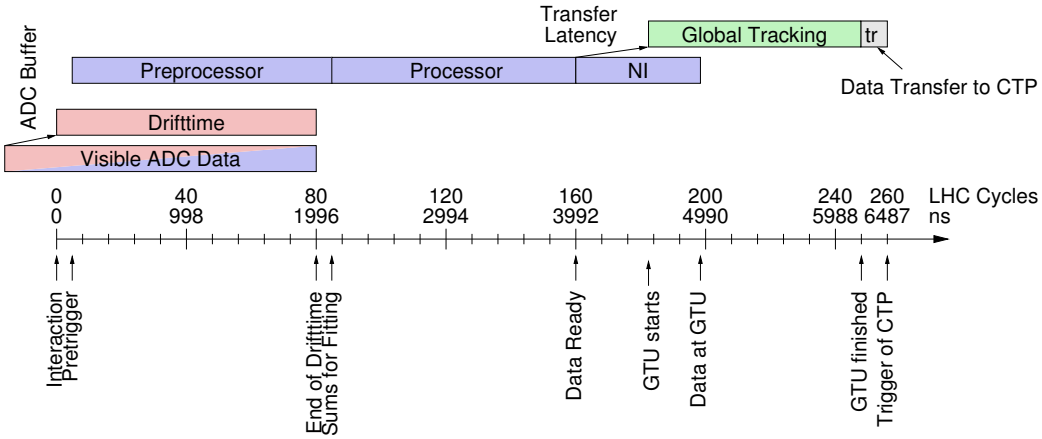

Abbildung 2.6: Zeitdiagramm für den TRD. Für die Auslese der Daten steht 1 µs zur Verfügung. Nach etwa 600 ns erreichen die ersten Daten (spätestens) die GTU, die somit starten kann. Nach weiteren 400 ns ist die Auslese fertig.

Arbeitsweise der GTU kann das globale Tracking noch während der Auslese nach etwa 4,2 µs starten. Hierdurch ist es notwendig, dass sowohl die initiale Latenz als auch die Dauer für die Übertragung der Daten möglichst gering sind. Die Ergebnisse der GTU werden nach 6,4 µs an den CTP gegeben, der dann den L1-Trigger auslöst.

3 Simulation

Für die Simulation steht ein umfangreiches Softwarepaket (AliRoot, [36]) zur Verfügung, mit dessen Hilfe man das erwartete Verhalten des gesamten Experiments nachbilden kann. Es beinhaltet die Geometrie der Detektoren, Ereignisgeneratoren, die Nachbildung der physikalischen Wechselwirkungen der Teilchen mit den Detektoren und die Verarbeitung der erhaltenen Messdaten. Als Besonderheit ist in dieser Softwareumgebung nicht nur die Simulation des Experimentes im Betrieb, sondern gleichzeitig die gesamte Offline-Analyse integriert.

Dieses Kapitel beschreibt die Simulation des TRDs, in der die Anforderungen an die lokalen Tracking-Einheiten (LTUs) unter Berücksichtigung der Trigger-Funktionalität untersucht werden. Die Implementierung der LTU erfolgt durch den Tracklet-Prozessor (TRAP) und ist in Kapitel 4 näher beschrieben.

Linux-Kernel	2.4.0
SuSE Linux	7.0
GCC	2.95.2 19991024
ROOT	2.25_03
AliRoot	3.04

Tabelle 3.1: Alle nachfolgenden Simulationsergebnisse wurden mit den hier dargestellten Programmversionen erzeugt. Weitere Informationen können Anhang B entnommen werden.

Die vorgestellten Ergebnisse sind mit den in Tabelle 3.1 genannten Softwareversionen erstellt worden. Es gibt inzwischen neuere Versionen, die die Physik noch genauer beschreiben, und es haben sich Änderungen in der Geometrie des TRDs ergeben. Allerdings spielen diese für die Entwicklung des Konzeptes keine entscheidende Rolle. Es wird an den entsprechenden Stellen darauf hingewiesen.

3.1 Motivation

Der TRD wandelt zunächst Spuren geladener Teilchen, die durch eine Driftkammer fliegen, in elektrische Signale um. Die hier gezeigte Simulation setzt an dieser Schnittstelle an. Dies entspricht in der daraus folgenden Implementierung den Eingängen der ADCs des TRAPs (Abbildung 2.5). Die Hauptaufgabe des TRDs ist es, als Trigger zu dienen. Das Verhältnis aus Verarbeitungsgeschwindigkeit und Präzision muss daher daraufhin optimiert werden, dass der Trigger in der sehr kurzen Zeit von 6,5 µs seine Aufgabe adäquat erfüllt.

Die spezielle Aufgabenstellung an diese Simulation ist es, nicht nur eine Hardware-nahe Beschreibung der Funktionsweise des Tracklet-Prozessors zu entwickeln, sondern vielmehr das Zusammenspiel der LTUs, des Auslesenetzwerks und der globalen Tracking-Einheit (GTU) zu untersuchen. Daraus ergeben sich jeweils Anforderungen an die einzelnen Module, angefangen vom dynamischen Bereich der ADCs bis hin zur Verarbeitungsbandbreite der GTU.

Die wichtigsten Konzepte der LTU waren zu Beginn der Untersuchungen vorhanden und mussten auf die Gesamtfunktionalität hin untersucht und optimiert werden. Dabei muss besondere Rücksicht auf die Implementierbarkeit in Hardware genommen werden. Das Auslesenetzwerk bildet die Schnittstelle zwischen den LTUs und der GTU. Für einen Entwurf dieses Netzwerks ist es notwendig, ein grundlegendes Konzept für die GTU zu entwickeln und daraufhin das Auslesenetzwerk und die dafür benutzte Netzwerkschnittstelle des TRAPs zu entwerfen.

Der Tracklet-Prozessor ist mithilfe der Hardware-Beschreibungssprache VHDL[1] entworfen. Ohne großen Aufwand ist es nicht möglich, ein Verhaltensmodell in dieser Form der Beschreibung in die AliRoot-Umgebung zu integrieren. Es besteht also das Problem der möglichen Inkonsistenz zwischen Simulation und entwickelter Hardware. Da zunächst die Ergebnisse der Simulation die Architektur der Hardware bestimmen, wird wie folgt verfahren: es wird mithilfe von AliRoot nach geeigneten Algorithmen für die einzelnen Verarbeitungsschritte gesucht und diese werden auf ihre Wirkung für die Gesamtfunktionalität überprüft. Dieser Schritt der Entwicklung wird in diesem Kapitel beschrieben. Die Algorithmen werden mittels VHDL in die entsprechenden Architekturen umgesetzt. Dabei ergeben sich gegebenenfalls Änderungen (bedingt durch Implementierbarkeit, Geschwindigkeit, Platzbedarf, usw.), die wieder in die Simulation übernommen und geprüft werden. Die Implementierung der Module wird im Kapitel 4 beschrieben. Weiterhin werden Untersuchungen der aktuellen Verhaltensmodelle am Physikalischen Institut der Universität Heidelberg [20] angestellt.

3.2 AliRoot-Softwareumgebung

Die für ALICE entwickelte Softwareumgebung AliRoot ist ein objekt-orientiertes Framework für Simulation, Rekonstruktion und Analyse. Es basiert auf dem Datenanalyse-Framework ROOT. AliRoot beinhaltet die Simulation der primären physikalischen Ereignisse und den Teilchen-Transport bzw. die Reaktionen der Teilchen mit den Detektoren. Dieser Teil der Simulation wird als Ereigniserzeugung bezeichnet und greift auf externe Module wie z. B. Pythia oder Geant zurück, die in den folgenden Abschnitten näher beschrieben werden. Zusammen mit der ebenfalls in AliRoot enthaltenen Simulation der Antwort der Detektoren und der Verarbeitung der Daten sind damit alle während des Betriebs aktiven Komponenten von ALICE beschrieben. Zusätzlich findet hier die Analyse der gespeicherten Daten statt. Damit ist es möglich, sowohl die simulierten Ereignisse als auch die im Betrieb aufgenommenen Daten in derselben Umgebung auszuwerten. Da die Generierung

[1] V(ery High Speed Integrated Circuit) Hardware Description Language

eines Ereignisses sehr rechenaufwendig ist und eine vollständige Bearbeitung bis zu einigen Stunden dauern kann, ist außerdem eine Schnell-Simulationsumgebung implementiert. Man stellt hierzu Erkenntnisse aus der Simulation und Messungen in vereinfachter Form zur Verfügung und erreicht so – auf Kosten der Genauigkeit – eine bessere Statistik.

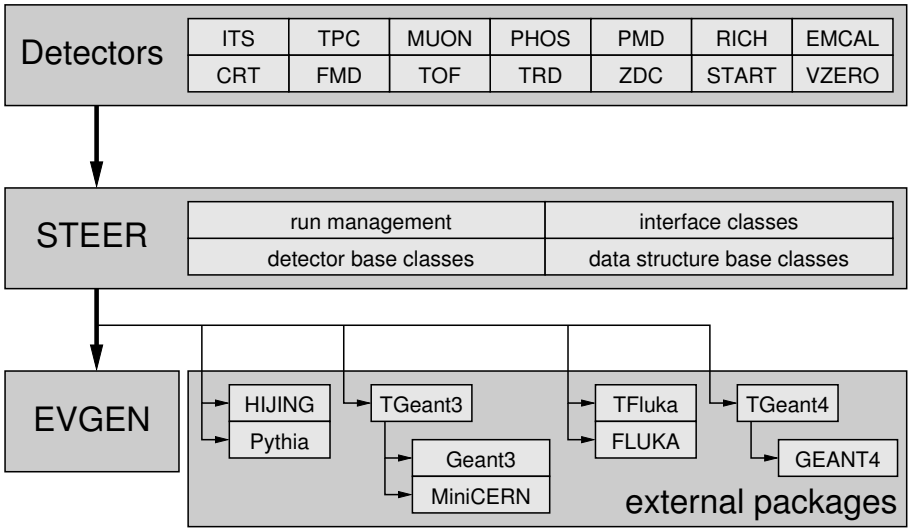

Abbildung 3.1: Komponenten der AliRoot-Simulationsumgebung. Quelle: [12]

Das zentrale Modul der AliRoot-Umgebung (Abbildung 3.1) ist STEER. Es stellt die Schnittstellen zwischen den externen Modulen und den Detektorbeschreibungen und internen Datenstrukturen zur Verfügung. Außerdem verwaltet STEER den Ablauf der Ereigniserzeugung.

Die Detektormodule sind in der Simulation in ihre Unterdetektoren gegliedert und enthalten jeweils die Geometrieinformationen, die Analyseumgebung und Methoden für die Wandlung der Eingangsdaten, die aus der Monte-Carlo-Simulation stammen. Jedes dieser Module ist unabhängig von den anderen, d. h. es gibt keine Datenabhängigkeiten zwischen den Detektorklassen. Somit können die Module separat voneinander entwickelt werden.

Die externen Module beinhalten die Ereigniserzeugung. Das HIJING[2]- und das Pythia[3]-Paket [14][15] sind Bibliotheken zur Beschreibung primärer physikalischer Ereignisse. AliRoot bietet zudem die Möglichkeit, „Cocktails“ aus unterschiedlichen Erzeugern zu produzieren. Damit ist es möglich, die Vorteile verschiedener Pakete zu vereinen und zu Testzwecken und zur Konstruktion von Detektoren angepasste Ereignisse zu erzeugen. Zusätzlich können Benutzer über das EVGEN-Modul eigenen Kode zur Ereigniserzeugung hinzufügen.

[2]Heavy Ion Jet INteraction Generator, http://www-nsdth.lbl.gov/∼xnwang/hijing
[3]http://www.thep.lu.se/∼torbjorn/Pythia.html

3.2.1 HIJING und Pythia

In AliRoot werden die Monte-Carlo-Generatoren HIJING und Pythia eingesetzt. Mit ihnen werden physikalische Prozesse beschrieben, speziell Kollisionen hochenergetischer Teilchen, indem gemäß vorher definierten bzw. theoretisch bestimmten Verteilungsfunktionen zufällige Ereignisse generiert werden. Häufig sind diese Funktionen Energie-, Impuls- und Winkelverteilungen. Während Pythia zur Beschreibung von Elementarteilchen wie Elektronen oder Protonen optimiert ist, werden mit HIJING bevorzugt Wechselwirkungen von Schwerionen simuliert.

3.2.2 Geant

Um konkrete Aussagen über die von HIJING und Pythia generierten Ereignisse zu machen, muss studiert werden, wie die Teilchen bzw. deren Zerfallsprodukte mit dem Detektormaterial wechselwirken. Hierzu wird häufig – wie auch in diesem Fall – die Geant3-Software verwendet. Als Optionen sind in AliRoot die weiteren Pakete GEANT4 und FLUKA für zukünftige Simulationen vorgesehen.

Die Eingangsdaten für Geant sind die monte-carlo-generierten Teilchendaten und die Geometriedaten aus den Detektormodulen. Geant simuliert dann das Passieren der Elementarteilchen durch die Materie, wobei es die physikalischen Eigenschaften, die Wechselwirkung mit der Materie und das Umfeld, wie z. B. ein magnetisches Feld, berücksichtigt. Außerdem bietet Geant eine Schnittstelle zur Visualisierung der Detektoren und der Teilchen-Trajektorien. Bei dieser detaillierten Analyse sind aufwendige Rechenoperationen nötig, so dass schon für niedrige Statistik viel Rechenzeit in Anspruch genommen wird.

Als Eingangsdaten für die Analyse innerhalb der Detektormodule erzeugt Geant so genannte „Hits“[4]. Sie enthalten genaue Informationen eines Teilchens an einer Übergangstelle von zwei verschiedenen Materien. Damit kann eine Antwort des Detektors auf ein stattgefundenes Ereignis erzeugt werden. Ein Detektormodul wie z. B. der TRD simuliert so die Driftkammer und erzeugt für ein Volumenelement einen entsprechenden Zahlenwert, das „Digit“. Im Falle des TRDs ist hierin schon der Drift, der Lorentz-Winkel, Rauschen, Vorverstärkung und die Analog-Digital-Wandlung enthalten. Die Digits entsprechen in diesem Fall also den ADC-Werten des Tracklet-Prozessors.

3.2.3 ROOT

ROOT[5] ist ein objekt-orientiertes Datenanalyse-Framework, das speziell für die Bedürfnisse von Hochenergieexperimenten entwickelt ist. Mit ihm lassen sich sehr große Datenmengen verwalten und analysieren, und es bietet Schnittstellen zu den in diesem Umfeld

[4] Diese „Hits“ sind nicht zu verwechseln mit dem ebenso bezeichneten Zusammenfassen dreier ADC-Werte innerhalb des Präprozessors, wie es in Kapitel 3.3 beschrieben ist.

[5] http://root.cern.ch

häufig benutzten Programmen. ROOT ist frei verfügbar und kann auf allen gängigen Systemen installiert werden. Häufig benutzte Klassen sind z. B.: Histogramme und Fitting, Eingabe/Ausgabe (Datenstromerstellung), Grafische Benutzerschnittstellen und 2D/3D-Grafik.

Außerdem bietet ROOT mit dem von Masaharu Goto entwickelten CINT, einem C++-Kommandozeilen-Interpreter, eine komfortable Schnittstelle, die in das C++-Framework integriert ist.

Das ROOT-Framework wurde von René Brun und Fons Rademakers Mitte der Neunziger im Zusammenhang mit dem NA49[6]-Experiment am CERN entwickelt. NA49 hat – gerade für damalige Verhältnisse – enorme Datenmengen produziert. Für jeden Lauf mussten mehr als 10 Terabyte an Daten verarbeitet und analysiert werden.

3.2.4 Simulationsklassen und Skripte

Im Folgenden wird auf die Klassen und Skripte der TRD-Umgebung eingegangen, die für die dargestellten Simulationen relevant sind. AliTRDgeometry enthält die für das externe Modul Geant benötigten Geometriedaten. Geant erstellt hiermit so genannte „Hits“, die in der von AliHit abstammenden Containerklasse AliTRDhit gespeichert werden. Eine Hit-Instanz enthält die Koordinaten des Hits, die Ladung und die Identifikation des Tracks bzw. Informationen über die Ursache der Erzeugung. Durch AliTRDdigitizer wird das Verhalten der Kammer simuliert, indem die Hits für ein jeweiliges Volumenelement integriert werden.[7] Ein Volumenelement ergibt sich dabei aus der Fläche des betroffenen Pads und einem Zeitabschnitt. Das Ergebnis entspricht dann einem ADC-Wert zu einem Zeitpunkt und wird in AliTRDdigit verwaltet. Wie alle Containerklassen enthält auch diese Referenzen auf die Daten, aus denen ihre Inhalte erzeugt werden. Die bisher genannten Klassen sind fester Bestandteil von AliRoot und werden für die folgenden Untersuchungen als Framework genutzt.

Alle Containerklassen besitzen eine Streamer-Funktion, mit der sich die Dateninhalte der Klasse in eine ROOT-Datei speichern lassen. ROOT stellt damit die Möglichkeit zur Verfügung, einen Zustand der Simulation zu speichern und durch Laden dieser Datei später wieder an dieser Stelle fortzufahren. Da die Simulation eines Ereignisses bis zu einigen Stunden dauern kann, werden auf diese Weise Ereignis-Dateien erzeugt, die alle Informationen bis einschließlich der Digits enthalten. Hiermit kann die Ereigniserzeugung mittels Geant und AliTRD (Abbildung 3.2, links) von der Analyse getrennt werden.

Alle Simulationen sowie die anschließenden Analysen werden durch ROOT-Skripte gesteuert. Das Skript liest zunächst die ROOT-Datei und die Bibliotheksklassen ein, die nicht Bestandteil von AliRoot sind. In diesem Fall sind das die Klassen AliTRDfit und AliTRDgtu. Der größte Teil der Analyse findet in den Skripten selbst statt, indem die Daten aus AliRoot entweder direkt ausgewertet oder an die eingebundenen Klassen weitergegeben und

[6] siehe http://na49info.cern.ch

[7] Inzwischen wurden genauere Modelle der Driftkammer entwickelt, die den durch das Magnetfeld bewirkten Lorentz-Winkel und auch exaktere Pad-Antwort-Funktionen enthalten.

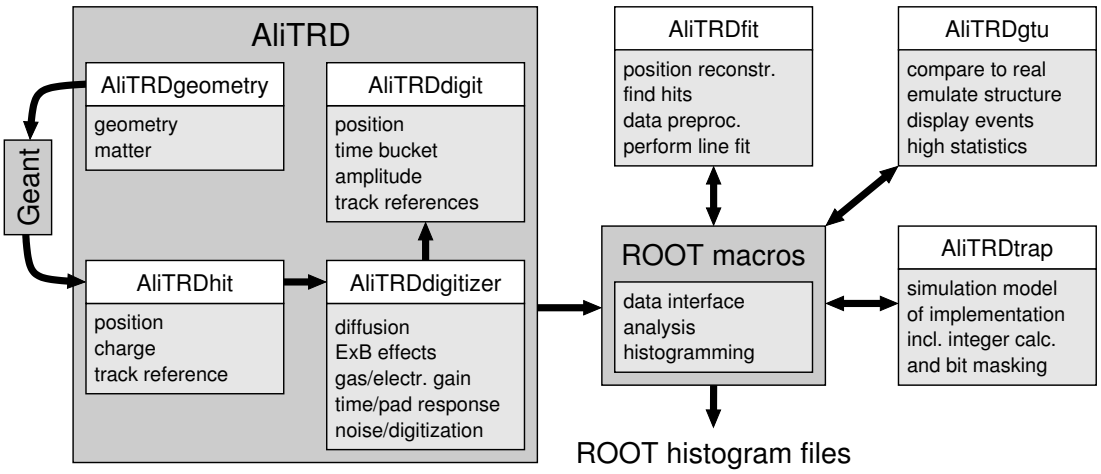

Abbildung 3.2: Klassen und Container in Aliroot und die externen Klassen zur Beschreibung der LTU und GTU. Gezeigt sind der Datenfluss und die Simulationsmodule und deren wichtigste Aufgaben bzw. Inhalte.

die Ergebnisse ausgewertet werden. Da es sich bei den Analysen zum Teil um sehr rechenaufwendigen Kode handelt und die eingebundenen Klassen auch Zugriff auf alle Daten in AliRoot haben, sind Teile der Analyse in diesen vorkompilierten Klassen implementiert.

Im Allgemeinen erfolgt die Ausgabe der Ergebnisse der Analysen in Form von Histogrammen wie z. B. in Abbildung 3.4. ROOT bietet eine umfangreiche Umgebung zur Verwaltung dieser Histogramm-Klassen und die Daten lassen sich ebenfalls als ROOT-Dateien speichern. Dadurch lassen sich solche Ausgaben leicht formatieren, überlagern, etc. oder sogar zu weiteren Analysen verwenden. Eine Liste der ROOT-Dateien (siehe Datenträger) aller Histogramme findet sich in Anhang B.2.

3.3 Funktion der lokalen Tracking-Einheit

Die lokale Tracking-Einheit (LTU) wandelt die analogen Eingangssignale in digitale Werte und speichert diese. Die Hauptaufgabe der LTU als Bestandteil des Triggers ist es, Spuren mithilfe der Zeit-Ort-Korrelation im Zweidimensionalen zu rekonstruieren. Diese Spuren werden im Folgenden Spursegmente genannt. Im lokalen Koordinatensystem entspricht die y-Achse der Position entlang einer Padzeile, die x-Achse der Driftrichtung der ionisierten Gasteilchen. Die Driftgeschwindigkeit der Teilchen lässt sich als konstant annehmen und somit die x-Position eines Wertes über die Zeit ermitteln. Abbildung 3.3 zeigt ein Beispiel für eine Padzeile in der Simulation. Ein Rechteck stellt einen ADC-Wert dar. Die roten Linien markieren den tatsächlichen Eintritts- und Austrittspunkt eines Tracks aus der Simulation.

Der in Kapitel 4.1 beschriebene Tracklet-Prozessor realisiert die Funktionalität einer LTU und verarbeitet 18 Kanäle in einer Padzeile. Im Folgenden wird als LTU das funktionale System mit dieser Kanalanzahl bezeichnet.

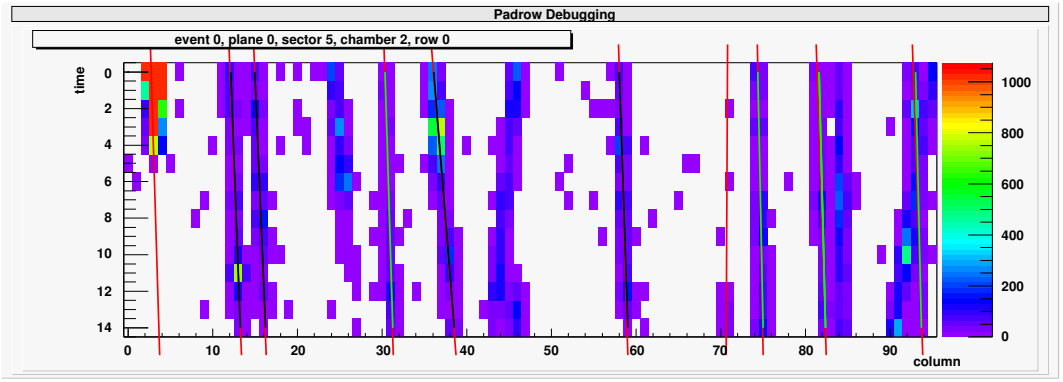

Abbildung 3.3: Ereignisdarstellung für eine Padzeile. Dargestellt sind die ADC-Werte über der Zeit und den Pads. Die Werte sind durch die Farbskala kodiert. Die roten Linien markieren den tatsächlichen Track, d.h. wie er von der Monte-Carlo-Simulation generiert wurde. Die grünen und schwarzen Linien stellen die durch die LTUs gefundenen Spursegmente dar. Die schwarzen Linien sind dabei Spursegmente, die z. B. aufgrund von Beeinflussung durch Nachbarspuren oder niedriger ADC-Werte zu ungenau sind und verworfen werden.

Die LTU bestimmt zu jedem Zeitabschnitt zusammengehörige ADC-Werte (Hits) und ermittelt aus diesen Werten durch Ausnutzung der Ladungsteilung an den Pads die Position eines Hits. Maximal können bis zu vier Hits pro LTU erzeugt und ausgewertet werden. Die Spuren werden mittels eines linearen Regressionsfits an diese Punkte rekonstruiert.

Gesucht werden Spuren, die möglichst steif sind (Kapitel 2.3), die also vom Vertex kommen und eine nahezu gerade Linie bilden. Bei einem linearen Fit lässt sich die Qualität der Spur, also wie gerade eine Spur ist, an den Residuen bzw. der Standardabweichung des Fits erkennen. Um festzustellen, ob eine Spur vom Vertex kommt, wird die ermittelte Steigung der Spur mit der Position innerhalb des Moduls verglichen. Aufgrund der Geometrie des TRD (Kapitel 2.3.2) trifft bereits ein sehr steifes Teilchen mit einem Winkel von bis zu $\pm 10°$ auf ein Modul. Misst man den Winkel bzw. die Steigung der Spur relativ zu dem durch die Geometrie bedingten Winkel, kann man eine Aussage über den Transversalimpuls und das Vorzeichen der Ladung des Teilchens machen. Allerdings können auch sekundäre Teilchen mit einem Transversalimpuls kleiner als 1 GeV in einem vermeintlich interessanten Winkel auf ein Detektormodul treffen. Diese lassen sich lokal nur durch die Krümmung und damit anhand der Qualität der Spur erkennen.

In der Simulation verwendet man an Stelle des Winkels bzw. der Steigung die Ablenkung eines Teilchens. Diese entspricht dem Produkt aus der Steigung und der Anzahl der Zeitabschnitte (hier 15). Abbildung 3.4 zeigt die Differenz der Ablenkungen eines betrachteten Teilchens zu einem Teilchen, das unendlichen Transversalimpuls hat. Für Teilchen, die vom Primärvertex stammen, lässt sich somit ein eindeutiger Zusammenhang zwischen der Differenz der Ablenkung und dem Transversalimpuls zeigen. Die Spiegelsymmetrie des Histogramm zur p_t-Achse rührt von der unterschiedlichen Ladung der Teilchen her. Erkannte Spuren außerhalb der beiden Kurven stammen entweder von Sekundärvertices oder wurden durch Überlagerung falsch erkannt. Idealerweise sollte es für $p_t < 0,2\,\text{GeV}$ keine Teilchen geben, da diese durch ihren kleinen Radius gar nicht den TRD erreichen. In Abbildung 3.5 ist

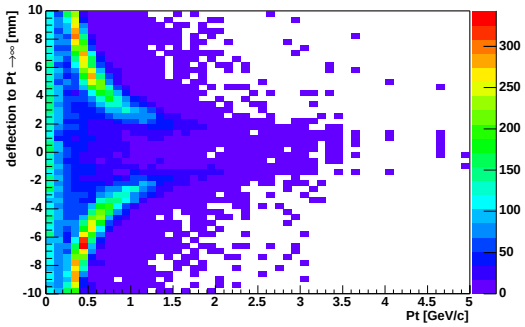

Abbildung 3.4: Ablenkung eines Spursegments zu einer geraden Spur vom Vertex über dem Transversalimpuls.

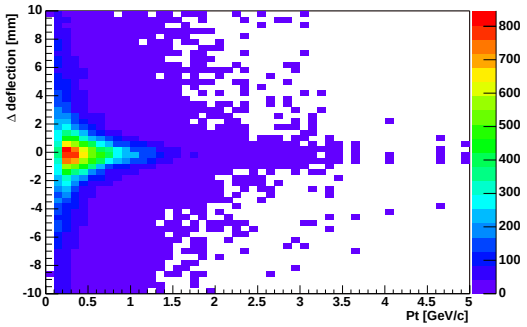

Abbildung 3.5: Differenz der Ablenkung zum errechneten Wert über dem Transversalimpuls.

der Transversalimpuls über der Differenz der gemessenen und der errechneten Ablenkung aufgetragen.

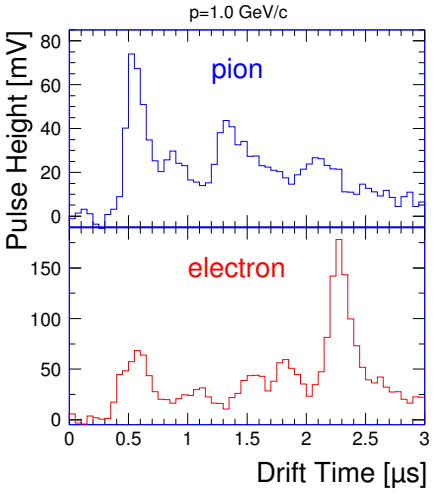

Abbildung 3.6: Beispiel für eine Ladungsverteilung über die Driftzeit für ein Pion und ein Elektron. Man beachte die verschiedenen Skalen für die Pulshöhe. Quelle: [5]

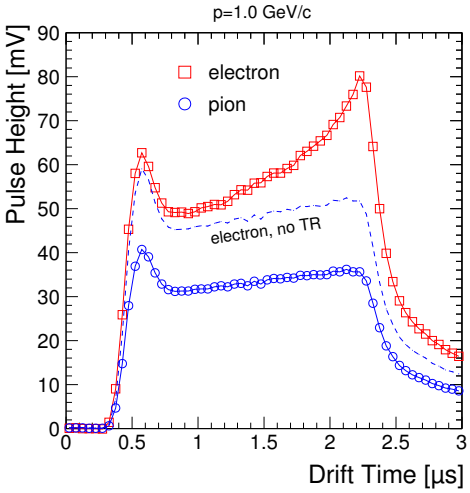

Abbildung 3.7: Mittlere Ladungsverteilung entlang der Driftzeit für Pionen und Elektronen. Quelle: [5]

Die Trennung von Pionen und Elektronen findet anhand der verschiedenen Ladungsverteilungen statt. Abbildung 3.6 zeigt eine typische Ladungsverteilung über der Driftzeit. Sowohl beim Pion als auch beim Elektron gibt es zu Beginn der Driftzeit durch eine erhöhte Ladungsbildung in der Verstärkungszone einen Peak. Insgesamt erhält man durch ein Elektron im Mittel eine höhere Ladung (Abbildung 3.7). Am Ende der Driftzeit (entspricht im Raumbereich nahe am Radiator) kann man außerdem die erhöhte Ladung erkennen, die bei hochenergetischen Elektronen durch ein Übergangsstrahlungsphoton erzeugt wird. Der leichte Anstieg des Signals, sowohl bei Pionen als auch bei Elektronen ohne erzeugter Übergangsstrahlung, wird durch einen Ionenschweif hervorgerufen und durch den Energieverlust

durch Ionisation in der Driftkammer nach der Bethe-Bloch-Formel [16][17] noch verstärkt. In der Verstärkungsregion (Kapitel 2.3.1) driften die positiv geladenen Ionen zur Kathode. Durch diese Vergrößerung des Abstandes zu den Pads nimmt die induzierte Ladung ab. Einem relativ schnellen Anstieg eines Signals folgt demnach ein langsamer Abfall, was sich hier nach Ende der Driftzeit von 2 µs gut beobachten lässt. Während der Driftzeit wird dieser Effekt durch die Ionenschweifunterdrückung (Kapitel 4.1.2) weitgehend kompensiert.

Bei der Entwicklung der LTU wird unterschieden, welche Berechnungen während der Driftzeit und welche danach durchgeführt werden. Bei einem linearen Regressionsfit nach den Gleichungen 3.1–3.3 können während der Aufnahme der Werte (hier Position und Zeit) schon die Summenterme berechnet werden, die dann nach der Driftzeit weiterverarbeitet werden. Die Bestimmung dieser Summen ist Teil des Präprozessors. Nach der Datenaufnahme kann der Präprozessor abgeschaltet werden, und der Prozessor übernimmt die vorverarbeiteten Werte, bestimmt die endgültigen Spurparameter und kodiert diese (Kapitel 3.5.4).

3.3.1 Positionsbestimmung

Unter der Positionsbestimmung versteht man die Rekonstruktion der Position eines Spurabschnitts in einem Zeitabschnitt. Die Größen der Pads sind im Verhältnis zu den Ladungswolken, die durch Ionisation eines Teilchens erzeugt werden, so gewählt, dass etwa 90 % der Ladung über den Bereich einer Padbreite deponiert werden. Auf diese Weise lässt sich mithilfe der Ladungsteilung und Ermittlung des Schwerpunktes (center of gravity, COG) eine genauere Position in y-Richtung ermitteln (Abbildung 3.9). Nimmt man als Referenzpunkt das Pad mit dem größten Wert, kann in erster Näherung die Position durch $y_m = \frac{R-L}{L+C+R}$ berechnet werden (siehe auch Gleichung 3.6). Die so errechnete Position wird dann mittels einer programmierbaren Tabelle korrigiert. Zum einen lässt sich die reale Pad-Antwort-Funktion nicht durch eine einfache Funktion annähern, zum anderen sind durch die Geometrie des Detektors nicht alle Pads gleich breit, so dass die Ladungsteilung unterschiedlich ist (Abbildung 3.8).

Die benötigte Auflösung des Detektors für eine adäquate Bestimmung des Transversalimpulses zum Triggern der Auslese beträgt etwa 400 µm [5]. Die Größe der Pads in y-Richtung ist gegeben durch die benötigte Ladungsteilung. In z-Richtung ist die Größe limitiert durch die Anzahl der erwarteten Spuren im Detektor. Für eine saubere Ortsbestimmung dürfen die Ladungswerte nicht durch Werte anderer Spuren „verschmutzt“ sein, d. h. es gibt eine maximale Belegung [5], bei der der Detektor ideal arbeitet. Diese Ausnutzung des Detektors (occupancy) liegt bei etwa 20 %. Die Pads haben, je nach Position im Detektor, eine Größe von 0,65–0,80 cm in y- und 7,53–10,11 cm in z-Richtung.

Die Bestimmung des Ortes wird durch mehrere Faktoren negativ beeinflusst. Neben den physikalischen Effekten, wie z. B. den Tails und dem Übersprechen (Kapitel 4.1.2), verfälschen Nichtlinearitäten der Vorverstärker und ADCs die Messungen. Diese können mithilfe

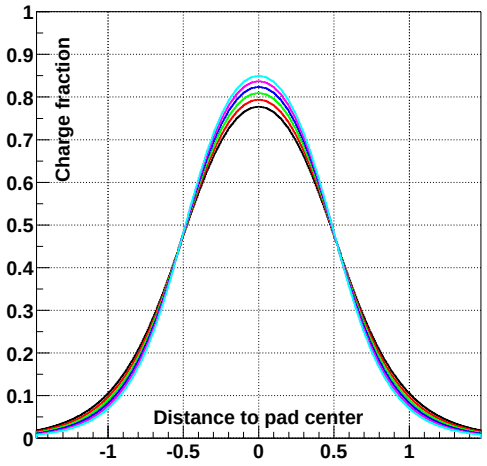

Abbildung 3.8: Pad-Antwort-Funktion für verschiedene Padbreiten (Lagen). Quelle: [5]

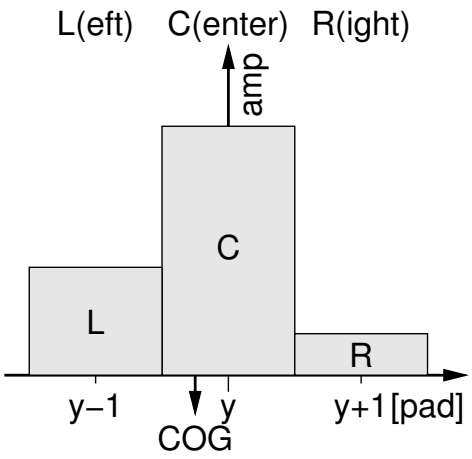

Abbildung 3.9: Positionsbestimmung mittels Ladungsteilung.

von Tabellen weitgehend korrigiert werden. Die entsprechenden Messungen und Analysen werden im Physikalischen Institut in Heidelberg [20] durchgeführt und den aktuellen Änderungen der Kammer angepasst.

3.3.2 Spursegment-Berechnung

Die Spurabschnitte werden durch einen linearen Regressionsfit angenähert. Da nur sehr gerade Spuren erwartet bzw. gesucht werden, reicht diese Form der Annäherung vollkommen aus. Sie bietet den Vorteil, mit einfachen Operationen berechnet werden zu können. Außerdem lassen sich die wichtigsten Terme für die Berechnung bereits während der Aufnahme der Messwerte (Driftphase) ermitteln. Dies führt zu einer enormen Verkürzung der Berechnungsphase, die zwischen der Drift- und der Auslesephase liegt (siehe auch Abbildung 2.6). In den für die Berechnung gezeigten Gleichungen für die Gerade entspricht y_i der Position in y-Richtung (entlang einer Padreihe) und x_i dem Zeitabschnitt (Timebin), was der Position in x-Richtung entspricht. Noch während der Driftphase, die etwa 2 µs dauert, können die Werte für $\sum x_i$, $\sum y_i$, $\sum x_i^2$ und $\sum x_i y_i$ berechnet werden. Da die Werte der Messreihe nicht zu jedem Zeitpunkt die Kriterien für einen Hit erfüllen, kann die Anzahl der Messwerte N geringer sein als die Anzahl der Messungen.

In der Regressionsgeraden $y_{(x)} = a + bx$ sind der Achsenabschitt und die Geradensteigung gegeben durch [38]:

$$a = \frac{1}{\Delta} \begin{vmatrix} \sum y_i & \sum x_i \\ \sum x_i y_i & \sum x_i^2 \end{vmatrix} = \frac{1}{\Delta} \left(\sum x_i^2 \sum y_i - \sum x_i \sum x_i y_i \right) \tag{3.1}$$

$$b = \frac{1}{\Delta} \begin{vmatrix} \sum N & \sum y_i \\ \sum x_i & \sum x_i y_i \end{vmatrix} = \frac{1}{\Delta} \left(N \sum x_i y_i - \sum x_i \sum y_i \right) \tag{3.2}$$

$$\Delta = \begin{vmatrix} \sum N & \sum x_i \\ \sum x_i & \sum x_i^2 \end{vmatrix} = N \sum x_i^2 - \left(\sum x_i \right)^2 \tag{3.3}$$

Der Prozessor (Kapitel 4.1.4) kann bis zu vier solcher Datensätze gleichzeitig bearbeiten und somit bis zu vier Spurabschnitte erzeugen. Hierzu werden die Daten nach Kriterien wie z. B. Anzahl der Messwerte (Hits) untersucht und ausgewählt. Nach der Driftphase sind also lediglich die Multiplikationen und eine Division notwendig, um die endgültigen Spurparameter zu bestimmen. Um eine Aussage über die Qualität des Fits machen zu können, benutzt der Präprozessor den auch für den Fit benötigten Term $\sum y_i^2$. Die Varianz für die Regression mit zwei Freiheitsgraden ist [38]:

$$\sigma^2 \simeq \frac{1}{N-2} \sum (y_i - a - bx_i)^2 \tag{3.4}$$

$$= \frac{1}{N-2} \left(\sum y_i^2 + Na^2 + b^2 \sum x_i^2 - 2a \sum y_i - 2b \sum x_i y_i + 2ab \sum x_i \right) \tag{3.5}$$

Mithilfe einer vereinfachten Gleichung 3.5 [21] für die Varianz lässt sich nach der Bestimmung der Fitparameter durch wenige einfache Operationen ein Qualitätsmaß für die Regression bestimmen. Dieses Maß und der mithilfe der Steigung abgeschätzte Transversalimpuls des Teilchens entscheiden, ob der Spurabschnitt zur Weiterverarbeitung an die globale Tracking-Einheit geschickt oder verworfen wird.

3.3.3 Fehlerbetrachtung

Quantisierungsfehler

Die ADCs arbeiten mit einer Halbbitgenauigkeit, d. h. jeder Wert hat einen maximalen Fehler von 0,5. Im ungünstigsten Fall wird also ein möglicher Ausgangswert übersprungen (Abbildung 3.10). Diese Art des Quantisierungsfehlers, der die Kennlinie beeinflusst, ist für jeden ADC ein systematischer Fehler. Ein weiterer systematischer Fehler wäre z. B. eine Nullpunktverschiebung durch eine ungenaue Referenzspannung.

Für eine große Anzahl an ADCs kann man davon ausgehen, dass diese systematischen Fehler eine gaußsche Verteilung aufweisen und man den statistischen Fehler mit einer Standardabweichung von $\sigma = 0,5$ hinreichend gut beschreiben kann. Dieser Fehler ist nicht zu verwechseln mit einem möglichen rechteckverteilten statistischen Fehler für das LSB eines einzelnen ADCs im Bereich $[-0,5; 0,5]$. Hierfür wäre die Standardabweichung $\sigma = \frac{1}{\sqrt{12}}$. Dieser Quantisierungsfehler (Abbildung 3.11) kann ebenfalls durch eine gaußsche Verteilung angenähert werden.

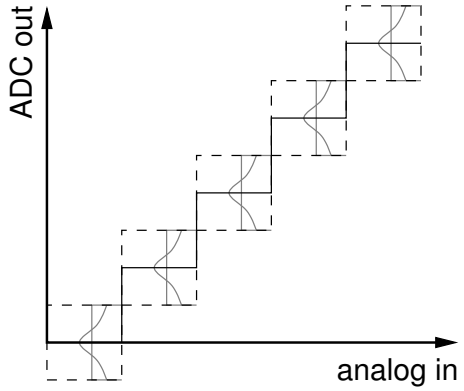

Abbildung 3.10: Mögliche Kennlinienfehler durch die Halbbitgenauigkeit.

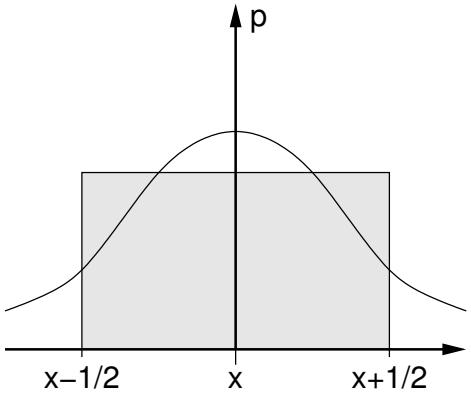

Abbildung 3.11: Quantisierungsfehler für einen Kanal eines ADCs.

Hit-Positionsfehler durch Quantisierung

Für den Trigger wird die Pad-Antwort-Funktion (Abbildung 3.8) durch die rationale Funktion nach Gleichung 3.6 angenähert. Der Fehler, der durch diese Annäherung entsteht, bleibt im Folgenden außer Acht, da er mithilfe von Tabellen korrigiert werden kann. Die Amplitude des zentralen Pads und die seines rechten und linken Nachbarn sind mit C, R und L bezeichnet (Abbildung 3.9). Die erste Näherung der Bestimmung der Position $y_m(L, C, R)$ erfolgt durch:

$$y_m = \frac{d}{s} \quad , \quad d = R - L \quad , \quad s = L + C + R \tag{3.6}$$

Für die Hit-Summe s wird ein bestimmter Mindestwert vorausgesetzt.

Mit den gaußverteilten Fehlern der Eingangsgrößen $\Delta\overline{L}$, $\Delta\overline{C}$ und $\Delta\overline{R}$ ergibt sich der Positionsfehler zu:

$$\Delta\overline{y_m} = \sqrt{\left(\frac{\partial y_m}{\partial L}\Delta\overline{L}\right)^2 + \left(\frac{\partial y_m}{\partial C}\Delta\overline{C}\right)^2 + \left(\frac{\partial y_m}{\partial R}\Delta\overline{R}\right)^2} \tag{3.7}$$

$$\Delta\overline{y_m} = \sqrt{\left(\frac{-s-d}{s^2}\Delta\overline{L}\right)^2 + \left(\frac{-d}{s^2}\Delta\overline{C}\right)^2 + \left(\frac{s-d}{s^2}\Delta\overline{R}\right)^2} \tag{3.8}$$

Schätzt man die Standardabweichung der Eingangswerte mit

$$\Delta\overline{L} = \Delta\overline{C} = \Delta\overline{R} =: \Delta ADC \tag{3.9}$$

ab, erhält man für $\Delta\overline{y_m}$ den vereinfachten Ausdruck:

$$\Delta\overline{y_m} = \frac{1}{s^2}\sqrt{2s^2 + 3d^3} \cdot \Delta ADC \approx \frac{\sqrt{2}}{s} \cdot \Delta ADC \tag{3.10}$$

Regressionsfehler allgemein

Wie in den Gleichungen 3.2–3.3 dargestellt, ist in der Regressionsgeraden $y_{(x)} = a + bx$ die Geradensteigung gegeben durch:

$$b = \frac{N \sum y_i x_i - \sum x_i \sum y_i}{N \sum x_i^2 - \left(\sum x_i\right)^2} \tag{3.11}$$

Es werden N Messungen durchgeführt ($i = 1, 2, ...N$). Der diskrete Wert x_i entspricht dem Zeitabschnitt und lässt sich direkt in eine Position innerhalb der Driftkammer umrechnen. Die Position der gemessenen Ladung ist y_i. Mit der Abschätzung des Vertrauensbereiches des Mittelwertes der einzelnen Messgrößen

$$\Delta\overline{x_i} = 0 \quad , \quad \Delta\overline{y_i} = 400\,\mu\mathrm{m} \tag{3.12}$$

reduziert sich der mittlere Fehler des Funktionswertes b auf:

$$\Delta\bar{b} = \sqrt{\sum_{m=1}^{N} \left(\frac{\partial b}{\partial y_m} \Delta\overline{y_m}\right)^2} \tag{3.13}$$

Mit der partiellen Ableitung von b nach y_m

$$\frac{\partial b}{\partial y_m} = \frac{N x_m - \sum x_i}{N \sum x_i^2 - \left(\sum x_i\right)^2} \tag{3.14}$$

ergibt sich der mittlere Fehler von b zu:

$$\Delta\bar{b} = \sqrt{\sum_{m=1}^{N} \left(\frac{N x_m - \sum x_i}{N \sum x_i^2 - \left(\sum x_i\right)^2} \Delta\overline{y_m}\right)^2} \tag{3.15}$$

Regressionsfehler speziell

Durch die Gleichungen für die Regressionsgerade (3.1–3.3) ergeben sich die Varianzen für a und b [38]:

$$\sigma_a^2 \simeq \frac{\sigma^2}{\Delta} \sum x_i^2 \quad , \quad \sigma_b^2 \simeq N \frac{\sigma^2}{\Delta} \tag{3.16}$$

Auf lokaler Ebene wird allein die Steigung als Qualitätsmaß benutzt, da man mit ihr auf den Transversalimpuls der Teilchen schließen kann. Für n zusammenhängende Stützstellen lässt sich die Varianz der Steigung durch die (konstante) Varianz der Einzelmessungen annähern mit

$$\sigma_b^2 \simeq \frac{\sigma^2}{F_n} \left[\frac{\mathrm{mm}^2}{\mathrm{timebin}^2}\right] \tag{3.17}$$

wobei F_n Tabelle 3.2 entnommen werden kann.

n	F_n	n	F_n
15	280	12	143
14	227,5	11	110
13	182	10	82,5

Tabelle 3.2: Zur Abschätzung des Steigungsfehlers für n zusammenhängende Stützstellen nach Formel 3.17.

Da zusammenhängende Stützstellen enger zusammenliegen als verteilte und es sich um einen Fit mit zwei Freiheitsgraden handelt, erhält man somit eine obere Abschätzung für den Steigungsfehler. Beispiel: Der Fehler für die Ablenkung über eine Kammer (Kapitel 3.3) einer Spur mit 12 Stützstellen und 15 Zeitabschnitten ist:

$$\sigma_b \simeq \sqrt{\frac{(400\,\mu\mathrm{m})^2}{143}} \cdot 15 = 502\,\mu\mathrm{m} \tag{3.18}$$

3.3.4 Korrektur von Lorentz-Winkel und „Tilted Pads“

Durch das elektrische Feld $\vec{E}$ driften die Elektronen zunächst parallel zur Richtung des $\vec{E}$-Feldes, und die Spur des Teilchens kann direkt aus der Orts-Zeit-Korrelation rekonstruiert werden. Der gesamte Detektor wird jedoch zusätzlich von einem Magnetfeld $\vec{B}$ in z-Richtung durchdrungen. Auf die durch das $\vec{E}$-Feld beschleunigten Elektronen wirkt dadurch zusätzlich die Lorentz-Kraft.

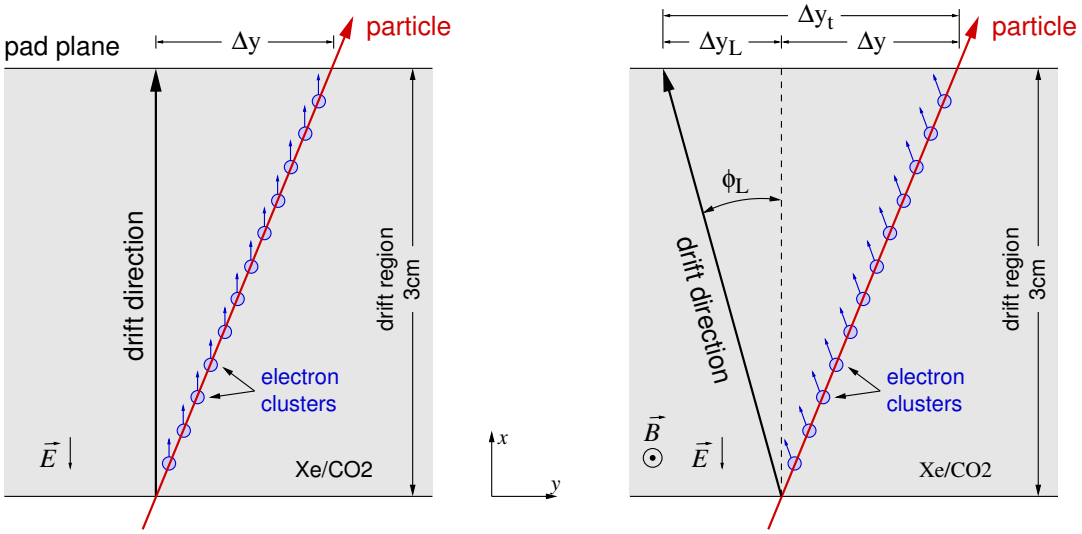

Abbildung 3.12: Wirkt allein das elektrische Feld $\vec{E}$, steht die Driftrichtung der Elektronen senkrecht zur Modul-Oberfläche. Durch das Magnetfeld $\vec{B}$ wird die Richtung um den Winkel Φ_L abgelenkt.

Durch die Stöße der Elektronen mit Gasmolekülen werden diese immer wieder abgebremst und es ergibt sich eine konstante Driftgeschwindigkeit. Die mittlere Zeit zwischen zwei Stößen wird mit τ bezeichnet. In dieser Zeit werden die Elektronen durch das $\vec{E}$-Feld beschleunigt und beschreiben in Kombination mit dem $\vec{B}$-Feld eine Spiralbahn bis zur nächsten Kollision. Als Resultat driften die Elektronen verlangsamt um den Lorentz-Winkel $\Phi_\mathrm{L} = \arctan \frac{eB\tau}{m}$ zum $\vec{E}$-Feld.

Hierdurch erscheint die eigentliche Ablenkung Δy einer Spur um einen konstanten Betrag Δy_L in stets dieselbe Richtung verändert. Man rechnet in diesem Experiment mit einem Lorentz-Winkel von 7°, was für den möglichen Ablenkungsbereich berücksichtigt werden muss. Da es sich bei der Korrektur der Ablenkung um eine einfache Addition einer Konstanten handelt, lässt sie sich einfach und schnell innerhalb des TRAPs implementieren.

Durchquert ein Teilchen ein Detektor-Modul, kann die y-Position mithilfe der Ladungsverteilung sehr genau bestimmt werden. Zur Rekonstruktion des Transversalimpulses spielt die Position des Tracks in z-Richtung keine entscheidende Rolle. Dieses Auflösungsvermögen bestimmt sich für eine einzelne Lage durch die Länge (etwa 10 cm) eines Pads in z-Richtung. Betrachtet man mehrere Lagen und setzt voraus, dass die Spuren in der xz-Ebene eine Gerade beschreiben, kann man die Spurparameter auch genauer bestimmen.

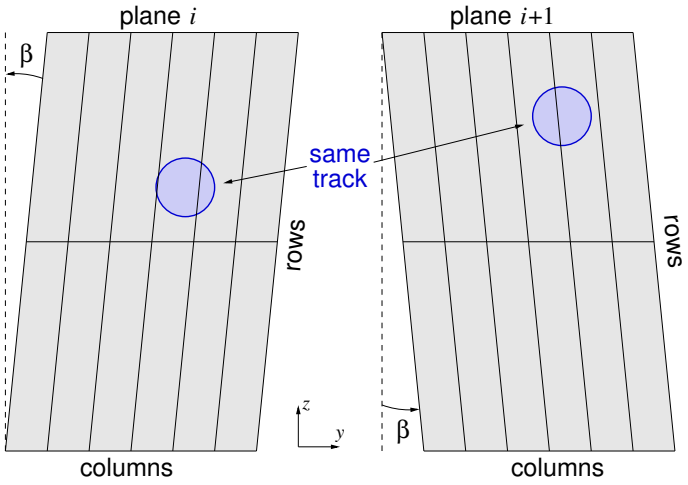

Abbildung 3.13: Zur Verbesserung der Ortsauflösung in z-Richtung sind die Pads in den verschiedenen Lagen gegeneinander geneigt. In Kombination mit der Berechnung des Transversalimpulses lässt sich auf die Position einer Spur innerhalb einer Zeile schließen.

Indem man die Padreihen der verschiedenen Lagen gegeneinander neigt (Abbildung 3.13), kann man in der späteren Analyse durch einen mehrdimensionalen Fit unter Berücksichtigung aller Lagen eine Auflösung von etwa 1 cm in z-Richtung [20] erreichen. Durchquert z. B. ein beliebig steifes Teilchen zwei Lagen in derselben Reihe, lässt sich anhand der unterschiedlichen y-Positionen die genaue z-Position ermitteln. Die einzelnen Pads haben hierbei die Form von Parallelogrammen mit einem Neigungswinkel von $\beta = \pm 2°$ zur z-Richtung.

Diese Verschiebung der y-Position in Abhängigkeit von der z-Position gilt auch für jede gemessene Position innerhalb eines Moduls. Die gemessene Ablenkung eines Spursegmentes

muss also korrigiert werden. Hierzu kann ein konstanter Wert für die jeweilige Padreihe zu der gemessenen Ablenkung addiert bzw. von ihr subtrahiert werden, je nach Neigungsrichtung in der entsprechenden Lage. Auch diese Korrektur lässt sich mit einfachen Mitteln innerhalb des TRAPs bewerkstelligen.

Hinweis: Sowohl der Effekt des Lorentz-Winkels als auch die Geometrie der Tilted Pads sind nicht in den benutzten Versionen der Software (Tabelle 3.1) enthalten. Sie spielen jedoch für die grundlegende Architektur der Elektronik keine Rolle, da gezeigt wird, dass alle hierdurch entstehenden Effekte mittels einfacher Operationen innerhalb des Prozessors korrigiert werden können.

3.4 Funktion der globalen Tracking-Einheit

Über das Auslesenetzwerk (Kapitel 5) gelangen die einzelnen Spurabschnitte aller Module zur globalen Tracking-Einheit (GTU). Aufgabe der GTU ist es, zusammengehörige Spurabschnitte zu erkennen bzw. unnötige zu verwerfen und eine genauere Aussage über den Transversalimpuls einer gefundenen Spur zu machen. Diese Informationen werden für den L1-Trigger benötigt, d. h. für den Transport und die Verarbeitung der Daten stehen nur etwa 2 µs zur Verfügung. Die GTU wird aus programmierbarer Hardware (FPGAs) aufgebaut. Ziel ist es, das lokale Tracking, die Datenübertragung und die Arbeitsweise der GTU aufeinander abzustimmen. Nur so kann ein hoher Parallelisierungsgrad erreicht und können Berechnungsschritte für eine Hardwareimplementierung optimiert werden. Die Architektur der GTU wird im Rahmen einer Diplomarbeit [24] erarbeitet.

In diesem Kapitel wird ein grundlegendes Konzept der GTU entwickelt und gezeigt, dass die Umsetzung unter Berücksichtigung der Anforderungen möglich ist.

3.4.1 Ansätze zum globalen Tracking

Für den Datentransport und die Verarbeitung der Daten spielt die komplette Detektorgeometrie eine Rolle. Die eigentliche Aufgabe lässt sich jedoch anhand einer 2-dimensionalen Projektion verdeutlichen.

Die LTUs erzeugen für eine Spur im idealen Fall in jeder Ebene einen Spurabschnitt (Abbildung 3.14). Die gesuchten Spuren sind sehr steif, d. h. sie bilden nahezu eine Gerade mit dem Vertex als Ursprung. Projiziert man die Abschnitte auf eine Mittelebene, so treffen sich ihre Verlängerungen an einem Punkt. Eine Projektion einer anderen Spur würde denselben Punkt in einem anderen Winkel schneiden. Hat man damit die zusammengehörigen Spurabschnitte gefunden, kann mithilfe der max. 6 Geradenabschnitte ein Fit durch diese Punkte gelegt und eine adäquate Abschätzung für den Transversalimpuls errechnet werden.

Von entscheidender Bedeutung für die Architektur der Triggerelektronik und der Datenauslese im Speziellen ist der Zusammenhang zwischen der Detektorgeometrie und den Spursegmenten. Das globale Tracking findet ausschließlich auf Stack-Ebene (Abbildung 2.4) statt.

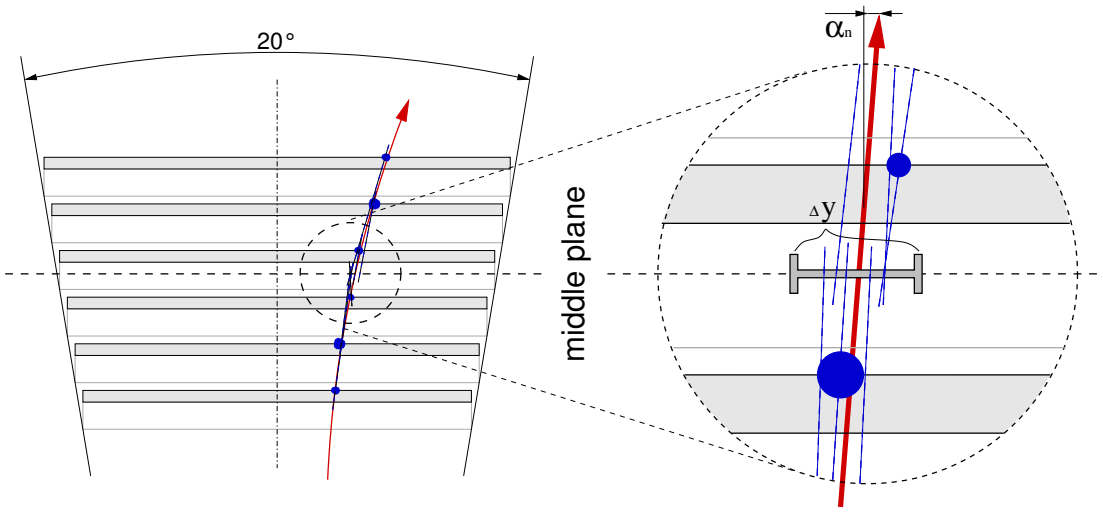

Abbildung 3.14: Gezeigt ist ein Schnitt durch einen Stack des Detektors. Die Prozessoren auf dem Detektor erzeugen Spurabschnitte, die auf eine Mittelebene projiziert werden.

Durch die mechanischen Elemente zwischen den Stacks kann es keine für die Triggerfunktion relevanten Spuren geben, die über mehr als ein Stack reichen. An den Rändern der Stacks in z-Richtung ist die Anordnung der Module so gewählt, dass selbst im ungünstigsten Fall eine Spur stets über mindestens vier Lagen innerhalb eines Stacks reicht.

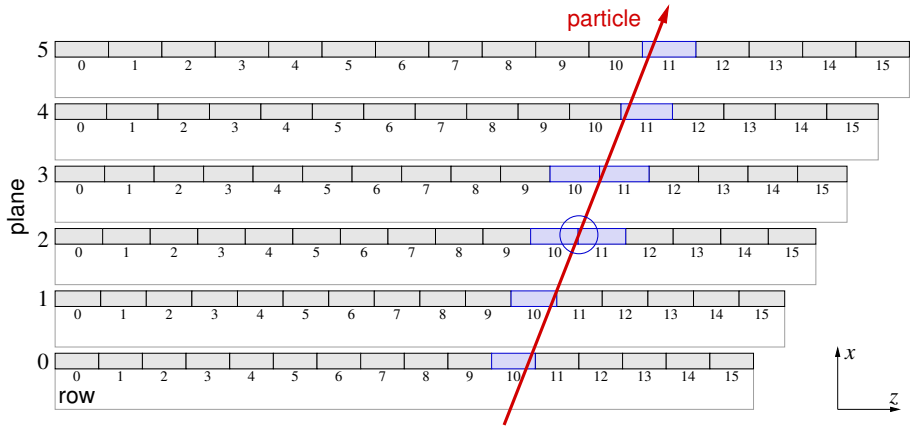

Abbildung 3.15: Gezeigt ist eine Spur vom Vertex, die in den verschiedenen Lagen unterschiedliche Zeilen schneidet. In Lage 2 geht sie außerdem von Zeile 10 in Zeile 11 über (Abbildung 3.21).

In Abbildung 3.15 ist ein Schnitt durch ein Stack in der xz-Ebene gezeigt. In dieser Ebene können die Spuren als gerade angenommen werden, da das Magnetfeld in z-Richtung verläuft. Aufgrund der Geometrie des Detektors im Zusammenhang mit den verschiedenen möglichen Positionen der Interaktion (Vertex) kann sich die betroffene Padzeile in den verschiedenen Lagen um bis zu drei unterscheiden. Innerhalb dieses Bereiches gibt es weiterhin die Möglichkeit, dass eine Spur die Grenze zwischen zwei Zeilen innerhalb einer Lage

schneidet. Durch diese Eingrenzung der möglichen Positionen zusammengehöriger Spursegmente in den verschiedenen Lagen, kann durch eine geeignete Auslesereihenfolge mit dem globalen Tracking noch während der Übertragung der Daten begonnen werden.

Auf dem Detektor findet die Verarbeitung und Übertragung der Daten der unterschiedlichen Lagen unabhängig voneinander statt. Werden von jeder Lage die Spursegmente zeilenweise, also in z-Richtung, an die GTU geleitet, kann diese mit der Rekonstruktion der ersten Spur beginnen, wenn die ersten (mind. drei) Zeilen einer jeden Ebene empfangen wurden. Die für die Übertragung benötigte Zeit hängt von der Anzahl der gefunden Spursegmente ab, die von Ebene zu Ebene variieren kann. Neben der initialen Übertragungslatenz (Abbildung 2.6) hängt es also auch von der Übertragungsrate ab, nach welcher Zeit die GTU starten kann. Durch die Integration der Auslese in die mehrstufige Verarbeitung (Pipelining) innerhalb der GTU kann die nachfolgende Übertragung der Daten parallel zu den Berechnungen erfolgen. In Kapitel 5 sind die Funktionalität der Auslese und deren Implementierung näher beschrieben.

Nach der Zuordnung der einzelnen Spursegmente zu einer Spur erfolgt die genaue Bestimmung des Transversalimpulses. Dieser kann aus dem Radius r und dem Magnetfeld $\vec{B}$ in z-Richtung über die Beziehung $p_\mathrm{t} = \mathrm{e} \cdot r \cdot B$ berechnet werden.

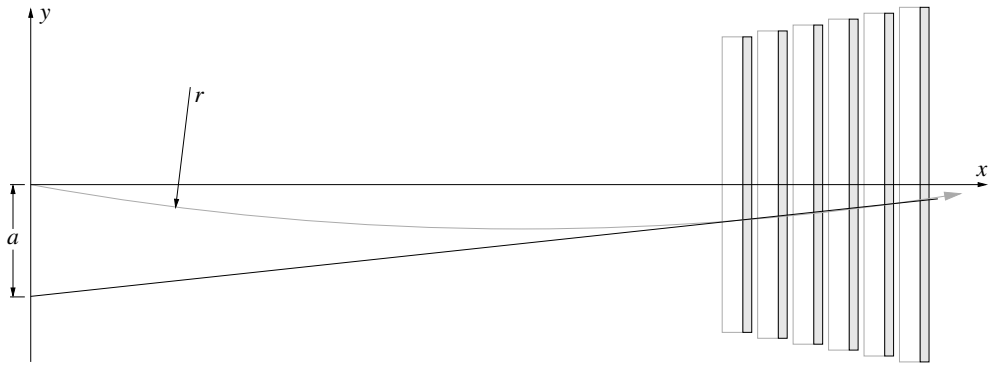

Abbildung 3.16: Durch Ausnutzung bekannter Randbedingungen lässt sich der Transversalimpuls hinreichend genau durch den dargestellten Achsenabschnitt a beschreiben. Damit kann auf die für eine Hardwareimplementierung aufwendige Berechnung trigonometrischer Funktionen verzichtet werden. Quelle: [24]

Zur genauen Bestimmung des Vertex wird zu den Positionen der Spursegmente die in der xy-Ebene exakt bekannte Position des Vertex hinzugezogen. Die Berechnung mithilfe trigonometrischer Funktionen wäre in Hardware sehr aufwendig und könnte in der zur Verfügung stehenden Zeit nicht erfolgen. Unter Ausnutzung bekannter Randbedingungen, wie z. B. eines großen Radius (geringe Krümmung) und fixer Positionen in x-Richtung, können die Berechnungen jedoch vereinfacht werden [24]. Durch einen für Hardware optimierten Geradenfit durch die Positionen der Spursegmente erhält man eine Sekante für den Bereich des Stacks und damit den Achsenabschnitt a in y-Richtung. Es kann durch Detektorsimulationen gezeigt werden, dass allein aufgrund dieser Größe a eine adäquate Abschätzung des Transversalimpulses für die Funktion als Trigger erfolgen kann.

3.5 Ausgewählte Ergebnisse

In diesem Kapitel werden ausgewählte Ergebnisse der Simulation vorgestellt, die direkte Auswirkungen auf die Architektur des Detektors und der Elektronik erkennen lassen. Wie in Anhang B.2 beschrieben, sind die Quellen und die ROOT-Dateien zu den gezeigten Histogrammen auf dem beigelegten Datenträger zu finden. Die in diesem Kapitel vorgestellten Ergebnisse sind mit den Skripten zur Konfiguration der Ereigniserzeugung und der Digitalisierung Config_bv04.C, grun_bv04.C und slowDigitsCreate_bv04.C erstellt. Die hier eingestellten Parameter lassen sich der Tabelle 3.3 entnehmen.

Generierung	parameters	HIJINGpara
	particles	20000
	PhiRange	0..360
	ThetaRange	45..135
	B	0.4 T
Digitalisierung	gasgain	4000
	noise	3000
	chipgain	5
	ADCoutRange	1023
	ADCinRange	2000
	ADCthreshold	1
	ExB	off
	timebins	15
Simulation	events	0..0
	planes	0..5
	sectors	0..17
	chambers	0..4
	threshold	15
	HitsMin	4

Tabelle 3.3: Zur Ereigniserzeugung, Digitalisierung und zur Simulation des TRDs sind für die nachfolgenden Ergebnisse die gezeigten Parameter benutzt.

Mit diesem Satz an Konfigurationsparametern für die Ereignissimulation mit Geant werden 20 000 primäre geladene Teilchen erzeugt. Dabei liegen 7 255 im Akzeptanzbereich des TRDs. Zusammen mit allen indirekt erzeugten Teilchen, also inklusive sekundärer Teilchen, werden bei einem Ereignis 63 644 Teilchen erzeugt, von denen 17 475 im TRD liegen und deren Spuren verarbeitet werden.

3.5.1 Dynamischer Bereich und Auflösung der ADCs

Die Auflösung des ADCs muss so bemessen sein, dass trotz Quantisierungsfehlern und differentieller Nichtlinearitäten eine adäquate Ortsauflösung von etwa 400 µm erreicht wird. Dies gilt sowohl für interessante Spuren als auch für solche, die aussortiert werden sollen.

Legt man für die Berechnung der Position y_m die Formel 3.6 zugrunde, erhält man je nach Ausnutzung des dynamischen Bereichs des ADCs unterschiedliche Ortsauflösungen.

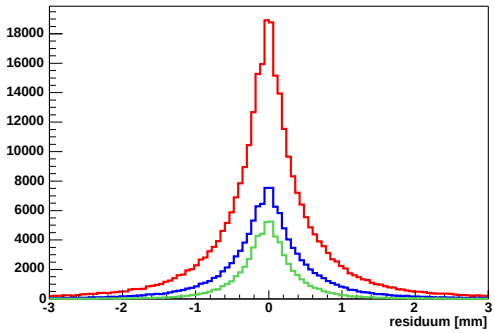

Abbildung 3.17: Residuenverteilung der Hits (rot: alle; blau: primäre; grün: hochenergetische primäre Teichen).

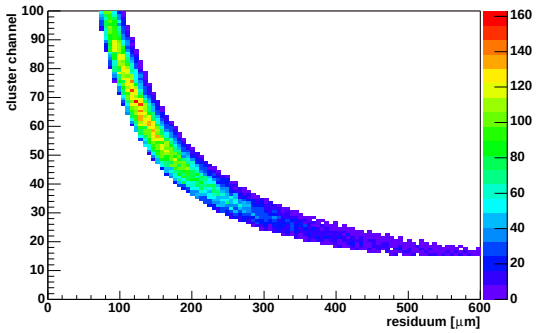

Abbildung 3.18: Durch die Quantisierung entstehender Positionsfehler über der mittleren Kanalsumme eines Hits.

Abbildung 3.18 zeigt die mittlere Kanalsumme eines Hits (Hit-Summe) über den in der Simulation erhaltenen Positionsfehler, der aufgrund der Quantisierung entsteht. Die Hit-Summe berechnet sich aus der Summe der bis zu drei ADC-Werte, die zur Bestimmung der Position herangezogen werden. Die Unschärfe des Histogramms rührt von den unterschiedlichen Ladungsverteilungen über die Pads und deren verschiedenen Breiten (je nach Lage) her. Dem Histogramm kann entnommen werden, dass bei einer Hit-Summe von etwa 40 mit einer Ortsungenauigkeit von weniger als 300 µm zu rechnen ist.

Außerdem möchte man größere Peaks, wie sie durch Übergangsstrahlungsphotonen hervorgerufen werden, in gleicher Auflösung messen. Im Beispiel für eine Ladungsverteilung über die Zeit in Abbildung 3.6 lässt sich erkennen, dass der Peak, hervorgerufen durch Übergangsstrahlung, etwa um einen Faktor 10 über dem Mittel liegt. Größere Peaks in der Hit-Summe sind bis zu einem Faktor 20 größer als die mittlere Summe. Hieraus erhält man den nötigen dynamischen Bereich, den die ADCs abdecken müssen. Mit einem 10-Bit-ADC (1024 Kanäle) sind eine hinreichend gute Ortsauflösung und die Erfassung von Werten im oberen Kanalbereich möglich.

3.5.2 Einflüsse der Detektorgeometrie

Die Detektorgeometrie, wie in Abbildung 2.4 dargestellt, wirkt sich in vielfacher Hinsicht auf die Architektur der Elektronik aus. Wird ein Modul betrachtet, entspricht die x-Richtung der Driftrichtung innerhalb der Kammern. Hiervon ist die Datenaufnahme beeinflusst. Das Magnetfeld durchfließt den Detektor in z-Richtung. Dadurch werden die geladenen Teilchen in der xy-Ebene abgelenkt. Für ein kurzes Spursegment innerhalb einer Kammer ist für die Auslösung des Detektors die y-Richtung entscheidend.

***x*-Richtung** Die Höhe bzw. Dicke einer Kammer kann als vorgegeben angesehen werden. Sie darf nicht zu groß sein, da ansonsten die Driftdauer zu groß wird (Kapitel 2.3.1). Auf die

Einflussnahme der Driftspannung und deren Abhängigkeit wird hier nicht weiter eingegangen. Bei zu kleiner Höhe können keine adäquaten Parameter der Spursegmente bestimmt werden. Die Driftdauer beträgt bei der betrachteten Geometrie etwa 2 µs. Der wählbare Parameter beschränkt sich auf die Abtastrate, was der Integrationszeit der in die Vorverstärker induzierten Ladung entspricht. Wird diese Zeit zu kurz gewählt, schränkt man den dynamischen Bereich und damit die Auflösung ein (Kapitel 3.5.1). Bei zu niedriger Abtastrate erhält man zu wenig Stützpunkte für einen hinreichend genaue Spurbestimmung. Alle Simulationen sind mit einer Abtastrate von 10 MHz vorgenommen, was 15 Zeitschritten entspricht.

y-Richtung Die Positionsbestimmung für einen Zeitabschnitt findet wie in Kapitel 3.3.1 beschrieben über die Ladungsverteilung eines zentralen und dessen beider benachbarter Pads statt. Das gesamte Spursegment, das aus bis zu 15 Zeitabschnitten besteht, kann die Grenze zwischen zwei Pads überschreiten. Der maximale Winkel für interessante (steife) Spuren ergibt sich aus der Detektorgeometrie, 10° am Rand einer Kammer, dem Lorentz-Winkel (konstant 7°) und dem Transversalimpuls. Abbildung 3.19 zeigt die Anzahl der Spalten bzw. Pads, die von einem Spursegment betroffen sind. Während Segmente, die von sekundären Spuren (rot) herrühren, auch über drei Pads reichen können, überschreiten primäre Spuren (blau) und Spuren hochenergetischer primärer Teilchen (grün) maximal eine Padgrenze.

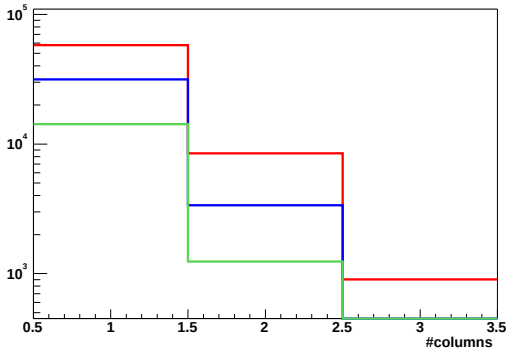

Abbildung 3.19: Anzahl der Spalten (Pads), die durch ein Spursegment involviert sind.

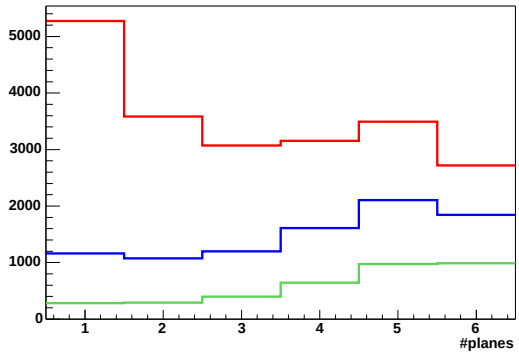

Abbildung 3.20: Anzahl der Lagen, in denen für eine Spur ein Spursegment gefunden wird.

Mit dieser Voraussetzung ist es ausreichend, zwei benachbarte Kanäle zur Bestimmung eines Spursegments zu betrachten. Ein MCM besitzt 18 aktive Kanäle. Da jedoch Spuren über jeweils zwei Pads reichen können, werden Kanäle der benachbarten MCMs berücksichtigt. Hiermit werden Verluste in y-Richtung vermieden. In Kapitel 4.1.3 wird das hierzu verwendete Verfahren vorgestellt. Aufgrund der projektiven Geometrie der Module ist es nicht erforderlich, auch Modulgrenzen zu berücksichtigen.

z-Richtung In dieser Richtung sind die Module des Detektors nur bedingt projektiv angeordnet. Der mittlere Stack, der allerdings recht schmal ist, hat über alle sechs Lagen in

z-Richtung dieselbe Ausdehnung. Die Grenzen zwischen den beiden äußeren Modulen sind bis auf die äußeren beiden Lagen projektiv zum Vertex. In beiden Fällen werden bei einer Spur eines primären Teilchens mindestens vier Ebenen desselben Moduls durchquert, was zur Bestimmung der Spur auf globaler Ebene ausreichend ist (Abbildung 3.20).

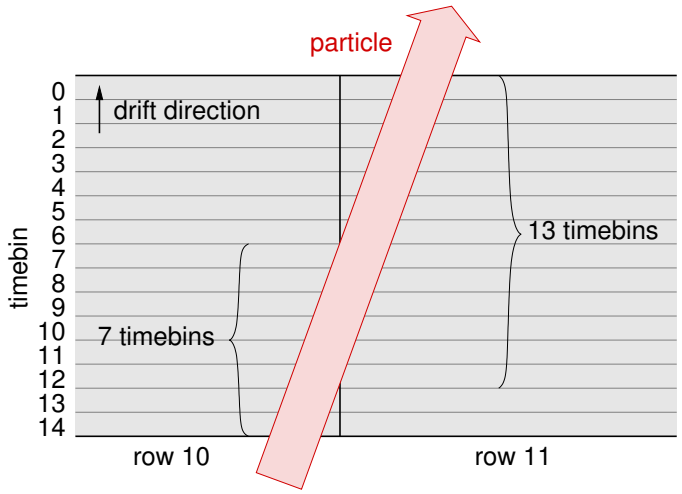

Abbildung 3.21: Die dargestellte Spur (Abbildung 3.15) schneidet die Grenze zwischen zwei Zeilen und erzeugt für beide Zeilen unabhängige Spurabschnitte.

Wie schon in Kapitel 3.4.1 beschrieben, können Spursegmente innerhalb einer Lage die Grenze zwischen Zeilen durchschreiten. Wie in Abbildung 3.21 dargestellt, kann durch die räumliche Ausdehnung der Ionisationsspuren die Anzahl der verwendbaren Zeitabschnitte durchaus groß genug sein, um trotzdem in einer oder beiden Zeilen separat eine Spurerkennung durchzuführen.

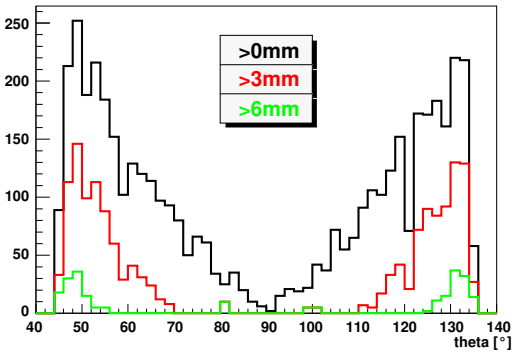

Abbildung 3.22: Verteilung der Spursegmente, die mehr als 0 mm, 3 mm, ... in einer anderen Padzeile liegen über dem Winkel Theta θ.

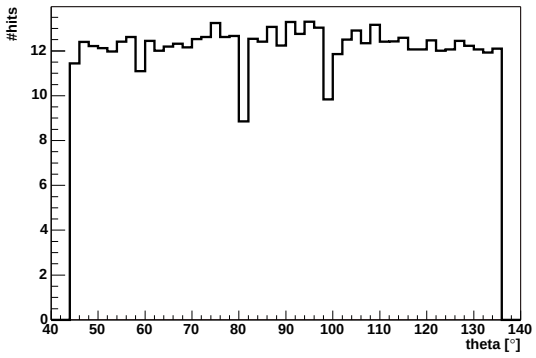

Abbildung 3.23: Anzahl der involvierten Hits (th=15) pro Spursegment über dem Winkel Theta θ.

Die Häufigkeit solcher Spursegmente hängt von der z-Position im Detektor, also dem Winkel θ der Spur ab. Abbildung 3.22 zeigt die Verteilung solcher Spuren über θ für unterschiedliche Längen, die von dem Segment in die benachbarte Zeile fallen. Entscheidend für die

Genauigkeit bei der Bestimmung eines Spursegments innerhalb eines MCMs ist die Anzahl der verwendbaren Stützpunkte (Hits). Als verwendbar werden Hits bezeichnet, deren Ladungssumme in ADC-Kanälen mindestens 15 beträgt. Wie in Abbildung 3.23 zu erkennen, ist die Anzahl der gefundenen Stützpunkte über den gesamten Winkelbereich von θ nahezu konstant. Die Einbrüche entstehen an den Modulgrenzen. Eine Verteilung oder ein Austausch von Daten über Zeilen hinweg ist also nicht notwendig. Die Geometrie der Kammer ist so gewählt, dass diese Teilung eines Spursegments nur in einer Lage für dieselbe Spur vorkommen kann. Ein eventuell doppelt gefundenes Spursegment innerhalb einer Lage kann später während des globalen Trackings verworfen werden. Hierbei wird die Spur mit der höheren Qualität gewählt.

3.5.3 Einfluss auf die Auslesearchitektur

Eines der wichtigsten Simulationsergebnisse ist die Effizienz des Systems, in diesem Fall der lokalen Tracking-Einheit. Dieses Ergebnis bestimmt die Architektur der nachfolgenden Systeme. Ziel des lokalen Trackings ist es, möglichst alle Spursegmente mit einer geringen Ablenkung bzw. einem hohen Transversalimpuls zu finden und an die globale Tracking-Einheit weiterzugeben. Gleichzeitig ist es für eine schnelle Weiterverarbeitung der Daten wichtig, dass möglichst wenig uninteressante Spuren darin enthalten sind. Die Definition der Effizienz ist nicht eindeutig, was die Bestimmung der Leistung des Systems erschwert.

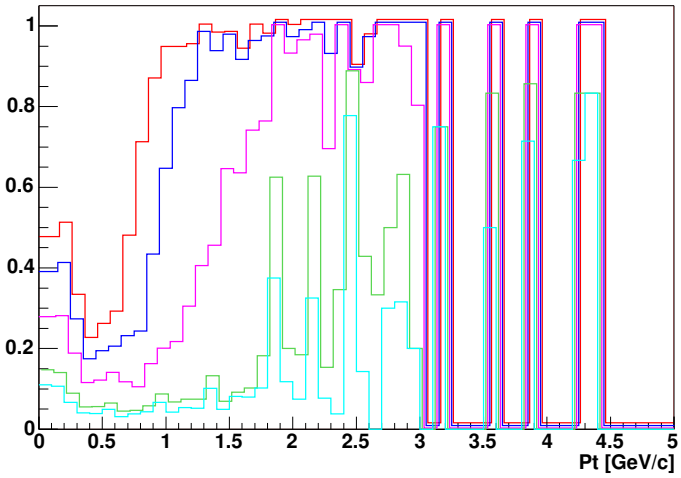

Abbildung 3.24: Dargestellt ist die relative Anzahl der gefundenen Spursegmente für verschiedene zugelassene maximale Ablenkungen (8 mm: rot, 6 mm: blau, 4 mm: pink, 2 mm: grün, 1,5 mm: cyan).

Abbildung 3.24 zeigt die relative Anzahl gefundener Spursegmente für verschiedene zugelassene maximale Ablenkungen. Interessante Spuren haben einen Transversalimpuls von mindestens 2 GeV, was einer Ablenkung von etwa 2 mm (grün) entspricht. Erhöht man die zulässige Ablenkung auf z. B. 6 mm (blau), werden fast 95 % aller Spursegmente von Teilchen mit mehr als 1,5 GeV gefunden. Dafür sind allerdings dann auch viele uninteressante Spuren darin enthalten.

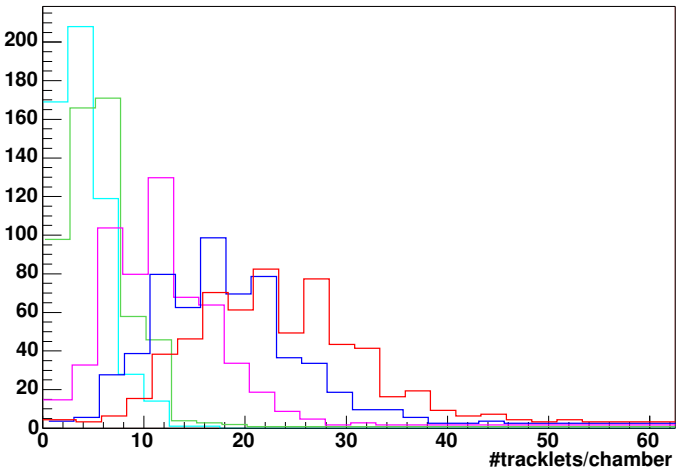

Abbildung 3.25: Anzahl gefundener Spursegmente in einer Kammer für verschiedene zugelassene Ablenkungen. Die farbliche Zuordnung entspricht der aus Abbildung 3.24.

Für die Auslese ist es von Bedeutung, wie viele Spursegmente maximal zu erwarten sind, auch wenn der zugelassene Ablenkungsbereich vergrößert wird. Die zur Verfügung stehende Zeit für die Übertragung der Daten (Abbildung 2.6) beträgt etwa 400 ns, zuzüglich der Übertragungslatenz. In Abbildung 3.25 ist zu erkennen, dass bei einem Ablenkungsbereich von 6 mm (blau) im Mittel etwa 18 und maximal etwa 40 Spursegmente pro Kammer zu erwarten sind. Die in Kapitel 5 beschriebene Architektur der Auslese ist hierauf optimiert.

3.5.4 Kodierung und Transformation

Für die Funktion des TRDs als Trigger werden von den lokalen Tracking-Einheiten Spursegmente an die globale Tracking-Einheit geschickt. Die Geradengleichungen werden hierfür so optimiert, dass sie den Auflösungsanforderungen entsprechen und dabei möglichst effizient kodiert sind. Die gewünschte Breite eines Datenwortes ist 32 Bit, da es der Datenbusbreite der Prozessoren entspricht und auch die Netzwerkschnittstelle auf diese Paketgröße ausgelegt ist. Die große Anzahl parallel arbeitender Prozessoreinheiten auf lokaler Ebene legt es nahe, möglichst viele Rechenschritte schon vor der Übertragung stattfinden zu lassen, um Zeit zu sparen. Schließlich werden die Parameter eines Spursegmentes an die Arbeitsweise der GTU angepasst. Es ergeben sich folgende Parameter:

- Die y-Position bzw. der Achsenabschnitt der Geraden. Dieser befindet sich auf der Oberseite einer Kammer, da hier der erste Zeitabschnitt und damit der Ursprung der x-Achse liegt. Wegen der unterschiedlichen Breiten der Kammern je nach Lage in x-Richtung werden die Positionen in absolute Werte (Meter) umgerechnet und übertragen.
- Die Ablenkung. Sie entspricht der Steigung der Geraden. Sie wird als Differenz der y-Position der Außenseite zu der der Innenseite angegeben. Auch hier erfolgt eine

Skalierung an absolute Koordinaten, da die Pads in den verschiedenen Lagen unterschiedliche Breiten haben.

- Die Padreihe bzw. die z-Position innerhalb der Kammer. Dieses ist ein programmierbarer Wert für jedes MCM.
- Die Elektron-Signatur. Aus der gemessenen Ladungssumme und ihrer Verteilung über die Driftzeit lässt sich eine Wahrscheinlichkeit für ein Elektron als erkannte Spur ermitteln.
- Die Spurqualität. Als Ergebnis des Geradenfits durch eine lineare Regression erhält man durch die Residuen der Spurpunkte ein Qualitätsmaß. Dieses kann auf globaler Ebene z. B. bei der Beurteilung von doppelt gefundenen Spursegmenten herangezogen werden.

Parameter	Bit	Schrittweite	Wertebereich
Achsenabschnitt	13	160 µm	-643,2 mm .. 643,2 mm
Ablenkung	7	140 µm	-8,8 mm .. 8,8 mm
Padreihe	4	1	0 .. 15
Elektron-Signatur	6	1/63	0,0 .. 1,0
Qualität	2	1/3	0,0 .. 1,0

Tabelle 3.4: Kodierung der Spursegmente, die von den lokalen Tracking-Einheiten an die GTU gesendet werden.

Die in Tabelle 3.4 dargestellte Aufteilung der Datenbits auf die Spurparameter stellt nur eine Möglichkeit dar und zeigt, dass eine adäquate Aufteilung in 32 Bit möglich ist. Durch die Verarbeitung der Daten vor der Übertragung vom Detektor lassen sich die Genauigkeiten und die Aufteilung der Bits auch später noch beliebig anpassen. Die aktuelle Implementierung der globalen Tracking-Einheit sieht vor, dass die Felder „Elektron-Signatur“ und „Qualität“ zu einem 8 Bit breiten Feld zusammengefasst werden, das eine Wahrscheinlichkeit der Detektion eines Elektrons beschreibt.

4 Architektur & Implementierung

Im folgenden Kapitel wird die Architektur des Tracklet-Prozessors und der globalen Tracking-Einheit beschrieben, wie sie sich aus den Simulationen und Überlegungen in Kapitel 3 gebildet hat. Das Auslesenetzwerk und die dazugehörige Netzwerkschnittstelle werden in Kapitel 5 und 6 ausführlicher erläutert. In der zweiten Hälfte dieses Kapitels ist der Design-Fluss des Tracklet-Prozessor-Chips dargestellt.

4.1 Tracklet-Prozessor

Der Tracklet-Prozessor (TRAP) erfüllt die Funktion der in Kapitel 3.3 beschriebenen lokalen Tracking-Einheit (LTU).

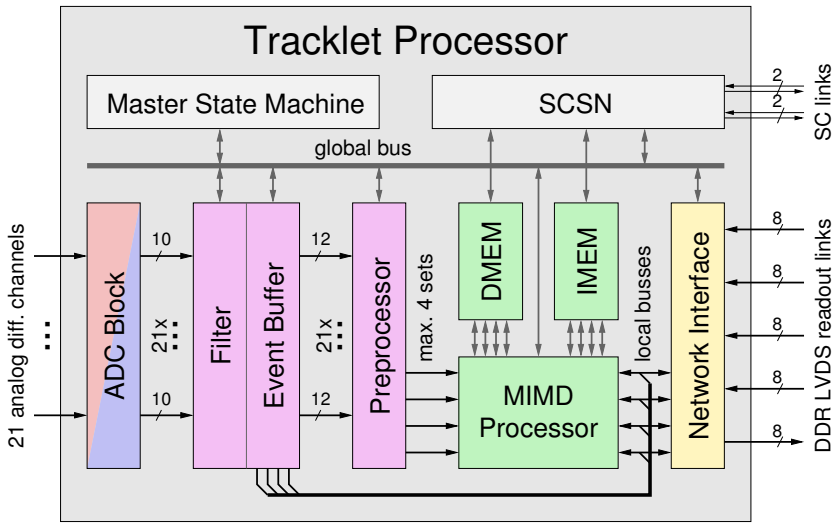

Abbildung 4.1: Alle Module des TRAP können über einen globalen Bus, mit der Konfigurations-Schnittstelle (SCSN) als Busmaster, konfiguriert und ausgelesen werden. Auch der Prozessor hat Lese- und Schreibrechte auf diesen Bus. Der Zugriff auf die Netzwerkschnittstelle und das Lesen aus den Ereignisspeichern erfolgt über vier schnellere lokale Busse der CPUs. Die Master-State-Machine verarbeitet die Trigger-Signale und kontrolliert den Ablauf.

Alle Zugriffe zur Konfiguration und Überwachung des TRAPs finden über einen seriellen Link (Slow Control Serial Network, SCSN) statt. Dieses Netzwerk [28] ist als Master-Slave-Ring aufgebaut, in dem allein der Master Pakete fester Länge erzeugen und verwerfen darf. Durch die Slaves (TRAPs) werden die Pakete entweder weitergeleitet oder bei Adressierung

des entsprechenden TRAPs modifiziert. Die Struktur dieses Netzwerks wird in Kapitel 5.2.4 näher beschrieben. Das SCSN verarbeitet einen 1-Bit-Datenstrom ohne Taktsignal, indem es durch geeignete Kodierung die Daten zum schnellen internen 120-MHz-Takt synchronisiert.

Das SCSN ist Busmaster auf den globalen Bus des TRAPs (Abbildung 4.1), der alle Module miteinander verbindet. Neben ihm hat lediglich der Prozessor Schreibrechte auf diesen Bus. Unabhängig hiervon lässt sich über das SCSN auch der Daten- und Instruktionsspeicher beschreiben und auslesen. Über den globalen Bus lassen sich alle Statusregister der Module lesen und die Konfigurationsregister sowie der Ereignisspeicher beschreiben und auslesen.

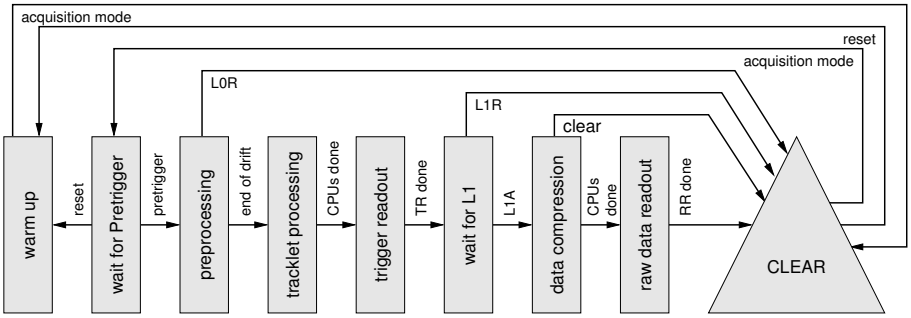

Abbildung 4.2: Die Master-State-Machine durchläuft in Abhängigkeit der Triggersignale und der Verarbeitungszustände des TRAPs die dargestellten Zustände und sorgt dafür, dass die jeweils aktiven Module mit einem Takt versorgt sind.

Die Master-State-Machine (MSM) verwaltet unter Berücksichtigung der Triggersignale die Abläufe (Abbildung 4.2) innerhalb des TRAPs. Da die Leistungsaufnahme des TRAPs ein wichtiges Entwurfskriterium ist, werden – von der MSM gesteuert – nur diejenigen Module mit einem Takt versorgt, die für den entsprechenden Verarbeitungsschritt benötigt werden. Befindet sich der TRAP im Wartezustand auf einen Prätrigger, so arbeiten z. B. nur der ADC-Block und Teile des Filters. Aus der in Kapitel 2.2 beschriebenen Triggerhierarchie lässt sich die Funktionalität der MSM in Bezug auf die Trigger-Signalverarbeitung herleiten.

4.1.1 Analog-Digital-Wandler

Die 21 Analog-Digital-Wandler (ADCs) des Tracklet-Prozessors sind zusammen mit weiteren Modulen in einem ADC-Block zusammengefasst und dieser ist an das Design des TRAPs angefügt. Der ADC-Block [29] wurde bei Prof. Dr. R. Tielert am Lehrstuhl für Mikroelektronik an der Universität Kaiserslautern entwickelt.

Abbildung 4.3 zeigt die Module des ADC-Blocks, die im Folgenden vorgestellt werden. Da der Takt extern erzeugt und über ein Detektor-Modul verteilt wird, kann es zu Jitter-Effekten kommen, so dass die Qualität nicht mehr den Anforderungen der ADCs entspricht. Daher erzeugt die PLL aus dem eingehenden 120-MHz-Takt einen sauberen Takt doppelter Frequenz. Die Referenzspannung (REF) wird von den ADCs intern zur Erzeugung von

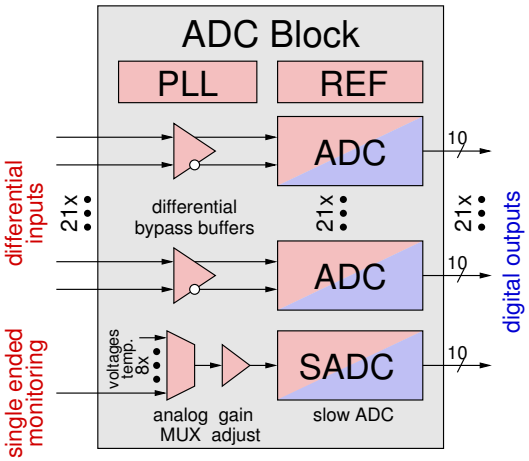

Abbildung 4.3: Neben den 21×10 Bit ADCs, die eine Sampling-Rate von 10 MHz haben, enthält der ADC-Block einen langsamen ADC zur Überwachung verschiedener Spannungen. Die ADCs verfügen über eine eigene 240-MHz-PLL und eine Referenz-Spannungserzeugung.

Vergleichsspannungen benötigt. Der 22-ste ADC ist für die Überwachung von internen und externen Spannungen und der Temperatur mittels eines integrierten Temperatursensors vorgesehen und arbeitet mit einer geringeren Wandlungsrate von 80 kHz. Die verschiedenen Eingänge für diesen zusätzlichen ADC werden über einen analogen Multiplexer ausgewählt. Da die verschiedenen Eingangsquellen unterschiedliche dynamische Bereiche besitzen, lässt sich der Ausgang des Multiplexers um einen konfigurierbaren Faktor dämpfen.

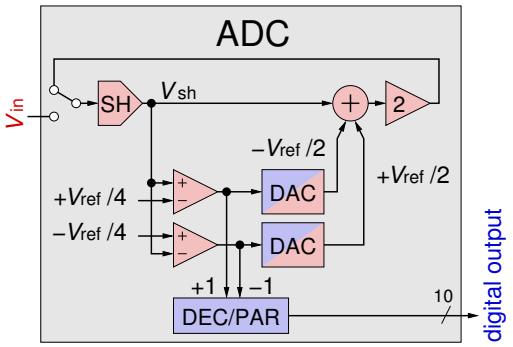

Abbildung 4.4: Im ersten Zyklus wird die Spannung V_{in} übernommen. In jedem Zyklus wird ein Summand des Polynoms erzeugt, das den Ausgangswert darstellt.

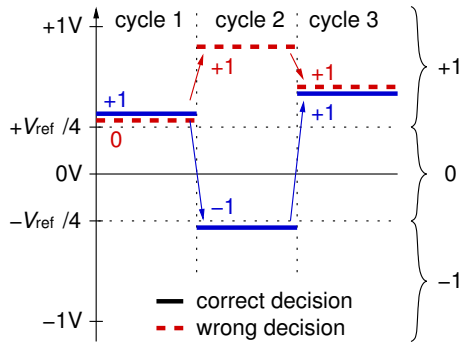

Abbildung 4.5: Die analoge Spannung wird mit zwei Schwellen verglichen und eine tertiale Entscheidung getroffen. Trotz einer Fehlentscheidung (gestrichelt) erhält man dasselbe Ergebnis.

Die ADCs, die die differentiellen Ausgangsspannungen der ladungsempfindlichen Vorverstärker (PASA) digitalisieren, haben eine Auflösung von 10 Bit und arbeiten mit einer Sampling-Rate von 10 MHz. Neben der Erfüllung der in Kapitel 3.5 hergeleiteten Anforderungen, wie z. B. an die Auflösung und die Samplingrate, sind die ADCs speziell auf Größe ($\sim$0,11 mm^2) und Leistungsaufnahme (12 mW) hin optimiert. Sie arbeiten nach einem zyklischen Arbeitsprinzip und nutzen die „switched-capacitor“ Technik. Alle 21 ADCs

sind dabei synchron, wobei eine Wandlungsphase 24 Perioden des 240-MHz-Takts dauert. Die 24 Zeitabschnitte werden für die vom ADC benötigten zehn Zyklen so aufgeteilt, dass die ersten etwas länger sind. Da die S&H-Schaltung für längere Zyklen genauer arbeitet, wird hiermit die Größe des aufsummierten Fehlers, der durch die S&H-Schaltung entsteht, gering gehalten und die Genauigkeit für die letzten Zyklen erhöht.

In die Sample&Hold-Schaltung (S&H) wird zunächst die Eingangsspannung V_{in} übernommen (Abbildung 4.4). Vereinfacht dargestellt wird für jeden weiteren Wandlungszyklus der Ausgang der S&H Schaltung V_{sh} mit den zwei Referenzspannungen $\pm V_{\text{ref}}/4$ verglichen und je nach Ergebnis V_{sh} oder $V_{\text{sh}} \pm V_{\text{ref}}/2$ verdoppelt und der S&H Schaltung wieder zugeführt. Die Gewichtung b des dem Zyklus i entsprechenden Summanden des Polynoms $\sum b \cdot 2^{10-i}$, das das Ergebnis darstellt, ist dabei:

$$\begin{aligned} -1 \quad &\text{für} \quad V_{\text{sh}} < -V_{\text{ref}}/4 \\ 0 \quad &\text{für} \quad -V_{\text{ref}}/4 \leq V_{\text{sh}} \leq +V_{\text{ref}}/4 \\ +1 \quad &\text{für} \quad V_{\text{sh}} > -V_{\text{ref}}/4 \end{aligned} \tag{4.1}$$

Wie in Abbildung 4.5 (blau) anhand von drei Verarbeitungszyklen dargestellt, wird bei korrekten Entscheidungen der ADC-Wert $+1 \cdot 2^2 - 1 \cdot 2^1 + 1 \cdot 2^0 = 3$ errechnet. Trotz einer Fehlentscheidung (rot) für den zweiten Zyklus – die Eingangsspannung wird fälschlicherweise als kleiner als $V_{\text{ref}}/4$ erkannt – erhält man ebenfalls den Wert $0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 3$. Durch diese Redundant-Signed-Digit-Technik (RSD) werden die Anforderungen an die Komparatoren stark entspannt und mögliche Fehlentscheidungen kompensiert.

4.1.2 Filter & Ereignisspeicher

Die Eingangswerte werden durch physikalische und architekturell bedingte Effekte beeinflusst, so dass eine Korrektur bzw. Filterung der Werte [23] notwendig ist. Diese Effekte existierten in der in Kapitel 3 beschriebenen Simulationsumgebung nicht, und die Eingangswerte werden dort als ideale Umwandlung der in der Driftkammer deponierten Ladung angenommen.

Die in Abbildung 4.6 dargestellte Filtereinheit besteht aus den Filterstufen, dem Ereignisspeicher und einer Konfigurationseinheit, über die die Filterparameter gesetzt werden können und über die der Speicher zugänglich ist.

Alle Eingangswerte der Kanäle werden in einem Ereignisspeicher zwischengespeichert. Wird entschieden, dass diese Daten für einen Trigger höheren Grades benötigt werden, erfolgt eine Rohdatenauslese. Der Speicher lässt sich außerdem als Datenquelle für die Filterstufen nutzen. Auf diese Weise kann der Filter mit dedizierten Werten gespeist und überprüft werden. Da sich alle Stufen des Filters separat ausschalten lassen, können somit auch die aktuellen Detektoreigenschaften überprüft und die einzelnen Filterstufen optimiert werden.

Sowohl die Vorverstärker als auch die ADCs weisen differentielle und integrale **Nichtlinearitäten** auf. Nach der Digitalisierung der Werte kann man diese Fehler gemeinsam

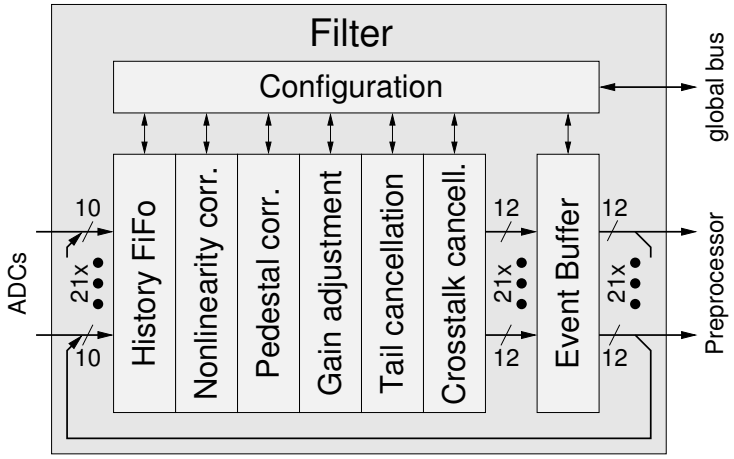

Abbildung 4.6: Stufen der Filtereinheit mit dem Ereignisspeicher, der gleichzeitig als Datenquelle dienen kann. Auf Filterparameter und Daten des Ereignisspeichers kann über den globalen Bus zugegriffen werden.

durch eine Wertetabelle (Look Up Table, LUT) korrigieren. Für die Positionsbestimmung durch Ladungsteilung ist es besonders wichtig, dass die Sockelwerte und die Verstärkung benachbarter Kanäle identisch sind. Hierfür ist eine dynamische **Pedestal-Korrektur** implementiert. Da es durch das Übersprechen benachbarter Kanäle zu Unterschwingern kommen kann und diese zu Korrektur erfasst werden müssen, kann zusätzlich ein konstanter Wert für alle Kanäle aufaddiert werden. Allein durch die Variation der Vorverstärkerkanäle wird mit einen Unterschied von 5 % für die lineare Verstärkung gerechnet. Durch die **Gain-Korrektur** lässt sich der Verstärkungsfaktor eines jeden Kanals durch eine Multiplikation mit einer Konstanten anpassen.

Wie in Kapitel 2.3 beschrieben, wird an den Kathodenpads die induzierte Ladung der positiv geladenen Ionen gemessen. Diese driften relativ langsam aus dem Bereich des Anodendrahtes zum Kathodendraht. Mit wachsendem Abstand verringert sich der Einfluss auf die Kathodenpads. Einem schnellen Anstieg eines Signals folgt somit ein langsamer Abfall. Die **Ionenschweifunterdrückung** wird durch ein IIR-Filter (Infinite Impulse Response) zweiten Grades vorgenommen.

Durch die Größe eines Pads (ca. $0,7\,\mathrm{cm}\times 8\,\mathrm{cm}$) bildet es sowohl zur Masse als auch zu seinen Nachbarpads eine Kapazität. Die Kapazität zu einem Nachbarpad ($\sim 6\,\mathrm{pF}$) bewirkt ein Nebensprechen von etwa 6 % des Hauptsignals. Aufgrund der zur Signaländerung geringen Abtastrate lässt sich das Nebensprechen nur bedingt filtern. Die **Nebensprechunterdrückung** ist im TRAP durch ein FIR-Filter (Finite Impulse Response) realisiert.

Die Abbildungen 4.7 und 4.8 zeigen Ergebnisse, die mit einem der Implementierung entsprechenden Simulationsmodell (AliTRDtrap, Kapitel 3.2.4) erstellt wurden. Der Filter wird durch einen Impuls mit Tail am Eingang stimuliert. Für die weitere Verarbeitung werden diese Werte um 2 Bit erweitert. Der Ausgang des Nichtlinearitäts-Filters (schwarz) stellt damit die um einen Faktor 4 erhöhten Eingangswerte dar. Die Kompensierung des Tails (grün) hat als Nebeneffekt zur Folge, dass sich der Sockelwert um einen konstanten Faktor verrin-

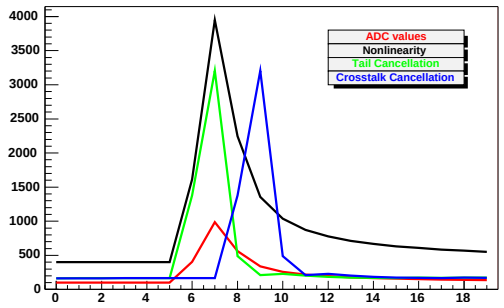

Abbildung 4.7: Antworten der verschiedenen Filterstufen auf einen typischen Impuls mit Tail (rot) am Eingang (rot).

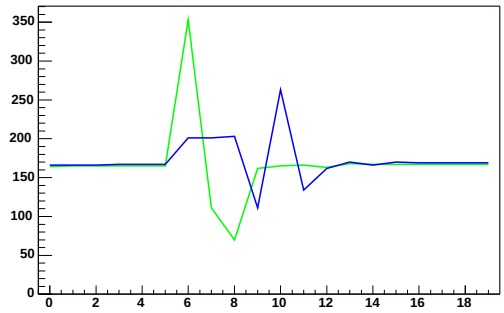

Abbildung 4.8: Die Auswirkung des Übersprechens (grün) wird durch das Übersprechfilter kompensiert (blau).

gert, der von den entsprechenden Filterparametern abhängt. In der FIR-Implementierung der Crosstalk-Unterdrückung (blau) werden die letzten beiden Zeitabschnitte berücksichtigt. Die Verzögerung des Ausgangs dieser Filterstufe lässt sich deutlich erkennen. Der Einfluss des Übersprechens auf die benachbarten Kanäle ist in Abbildung 4.8 (grün) dargestellt. Durch das Filter wird dieser Effekt um etwa einen Faktor zwei (blau) reduziert.

4.1.3 Präprozessor

Der Präprozessor verarbeitet in jedem Zeitabschnitt (während der Driftzeit) die Werte von 21 ADCs und stellt am Ende der Driftzeit den CPUs die Parameter für den linearen Regressionsfit zur Verfügung. Die Parameter sind die in Kapitel 3.3.2 beschriebenen Summen $\sum x_i$, $\sum y_i$, $\sum x_i^2$, $\sum y_i^2$, $\sum x_i y_i$ und $\sum Q_i$, wobei x_i der Zeitabschnitt (entspricht der x-Richtung) und y_i die Position in y-Richtung ist. Die Gesamtladung Q wird zur Unterscheidung von Elektronen und Pionen genutzt (Abbildung 3.7). Um Granularitätsverluste in y-Richtung zu vermeiden, werden die 18 aktiven Kanäle, für die ein MCM zuständig ist, um drei Kanäle erweitert – zwei links (in negativer y-Richtung) und einen rechts. Hierbei werden die beiden äußersten ausschließlich zur Bestimmung der Hit-Summe benutzt. Über den zweiten zusätzlichen linken Kanal ist es möglich auch solche Spuren zu erkennen, die über zwei Pads von benachbarten MCMs reichen. Dabei wird die doppelte Erkennung von Spuren in beiden MCMs durch die asymmetrische Aufteilung ausgeschlossen.

Die Rekonstruktion der Position einer Ladungswolke in der Driftkammer zu einem Zeitabschnitt erfolgt mithilfe der Ladungsteilung (Kapitel 3.3.1). Dazu werden drei ausgewählte benachbarte Kanäle zusammengefasst und als „Hit“ gekennzeichnet. Die Hit-Erkennungseinheit vergleicht dazu die Summe der ADC-Werte der drei Kanäle, markiert die Position des zentralen Kanals, dessen Wert größer-gleich als der seines linken und größer als der seines rechten Nachbarn ist, und gibt diese Positionen an die Hit-Auswahl weiter.

Die Hit-Auswahl vergleicht bis zu sechs Hits, die durch einen Prioritätsenkoder ausgewählt werden. Von diesen Hits werden diejenigen vier ausgewählt, die die größte Ladungssumme aufweisen. Die so ausgewählten (bis zu) vier Kanäle werden zusammen mit den jeweiligen

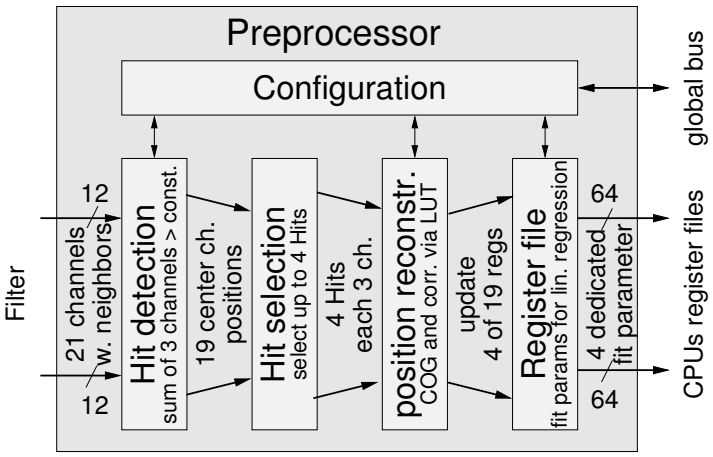

Abbildung 4.9: Aus den 19 zentralen Kanälen (21 mit Nachbarn) werden bis zu vier ausgewählt (Hits) und die Position rekonstruiert. Den CPUs werden die Parameter für den linearen Fit zur Verfügung gestellt.

Nachbarkanälen zur Positions-Rekonstruktion weitergeleitet. Aufgrund der Okkupanz des Detektors werden durchschnittlich etwa zwei Hits pro 21 Kanäle erwartet.

Im Gegensatz zu der in Kapitel 3.3.1 vorgestellten Methode des COG zur Positionsbestimmung werden im Präprozessor die Nachbarkanäle in der Gewichtung vernachlässigt und die Position mit $y_m = \frac{R-L}{C}$ berechnet. Da sich die Pad-Antwort-Funktion nicht durch eine einfache rationale Funktion adäquat annähern lässt, wird die errechnete Position über eine Wertetabelle korrigiert. Aufgrund der definierten Antwort-Funktion und der bekannten Verteilung der Ladung über die Pads (in Abhängigkeit der Position), lässt sich außerdem ein Qualitätsmaß errechnen, mithilfe dessen „unsaubere“ Hits aussortiert werden können. Dies ist der Fall, wenn z. B. zwei Spuren sehr nahe beieinander liegen. Hierfür wird das Verhältnis $\theta = \frac{L \cdot R}{C^2}$ benutzt.

Die vier ermittelten Positionen werden in der letzten Stufe, dem Register-File, wieder dem ursprünglichen Kanal zugeordnet. Hier werden die oben genannten Summen für jeden Kanal gebildet, die direkt für den linearen Fit genutzt werden können. Zusätzlich wird die Anzahl der Hits, die in diesem Kanal bearbeitet wurden, gespeichert. Da sich ein interessantes Spursegment über maximal zwei Pads erstecken kann, wird für die Auswahl der Spurparameter, die an die CPUs weitergegeben werden, jeweils ein Kanalpaar herangezogen. Als Kriterium wird die Summe der Hits eines Paares benutzt, und die bis zu vier Paare mit den meisten Hits werden weiterverarbeitet.

4.1.4 Prozessor

In Anwendung des TRDs als Trigger verarbeitet der Prozessor die durch den Präprozessor erzeugten Parameter für den linearen Fit. Er berechnet die durch die Gleichungen 3.1-3.3

beschriebenen Achsenabschnitte und Steigungen der bis zu vier Spursegmente. Diese Ergebnisse werden, wie in Kapitel 3.5.4 beschrieben, kodiert und an die Netzwerkschnittstelle übergeben.

Durch die Flexibilität eines Prozessors ist es nicht nur möglich, die Kriterien zur Spurfindung oder die Kodierung und Transformation zu variieren, sondern auch beliebige weitere Aufgaben zu übernehmen. So können z. B. Selbsttests der Elektronik vorgenommen werden oder die Daten des 22-sten ADCs (Kapitel 4.1.1), der Spannungen und Temperatur überwacht, ausgewertet werden.

Der Prozessor [21][22] besteht aus vier unabhängigen zentralen Prozessor-Einheiten (CPUs), deren Datenpfade über ein globales Register-File (GRF) und den Datenspeicher (DMEM) gekoppelt sind. Diese beiden Speicher verfügen jeweils über vier Ports, die einzeln adressiert werden können. Hiermit ist es möglich einfach und schnell Daten zwischen den CPUs auszutauschen bzw. auf die CPUs zu verteilen. Eine Anwendung des DMEMs ist es, z. B. dynamische Wertetabellen zur Verfügung zu stellen, die für alle CPUs gleich und somit effizient implementiert sind.

Zum Anschluss anderer Baugruppen an den Prozessor verfügt jede CPU über eine eigene Schnittstelle (Local I/O, LIO) mit einem Adressraum von 16 Bit. Ein Pseudogerät dieser Schnittstellen ist der Arbiter für den globalen Bus (GIO). Da dieser Arbiter neben den Adress- und Datenleitungen einige ausgezeichnete Steuerleitungen zu den CPUs benötigt, wird er als Teil des Prozessors gesehen.

Im Folgenden wird auf die Architektur einer einzelnen CPU eingegangen. Die Harvard-Architektur – Instruktions- und Datenspeicher sind getrennt – erlaubt eine konfliktfreie Abarbeitung von Programmen, auch bei häufigem Zugriff auf den Datenspeicher. Der Instruktionsspeicher ist für jede CPU separat ausgeführt und ermöglicht so eine bessere Optimierung der Algorithmen. Die CPU verfügt über einen 32 Bit breiten Datenpfad und ist in zwei Pipelinestufen geteilt. In der ersten Stufe wird die Instruktion aus dem Instruktionsspeicher (IMEM) gelesen, dekodiert und in das Pipelineregister geschrieben[1]. Während des zweiten Zyklus werden die Operanden von den verschiedenen Datenquellen ausgewählt, die Berechnung durchgeführt und das Ergebnis zurückgeschrieben oder es erfolgt ein Speicherzugriff auf das externe DMEM[2]. Die Operanden können dabei von dem GRF, dem privaten Register-File (PRF), dem Konstanten-Register-File (Const) oder von dem in die CPU abgebildeten Register-File des Präprozessors (Fit) gelesen werden. Das PRF enthält die 16 Arbeitsregister einer CPU. Die ebenfalls 16 Konstantenregister enthalten zur einen Hälfte häufig benutzte Werte wie z. B. die CPU-Identifikationsnummer, eins und minus eins. Die andere Hälfte lässt sich über die CPUs und über das SCSN konfigurieren.

Arithmetisch-logische Einheit (ALU)

Die zentrale Einheit einer CPU ist die Arithmetisch-logische Einheit [25], die Operanden miteinander verknüpft. Die ALU ermöglicht die 32-Bit-Operationen:

[1] instruction fetch – instruction decode
[2] operand fetch – execution – write back *or* memory access – write back

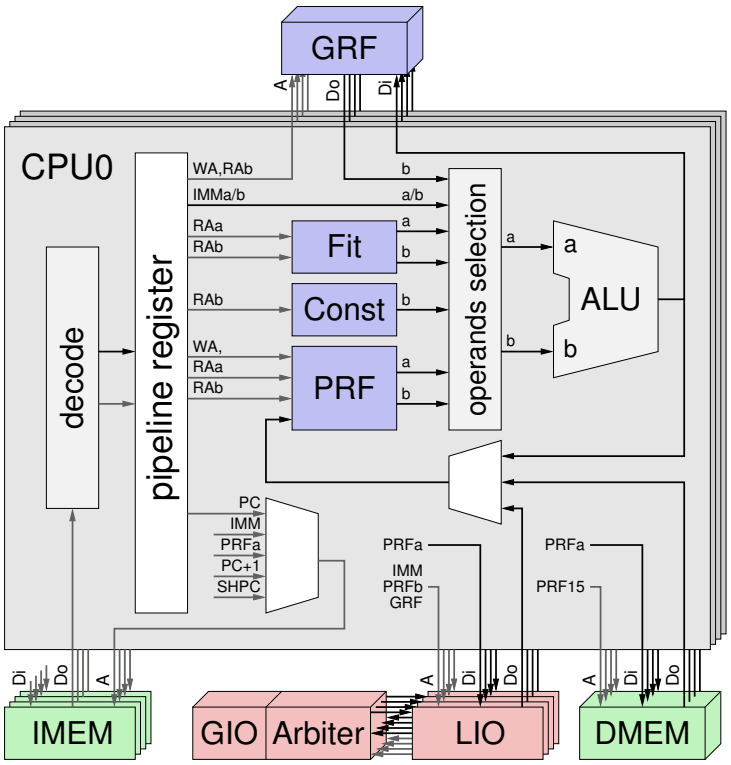

Abbildung 4.10: Der Prozessor besteht aus vier unabhängigen CPUs, die ihre Daten über das globale Register-File (GRF) und den Datenspeicher (DMEM) teilen, die mit vier Lese- und Schreib-Ports ausgestattet sind. Jede CPU ist als Harvard-Architektur mit zwei Pipeline-Stufen aufgebaut. Die Möglichkeiten der Adressierung und Operandenauswahl ist durch a und b gekennzeichnet, die die Eingänge der ALU bilden.

- **Addition** und **Subtraktion**, jeweils mit oder ohne Überlauf
- **Exklusives Oder**, **Und**, **Oder** und **Zweier Komplement**
- **Multiplikation** (32 Bit → 64 Bit) und **Division** (64 Bit → 32 Bit)
- **Schieben**, **Arithmetisches Schieben**, **Rotieren** und **Einer Komplement**

Mit Ausnahme der Division werden alle diese Operationen in einem Verarbeitungsschritt ausgeführt. Mit dem Aufruf eines Divisionsbefehls werden die Operanden übernommen, und die Division erfolgt parallel zum weiteren Programmverlauf. Ist die Division abgeschlossen (nach 18 Taktzyklen), signalisiert die ALU der CPU, dass das Ergebnis in den entsprechenden Registern zur Verfügung steht

Quad-Port-Memory (QPMEM)

Der Datenspeicher des TRAPs mit seinen vier unabhängigen Schreib- und Leseports wurde für den benutzten UMC-0,18-µm-Prozess im Rahmen einer Diplomarbeit [26] und Disserta-

tion entworfen und implementiert. Das DMEM erlaubt direkten Zugriff auf die 1024 32 Bit breiten Einträge innerhalb eines Takts. In der Arbeitsumgebung des Speichers, also während des Experiments, kann es durch ionisierende Strahlung in seltenen Fällen zu statistisch verteilten Bitfehlern kommen. Um die Fehlertoleranz der Speicherzellen zu erhöhen, ist das DMEM mit einer Hamming-Kodierung versehen, das die 32-Bit-Wörter intern auf 39 Bit erweitert und somit 1-Bit-Fehler korrigieren und 2-Bit-Fehler erkennen kann.

4.1.5 Netzwerkschnittstelle

Die Netzwerkschnittstelle bildet die Verbindung des TRAPs zum Auslesenetzwerk (Kapitel 5). Die Anbindung zum Prozessor und eine ausführliche Beschreibung der Architektur und Funktionsweise ist in Kapitel 6 zu finden.

4.2 Design-Fluss am Beispiel des TRAPs

Ein wichtiger Gesichtspunkt beim Entwurf eines Systems mit hoher Komplexität und Integrationsdichte, wie es bei TRAP der Fall ist, ist der Design-Fluss und die Verifikationsmöglichkeit in den verschiedenen Entwicklungsstadien. Die Spezifikation des System findet in diesem Fall in erster Linie über die Beschreibung der Funktionalität entsprechend der Anforderungen im Zusammenhang mit der Entwicklung eines Modells in der Simulationsumgebung (Kapitel 3) statt.

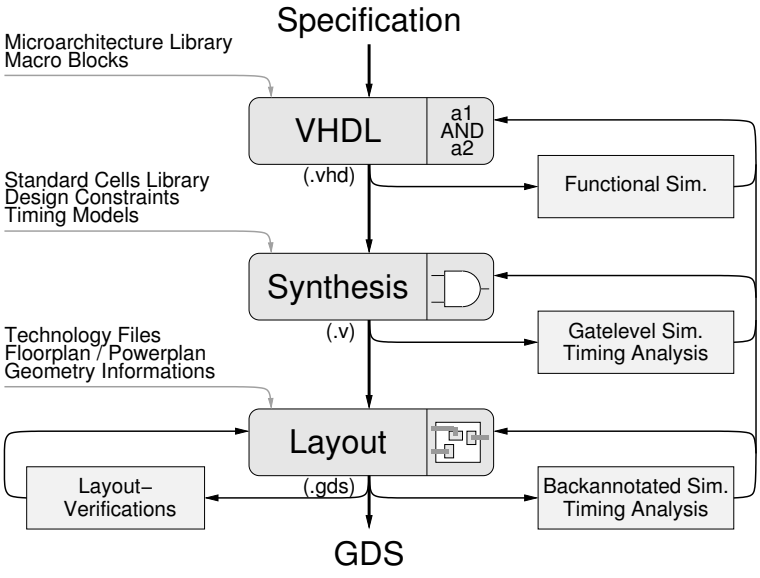

Abbildung 4.11: Die Abbildung zeigt den abstrakten Ablauf für einen digitalen Design-Fluss. Als Hardware-Beschreibungssprache ist hier VHDL gewählt. Durch die Synthese wird die Beschreibung zunächst in eine abstrakte Netzliste übersetzt und danach auf die gewählte Zielarchitektur abgebildet. Im Layout werden die Zellen platziert und verdrahtet.

In den folgenden Kapiteln werden die einzelnen in Abbildung 4.11 dargestellten Entwurfsstadien näher beschrieben. Dabei beziehen sich alle Angaben zu den Programmen, deren Funktionalität und Arbeitsweise oder den benötigten Dateiformaten auf den Design-Fluss des TRAPs. Eine Übersicht der benutzten Programme und Versionen ist in Tabelle 4.1 dargestellt.

Hersteller	Paket	Programm	Version	Funktion
Synopsys	DesignCompiler	dc_shell	2003.03	Synthese
	Primetime	pt_shell	2003.03	Timing Analyse
	Formality	fm_shell	2004.03	Netzlistenvergleich
Model	Modelsim	vsim	5.7c	Simulation
Cadence	FirstEncounter	encounter	3.20	Automatisiertes Layout
	Custom IC Design	icfb	5.0.32	Full-Custom-Layouts
Mentor	Verifikation	calibre	9.3	DRC, Extraktion, LVS

Tabelle 4.1: Verwendete Softwareversionen im Design-Fluss des TRAPs.

4.2.1 Hardwarebeschreibung mit VHDL

Um komplexe digitale Schaltungen entwerfen und handhaben zu können, werden Hochsprachen zur Hardwarebeschreibung, wie z. B. VHDL[3], eingesetzt. Es ist eine menschenlesbare Sprache mit ausführlich gehaltener und selbsterklärender Syntax. VHDL ist programm- und technologieunabhängig und durch die Definition der Sprache als IEEE-Standard (später auch als ANSI-Standard) sehr weit verbreitet. Als Nachteil sollte genannt werden, dass VHDL keine Konstrukte zur Modellierung analoger Systeme bereitstellt. Außerdem ist VHDL eine Beschreibungssprache, d. h. das Verhalten lässt sich zwar simulieren, aber nicht unbedingt auch synthetisieren (Kapitel 4.2.3).

Die Beschreibung eines VHDL-Modells besteht im wesentlichen aus einer Schnittstellenbeschreibung und der Architektur. Die Schnittstelle (Entity) beschreibt das Aussehen des Moduls nach außen, also Ein- und Ausgänge sowie Konstanten und Unterprogramme. Die Architektur (architecture) enthält die Funktionalität des Modells.

Die für VHDL übliche Standard-Logik-Bibliothek (1164) beinhaltet die Definition von Verbindungsdatentypen („1“, „0“, „hoch-ohmig“, „undefiniert“, ...) und alle grundlegenden Logik-Operatoren („nicht“, „und“, „oder“, ...). In VHDL wird zwischen einer Verhaltensmodellierung (behavioral) und einer strukturalen Modellierung (structural) unterschieden. Bei einem Verhaltensmodell wird die Reaktion der Ausgangssignale auf Änderung der Eingangssignale beschrieben. Dies kann durch nebenläufige (Signalzuweisungen, Logikelemente) und sequentielle Anweisungen (Verzweigungen, Schleifen) geschehen. In einer strukturalen Modellierung werden die Eigenschaften eines Modells durch seinen inneren Aufbau dargestellt. Hiermit lassen sich eingebundene Unterkomponenten verdrahten und eine Hierarchie erzeugen. Auf diese Weise sind bei TRAP unter anderem die großen Module, wie z. B. der Prozessor oder die Netzwerkschnittstelle, nebeneinander entwickelt und

[3] V(ery high speed integrated circuits) Hardware Description Language

synthetisiert (Kapitel 4.2.3). In einem übergeordneten strukturalen Modell sind dann diese Module zusammengefasst und verdrahtet. Dies hat nicht nur den Vorteil, dass unabhängig voneinander entwickelt werden kann, sondern auch, dass die Komplexität zur Entwicklung (Simulationen, Synthese) begrenzt wird.

Neben den eigenen in VHDL entwickelten Unterkomponenten stehen zudem so genannte Mikroarchitektur-Bibliotheken zur Verfügung, die z. B. Komponenten wie Addierer oder FIFOs bis hin zu Trigonometrischen Funktionen oder die Erzeugung zyklischer Kodes enthalten. Diese Bibliotheken sind von dem benutzen Synthese-Programm abhängig.

Um unabhängig entwickelte Makroblöcke wie z. B. Speicher, ADCs oder eine PLL einzubinden, benötigt man für die Beschreibung in VHDL lediglich die exakte Schnittstellenbeschreibung. Erst bei der Synthese oder der Simulation werden dann die Modelle für das zeitliche bzw. funktionale Verhalten geladen.

4.2.2 Design Bibliotheken

Jedes Programm benötigt neben der Ausgabe der jeweils vorherigen Designstufe zusätzliche Informationen, z. B. Bibliotheken zur Technologie. Abbildung 4.12 zeigt eine Übersicht zu diesen Dateien für den Design-Fluss des TRAPs.

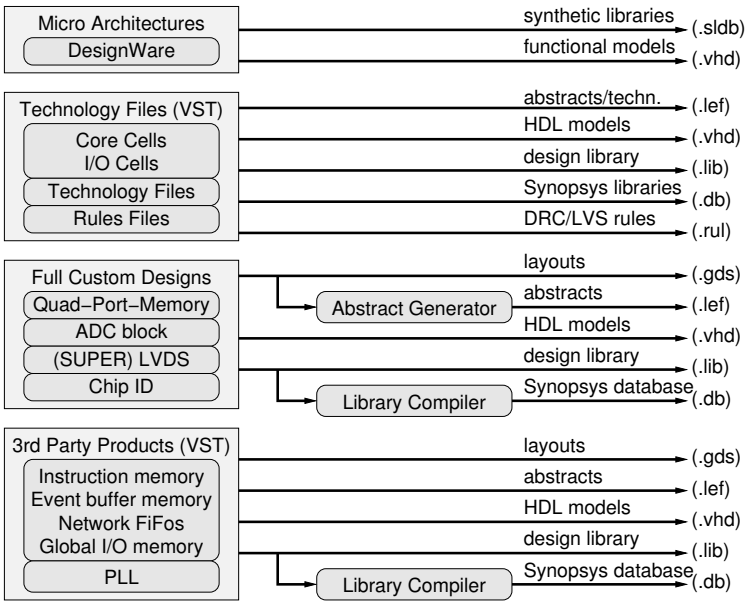

Abbildung 4.12: Das Diagramm zeigt die im Design-Fluss des TRAPs benutzten Bibliotheken getrennt nach ihren Quellen. Die benutzten Dateien liegen für jedes Modul separat vor oder müssen erzeugt werden.

Die verwendete Mikroarchitektur-Bibliothek von Synopsys (DesignWare IP) wird vom Syntheseprogramm und zur funktionalen Simulation benutzt. Sie enthält komplexere Komponenten (Kapitel 4.2.1), die direkt in den VHDL-Kode eingebunden und während der Synthese auf Standardzellen abgebildet werden.

Für den Prozess erhält man die Standardzellen, Technologiedaten und Regeldateien. Im Fall des TRAPs handelt es sich um einen $0,18\,\mu m$ CMOS Prozess der Firma United Microelectronics Company (UMC), der von IMEC vertrieben wird, einem Design Serviceunternehmen für Mikroelektronik. IMEC beteiligt sich am Europractice IC Service, wodurch es für Universitäten und Forschungseinrichtungen möglich ist vollständige Design-Umgebungen – erschwinglich – zu nutzen. Das Design-Paket beinhaltet eine Zusammenstellung aller notwendigen Bibliotheken und Dateien, die von den Programmen benötigt werden. Die Standardzellen-Bibliothek stammt von Virtual Silicon Technologies (VST), einem Anbieter für Halbleiter IPs (Semiconductor Intellectual Property, SIP). Im Allgemeinen werden die Layouts der Standardzellen nicht mitgeliefert. Die entsprechenden Abstracts enthalten alle für ein Layout notwendigen Informationen, wie die Geometrie einer Zelle oder die Positionen der Pins und Blockierungen, die ein falsches Verdrahten verhindern. Vor der Fertigung werden von dem Anbieter diese Abstracts durch die Layouts der Standardzellen ersetzt und finale Test an dem vollständigen Layout durchgeführt. Bei TRAP werden diese Schritte selbst durchgeführt, da die Layouts zur Verfügung stehen.

Die fertigen eigenentworfenen (analogen) Komponenten des Designs bestehen zunächst aus dem Schaltplan und dem Layout. Um die Komponenten in die VHDL Beschreibung einbinden zu können, werden von Hand Bibliotheken erzeugt, die dann mithilfe eines Programms (LibraryCompiler) in ein Synopsys Datenbank Format gewandelt werden. Hiermit werden alle Pins und deren Kapazitäten und zeitliches Verhalten beschrieben. Zur Simulation wird außerdem ein VHDL Verhaltensmodell entworfen, um die Schnittstellenlogik zu prüfen. Das Layout Programm benötigt zudem, wie für die Standardzellen, die Abstracts der Blöcke. Diese lassen sich automatisch aus dem Layout generieren.

Für das Design des TRAPs werden automatisch generierte Speicherblöcke und PLLs benutzt. Der Dritthersteller VST bietet Compiler an, mit denen über eine einfach zu bedienende Oberfläche diese Komponenten mit allen benötigten Dateien erzeugt werden können.

Zur besseren Übersicht sind in Tabelle 4.2 alle benutzten Typen von Bibliotheken mit Suffix, dem Inhalt der Dateien und den Programmen, von denen sie benötigt werden, aufgelistet.

Typ	Suffix	Inhalt	benötigt für
Synthetische Bibliothek	.sldb	Mikroarchitekturen	Synthese
Verhaltensmodell	.vhd	Simulationsmodell, VITAL Modell	Simulation
Design Bibliothek	.lib	Zeitverhalten, Pin-Kapazitäten, Verlustleistung, Fläche	Library Compiler, Layout.
Synopsys Datenbank	.db	Binäres Synopsys Format von .lib	Synthese, Timing-Analyse.
Abstracts	.lef	Geometrie, Pin Positionen	Layout
Technologie Datei	.lef	Lagen-/Routing Informationen, VIAs, Kapazitätsbeläge	Layout
Regeln	.rul	Layout-Regeln, Extraktions-Regeln	Layout-Verifikationen
Layout	.gds	Layout-Informationen zur Chipfertigung	Design Checks, Fertigung

Tabelle 4.2: Dateiformate der Design Bibliotheken.

4.2.3 Synthese

Unter der Synthese versteht man den Übergang von der Verhaltensbeschreibung zu einer dieses Verhalten realisierenden Struktur. Dies erfolgt in zwei Schritten. Zunächst werden die Register-Transfer-Ebenen und die algorithmischen Strukturen analysiert und mithilfe einer technologieunabhängigen Netzliste auf Logikebene dargestellt. Die dabei verwendeten Logikelemente sind programmabhängig und dienen nur zur internen Darstellung. Als zweiter Schritt werden alle Logikelemente durch technologiespezifische Gatter ersetzt. Durch lokales Neuarrangieren von Komponenten oder Verwendung von Bausteinen mit unterschiedlicher Anzahl von Eingängen wird versucht, möglichst passende Gatter zu finden. Dieser zweite Schritt ist ein iterativer Prozess, in dem das Syntheseprogramm sich den Vorgaben (constraints) des Entwicklers annähert. Solche Vorgaben bestehen aus dem erlaubten Platzbedarf, der erstrebten Geschwindigkeit und einer möglichst geringen Verlustleistung.

Die Standardzellen-Bibliothek enthält eine Liste aller in der Technologie vorhandenen Gatter mit allen für die Synthese notwendigen Informationen, wie Funktion, Verzögerungszeiten, Kapazitäten der Pins, Fläche und Stromverbrauch. Die Funktion der Zellen wird benötigt, um eine optimale Auswahl bei identischer logischer Funktion zu erreichen. Die Bibliotheken größerer Makroblöcke, wie z. B. der eigenentwickelte Speicher, enthalten diese Beschreibung der Funktion nicht. Die wichtigsten Vorgaben zur Optimierung einer Schaltung sind die Beschreibungen der Taktsignale. Hiermit legt man die maximale Verzögerungszeit durch die kombinatorischen Pfade fest. Die Ein- und Ausgänge der obersten Ebene (Ports), die gerade synthetisiert wird, nehmen hierbei eine besondere Stellung ein, da das zeitliche Verhalten der an die Ports angeschlossenen Logik berücksichtigt werden muss. Dies wird erreicht, indem die Verzögerungen der Signale des angeschlossenen Moduls ebenfalls über Vorgaben angegeben werden. Die Summe dieser jeweils externen und der internen Verzögerungen ergibt die Gesamtverzögerung des Pfades. Bei einem hierarchischen Synthese-Fluss ist die optimale Aufteilung oftmals ein iterativer Prozess. Für die Synthese ist es außerdem notwendig, dass unkritische Pfade (Reset, Konstanten, Mehrzyklen-Pfade) definiert sind und somit unnötige oder gar unmögliche Optimierungen vermieden werden.

Zur Synthese werden neben den Beschreibungen in VHDL die Mikroarchitektur-Bibliothek (Kapitel 4.2.1), die Standardzellen-Bibliothek, Vorgaben zur Optimierung und die in Abbildung 4.12 gezeigten Bibliotheken benötigt. Abbildung 4.13 zeigt das hierarchische Vorgehen bei der Synthese des TRAPs. Zur Synthese wird der DesignCompiler der Firma Synopsys verwendet. Die großen Module Präprozessor, Prozessor und Netzwerkschnittstelle (Kapitel 6), gegebenenfalls sogar Untermodule (Filter, Ereignisspeicher, etc.), werden separat voneinander synthetisiert und optimiert und danach in einem übergeordneten Design zusammengefasst. Eine Synthese, die alles enthält, wäre nicht sinnvoll, da die Rechenzeit zu lang wäre. Allein die Synthese des Präprozessors dauert mehrere Stunden. Auch das Einbinden der drei fertig synthetisierten großen Module als vollständige Designs in das übergeordnete Design (TopTI) hat sich als nicht praktikabel herausgestellt, da der DesignCompiler weiterhin das vollständige Design analysiert und versucht, seine Optimierungen dem vorhandenen Design anzupassen. Sie werden als leere Komponenten (black boxes) in das übergeordnete Design (TopTI) eingebunden. Auf diese Weise lassen sich die Haupt-

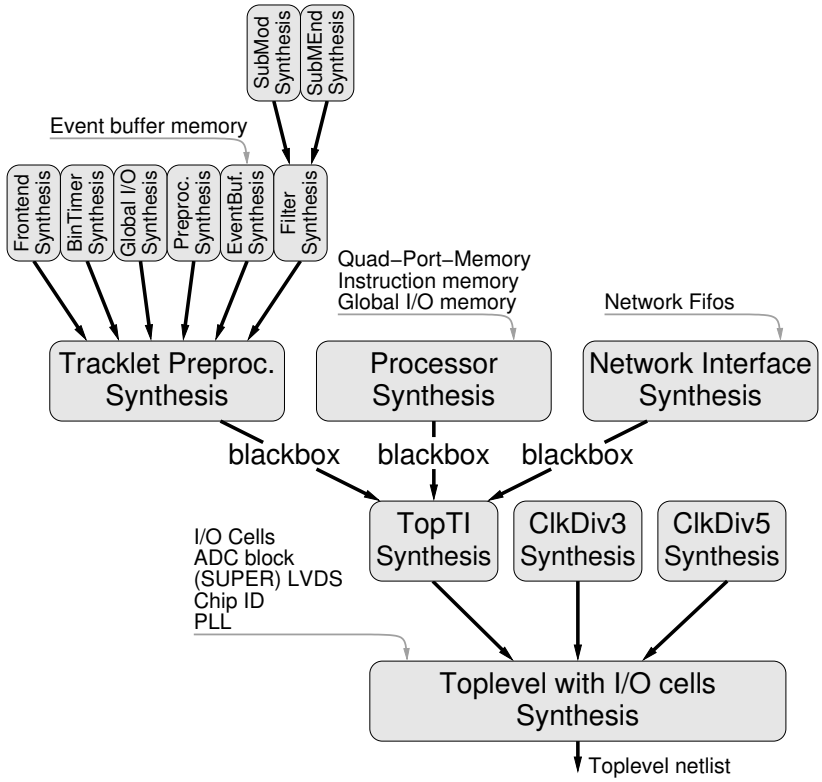

Abbildung 4.13: Die Synthese des TRAPs erfolgt in mehreren Stufen. Zur TopTI Synthese werden die drei großen Komponenten lediglich als leere Module behandelt. Die Bibliotheken über den hellgrauen Pfeilen entsprechen denen aus Abbildung 4.12.

komponenten unabhängig voneinander entwickeln und ohne großen Zeitaufwand eine neue Toplevel Netzliste erstellen. Die vollständige Netzliste, die alle Module enthält, wird vor dem Layout (Kapitel 4.2.6) durch Aneinanderhängen der einzelnen Netzlisten erzeugt.

Wie zu Beginn des Kapitels erwähnt, sind bei der Art der hierarchischen Synthese, wie sie bei TRAP verwendet wird, besondere zeitliche Vorgaben zu berücksichtigen. Es muss die Verzögerungszeit der Schnittstellen der angeschlossenen Module exakt angegeben werden, da eine Optimierung nach dem Zusammenfügen der Module nur sehr begrenzt möglich ist. Zwar besitzt jedes Modul eine eigene, durch die Master-State-Machine gesteuerte, Taktdomäne (Kapitel 4.1), die untereinander synchron sein müssen, dies muss jedoch erst während der Generierung des Taktbaums (Kapitel 4.2.6) berücksichtigt werden, da die Taktsignale während der Synthese als ideal angenommen werden.

4.2.4 Simulation

Die Simulation dient in erster Linie der funktionalen Verifikation eines Designs. Hierzu werden an ein Design, wie z. B. im Fall des Filters, Testvektoren angelegt und das Verhalten der VHDL Beschreibung mit dem Verhaltensmodell aus der AliRoot Simulation (Kapitel 3)

verglichen. Hat ein Design sehr viele Eingangsvektoren oder ein kompliziertes sequentielles Verhalten, lässt sich dieses oft nicht vollständig überprüfen. Hier beschränkt man sich auf die kritischen und einige typische Fälle. Um dennoch eine möglichst lückenlose Abdeckung der Untersuchung zu erreichen, wird nicht nur das vollständige Design (Toplevel) simuliert, sondern die Simulation wird zu einem Bestandteil jedes Entwurfschrittes und jedes Modul wird für sich getestet. Ein Design wird durch eine so genannte Testbench stimuliert. Dies ist ein das Design umgebendes VHDL Modell, das die Testmuster erzeugt oder aus einer Datei liest und an das Design, das als Komponente in die Testbench eingebunden ist, anlegt. Die Ergebnisse werden mit den erwarteten Werten aus dem Verhaltensmodell verglichen. Zur Suche nach Fehlern lassen sich die Wellenformen jedes beliebigen Signals anzeigen, so dass sich Signale auch innerhalb des Designs überprüfen lassen. Natürlich können auch mehrere Designs in einer Testbench eingebunden werden. Somit wird z. B. eine ganze Ausleseplatine (Kapitel 5.2.2) mit Leitungsverzögerungen, gegeneinander verschobenen Takten und serieller Konfiguration durch eine Kette von MCMs simuliert.

Bei der Simulation wird zwischen der funktionalen und der Netzlisten-basierten Simulation unterschieden. Bei der funktionalen Simulation wird der VHDL Quellkode gelesen und interpretiert. Durch die Simulation der Gatternetzliste kann überprüft werden, ob auch das synthetisierte Design mit dem Verhaltensmodell funktional übereinstimmt und ob die zeitlichen Anforderungen prinzipiell erfüllt werden können. Die genaueste Methode ist die Simulation der zurückannotierten Gatternetzliste. Während des Layouts wird die Netzliste aus der Synthese verändert, indem Treiber hinzugefügt oder gelöscht werden und auch Gatter (Core Zellen) durch andere ersetzt werden. Außerdem können die Einflüsse von Leitungskapazitäten aus dem Layout berücksichtigt werden. Diese Methode der Simulation ist sehr rechenaufwendig und speicherintensiv, so dass es z. B. nicht möglich ist eine vollständige Ausleseplatine auf diese Weise zu simulieren. Um dennoch eine Aussage über das Verhalten eines MCMs auf der Platine machen zu können, sind nur einzelne MCMs zurückannotiert und die übrigen funktional simuliert. Diese drei Möglichkeiten der Simulation werden nach den jeweiligen Entwurfschritten, wie in Abbildung 4.11 dargestellt, durchgeführt.

Tabelle 4.3 zeigt einige typische Simulations- und Ausführungszeiten für verschiedene Module. Da die zurückannotierte Gatternetzliste erst nach dem Layout zur Verfügung steht und das Zeitverhalten des gesamten Chips enthält, ist für die Netzwerkschnittstelle separat nur die funktionale und die Simulation der Gatternetzliste möglich. „Alle Simulationen“ der Netzwerkschnittstelle enthält neben Tests der verschiedenen Betriebsmodi auch zwei Varianten, in denen eine gesamte Ausleseplatine simuliert wird. Hierbei wird die Platine jedoch nur durch Netzwerkschnittstellen aufgebaut. In Kapitel 7.3 sind einige Simulationsergebnisse für die Netzwerkschnittstelle bzw. die Ausleseplatine vorgestellt.

Die Simulationszeit für den Tracklet-Prozessor bzw. die Ausleseplatine ergibt sich hauptsächlich aus der seriellen Konfiguration 4.1 über das SCSN. So werden z. B. von den 2 ms Simulationszeit nur etwa 20 µs Berechnungen auf dem Präprozessor und Prozessor durchgeführt. Der Simulator arbeitet jedoch Ereignis-basiert. Da während der Konfiguration die Taktsignale der meisten Module abgeschaltet sind, wird hierfür nur etwa die Hälfte der Ausführungszeit benötigt. Bei der Simulation der Ausleseplatine mit zurückannotierter

Modul	Simulationszeit	Methode	Ausführungszeit
Netzwerkschnittstelle (nur Triggerauslese)	1 µs	funktional	1,5 s
		synthetisiert	4 s
Netzwerkschnittstelle (alle Simulationen)	5 µs	funktional	22 s
		synthetisiert	2 m 50 s
Tracklet-Prozessor	2 ms	funktional	4 m
		synthetisiert	30 m
		zurückannotiert	1 h 30 s
Ausleseplatine	4,5 ms	funktional	1 h 30 m
		synthetisiert	2 h
		zurückannotiert*	3 h

Tabelle 4.3: Es sind für verschiedene Module die Zeiten gelistet, die simuliert werden und die jeweiligen Ausführungszeiten von Modelsim. Alle Simulationen der Netzwerkschnitte beinhalten fünf Tests der Schnittstelle und zwei für eine Ausleseplatine, die ausschließlich aus Netzwerkschnittstellen aufgebaut sind.
* Einer der 17 Tracklet-Prozessoren wird zurückannotiert, die übrigen funktional simuliert.

Gatternetzliste ist nur ein Tracklet-Prozessor mit dieser Netzliste instantiiert. Die übrigen 16 werden funktional beschrieben. Hierbei liegt die Limitierung im Speicher (4 GByte) des verwendeten Rechners.

4.2.5 Zeitverhalten

Durch die funktionale Simulation (Kapitel 4.2.4) ist es möglich, die Funktionalität einer Schaltung zu verifizieren. Das zeitliche Verhalten (Timing) bleibt dabei zunächst unberücksichtigt. Durch die Simulation der Gatternetzliste oder der zurückannotierten Netzliste können zwar auch solche Fehler entdeckt werden, aber nur für die getesteten Zustände der Schaltung. Statische Timing Analyse Programme arbeiten auf beiden genannten Arten von Netzlisten, indem sie alle möglichen Pfade durch die Schaltung auf zeitliche Verletzungen hin untersuchen. Die wichtigsten Pfade sind die von einem Speicher zum nächsten, deren maximale Verzögerung durch die Frequenz des angelegten Takts vorgegeben ist. Der langsamste Pfad bestimmt die maximale Betriebsfrequenz der Schaltung.

Abbildung 4.14 oben zeigt eine Schaltung, an der sich typische zeitliche Vorgaben und Abhängigkeiten erläutern lassen. Der Speicher reg1 übernimmt zur steigenden Flanke zum Zeitpunkt 0 die Eingangsdaten, die kurze Zeit später am Ausgang reg1Q anliegen. Durch die Logik A entstehen weitere Verzögerungen, so dass die Signale am Speichereingang reg2D erst nach der Zeit Tpd (Propagation-Delay) anliegen. Der Speicher reg2 übernimmt dann die Daten zum Zeitpunkt 2. Um die positive Taktflanke herum gibt es ein Zeitfenster, in dem das Datum stabil anliegen muss, damit das Flipflop sicher arbeitet. Ist das Datum zu spät relativ zur Taktflanke stabil, spricht man von einer „Setup-Time-Violation“. Diese Art der Verletzung entsteht in der Regel durch eine lange Verzögerung der Logik zwischen den Speicherelementen. Bleibt das Datum nicht lang genug stabil, spricht man von einer „Hold-Time-Violation“. Diese Verletzung tritt vor allem dann auf, wenn es nur sehr wenig

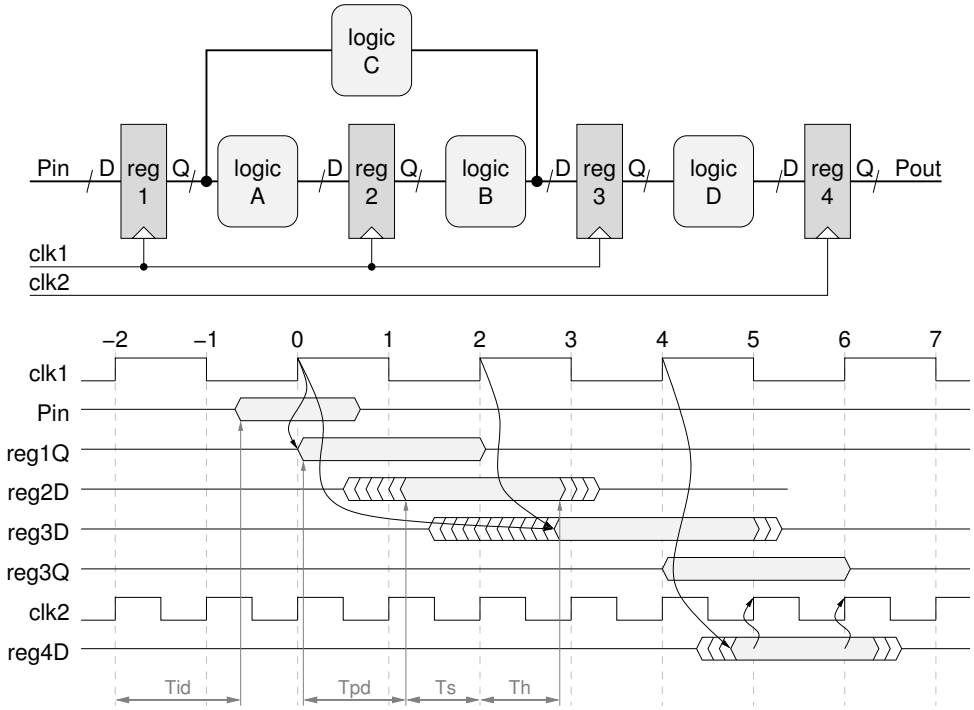

Abbildung 4.14: Im oberen Teil der Abbildung sind einige typische Anordnungen von Speichern und Logik dargestellt, die bei einer Timing-Analyse berücksichtigt werden müssen. Der untere Teil zeigt das Zeitdiagramm für ein Datenwort, das zum Zeitpunkt 0 von Pin in den Speicher reg1 übernommen wird.

oder gar keine Logik zwischen den Speichern gibt und die Taktsignale an den Speichern leicht gegeneinander verschoben sind. Die Unsicherheit im Takt nennt man Taktversatz. So kann der Taktversatz sich effektiv wie eine zusätzliche Verzögerung zur Logik, aber auch entgegengesetzt, auswirken.

Weiterhin können durch die Architektur eines Designs Anforderungen an das zeitliche Verhalten entstehen. So übernimmt in diesem Beispiel der Speicher reg3 die Daten, die unter anderem durch Logik C gehen. Diese Art von Pfaden heißen „Multi-Cycles-Paths“. Sie entstehen entweder, wenn sich die Logik nicht in mehrere kleinere Logikblöcke aufbrechen lässt (Pipelining), oder wenn unterschiedliche Teile des Designs mit verschiedenen Frequenzen arbeiten. Durch den Speicher reg4 kann hierfür ein weiterer typischer Fall veranschaulicht werden. Er arbeitet mit der doppelten Taktfrequenz clk2 relativ zu seiner Datenquelle reg3. Im Zeitdiagramm lässt sich erkennen, dass in diesem Fall der Speicher reg4 zu den Zeitpunkten 5 und 6 die Möglichkeit hat, stabile Daten zu übernehmen. Auch uninteressante Pfade, die entweder zeitlich nicht kritisch sind (z. B. ein statisches Reset Signal) oder logisch nicht vorkommen, können in der Timing-Analyse durch entsprechende Vorgaben berücksichtigt werden.

Die Vorgaben für das zeitliche Verhalten eines Designs werden dem Analyseprogramm durch eine Datei (Synopsys-Design-Constraints) übergeben. Im Fall des TRAPs enthält diese Datei ausschließlich Angaben zum Timing und wird auch als Eingang für das Layout

benutzt. TRAP beinhaltet zwei PLLs und mehrere Taktteiler, so dass insgesamt sieben unterschiedliche Typen von Takten definiert sind. Durch die vielen Taktdomänen ist es notwendig eine ganze Reihe von falschen Pfaden und Mehrzyklenpfaden zu definieren. So propagieren z. B. die Parameter für die Filter nur sehr langsam durch den Präprozessor, können allerdings als statisch angesehen werden, da diese Parameter nur einmal vor Beginn der Messungen gesetzt werden. Das Setzen solcher falschen Pfade kann sehr komplex werden und man muss das Design genau kennen, um nicht unbeabsichtigt auch relevante Pfade auszuschließen.

Die Analyse selbst wird mittels eines Skriptes ausgeführt. Es werden die Bibliotheken der Makroblöcke und Standardzellen, die Netzliste, Extraktionsdaten aus dem Layout und die Constraints-Datei gelesen. Das erste Ergebnis liefert schon das Einlesen und Überprüfen der Konsistenz der Constraints-Datei. Es wird eine Liste mit Pfaden oder Speichern ausgegeben, für die noch keine Vorgaben gemacht sind. Dies ist besonders wichtig, da diese Vorgaben auch vom Layout Programm benutzt werden, um danach seine Optimierungen vorzunehmen. Das Layout Programm kann z. B. fehlende Constraints nicht erkennen und würde deshalb weder optimieren noch zeitliche Verletzungen feststellen. Die beiden wichtigsten Tests bei der Analyse sind die kritischen Pfade für die Setup- und Hold-Zeiten, d. h. die längsten und kürzesten Pfade zwischen zwei Speichern. Hierbei wird der Taktversatz und auch die Verzögerung durch den Taktbaum (Kapitel 4.2.6) berücksichtigt. Es lassen sich verschiedene Umgebungen simulieren und die Analyse erfolgt in „best-case“, „typical“ und „worst-case“ (Tabelle 4.4).

Umgebung	Temperatur	Spannung
best-case	0 °C	1,98 V
typical	25 °C	1,8 V
worst-case	125 °C	1,62 V

Tabelle 4.4: Umgebungen zur Timing-Analyse.

Während zur Aufdeckung von Setup-Time-Violations eine worst-case Analyse ausreichen würde, lassen sich zuverlässige Aussagen über Hold-Time-Violations nur durch die Simulation aller Umgebungen machen. Dies rührt daher, dass der Taktversatz hierbei eine wichtige Rolle spielt. So sind z. B. bei einer worst-case Analyse zwar die Verzögerungen zwischen den Speichern relativ groß (geringe Chance für Hold-Time-Violations) aber die größeren Verzögerungen durch den Taktbaum bewirken einen größeren Taktversatz, was den Datenverzögerungen entgegenwirken kann.

Die skriptbasierte Ausführung der Analysen ermöglicht es auch komplexere zeitliche Verhalten auszuwerten. Um benachbarte MCMs auf einer Ausleseplatine mit einem Takt zu versorgen, besitzt der TRAP vier Taktausgänge, die von der internen PLL gespeist werden. Über Pufferketten gelangt das Taktsignal von der PLL zu den örtlich verteilten I/O-Zellen. Natürlich wird ein möglichst sauberes Taktsignal an den Ausgängen gewünscht, d. h. steile Flanken, wenig Jitter und ein gutes Taktphasenverhältnis (Duty-Cycle). Gleiches gilt auch für die Taktausgänge der Netzwerkschnittstelle. Mittels der Timing-Analyse lassen sich

für jeden Pin und jede Leitung innerhalb des Designs hierüber genaue Aussagen machen (Abbildung 6.6) und daraufhin gegebenenfalls das Layout optimieren.

4.2.6 Layout

Das Ziel des Layouts ist es eine grafische Darstellungsform des Designs zu entwickeln, die die einzelnen Lagen des Prozesses darstellen und anhand derer die Fertigung des Chips erfolgen kann.

Die Erstellung des vollständigen Layouts bis hin zur Datei (GDSII[4]), die zur Fertigung des Chips benötigt wird, erfolgt in drei Schritten. Zunächst werden alle Zellen platziert und verdrahtet, was im Folgenden näher beschrieben wird. Im zweiten Schritt werden als „Full-Custom-Layout“ die Pads und die Scribeline, die die Schnittkante eines Chips definiert, hinzugefügt. Dieses Design wird an den Anbieter des Prozesses übergeben, der in einem letzten Schritt die Abstracts der Standardzellen durch die Layouts ersetzt und Markierungen (Butterflies) auf allen Lagen anbringt. Im Fall des TRAPs und des benutzten Prozesses stehen die Layouts zur Verfügung und werden nach dem Verdrahten der Zellen eingesetzt. Die Markierungen werden von dem Hersteller benötigt, um die Masken für die Belichtung exakt zu positionieren.

Für das Layout des TRAPs wird FirstEncounter von Cadence verwendet. Der Design-Fluss von FirstEncounter erlaubt einen hierarchischen Ansatz zur Partitionierung des Designs. Um dies in vollem Umfang nutzen zu können, sollten die einzelnen Module klare Schnittstellen haben und nicht ineinander greifen. Da es bei TRAP eine ganze Reihe von Taktdomänen gibt und die Übergänge zeitlicher Abhängigkeiten nicht unbedingt mit den Schnittstellen der Module übereinstimmen, ist hier ein flacher Layoutansatz gewählt. Es wird deshalb nicht näher auf die Möglichkeiten der Partitionierungen in FirstEncounter eingegangen.

Im Folgenden werden die einzelnen Designschritte für das Layout des TRAPs näher beschrieben.

Import des Designs

Wie in Kapitel 4.2.3 beschrieben, werden die drei Blöcke Präprozessor, Prozessor und Netzwerkschnittstelle separat voneinander synthetisiert. Vor dem Layout werden diese Netzlisten zusammen mit den leeren Netzlisten der Makroblöcke in eine einzige zusammengefasst.

Der Designimport erfolgt mittels einer Konfigurationsdatei, die folgende Definitionen enthält:

- Verilog Netzliste (.v) mit dem Bezeichner für das Toplevel Modul.

[4] Graphical Design System II, ein Datenformat zur Beschreibung von IC Layouts, das seit über 20 Jahren von der Chip-Industrie als ein Quasistandard benutzt wird. Quelle: http://en.wikipedia.org/wiki/GDSII

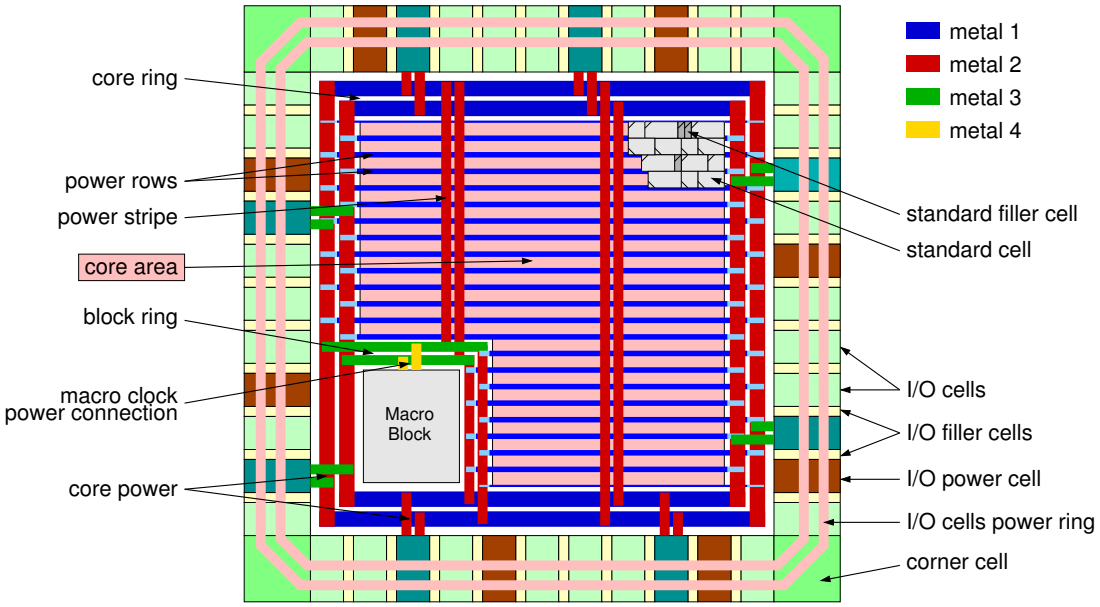

Abbildung 4.15: Schematische Darstellung eines typischen Layouts mit vier Metalllagen. (Dieses Layout entspricht nicht dem des TRAPs, sondern soll lediglich helfen die Struktur und verwendeten Begriffe darzustellen).

- Design Bibliotheken (.lib) der Technologie, der Standardzellen, der Full-Custom - Blöcke und der Blöcke von Drittanbietern (Abbildung 4.12).
- Abstracts (.lef) aller Design Bibliotheken.
- Vorgabendatei für das zeitliche Verhalten (timing constraints).
- wichtigste Informationen über die Größe des Chips, den Abstand zwischen I/O-Zellen und dem Core Bereich (Abbildung 4.15). Diese Angaben müssen zunächst nur ungefähr sein und werden durch den Floorplan exakt definiert.
- Bezeichner für die Power-Netze.

Floorplan

Mittels der Floorplan-Datei werden die exakten Maße des Chips (Core- und I/O-Bereich), die Anordnung der Rows, bevorzugte Verdrahtungsrichtungen und die Power-Netze definiert. Außerdem lassen sich Angaben zur Platzierung machen. Im Fall des TRAPs erfolgt die vollständige Platzierung, Mit Ausnahme der Standardzellen, durch den Floorplan. Dies beinhaltet die I/O-Zellen, die Power-Zellen, und alle Makroblöcke.

Der automatischen Platzierung lassen sich verschiedene Arten von Platzierungsvorgaben übergeben. Damit hat der Designer die Möglichkeit, durch zusätzliche Informationen über örtliche oder auch zeitliche Abhängigkeiten dem Platzierungsprogramm unter die Arme zu greifen. So genannte „Fences“ legen einen Bereich fest, innerhalb dessen bestimmte Teile

des Designs exklusiv, d.h. ansonsten keine anderen, platziert werden. Durch „Regions“ kann diese Vorgabe gelockert werden, indem erlaubt wird, auch Zellen anderer Teile des Designs innerhalb des Bereichs zu platzieren. So genannte „Guides“ erlauben es dem Platzierungsprogramm Zellen dieses Bereichs gegebenenfalls auch außerhalb zu platzieren. Es ist also eine Art Orientierungshilfe für den Platzierer.

Für TRAP wurden diese Möglichkeiten der Platzierungsvorgaben genutzt, um z. B. die Taktdividierer zusammen mit ihren Gates, die Resynchronisationseinheiten der Netzwerkschnittstelle oder den JTAG-Kontroller kontrolliert zusammenzuhalten.

Zu den Platzierungsvorgaben zählen auch die Bereiche, in denen keine Zellen platziert werden sollen oder in denen nicht verdrahtet werden soll. So sind bei TRAP z. B. solche „Routing-Blockages“ auf den entsprechenden Metall-Lagen im Core-Ring vorhanden. Wäre dies nicht der Fall, hätte der Router die Möglichkeit Signalleitungen zwischen den breiten Power-Bahnen des Rings zu verlegen, was bei der Fertigung des Chips zu Problemen führen kann.

Powerplan

Durch den Powerplan werden die Leitungen für die Spannungsversorgung aller Zellen definiert. Der Chip erhält seine Versorgungsspannung über die Power-Zellen. Hierbei wird unterschieden zwischen den Zellen, die den Core, und denen, die die I/O-Zellen versorgen. Die Versorgung der I/O-Zellen, die 1,8 V und 3,3 V benötigen, erfolgt über eigene Power-Ringe innerhalb der I/O Zellen. Durch I/O-Filler-Zellen und die Corner-Zellen (Abbildung 4.15) werden diese Ringe geschlossen. Im TRAP ist etwa ein Power/Ground Paar pro drei I/O-Zellen vorhanden.

Die Power-Zellen (24 Paare in TRAP) für den Core-Bereich, der mit 1,8 V arbeitet, werden direkt mit dem Core-Ring verbunden. Über diesen Ring werden alle Standardzellen und Makroblöcke versorgt. Um die Makroblöcke sind eigene Versorgungsringe gelegt. Dies ist notwendig, da sie außer der Spannungsversorgung des Makroblocks die Funktion des Core-Rings für den betroffenen Bereich übernehmen. Die Standardzellen werden über Power-Rows versorgt, die abwechselnd an Power und Ground angeschlossen sind. Jede Standardzelle besteht zu einem Teil aus diesen Rows. Durch das Spiegeln der Standardzellen in jeder zweiten Row (zu erkennen an der Dreiecksmarkierung in den Standardzellen in Abbildung 4.15) lassen sich die Zellen platzsparend aneinandersetzen. Um die Spannungsversorgung der Rows zu sichern werden vertikal über den Core-Bereich so genannte Power-Stribes gelegt. Hierdurch entsteht eine Art Strom- bzw. Spannung-Versorgungsnetz, das sich über den gesamten Chip erstreckt.

Üblicherweise befindet sich der Core-Ring auf den Metalllagen eins und zwei im Core-Bereich. Als Besonderheit sind hierfür im TRAP alle sechs Lagen verwendet. Darüberhinaus wurden alle I/O-Zellen so angepasst, dass sich zusätzliche Ringe zur Stromversorgung des Core-Bereichs im I/O-Bereich befinden. Damit lässt sich die Breite der Core-Ringe bei gleichem Querschnitt deutlich verringern und mehr Fläche für den Core schaffen. Da es bei der Fertigung von größeren Metallflächen zu Problemen kommen kann und damit die

Ausbeute der funktionsfähigen Chips (Yield) verringert wird, sind beim TRAP größere Abstände von 3 µm zwischen den Stromversorgungsleitungen und „Routing-Blockages“ auf den entsprechenden Metalllagen, so dass keine Signalleitungen dazwischen gelegt werden.

Place & EcoPlace

Da bei TRAP die Makroblöcke schon während des Floorplan platziert werden, geht es bei der automatischen Platzierung ausschließlich um die Standardzellen. Es werden immer die Platzierungsvorgaben und die Vorgaben für das zeitliche Verhalten berücksichtigt. Bei der ersten Platzierung erfolgt eine einfache Abschätzung der Leitungsverzögerungen anhand des Abstandes. Auch das Einfügen von Puffern und der Taktbaum (nachfolgende Kapitel) sind noch nicht vorhanden, aber der Platzierer versucht diese Einflüsse einzubeziehen. Diese erste Platzierung ist für den weiteren Design-Fluss sehr wichtig, da danach nur noch lokal begrenzte Optimierungsschritte vorgenommen werden. Zwar ist es möglich auch eine großflächige Optimierung der Platzierung (EcoPlace) vorzunehmen, dies hat allerdings den Nachteil, dass lokale Optimierungen verloren gehen. Bei TRAP konnte durch detaillierte Platzierungsvorgaben während des Floorplans auf diesen Designschritt verzichtet werden.

Nach der Platzierung ist es zum ersten Mal möglich, sich ein genaues Bild vom zeitlichen Verhalten der Schaltung zu machen. FirstEncounter bietet die Möglichkeit für jeden Design-Schritt, z. B. nach der Platzierung, eine einfache Art der Verdrahtung durchzuführen. Diese Art der Verdrahtung (trialRoute) ist nicht optimal und entspricht nicht allen Designregeln, aber sie ist sehr schnell und erlaubt eine Abschätzung der Leitungskapazitäten bzw. -verzögerungen und der Verdrahtungsdichte. Bei sehr hoher Verdrahtungsdichte werden die entsprechenden Stellen des Designs nicht entflochten, sondern nur für die weiteren Designschritte markiert (Congestion Map). Je nach Ergebnis dieser Abschätzungen werden dann Optimierungsschritte wie z. B. eine Neuplatzierung von Regionen, Logikoptimierungen oder Pufferdimensionierungen vorgenommen.

Im Design-Fluss des TRAPs wird ausschließlich eine endgültige Platzierung (amoebaPlace) durchgeführt. Bei dieser Methode der Platzierung werden gegebenenfalls schon Puffer eingefügt und Ausgangsstufen von Zellen dimensioniert. Nur damit ist es dem Platzierer möglich, kritische Netze global zu entschärfen. Dies kann auch für einzelne Netze sehr wichtig sein, da von diesen Netzen auch größere Teile des Designs anhängig sein können. Zwar sind weitere Optimierungsschritte notwendig, aber sie basieren bei TRAP alle auf dieser Platzierung und es werden nur noch sehr lokal Änderungen zugelassen.

Einfügen von Puffern (IPO) und Logikoptimierung

Es gibt zwei Arten von Netzen, die eine hohe Verzögerung verursachen: sehr lange Leitungen und Netze mit einem hohen Verzweigungsgrad (Fanout). Je höher die Last ist, die eine Ausgangsstufe einer Zelle treiben muss, desto länger dauert es, das Potential dieser Last zu ändern. Prinzipiell ist es möglich, einen großen Fanout schon während der Synthese zu

erkennen und durch das Benutzen großer Ausgangsstufen oder das Einfügen von Puffern die Verzögerungszeiten zu reduzieren.

Vor allem in DSM[5]-Prozessen spielt die Verdrahtung der Leitungen eine immer wichtigere Rolle. Während nach Sylvester und Keutzer [32][33] die durchschnittliche Leitungslänge einen festen Anteil der Seitenlänge eines Layouts darstellt, nimmt die Anzahl der Transistoren ständig zu. Hierdurch wird nicht nur die Komplexität der Verdrahtung erhöht; durch die kleineren Querschnitte und die geringeren Abstände der Leiterbahnen kommt es außerdem zu höheren Leitungswiderständen und größeren Kapazitäten zwischen den Leitungen.

Genaue Aussagen über die Last, die ein Netz liefert, lässt sich demnach erst während des Layouts machen. FirstEncounter bietet nicht nur die Möglichkeit, Zellen durch funktional gleiche Zellen mit einer anderen Ausgangsstufe zu ersetzen oder Puffer in ein Netz einzufügen, sondern auch Logik lokal zu verändern (In Place Optimization, IPO). In erster Linie werden hierbei Standardzellen, die komplexere Logik erfüllen (z. B. ein AND-OR Gatter) durch einzelne Gatter ersetzt, die örtlich optimal verteilt werden. Gleiches geschieht natürlich auch umgekehrt, wenn erkannt wird, dass Zellen nahe beieinander liegen oder womöglich überdimensioniert sind.

Wegen dieser Funktionalität von FirstEncounter wird auf Berücksichtigung von Lasten und Fanouts während der Synthese von TRAP verzichtet. Außerdem führt bei der benutzten Version von FirstEncounter das Entfernen von Puffern vereinzelt zu Problemen. Durch die zum Teil sehr komplexe Methodik, Logik zu ersetzten, ist es auch nicht notwendig nach den ersten Abschätzungen über das Zeitverhalten in den Synthese-Schritt zurückzukehren (Abbildung 4.11), um dort mit den aus dem Layout gewonnenen Informationen die Logik zu optimieren.

Bei TRAP werden die Möglichkeiten lokaler Optimierungen umfangreich genutzt, da auf regionale oder globale Optimierungsschritte, wie im vorhergehenden Kapitel beschrieben, verzichtet werden kann. Zunächst werden z. B. in einem separaten Schritt die langsamen asynchronen Netze auf ihre Übergangszeit (transition time) hin optimiert. Verletzungen der Setup- und Hold-Zeiten in den Datenpfaden, wie sie nach der ersten Platzierung noch vorhanden sind, werden gleich durch mehrere Iterationen lokaler Optimierungen beseitigt. Gleiches geschieht auch nach der Generierung der Taktbäume (nachfolgendes Kapitel), die im bisherigen Design-Fluss noch mit einem idealen zeitlichen Verhalten angenommen wurde. Durch die Implementierung der Taktbäume kann es deshalb wiederum zu Verletzungen in den Datenpfaden kommen.

Taktbaum

Bei der Verteilung eines Takts verhält es sich ähnlich wie bei einem Signal mit sehr großem Fanout. Die treibende Zelle, also die Taktwurzel (clock root), muss gegebenenfalls tausende von Zellen treiben, die über den gesamten Chip verteilt liegen können. Diese verteilten

[5] Deep Sub Micron, Prozesse mit einer Gatelänge kleiner 0,25 µm

Zellen sind im Allgemeinen Flipflops, die Blätter des Taktbaums (leaf cells). Wie in Kapitel 4.2.5 beschrieben, bestehen jedoch zusätzliche Anforderungen an ein Taktsignal. Der Taktversatz innerhalb eines Taktbaums, aber auch der Versatz zwischen verschiedenen Bäumen, trägt zur Setup- und Hold-Zeit der Datenpfade bei.

Der 120-MHz-Takt innerhalb des TRAPs treibt 12 092 Blätter, wovon die meisten Register sind. Die Verteilung erfolgt mittels 485 Puffern, die über 16 Level einen Baum bilden. Auf diese Weise wird ein Taktversatz von etwa 500 ps erreicht, wobei die Verzögerung durch den Taktbaum sich im Bereich von 3 ns bewegt. Die Generierung solcher Taktbäume erfolgt automatisch. Durch eine Datei wird dem Takt-Synthese-Programm mitgeteilt, welches der treibende Takt-Pin ist und welche Anforderungen an den Takt gestellt werden. Diese Funktionalität wird im TRAP außerdem für verschiedene andere Signale verwandt. So wird damit z. B. der Versatz zusammengehöriger Datensignale der Eingangsports der Netzwerkschnittstelle gering gehalten. Um ein optimales Ergebnis zu erzielen, werden die Taktbäume vor den übrigen Zellen verdrahtet.

Verdrahten

Beim finalen Verdrahten (globalDetailRoute) werden alle Pins miteinander verbunden. Dabei werden im Gegensatz zum „trialRoute“ alle Designregeln berücksichtigt: Abstände der Leiterbahnen, Kurzschlüsse, Antennen und Zeitvorgaben. Die Rechenzeit für diesen letzten Designschritt kann durchaus mehrere Stunden betragen. Erst danach stellt sich heraus, ob sich die Optimierungen der letzten Schritte ausgezahlt haben. Nach dem Verdrahten werden nochmals eine detaillierte Extraktion und alle Arten der Timing-Analysen innerhalb von FirstEncounter durchgeführt.

Ausgaben

Ähnlich der I/O-Filler-Zellen (Powerplan) wird direkt vor der Ausgabe des Layouts der Core-Bereich mit Core-Filler-Zellen gefüllt (Abbildung 4.15). Dadurch werden die Power-Rows und die Wannen der P-Kanal Transistoren verbunden. Es gibt eine Design-Regel, die festlegt, wieviel einer Lage mindestens mit Metall gefüllt sein muss. Dieser Schritt erfolgt entweder direkt vor dem Herausschreiben des fertigen Layouts oder wird von dem Anbieter des Prozesses vor der Fertigung ausgeführt.

Während des Erzeugens der Layout-Datei (GDSII) werden die Abstracts der Makroblöcke, Standardzellen und I/O-Zellen durch die Layouts ersetzt. Diese Layouts liegen ebenfalls im GDSII Format vor. Da sich durch das Einfügen von Puffern und der Logikoptimierung die Netzliste ändert, wird eine finale Gatternetzliste erzeugt, die dem endgültigen Layout entspricht. Durch das Einfügen des Füllmetalls können sich durch die kapazitiven Kopplungen leichte Änderungen im Zeitverhalten der Schaltung ergeben. Vor dem Herausschreiben der Extraktions- und Timing-Dateien wird nochmals eine genaue 3D-Extraktion und Timing-Analyse durchgeführt. Die beiden wichtigsten Dateien neben dem Layout selbst und der

Netzliste sind das „Standard Parasitics Exchange File“ (SPEF), das die Extraktionsergebnisse enthält und als Eingang für die Timing-Analyse mit Primetime dient, und das „Standard Delay File“ (SDF), mit dem die zurückannotierte Simulation erfolgt.

TRAP optimierter Ablauf

Der speziell für den TRAP entwickelte Ablauf des Layouts basiert allein auf Make-Skripten[6] und wird inzwischen auch für verschiedene andere Projekte weiterverwendet. Die folgende Auflistung gibt einen Überblick über die einzelnen Entwurfschritte des Layouts für TRAP, so wie sie im Make-Skript abgearbeitet werden:

floorplan: Platzierung aller I/O-Zellen und Makroblöcke, exakte Vorgaben von Regionen zur Platzierung, vollständiger Powerplan inklusive des Anschlusses aller Makroblöcke.

placeECO: Wiederherstellen der Platzierung, grobe Abschätzung des Zeitverhaltens der manuellen Platzierung von Regionen

place: Platzierung der Standardzellen, optimiert für das Zeitverhalten

IPOtran: Lokale Optimierungen, um langsame Übergangzeiten zu vermeiden. Diese entstehen bei einer großen Verzweigungsanzahl, wie z. B. „Reset“ oder sehr langen Pfaden.

IPOpreCTS: Lokale Optimierungen der Setup-Zeiten. Diese Methode der Optimierung wird mehrmals nacheinander ausgeführt, um optimale Ergebnisse zu erreichen.

CTS: Synthese der Taktbäume. Da sich die Platzierung durch das Hinzufügen von Taktpuffern lokal ändern kann, wird vor und nach dieser Synthese eine Analyse des Zeitverhaltens gemacht.

IPOpostCTS: Nochmalige Optimierungen der Setup-Zeiten nach dem Einfügen der Taktbäume. Auch hier wird dieser Optimierungsschritt mehrfach nacheinander ausgeführt.

RouteCLK: Um ein bestmögliches Ergebnis bezüglich der Taktversätze zu erreichen, werden die Taktbäume vor dem übrigen Design verdrahtet, wenn noch alle Routing-Ressourcen vorhanden sind.

RouteFirst: Erstes globales Routing, bei dem Fehler der Setup-Zeiten korrigiert werden. In einem zweiten Routing-Schritt werden lokale Änderungen der Treiberstufen von Standardzellen vorgenommen. Es werden jedoch nur Änderungen gemacht, bei denen die Zellen die gleichen Maße beibehalten, sich die Platzierung also nicht ändert.

RouteFinal: Mehrere globale Routings, bei denen Geometriefehler korrigiert und Rauschen berücksichtigt werden. Abschließend wird mehrfach ein Routing ausgeführt, bis keinerlei Verletzungen der Geometrie oder des Zeitverhaltens mehr vorhanden sind.

[6] http://www.gnu.org/software/make/

GDSout: Der Core-Bereich wird mit Füllerzellen aufgefüllt und leere Bereiche werden mit Metallflächen gefüllt, die mit Masse verbunden werden. Zusammen mit den GDS2-Daten der Makroblöcke wird eine GDS2-Datei, die den gesamten Chip enthält, erstellt.

Timing: Zur Analyse des zeitlichen Verhaltens in separaten Programmen, wie z. B. Primetime, und zur Simulation der zurückannotierten Gatternetzliste werden mit verschiedenen Varianten der Extraktion Dateien erzeugt, die die Kapazitäten oder Verzögerungen der Leitungen beinhalten.

4.2.7 Layout Verifikationen

Das fertige Layout aus FirstEncounter wird als GDSII Datei in das Custom IC Design Programm (icfb) eingelesen und um die Pads und die Scribeline ergänzt. Bevor dieses Design zur Fertigung an den Anbieter übergeben wird, werden noch zwei weitere Verifikationsschritte am Layout durchgeführt. Es wird getestet, ob das Design alle Layout-Regeln einhält und ob das Layout der erzeugten Netzliste entspricht.

Beim „Design Rule Check“ (DRC) wird das GDSII Layout daraufhin untersucht, ob alle Design-Regeln (Abstände von Leiterbahnen usw.), die vom Prozessanbieter vorgegeben werden, erfüllt werden. Hierzu wird vom Anbieter eine Regel-Datei mitgeliefert, mit der das DRC Programm (Calibre) das Design verifizieren kann. Werden nur sehr wenige Fehler erkannt, besteht eventuell die Möglichkeit, diese von Hand zu reparieren.

Bei der Timing-Analyse mit Extraktionsdaten und der Simulation mit zurückannotierter Netzliste wird davon ausgegangen, dass die Gatternetzliste aus FirstEncounter dem Layout entspricht. Um dies sicherzustellen wird mithilfe des „Layout Versus Schematic“ (LVS) – ebenfalls Calibre – das Layout mit der Netzliste verglichen. Bei einem LVS auf Makroblock-Ebene, wie es bei einem Standardzellen-Design ausreicht, wird eine Spice-Netzliste erzeugt, die direkt mit der Gatternetzliste nach Umwandlung in eine Spice-Netzliste verglichen werden kann. Bei Full-Custom-Design erfolgt die Extraktion bis auf die Ebene von Transistoren, Widerständen und Kapazitäten.

FirstEncounter verändert durch Logikoptimierungen, Einfügen von Puffern oder der Taktbäume, die Netzliste. Eine zusätzliche Möglichkeit der Verifikation ist es, die Gatternetzliste aus dem Layout mit der aus der Synthese logisch zu vergleichen. Dabei wird überprüft, ob zwischen zwei beliebigen Speichern dieselbe logische Operation durchgeführt wird. Hierzu wird bei TRAP das Programm Formality von Synopsys verwendet. Beschränkt man die Optimierungen während des Layouts auf das Ändern der Treiberleistung einer Zelle oder das Hinzufügen von Puffern oder Invertern, funktioniert diese Art der Verifikation einwandfrei. Es stellt sich allerdings heraus, dass die Logikoptimierungen von FirstEncounter dazu führen, dass Formality mit dieser Komplexität nicht mehr zurechtkommt. Für TRAP erfolgt daher die Verifikation durch umfangreiche Simulationen mit der zurückannotierten Gatternetzliste.

5 Auslesenetzwerk

Jedes der signalverarbeitenden 65 664 Multi-Chip-Module, die auf dem gesamten Detektor über eine Oberfläche von mehr als $700\,\mathrm{m}^2$ verteilt sind, produziert Daten, die außerhalb des Detektors weiterverarbeitet oder gespeichert werden. Auch hierbei wird zwischen der zeitkritischen Auslese für die Funktion des Detektors als Trigger (Triggerauslese) und der Auslese der ADC-Werte (Rohdatenauslese) unterschieden. Diese beiden Modi haben vollkommen verschiedene Anforderungen an die Auslese. Die besondere Herausforderung besteht darin, mit angemessenem Aufwand die weit verstreuten Daten für die Triggerauslese schnellstmöglich (innerhalb etwa 1 µs, Abbildung 2.6) an die globale Tracking-Einheit (GTU) weiterzugeben. Im Folgenden werden die Anforderungen, die Idee zu einer Lösung und die Implementierung vorgestellt. Aus der Architektur der Auslese ergeben sich außerdem die Anforderungen an die Netzwerkschnittstelle, deren Implementierung im Kapitel 6 beschrieben ist.

5.1 Anforderungen

In Kapitel 3.5.3 ist als ein wichtiges Simulationsergebnis, das sich direkt auf die Auslese auswirkt, dargestellt, wie viele Spursegmente pro Kammer maximal erwartet werden. Neben der enormen Anforderung an die Latenz und Geschwindigkeit der Auslese sind eine ganze Reihe von Faktoren zu berücksichtigen, die aufeinander abgestimmt und optimiert werden müssen. In den nachfolgenden Unterkapiteln sind die wichtigesten Parameter zusammengefasst.

5.1.1 Datenvolumen

In Kapitel 3.5.3 wird der Einfluss der Simulation auf die Auslese beschrieben. Ein wichtiges Ergebnis ist hierbei die maximal erwartete Anzahl von 40 Spursegmenten in jeder der 540 Kammern. Hierbei wird angenommen, dass die Spursegmente gleich verteilt sind.

Ein weiteres Ergebnis ist die erforderliche Genauigkeit in der Beschreibung eines Spursegmentes (Position, Ablenkung, ...). Wie in Kapitel 3.5.4 erläutert, kann ein Segment als 32 Bit breites Wort mit adäquater Genauigkeit dargestellt werden.

Aus diesen beiden Größen lässt sich das für die Triggerauslese maximal erwartete Datenvolumen für eine Kammer (160 Byte) bzw. den gesamten Detektor (84 KByte) bestimmen.

5.1.2 Latenz

Durch die gegebene Geometrie und die Methode der globalen Spurrekonstruktion ist es möglich, dass die GTU mit einer Teilmenge der Daten zu arbeiten beginnen kann. Dies setzt voraus, dass die Spursegmente einer Kammer in einer definierten Reihenfolge an die GTU weitergegeben werden. Damit die GTU möglichst früh anfangen kann zu arbeiten, ist es das Ziel, die Latenz möglichst gering zu halten.

Die Latenz setzt sich zusammen aus der Auslese auf dem Detektor, der Übertragung der Daten vom Detektor zur GTU und der empfangsseitigen Logik. Die GTU steht möglichst nahe am Experiment unterhalb des Dipol-Magneten (Abbildung 2.2) und es kann mit einer Kabellänge von ungefähr 40 m gerechnet werden. In Tabelle 5.1 ist die Latenz für die Übertragung und die Empfangslogik abgeschätzt. Die ersten drei Zeilen repräsentieren die Sendelogik, die hier nicht als Bestandteil des Auslesebaums gerechnet wird. Hierzu wurde ein Test-Design entworfen und die resultierenden Zeiten übernommen (siehe auch Kapitel 5.2.2). Zur Bestimmung der empfangsseitigen Latenz wurde das RocketIO von Xilinx [42] herangezogen. Die Receiver sind Bestandteil der Programmierbaren Logik (FPGAs), aus der die GTU aufgebaut wird.

CPLD	33,33 ns
Transmitter	15,83 ns
Wortlänge (16 Bit)	16,67 ns
Glasfaser (40 m)	200,00 ns
RocketIO Receiver	200,00 ns
Summe	∼466 ns

Tabelle 5.1: Übertragunslatenz der Triggerauslese vom Auslesenetzwerk zur globalen Tracking-Einheit. Zur Ermittlung der Wandlungszeiten wurde das RocketIO von Xilinx [42] mit folgenden Parametern zugrunde gelegt: 2,4 GBit/s, 8B/10B-Kodierung, 2-Byte-Schnittstelle bei 120 MHz, Low-Latency-Mode. Die Latenz für die elektrisch-optische Wandlung [44] kann vernachlässigt werden.

Werden diese Zahlen zugrundegelegt, beträgt die Latenz ohne die des Auslesebaums 466 ns. Unter Betrachtung des in Abbildung 2.6 dargestellten Zeitdiagramms für den TRD ergibt sich als adäquater Wert der zu erreichenden zusätzlichen Latenz eine Zeit von etwa 150–200 ns.

5.1.3 Durchsatz

Die Übertragungsgeschwindigkeit muss der Anforderung genügen, alle Spursegmente des Detektors, bei erwarteten 40 Segmenten pro Kammer, inklusive der Latenz innerhalb von etwa 1 µs an die GTU weiterzugeben. Dies entspricht nach Abzug der Latenz einer Übertragungszeit von etwa 400 ns bzw. Datenübertragungsrate pro Kammer von 3,2 GBit/s.

Die Datenübertragungsrate innerhalb des Auslesenetzwerkes hängt im Wesentlichen von der Übertragungsfrequenz und der Busbreite ab, deren Zusammenspiel im Folgenden beschrieben wird. Zum Erreichen einer hohen effektiven Bandbreite ist es außerdem wichtig

einen möglichst kontinuierlichen Datenstrom zu haben, das bedeutet ohne Pausen oder Leerwörter, mit einer schnellen Datensynchronisation und einem möglichst einfachen Protokoll.

Übertragungsfrequenz

Der TRAP arbeitet intern mit einem Takt von 120 MHz. Von der Verwendung einer PLL zum Hochtakten des internen Takts wurde zunächst abgesehen. Die Befürchtung war, dies würde schwer abschätzbare Störsignale bedeuten, die die ADCs und die Vorverstärker beeinflussen könnten. Wie sich mit den ersten Prototypen von TRAP herausstellte, hatte diese Art von Störung jedoch keinen nennenswerten Einfluss, und in der dritten Version des TRAP Chips wurde sogar eine 240-MHz-PLL für die ADCs eingesetzt (Kapitel 4.1.1). Zur Minimierung der Stromaufnahme ist es außerdem vorgesehen, geschaltete Takte für die verschiedenen Module, also auch die Netzwerkschnittstelle, zu nutzen, da diese immer nur für eine kurze Zeit aktiv sind. Bei der Benutzung einer PLL müsste zumindest diese durchlaufen. Eine 120 MHz zu 480 MHz PLL [45] für den verwendeten Prozess hat alleine eine Leistungsaufnahme von etwa 4 mW. Das also bedeutet, die maximale Datenrate der Schnittstelle nach außen beträgt 240 MWörter/s bei Nutzung der doppelter Datenrate (DDR).

Busbreite

Wie im nachfolgenden Kapitel 5.2 erläutert wird, erfolgt die gesamte Auslese durch die TRAP Chips selbst. Die Breite eines Busses im Zusammenhang mit der Anzahl der Ports ist limitiert durch die Größe bzw. den Umfang des Chips. Die Anzahl der Ports wiederum wird durch die gewählte Auslesestruktur bestimmt.

5.1.4 Fehlertoleranz

Das Auslesenetzwerk ist ein Teil des Detektors. Das Experiment wird über viele Jahre laufen und in dieser Zeit wird es praktisch unmöglich sein, Defekte durch einen manuellen Eingriff zu beheben. Dies bedeutet, das Auslesenetzwerk muss über eine selbständige Fehlerdiagnose verfügen und es muss möglich sein, erkannte Fehler zu beheben oder zumindest defekte Baugruppen auszugrenzen. Der Aufwand für diese Form der Fehlertoleranz ist mit den übrigen Anforderungen abzuwägen.

Durch die hohe Dichte der Bauteile auf dem Detektor ist während der Verarbeitungszeit der Prozessoren und der Auslese mit einem hohen Grad an elektromagnetischen Störungen zu rechnen. Um Fehler durch Einkopplungen auf die Datenleitungen zu vermeiden und gleichzeitig die Möglichkeit zu haben über längere Distanzen ($\sim$1,5 m) eine Übertragungsfrequenz von 240 MHz bzw. 240 MWörter/s zu erreichen ist als Übertragungsebene der LVDS-Standard bzw. eine angepasste Variante davon vorgesehen. Dies hat allerdings

den Nachteil, dass nur halb so viele Datenleitungen zur Verfügung stehen und die doppelte Anzahl an Leiterbahnen gegenüber einpoligen Leitungen geroutet werden muss.

5.1.5 Mechanik

Alle Kammern sind mechanisch voneinander getrennt; es gibt keine Möglichkeit Datensignale zwischen Kammern auszutauschen. Eine einzelne Kammer hat eine Oberfläche von etwa $1\,\mathrm{m}^2$. Die genauen Abmaße hängen von der Position der Kammer ab. Platinen dieser Größe wären nicht handhabbar und teuer in der Produktion. Außerdem würde ein einzelner Fehler auf der Platine, z. B. bei der Bestückung, bedeuten, dass die Platine unbrauchbar ist. Aus diesen Gründen und um die Platinen möglichst modular zu halten sind mehrere Platinen pro Kammer notwendig. Dies bringt mit sich, dass Signale/Daten zwischen den Platinen innerhalb der Kammer ausgetauscht werden müssen.

5.1.6 Modularität

Trotz der unterschiedlichen Abmessungen der Kammern ist es das Ziel, möglichst wenige Arten von Ausleseplatinen zu benötigen. Allerdings würde eine zu hohe Granularität bedeuten, dass sehr viele Signale zwischen den Platinen ausgetauscht werden müssen. Größere Platinen hingegen haben den Nachteil, dass sie schwer zu handhaben sind und ein modularer Aufbau nur bedingt möglich ist. Die optimale Größe einer Platine steht außerdem im direkten Zusammenhang mit der gewählten Auslesestruktur und daher sind diese aufeinander abzustimmen.

5.1.7 Stromverbrauch

Für die Architektur der Auslese muss die Möglichkeit berücksichtigt werden sowohl die Netzwerkschnittstelle als auch die LVDS-Zellen ausschalten zu können ohne Inkonsistenzen im Netzwerk zu erhalten. Wie in Kapitel 4.1 beschrieben, verwaltet die Master-State-Machine die Taktsignale für die einzelnen Module des TRAP. Während der relativ kurzen Triggerauslese kann das Netzwerk vollständig aktiv sein. Für die deutlich längere Rohdatenauslese sollten zumindest die LVDS-Zellen der Datensignale für inaktive MCMs abgeschaltet werden können. Für die Stromaufnahme einer Ausleseplatine ist weniger der maximale als vielmehr der durchschnittliche Strom interessant. Dies rührt daher, dass die Stromverteilung des Detektors über Kapazitäten, also Ladungspuffer, erfolgt. Diese sind auf den Ausleseplatinen hinter den Spannungsregulatoren platziert und können kurzzeitig eine höhere Leistung als die Regulatoren abgeben.

Die LVDS-Zellen sind bei Prof. Dr. R. Tielert am Lehrstuhl für Mikroelektronik an der Universität Kaiserslautern eigens für TRAP entwickelt worden. Ein LVDS-Ausgang hat

eine Stromaufnahme von nur etwa 4 mA was im Wesentlichen dem Ausgangsstrom entspricht. Damit liegt die treibende Stromstärke zwar über den nominalen 3,5 mA des LVDS-Standards, aber noch deutlich innerhalb der Spezifikation. So konnte ein höherer Störabstand erreicht werden und TRAP kann weiterhin auch mit kommerziellen Bauteilen kommunizieren, die den LVDS-Standard unterstützen. Die Verlustleistung eines Empfängers ist vernachlässigbar. Die stromtreibenden Sender sind auch abschaltbar. In diesem Zustand haben sie eine minimale Stromaufnahme. Dabei sind sie so entworfen, dass während des Abschaltvorgangs keine Störimpulse erzeugt werden. Die Empfänger sind darauf ausgelegt, dass sie, wenn der Eingang nicht getrieben wird, eine logische Eins erkennen. Die Sender sind innerhalb eines Taktzyklus nach dem Anschalten wieder betriebsbereit. Durch diese Funktionalität können die Zellen gezielt nur dann angeschaltet werden, wenn die Auslese des entsprechenden MCMs aktiv ist. Als Besonderheit ist außerdem in allen LVDS-Zellen die Boundary-Scan Logik (JTAG) integriert und sie enthalten die in Kapitel 4.2.6 beschriebenen Ringe zur Stromversorgung des Core-Bereichs.

5.2 Struktur des Auslesebaums

Durch die mechanische Abschottung der Module ist es für die Struktur der Auslese ausreichend ein einzelnes Modul zu betrachten. In Kapitel 3.5.2 ist beschrieben, dass auch für das Tracking, sowohl in ϕ- als auch in z-Richtung, benachbarte Module nicht berücksichtigt werden müssen. Ausschließlich übereinander liegende Module, also die Module eines Stacks, sind von derselben (interessanten) Spur betroffen. Es ist die Aufgabe der GTU die Spursegmente einer Spur zusammenzuführen und den genauen Transversalimpuls zu bestimmen. Da zusammengehörige Spursegmente der unterschiedlichen Lagen in ähnlicher z-Position sein werden, kann durch eine geeignete Reihenfolge der Auslese das globale Tracking beschleunigt werden (Kapitel 3.4.1). Zum einen kann die GTU auf einer Teilmenge der Daten anfangen zu arbeiten, also noch während Daten an die GTU übertragen werden, zum anderen sind die Daten schon vorsortiert.

Der TRAP als einer der beiden Chips eines MCMs beinhaltet die gesamte Funktionalität von der analog/digital Wandlung bis hin zur Berechnung von Spursegmenten mittels eines Prozessors. Der Chip hat damit eine Fläche von etwa 35 mm^2. Der ADC Block (Kapitel 4.1.1) mit seinen analogen Eingängen belegt eine Seite des TRAPs. Die übrigen drei Ränder werden bisher von wenigen I/O-Zellen zur Konfiguration und den Zellen zur Stromversorgung belegt. Alles spricht dafür, das Auslesenetzwerk als eine weitere Funktionalität in den TRAP zu integrieren. Die notwendige Logik wird relativ wenig Platz in Anspruch nehmen und es werden viele der bisher ungenutzten I/O Bereiche benötigt. Allein dies würde einen separaten Chip unrentabel machen, da er durch eine hohe Anzahl an I/O Zellen sehr groß würde und dabei nur wenig Logik enthielte. Durch ein Verschmelzen der Netzwerkschnittstelle mit dem Prozessor bzw. den vier CPUs kann außerdem direkt auf das Netzwerk zugegriffen und damit flexibel reagiert werden.

Die Datenübertragung von einem Modul zur GTU findet über optische Glasfasern statt. Nach dem heutigen Stand der Technik haben verfügbare Transmitter üblicherweise Über-

tragungsbandbreiten von 1,25 oder 2,5 oder 10 GBit/s. In Anbetracht der hohen Empfindlichkeit und des Umfeldes der Anwendung erscheint die 10 GBit/s Variante zu schwer handhabbar, da sehr hohe Anforderungen an die physikalischen Eigenschaften, das Routing und die Abschirmung der Platine gestellt werden. Betreibt man einen 2,5-GBit/s-Transmitter mit nur 2,4 GBit/s, lassen sich bei der Arbeitsfrequenz von 120 MHz des TRAP genau 16 Bit pro Taktperiode bzw. 8 Bit pro Halbtakt (DDR) übertragen. Dies rührt daher, dass die Transmitter und Receiver eine 8B/10B-Kodierung benutzen, die notwendig ist, um auf der Empfangsseite den Takt zu regenerieren und Wörter aus dem Datenstrom zu extrahieren.

Werden die maximal erwarteten 40 Spursegmente, dargestellt durch jeweils 4 Byte, am Stück über einen 2,4-GBit/s-Transmitter übertragen, entspricht dies einer Übertragungszeit von $40 \cdot 2 \cdot 8{,}33\,\text{ns} = 667\,\text{ns}$. Um die erstrebten 400 ns für die Übertragung der Daten während der Triggerauslese zu erreichen, sind zwei Transmitter pro Modul notwendig, berücksichtigt man einen möglichen Overhead von bis zu 20 %.

Die gewünschte Auslesereihenfolge innerhalb eines Moduls, damit die GTU optimal arbeiten kann, ist in z-Richtung. Wird ein Transmitter für die Auslese eines halben Moduls eingesetzt, so kann dies berücksichtigt werden, indem das Modul in zwei für die Auslese separate Abschnitte längs der z-Achse geteilt wird. Damit kann bei voller Nutzung der Bandbreite der Transmitter die Auslesereihenfolge gewahrt werden. Dieser Entwurfschritt ist unabhängig von der weiteren Struktur. In der Abbildung 5.1 lassen sich die beiden Bereiche mit den dazugehörigen Transmittern erkennen.

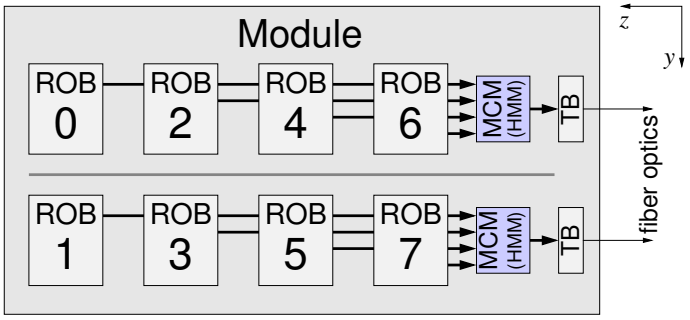

Abbildung 5.1: Zwei optische Transmitter übertragen die Daten eines Modul an die GTU. Unter Berücksichtigung der Auslesereihenfolge ist jeweils einer für eine Hälfte des Modul zuständig. Ein MCM (HMM) sammelt die Daten von vier Ausleseplatinen (ROBs) und leitet sie an die Transmitterplatine (TB) weiter.

Die Breite der Busse bzw. Schnittstellen beträgt 8 Bit (s. o.). Um eine hohe Fehlertoleranz, sowohl für die Fertigung als auch für einen längeren Betrieb, zu erreichen, sind neben den Datenleitungen noch weitere Leitungen vorzusehen, die im Fehlerfall eine Datenleitung ersetzen können. Die Anzahl dieser zusätzlichen Leitungen sollte möglichst gering sein. In erster Linie wird mit statischen Bitfehlern durch die Fertigung (z. B. kalte Lötstellen, fehlerhafte Bondverbindungen) und alterungsbedingten Erscheinungen (z. B. Bruch einer Lötstelle) gerechnet. Dies macht es möglich auf eine dynamische Fehlerkorrektur durch z. B. eine lineare Kodierung zu verzichten. Allein um Einbitfehler korrigieren und Zweibitfehler

erkennen zu können wären hierzu schon vier weitere Leitungen nötig. Anstelle einer Kodierung werden zwei zusätzliche Leitungen vorgesehen, die jeweils eine beliebige Datenleitung ersetzen können. Eine dieser Leitungen kann außerden – wenn sie nicht als Datenleitung fungiert – als Paritätsbit dienen, um Fehler in der Datenübertragung zu erkennen.

Es kann nicht sichergestellt werden, dass der Takt den einzelnen MCMs synchron zueinander ist. Dies macht es notwendig zusammen mit den Daten ein Taktsignal zum jeweils empfangenden MCM zu senden. Dieses Signal wird ausschließlich dazu genutzt die Daten empfangsseitig in ein Register zu übernehmen um sie dann zu dem internen Takt zu synchronisieren. Damit sind bisher 11 Leitungen (8 Daten + 2 Fehlertoleranz + 1 Strobe) für einen Bus notwendig.

Die besondere Herausforderung an die Triggerauslese ist die geringe Latenz und die hohe Übertragungsrate. Unabhängig von der Struktur der Auslese bietet es sich hierfür an, die Datenübertragung ohne ein bidirektionales Protokoll zu implementieren. Dies wird möglich durch einen statisch definierten Datenfluss und das Wissen um die Menge der Daten. Die Latenz ergibt sich wie nachfolgend beschrieben in erster Linie aus der Struktur des Netzwerkes. Um einen maximalen Durchsatz, also einen Datenstrom mit möglichst wenigen Unterbrechungen zu erhalten, schicken alle MCMs gleichzeitig ihre Daten an den nachfolgenden MCM und dieser speichert die Daten zwischen. Das bedeutet, das zusätzliche MCM vor dem Transmitter muss die Fähigkeit besitzen, die Daten seiner zugehörigen halben Kammer zu speichern. Bei den erwarteten maximalen 40 Spursegmenten pro Kammer ist dies allerdings kein Problem. Bis auf den fehlenden Vorverstärkerchip (PASA), der für diese Funktionalität nicht benötigt wird, entspricht das zusätzliche MCM (Half Module Merger, HMM) allen auch sonst verwendeten MCMs. Da für die Rohdatenauslese nicht diese kritischen Zeitanforderungen bestehen und es sich um viel mehr Daten handelt, ist die beschriebene Funktionsweise hierfür nicht angebracht. Zur Implementierung dieser Auslese wird eine weitere Leitung als Kontrollsignal (Handshake) entgegen der Richtung der Daten vorgesehen. Insgesamt ist ein Bus zwischen zwei MCMs also 12 Bit breit. In Kapitel 5.2.3 wird anhand eines vereinfachten Beispieles der Datenfluss für die beiden Auslesearten veranschaulicht.

Um die Daten mit möglichst geringer Latenz der über ein halbes Modul verteilten MCMs zu sammeln um diese an den Transmitter weiterzugeben, bietet sich eine Baumstruktur an. Die Breite und die daraus resultierende Tiefe des Baums ist ein Optimierungsschritt, der von mehreren Parametern abhängt. Die ungefähre Anzahl der Schnittstellen eines TRAP und damit die Breite des Baums ergibt sich aus der Breite eines Busses und der zur Verfügung stehenden I/O-Zellen. Unter Berücksichtigung der verschiedenen Modularten mit 12 und 16 Rows, der Modularisierbarkeit, der praktikablen Größe einer Ausleseplatine und der Auslesereihenfolge, ist eine Breite des Auslesebaums von vier gewählt worden. Dies bedeutet, die Netzwerkschnittstelle eines TRAP hat vier Eingangsports und einen Ausgangsport. Im Folgenden wird diese Struktur näher beschrieben.

5.2.1 Modul Struktur

Es gibt zwei Arten von Modulen mit entweder 12 oder 16 Rows in z-Richtung. Die kleineren sind mit 6, die größeren mit 8 Ausleseplatinen (ROBs) bestückt. In Abbildung 5.1 ist die größere Variante dargestellt.

Die Daten eines Moduls werden über zwei optische Links mit jeweils 2,4 Gbit/s zur GTU übertragen. Die beiden Auslesebäume, die in diesen Links enden, sind vollkommen unabhängig voneinander.

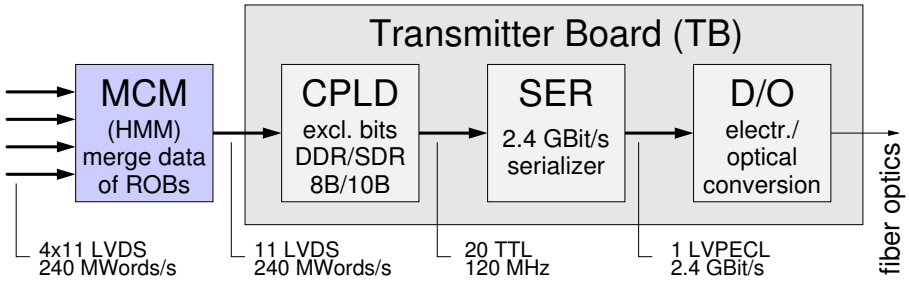

Abbildung 5.2: Ein MCM (HMM) vereint die drei/vier Datenströme, ein programmierbarer Logikbaustein (CPLD) bereitet die Daten auf und gibt sie an einen Serialisierer weiter. Der optische Transmitter nimmt danach die elektrische/optische Wandlung vor.

Ein zusätzliches MCM (HMM) sammelt die Daten von vier ROBs und gibt diese an eine Transmitterplatine (Abbildung 5.2) weiter. Ein programmierbaren Logikbaustein (CPLD) maskiert die beiden redundanten Datenbits aus, resynchronisiert die Daten und führt eine 8B/10B-Kodierung durch. Der Serialisierer (SER) erwartet einen 20 Bit breiten Datenstrom mit einem 120-MHz-Takt und einem Gültigkeitssignal. Die hochfrequenten serialisierten Daten werden dann als LVPECL Signal an den Transmitter gegeben, in dem die elektrische/optische Wandlung stattfindet.

In der Implementierung ist das MCM (HMM), das die Daten der ROBs sammelt, Bestandteil von z. B. ROB 4 (Abbildung 5.1). Die Daten von ROB 2 und ROB 6 werden an ROB 4 gegeben, die von ROB 0 werden durch ROB 2 hindurch weitergeleitet. Die Transmitterplatine (TB) ist als Aufsteckplatine für ROB 4 implementiert und erhält die Daten von dem zusätzlichen MCM (HMM). So gibt es unterschiedliche Varianten des ROB, wie z. B. solche, die Daten durchschleifen oder solche, die ein weiteres MCM (HMM) und die TB beherbergen.

5.2.2 Ausleseplatine

Auch auf der Ausleseplatine (ROB) erfolgt der Datenfluss über eine Baumstruktur, die einer optimierten Auslesereihenfolge angepasst ist.

Das MCM 16 (blau) in Abbildung 5.3, das allein zum Bilden des Auslesebaums genutzt wird, sammelt die Daten der vier rot markierten MCMs. Diese fungieren sowohl als Datenquelle

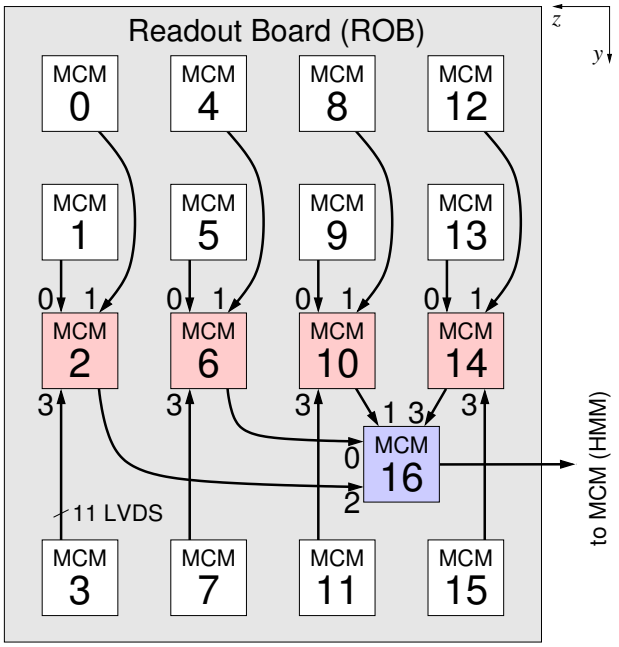

Abbildung 5.3: Der Auslesebaum wird durch die MCMs aufgebaut. Die weiß dargestellten MCMs sind ausschließlich Datenquelle, die blauen akquirieren keine Daten, sondern bilden ausschließlich den Auslesebaum, die roten vereinigen beide Funktionalitäten.

als auch zum Sammeln der Daten der drei benachbarten MCMs in derselben Padrow. Diese weiß markierten MCM sind allein Datenquelle.

Abbildung 5.3 zeigt die einfachste Variante eines ROB ohne zusätzliche Logik. Darüber hinaus gibt es ROBs, die noch ein weiteres MCM besitzen und mit einer Transmitterplatine bestückt werden. Hierbei werden die Daten von MCM 16 und die der drei benachbarten ROBs an das zusätzliche MCM (HMM) gesendet und danach an den Transmitter übergeben. Ein anderer Typ von ROBs wird mit einer Kontrollkarte (DCS) bestückt (Abbildung 5.7), die das SCSN (Kapitel 4.1) bedient, den LHC Takt verteilt und Messwerte, wie z. B. Temperaturen und Spannungen aufnimmt und verarbeitet.

Baumtiefe	4	3	2	1 (root)	TB	Summe
Taktzyklen [120 MHz]	4+1	3+2	4+2	4+2	4+1,9+2	28,9
Latenz [ns]	41,67	41,67	50,00	50,00	65,83	249,17

Tabelle 5.2: Durch die Funktionsweise der Netzwerkschnittstelle entstehen für die in Abbildung 5.4 gezeigten Stufen des Auslesebaums die dargestellten maximalen (theoretischen) Latenzen. Zudem ist hier nochmals die Latenz des Transmitters (TB) aufgeführt, wie schon in Tabelle 5.1.

In Abbildung 5.4 ist die vollständige Hierarchie des Auslesebaums dargestellt. Neben den 65 664 signalverarbeitenden MCMs werden noch 4104 MCMs (eines pro Ausleseplatine) und 1080 (eines pro halbes Modul, HMM) benötigt. Die Auslese wird damit aus insgesamt 70 848 MCMs aufgebaut. Das MCM der Tiefe eins (HMM), also die Wurzel des Baums, ist

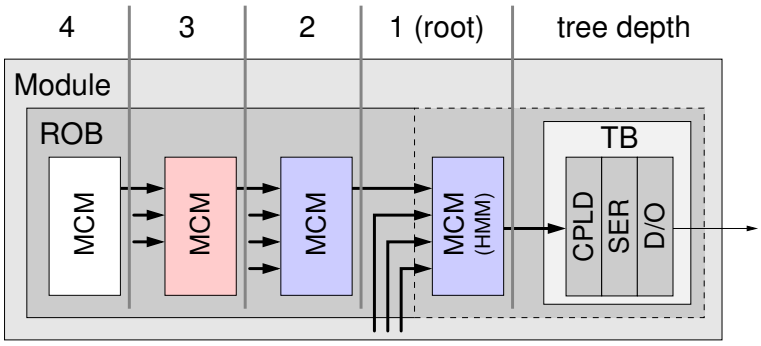

Abbildung 5.4: Der durch MCMs aufgebaute Auslesebaum, der die Daten eines halben Moduls sammelt, hat eine Tiefe und eine Breite von vier. Ein ROB kann optional mit einem MCM (HMM) der Tiefe 1 (root) und einer Transmitterplatine bestückt werden.

zusammen mit der Transmitterplatine optional für spezielle Ausleseplatinen.

Das Zustandekommen der für die Triggerauslese wichtigen Latenz der Daten ist in der Tabelle 5.2 der Baumtiefe nach aufgeschlüsselt. So haben z. B. die MCMs der Tiefe drei eine Latenz von 3+2 Taktperioden, um Daten weiterzuleiten. Die zwei Perioden ergeben sich aus dem Eingangsregister bzw. der Datenresynchronisation und dem Ausgangsregister. Die drei Perioden rühren von der Architektur der Netzwerkschnittstelle her. Die Minimierung der Latenzen innerhalb eines MCMs wird in Kapitel 6.2.6 näher erläutert. Die Latenz der Transmitterplatine lässt sich auch in den ersten drei Zeilen der Tabelle 5.1 wiederfinden. Die vier Taktperioden ergeben sich aus einem Testdesign für ein CPLD, das Resynchronisation, Bitausmaskierung und 8B/10B-Kodierung ausführt. Für die Serialisierung der Daten wurde der Tranceiver TLK2501 [43] von Texas Instruments zugrunde gelegt. Dieser hat eine maximale Latenz von 38 Taktperioden der 2,4 GHz des seriellen Ausgangs. Dies entspricht umgerechnet 1,9 Perioden des 120-MHz-Takts. Zwei weitere Perioden Latenz entstehen durch die Übertragung eines 16-Bit-Wortes selbst.

Zu beachten ist hierbei, dass die resultierende Latenz von fast 250 ns einen theoretischen Maximalwert darstellt. Diese Latenz entsteht nur dann, wenn sich für die Triggerauslese nur ein Spursegment in der Kammer befindet und dieses wiederum in dem MCM, das das letzte der Auslesereihenfolge bildet. In diesem Fall wäre zum Ausgleich die Übertragungszeit und natürlich auch die weitere Verarbeitung erheblich kürzer. Befinden sich mehrere Spursegmente in der Kammer, wird davon ausgegangen, dass diese etwa gleich verteilt sind und damit auch die Latenz geringer wird. Die MCMs, die in der Auslesereihenfolge früher kommen, haben auch eine geringere Latenz. Dies hängt mit der Implementierung der Netzwerkschnittstelle zusammen.

5.2.3 Datenfluss während der Trigger- und Rohdatenauslese

Nachfolgend werden jeweils ein vereinfachtes Beispiel für eine Trigger- und eine Rohdatenauslese vorgestellt. Durch den Verzicht auf ein Protokoll ist die Triggerauslese optimiert für

eine möglichst geringe Latenz und einen hohen Datendurchsatz, der durch einen lückenlosen Datenstrom erreicht wird (Kapitel 5.2). Die Anforderungen an die Geschwindigkeit der Rohdatenauslese sind deutlich geringer. Hierfür wird ein Handshake-Protokoll genutzt, wobei zugleich berücksichtigt werden muss, dass in beiden Formen der Auslese der Stromverbrauch (Kapitel 5.1) möglichst gering gehalten wird. Hierzu werden alle unbenutzten MCMs bzw. unbeteiligten Module abgeschaltet, wie z. B. der Prozessor oder die LVDS-Zellen.

In den folgenden Beschreibungen werden nur drei MCMs mit einer Baumbreite von zwei statt vier betrachtet. Alle drei MCMs haben eigene Daten und sollen in der Reihenfolge 2→3→1 ausgelesen werden. Die hier vorgestellte Funktionalität der MCMs bestimmt im Wesentlichen die Anforderungen an die Netzwerkschnittstelle, die im Kapitel 6 beschrieben ist.

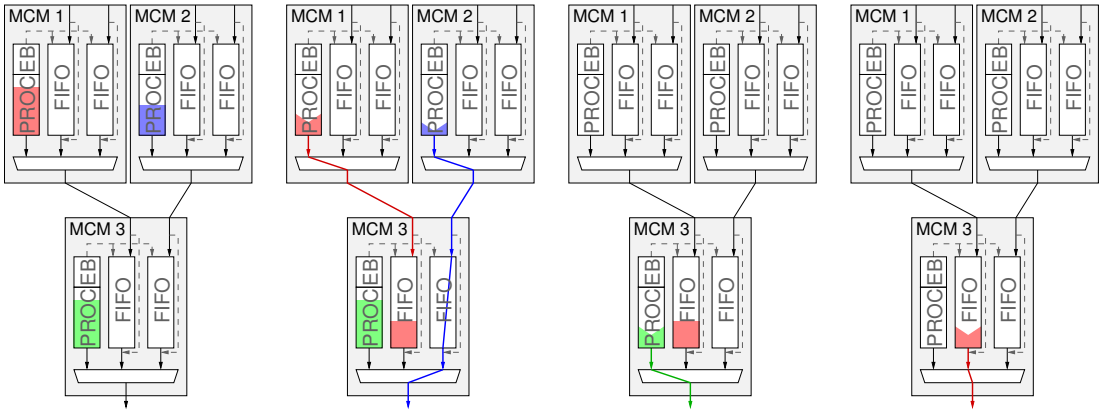

Abbildung 5.5: Die Prozessoren der drei MCMs enthalten Spursegmente für die Triggerauslese. Die Ausleseseihenfolge soll 2→3→1 sein. Alle MCMs senden gleichzeitig ihre Daten, bis auf MCM 3, das den direkten Ausgang bildet und zunächst die Daten von MCM 2 weiterleitet. Als letztes werden die in MCM 3 gepufferten Daten von MCM 1 ausgegeben.

Abbildung 5.5 zeigt das Beispiel für eine Triggerauslese. Zunächst halten die Prozessoren die zu übertragenden Spursegmente in ihren Registern. Danach senden alle MCMs gleichzeitig ihre Daten zum nachfolgenden MCM. MCM 3 hält zunächst seine eigenen Daten, weil es erst als zweites senden soll. Die Daten von MCM 1 werden im entsprechenden FIFO zwischengespeichert, die von MCM 2 können direkt an den Ausgang weitergeleitet werden. Danach sendet MCM 3 seine eigenen Daten und darauf die von MCM 1 gepufferten. Schon nach dem ersten Schritt können sich die beiden MCMs, die in der Hierarchie tiefer liegen, vollständig abschalten und durch die Pufferung der Daten entsteht ein kontinuierlicher Datenstrom am Ausgang von MCM 3. Um eine möglichst geringe Latenz zu erreichen, muss vor allem die Weiterleitung der Daten sehr schnell geschehen. Außerdem müssen die FIFOs alle Daten des vorherigen MCMs zwischenspeichern können. In diesem Beispiel ist dies nur eines, in der vorgestellten Auslesestruktur sind es 16 – also eine Ausleseplatine – pro FIFO.

Für die Rohdatenauslese, die in Abbildung 5.6 dargestellt ist, befinden sich die zu über-

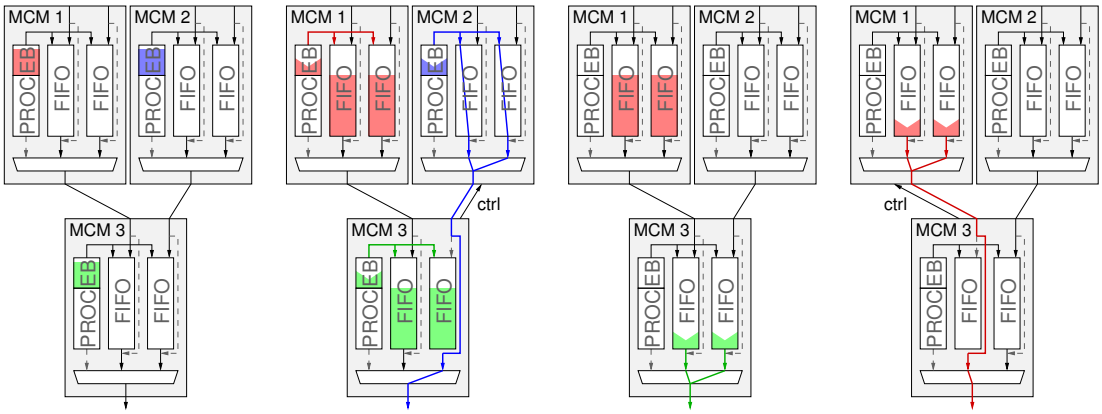

Abbildung 5.6: Als Beispiel für eine Rohdatenauslese wird die gleiche Reihenfolge wie in Abbildung 5.5 angenommen. Hier befinden sich die Daten im Ereignisspeicher (EB) und werden zunächst in den eigenen FIFOs zwischengespeichert. Danach können die Prozessoren abgeschaltet werden und über ein Handshake Signal werden von dem jeweiligen MCM die Daten angefordert.

tragenden Daten in den Ereignisspeichern (EB) des Filters (Kapitel 4.1.2). Dies wären bei z. B. 30 Zeitschritten der Akquisitionsphase 675 Byte pro MCM. Um das Schema der Triggerauslese hierzu zu verwenden, sind die Datenmengen zu groß und es wird ein Handshake-Protokoll benutzt. Die FIFOs sind so dimensioniert, dass sie die Menge der Daten des eigenen MCMs puffern können. Im ersten Schritt schreiben also alle MCMs ihre Daten in ihre eigenen FIFOs. Danach können sich die Prozessoren abschalten und lediglich die Netzwerkschnittstelle bleibt aktiv. Auch die I/O Zellen für die Datensignale der MCMs, die nicht senden, können abgeschaltet werden. Durch diese beiden Maßnahmen werden etwa 60% der sonst benötigten Leistung eingespart. Das MCM 3 signalisiert über eine Kontrollleitung dem MCM 2, dass es seine Daten senden kann und leitet diese direkt an den Ausgang weiter. Danach sendet es seine eigenen Daten und signalisiert dann MCM 1, dass es bereit ist die Daten zu empfangen. Erst jetzt müssen die I/O Zellen der Datensignale wieder eingeschaltet werden.

5.2.4 Spannungsversorgung und Kontroll- und Taktsignale

Alle MCMs auf einer Ausleseplatine, außer MCM 16, verfügen über vier vollkommen separate Versorgungsspannungen und Massen. Dies ist notwendig, damit die hochsensiblen analogen Baugruppen wie die Vorverstärker (PASA) und die ADCs nicht durch Spannungsschwankungen der Versorgung der digitalen Teile gestört werden.

Der PASA als ein eigener Chip des MCMs benötigt 3,3 V, während die ADCs, die Bestandteil des TRAP Chips sind, mit 1,8 V versorgt werden. Der digitale Teil des TRAPs benötigt sowohl eine 1,8 V Versorgung für den Logikteil als auch eine 3,3 V Versorgung für die I/O-Zellen. Für alle diese Spannungen sind Regulatoren auf dem ROB, die jeweils 8 MCM versorgen; das MCM 16 hat eine eigene Spannungsversorgung. Um Störungen zu minimieren werden außerdem diese vier Spannungen über eigene Lagen des ROBs verteilt.

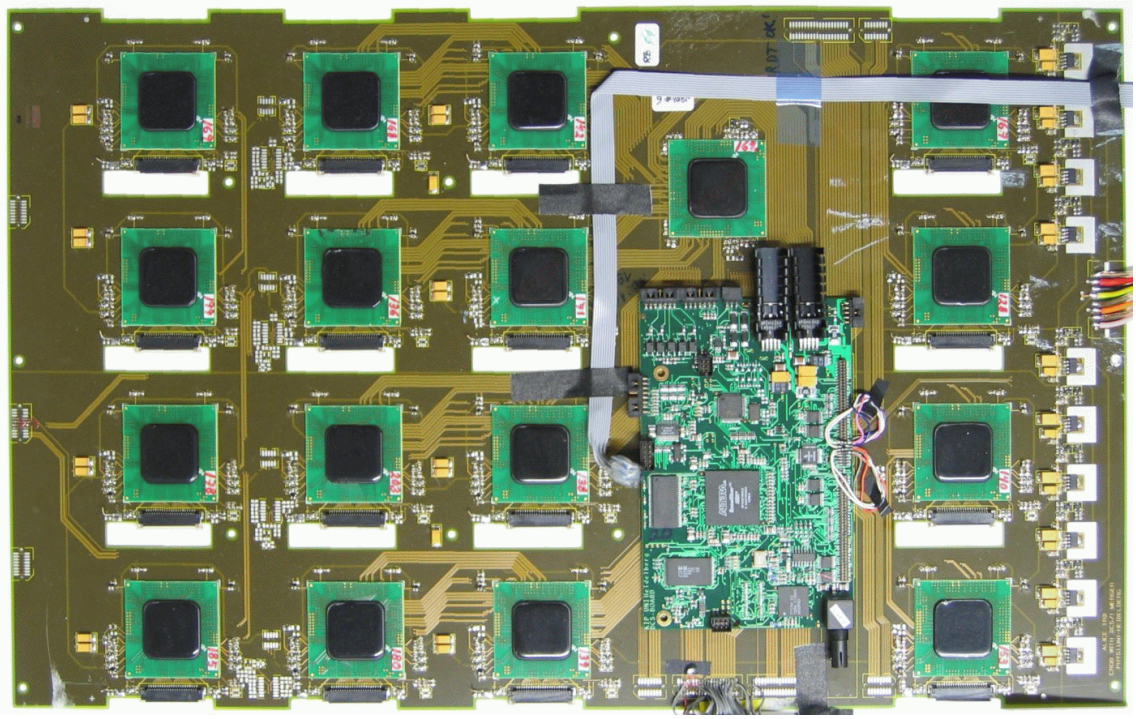

Abbildung 5.7: Das Foto zeigt eine Ausleseplatine, bestückt mit 17 MCMs und der DCS-Karte, die das SCSN bedient, den Takt verteilt und Überwachungsaufgaben erledigt.

Abbildung 5.7 zeigt eine Variante des ROB, die mit einer DCS[1]-Karte bestückt ist. Diese ist in erster Linie zuständig für die Verteilung der Taktsignale, die Konfiguration der MCMs und für Überwachungsaufgaben. Kern der Platine ist ein FPGA mit einem ARM Controller mit Linux als Betriebssystem. Zur Kommunikation nach außen ist eine Ethernet Schnittstelle implementiert. Auf diese Weise kann von zentraler Stelle aus über eine standardisierte Schnittstelle, wie sie jeder PC besitzt, die gesamte Konfiguration und Überwachung des Detektors erfolgen. Da auch beliebige Funktionen eigenständig von den Controllern auf den Detektoren ausgeführt werden, ist es möglich sehr schnell aktiv zu reagieren und lediglich Statusinformationen an die zentrale Kontrolle weiterzugeben.

Über einen TTC[2] System Empfänger wird der global über das Experiment verteilte LHC Takt vervielfacht und an die ROBs weitergegeben. Die Verteilung des Takts auf einem ROB selbst erfolgt dann ähnlich der Auslese über die TRAP Chips in einer Baumstruktur. Jedes MCM besitzt einen Takteingang, bereitet diesen Takt mithilfe einer PLL auf und kann damit weitere vier MCMs versorgen.

Die TRAPs besitzen eine Konfigurationsschnittstelle, das SCSN (Kapitel 4.1), über die auf den Programmspeicher des Prozessors und alle Konfigurations- und Statusregister zugegriffen werden kann. Die zugehörigen Master dieses Netzwerks sind im FPGA der DCS-Karte

[1] Detector Control System, http://alicedcs.web.cern.ch
[2] Timing, Trigger and Control, http://ttc.web.cern.ch

implementiert.

Darüberhinaus verfügt die DCS-Karte über ADCs mit denen die Versorgungsspannungen, Temperaturen und eventuell noch andere Größen, wie z. B. der Gasdruck der Kammern, überwacht werden können. Damit ist es möglich, auf dem Detektor über autark laufende Prozesse die Spannungsregler zu kontrollieren und in kritischen Fällen abzuschalten.

6 Netzwerkschnittstelle

Die Netzwerkschnittstelle ist eine Baugruppe innerhalb des Tracklet-Prozessors und bildet die Verbindung des Prozessors zum Auslesenetzwerk, das hiermit aufgebaut wird. Der Datenfluss – und damit die Netzwerkschnittstelle – sind optimiert auf die speziellen Anforderungen des TRDs in dem Experiment.

In diesem Kapitel wird die Architektur und Implementierung sowie eine genaue Beschreibung der Funktionalität zur Anwendung der Schnittstelle erläutert. Das zugehörige Programmiermodell mit einer Beschreibung aller Register ist in Anhang A zu finden. Die Registernamen werden im Text hervorgehoben dargestellt (z. B. **NMOD.m**).

6.1 Anforderungen

Die Anforderungen an das Auslesenetzwerk aus Kapitel 5 und die daraus folgende Struktur bestimmen im Wesentlichen die geforderten Eigenschaften der Netzwerkschnittstelle. Hierbei ist die Auslese der für den Level-1-Trigger (Kapitel 2.2) relevanten Daten (Triggerauslese) entscheidend in Bezug auf Latenz und Übertragungsrate. Während der kritischen Triggerauslese arbeiten auf dem gesamten Detektor alle Prozessoren parallel. Das bedeutet eine sehr hohe Leistungsaufnahme. Die Netzwerkschnittstelle ist so konzipiert, dass sich die individuellen CPUs jedes Prozessors möglichst früh ausschalten können, und die Auslese der Daten autonom erfolgen kann. Die Auslese der Rohdaten ist ebenfalls optimiert auf einen möglichst geringen Stromverbrauch. Die Rohdatenauslese nach einem Level-2-Trigger dauert aufgrund der größeren Datenmenge entsprechend länger, so dass nicht nur die Prozessoren während der Auslese abgeschaltet werden können, sondern auch der gerade nicht beanspruchte Teil des Auslesebaums inklusive der LVDS-Zellen.

Mithilfe der Netzwerkschnittstelle werden 4-zu-1 Auslesebäume aufgebaut. Gefordert sind eine möglichst geringe Latenz durch die einzelnen Hierarchien und ein kontinuierlicher Datenstrom an der Wurzel des Baums, um einen maximalen Datendurchsatz zu erreichen. Wie in Kapitel 5.2 beschrieben, ist es hierzu notwendig, Daten zwischenspeichern zu können. Die Übertragungsraten sind dabei durch die Taktfrequenz und die Breite des Busses bestimmt.

6.2 Architektur

Die Netzwerkschnittstelle ist ein Gerät des TRAPs, das an die vier lokalen Busse und den globalen Bus des Prozessors angeschlossen ist (Kapitel 4.1.4). Durch die lokalen Schnittstellen ist ein schneller Datenaustausch mit der entsprechenden CPU möglich. Über die globale Schnittstelle findet die Konfiguration der Netzwerkschnittstelle statt. Statussignale werden je nach Zugehörigkeit zur Verfügung gestellt. Die Schnittstelle nach außen bilden vier Eingangsports und ein Ausgangsport. Hiermit werden die Auslesebäume des Auslesenetzwerkes aufgebaut. Die Anzahl der Ports, die Bitbreite eines Ports und die Übertragungsrate sind durch das Auslesenetzwerk bestimmt.

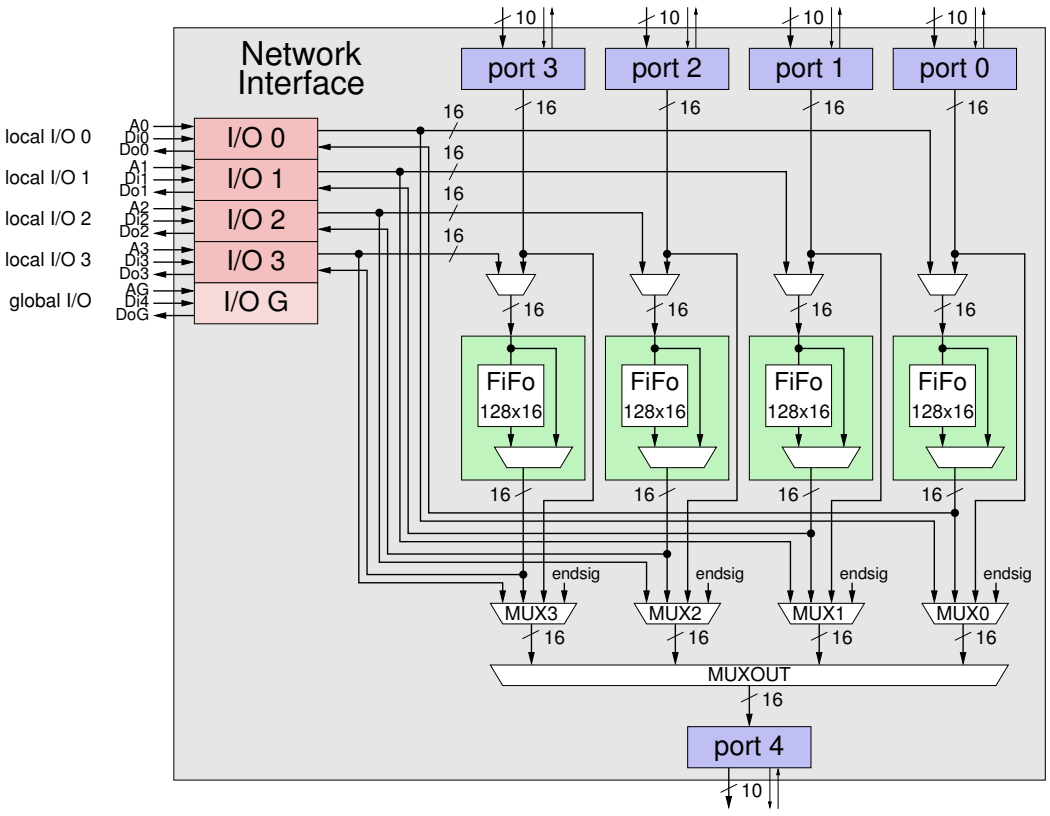

Abbildung 6.1: Datenpfaddiagramm der Netzwerkschnittstelle. Die I/O 0..3 und I/O G Schnittstellen bilden die Verbindung zum Prozessor bzw. zum globalen Bus. Vier Eingangsports und ein Ausgangsport sind die Schnittstelle nach außen. Sowohl über die Eingänge als auch über die CPUs können bis zu 128 Wörter in FIFOs gepuffert werden.

Während die Anbindung zu den Schnittstellen des Prozessors über die 32 Bit breiten Datenbusse erfolgt, arbeiten die Ports nach außen mit 8 Bit breiten Datenwörtern. Diese werden allerdings mit doppelter Datenrate (DDR[1]) übertragen, also jeweils ein Byte je steigender bzw. fallender Flanke. Für jeden internen Takt werden von den Ports also 16 Bit Daten verarbeitet. Die Netzwerkschnittstelle arbeitet intern mit 16 Bit breiten Datenbussen. Dies

[1] Double Data Rate

ist die kleinste Einheit, die von den Ports und FIFOs verarbeitet wird. Hierdurch lässt sich die Latenz für die Weiterleitung von Daten (von einem Eingangs- zu dem Ausgangsport) minimieren.

Zur weiteren Minimierung der Latenz ist die gesamte Architektur der Netzwerkschnittstelle als Einzyklenarchitektur implementiert. Der gesamte Datenpfad von einem synchronisierten Eingangsportregister durch das FIFO bis zum Ausgangsregister ist asynchron. Lediglich der Datenfluss in Richtung der I/O-Schnittstellen erfordert mehr als einen Takt, da hierbei die Umsetzung auf 32 Bit stattfindet. Dieser Datenfluss findet allerdings weder in der Trigger- noch in der Rohdatenauslese statt, ist somit also nicht kritisch.

Die FIFOs haben in mehrfacher Hinsicht eine besondere Bedeutung für die Funktionalität der Auslese. Die FIFOs sind so dimensioniert, dass sie während der Triggerauslese alle einkommenden Daten puffern können. Die FIFOs der Netzwerkschnittstelle, die die Wurzel eines Auslesebaums bildet, können somit die Triggerdaten einer halben Kammer zwischenspeichern. Hiermit wird nach dem Erreichen des ersten Datenwortes ein möglichst kontinuierlicher Datenstrom erzeugt. Während der Rohdatenauslese beschreiben die CPUs die FIFOs und schalten sich danach ab. Die Daten fließen dann nach und nach ab, wobei nur die aktiven MCMs die I/O Zellen der Datensignale zur Übertragung einschalten. Die Abläufe während der Auslesen sind in den Kapiteln 6.3.1 und 6.3.2 beschrieben.

6.2.1 Anbindung zum Prozessor

Um die bis zu vier Spursegmente pro MCM für die Funktion als Trigger zu verarbeiten, besteht der Prozessor des TRAPs aus vier CPUs (Kapitel 4.1.4). Jede CPU hat dabei über eine lokale Schnittstelle direkten Zugriff auf die Netzwerkschnittstelle. Die Architektur der Schnittstelle – im Zusammenspiel mit den FIFOs – ist dabei so gewählt, dass sich die CPUs nach der Übertragung der Daten abschalten können und die Auslese autonom erfolgen kann. Dies machen die Anforderungen an den Stromverbrauch des TRAPs (Kapitel 5.1) erforderlich. Während der Triggerauslese wird hierzu das 32 Bit breite Spursegment in das entsprechende Register der lokalen Schnittstelle der Netzwerkschnittstelle geschrieben. Für die Daten während der Rohdatenauslese werden dafür die FIFOs genutzt. Die lokalen Schnittstellen besitzen außerdem Statusregister, die für den jeweiligen Datenpfad spezifisch sind. Über die globale Schnittstelle kann jede CPU des Prozessors und das SCSN (Kapitel 4.1) Kontrollregister setzen und Statusregister auslesen.

Während der Prozessor mit einem 32 Bit breiten Datenpfad arbeitet, ist der interne Datenpfad der Netzwerkschnittstelle 16 Bit breit. Die Umsetzung von 32 auf 16 Bit und umgekehrt erfolgt in den lokalen Schnittstellen. Diese Umsetzung erfolgt nur für die zu übertragende Daten, für Kontroll- und Statuswörter ist sie nicht notwendig.

Globale Anbindung

Über die globale Anbindung, über die auch das SCSN Zugriff hat, werden alle Kontrollregister des NI gesetzt. Tabelle 6.1 zeigt einen Überblick über diese Register, deren Bit-weise

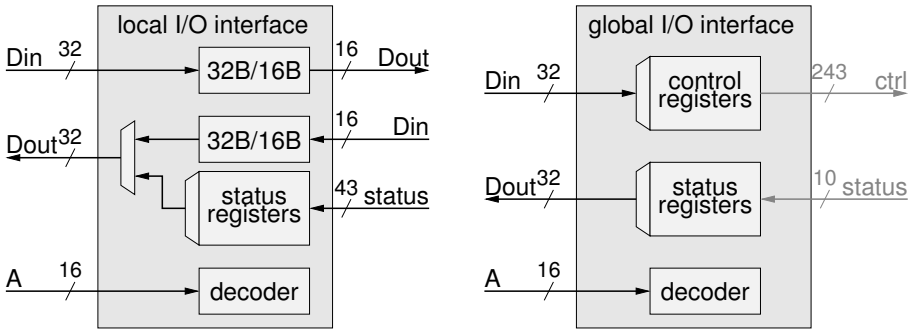

Abbildung 6.2: Über die lokalen Schnittstellen (links) können die vier CPUs Daten übertragen und Port-spezifische Statusregister abfragen. Alle Kontrollregister befinden sich innerhalb der globalen Schnittstelle (rechts), auf die alle CPUs Zugriff haben.

Aufschlüsselung im Programmiermodell in Anhang A beschrieben sind. Zusammengehörige Bitgruppen sind dort mit Buchstaben gekennzeichnet. So kann z. B. der Auslesemodus über das Register **NMOD**.**m** gesetzt werden, wobei **NMOD** einen Bezeichner für die Adresse darstellt und **m** die zugehörigen drei Bit für den Modus. Der globale Adressbereich beginnt bei 0x0D40 und umfasst 13 Wörter mit je 32 Bit Breite. Die Anzahl der effektiv genutzten Bits hängt von dem Register ab.

Register	Adresse	Beschreibung
NMOD	0x0D40	NI Modus Kontrolle
NDLY	0x0D41	Verzögerungen der Datenausgangbits
NED	0x0D42	Ausgangs-Bitpositionen und Verzögerungen
NTRO	0x0D43	Triggerauslesereihenfolge
NRRO	0x0D44	Rohdatenauslesereihenfolge
NES	0x0D45	Endsignaturen
NTP	0x0D46	Testmuster
NBND	0x0D47	FIFO Schwellen
NPn	0x0D48+n	Eingangsport n∈[0..3] Kontrolle
NCUT	0x0D4C	Abschneiden des Datenstromes

Tabelle 6.1: Globale Kontrollregister der Netzwerkschnittstelle.

Die Statusregister, auf die global zugegriffen werden kann, beinhalten den Kontrolleingang des Ausgangsports, eine Kopie der FIFO Fehlersignale und den kodierten Zustand der Netzwerkschnittstelle.

Register	Adresse	Beschreibung
NCTRL	0x0DC0	Kontrolleingang des Ausgangsports
NFIFO	0x0DC1	FIFO Fehler Signale
NFSM	0x0DC2	Zustand der Netzwerkschnittstelle

Tabelle 6.2: Globale Statusregister der Netzwerkschnittstelle.

Der auf das Statusregister **NFSM** abgebildete Zustand zeigt nicht jeden Zustand des Automaten der Netzwerkschnittstelle. Die Kodierung ist in der Beschreibung des Registers aufgelistet.

Lokale Anbindungen

Die wichtigsten Statussignale der einzelnen Module werden lokal abgebildet, damit die CPUs schnell und ohne Abhängigkeiten von den anderen CPUs darauf Zugriff haben. Wie man in Abbildung 6.1 erkennen kann, sind die Datenpfade der vier Ports und FIFOs über die lokalen Anbindungen eng mit jeweils einer der CPUs gekoppelt. Jede CPU hat Zugriff auf das ihr zugeordnete lokale Statusregister.

Register	Adresse	Beschreibung
NLF	0x00C0	FIFO Status Signale
NLP	0x00C1	Paritätsfehler- und Wortzähler
NLE	0x00C2	Zustand der Netzwerkschnittstelle

Tabelle 6.3: Lokale Statusregister der Netzwerkschnittstelle.

Der Kontrolleingang des Ausgangsports betrifft zwar alle vier CPUs, jedoch hängt die Bedeutung dieses Signals von dem jeweiligen Zustand des Ausgangsmultiplexers ab. Deshalb ist es für diejenige CPU, der gerade der Ausgangsport zugewiesen ist, ein Signal, auf das sie direkt reagieren können muss.

6.2.2 Ports

Die Ports bilden die Schnittstelle des Prozessors nach außen hin. Die Übertragung findet über 10 Bit breite Datenbusse, ein Strobe-Signal und ein Kontrollsignal in umgekehrter Richtung statt. Der Ausgangsport verfügt außerdem über einen Kontrollausgang, der zur Ansteuerung der Transmitterplatine (Kapitel 5.2.1) benötigt wird. Von den 10 Bit Datensignalen können zwei – frei maskierbar – anderweitig genutzt werden. Das eine ist ein reines Reservebit zur Fehlertoleranz, das andere wird als Paritätsbit genutzt. Fallen zwei Leitungen aus, kann auch das Paritätsbit als Datenleitung genutzt werden.

Der Eingangsport (Abbildung 6.3, links) resynchronisiert die mit doppelter Datenrate (DDR) einkommenden Daten zum internen Takt mit doppelter Breite und einfacher Datenrate. Auf diese Weise entstehen zwei Datenwörter mit jeweils 10 Bit und 120 MHz. Für jedes dieser Wörter werden daraufhin das Paritäts- und das Reservebit ausmaskiert und die Datenbytes auf Paritätsfehler geprüft. Die Anzahl der empfangenen Datenwörter und die Paritätsfehler der Bytes für jeweils die steigende und fallende Strobe-Flanke werden gezählt und stehen über die lokalen Statusregister **NLP** zu Verfügung.

Der Ausgangsport (Abbildung 6.3, rechts) berechnet die Parität und fügt diese zusammen mit dem Reservebit in den Datenstrom ein. Dies erfolgt äquivalent zum Eingangsport für die

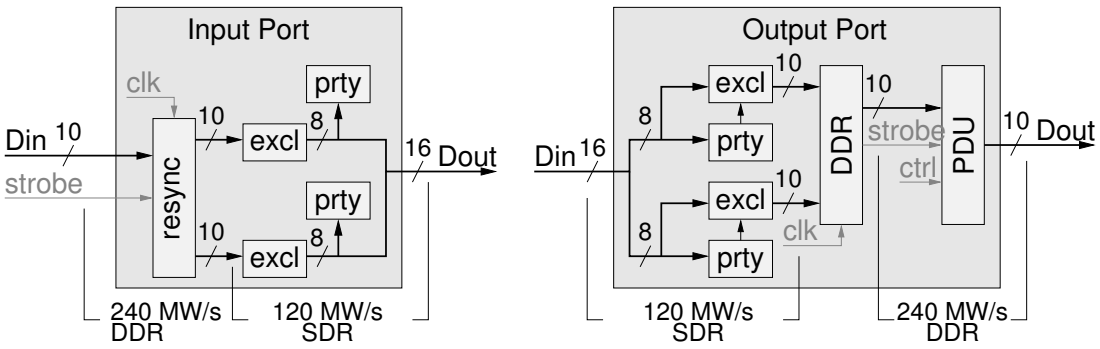

Abbildung 6.3: Der Eingangsport (links) resynchronisiert die DDR-Daten zum internen Takt und maskiert das Paritäts- und das Reservebit aus. Der Ausgangsport (rechts) berechnet die Parität, fügt die beiden Bits in den Datenstrom ein, führt eine DDR-Wandlung durch und verzögert die einzelnen Ausgangsbits.

beiden Bytes des internen Datenwortes separat. Darauf erfolgt die DDR-Wandlung. Der resultierende 10 Bit breite Datenstrom (bei 240 MWörtern/s) kann dann zusammen mit dem Strobe- und dem Ausgangskontrollsignal durch die programmierbare Verzögerungseinheit (PDU) bitweise verzögert werden.

Ausgangs- und Eingangskodierung

Wie in Kapitel 5.2 beschrieben, werden dem ursprünglich 8 Bit breiten Datenstrom zur Fehlertoleranz zwei weitere Bits hinzugefügt. Die Position dieser Bits kann frei gewählt werden. Dadurch lassen sich defekte Leitungen, wie sie z. B. durch kalte Lötstellen oder unsaubere Bondverbindungen entstehen können, ausmaskieren. Dafür ist zunächst ein Reservebit vorgesehen, das ansonsten keine Funktion hat. Das zweite Bit, das hierfür genutzt werden kann, ist das Paritätsbit. Mit diesem Bit können während des Betriebs fehlerhafte Übertragungen aufgespürt werden um dann entsprechend darauf zu reagieren. Wird das Paritätsbit für eine zweite defekte Leitung genutzt, besteht diese Möglichkeit natürlich nicht mehr.

Die Positionen der beiden Bits lassen sich für die Eingangsports über die Register **NPn**, für den Ausgangsport über das Register **NED** einstellen. In Abbildung 6.4 wurde als Beispiel die Position 6 für das Paritätsbit (Pa) und 5 für das Reservebit (Sp) gewählt. Zunächst wird Pa an Position 6 eingefügt; das bedeutet, die beiden Datenbits D6 und D7 ändern ihre Position. Im zweiten Schritt wird das Reservebit an die niedrigere Position 5 eingetragen. Dabei wandern sowohl die Datenbits D5-D7 als auch das Paritätsbit um eine Stelle nach links.

Zusammen mit dem Strobe- (St) und dem Ausgangskontrollsignal (Ctl) können alle Leitungen des Ausgangsports bitweise verzögert werden (Kapitel 6.2.3). Nach der eingangsseitigen Resynchronisierung werden diese wieder entsprechend ausmaskiert bzw. das Paritätsbit zum Vergleich weiterbenutzt. Natürlich ist darauf zu achten, dass die Positionen der Bits für den entsprechenden Ein- und Ausgangsport identisch sind.

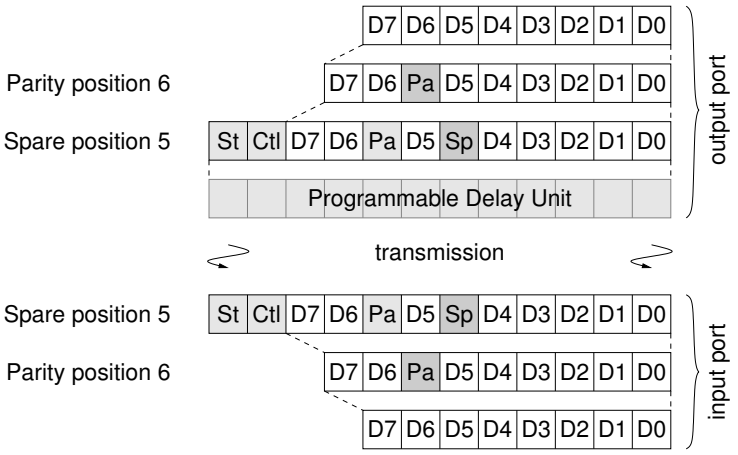

Abbildung 6.4: Der 8 Bit breite Datenstrom wird um ein Paritäts- und ein Reservebit ergänzt. Die Positionen dieser beiden Bits sind über die Register **NPn** und **NED** einstellbar. Dadurch lassen sich (bei Verzicht auf das Paritätssignal) bis zu zwei defekte Bitleitungen ausblenden. Über die programmierbare Verzögerungseinheit am Ausgangsport lassen sich der 10 Bit breite Datenstrom und das Strobe-Signal verzögern. Auf der Empfängerseite wird entsprechend dekodiert.

6.2.3 Programmierbare Verzögerungseinheit

Bei einer synchronen parallelen Übertragung ist es wichtig, dass die einzelnen Datensignale zeitlich nicht gegeneinander verschoben und in einer definierten Phase zum Takt bzw. Strobe sind. Bei einer Übertragung mit doppelter Datenrate (DDR) beträgt dieser Versatz idealerweise eine viertel Periode. Vor allem durch asymmetrisches Verdrahten der Leiterbahnen auf der Ausleseplatine kann es zu zu unterschiedlichen Verzögerungen zwischen den einzelnen Signalen kommen. Eine konventionelle Methode, dieses Problem in den Griff zu bekommen, ist die Länge der Leiterbahnen anzupassen (matched length). Das hat allerdings den Nachteil, dass im Mittel längere Leiterbahnen benötigt werden und damit der Platzbedarf steigt. Dies gilt nicht nur für die Ausleseplatine selbst, sondern für jedes angeschlossene Modul, also auch den HMM und die Transmitterplatine (Kapitel 5.2). Des Weiteren werden unterschiedliche Typen von Empfängern eingesetzt, bei deren Eingangszellen das zeitliche Verhalten berücksichtigt werden muss.

Um alle diese Verzögerungen zwischen den Datensignalen auszugleichen und um das Strobe-Signal an die optimale Position zu schieben (Kapitel 6.2.4), lässt sich im Ausgang eines Ports jede Bit-Leitung separat verzögern. Häufig werden diese Verzögerungseinheiten eingangsseitig ausgeführt, was den Vorteil hat, dass nur die tatsächlich benutzten Leitungen (Kapitel 6.2.2) betrachtet werden müssen. Im Falle der Netzwerkschnittstelle mit ihren vier Eingangsports und einem Ausgangsport hieße dies allerdings, dass vier Verzögerungseinheiten notwendig wären. Durch die Verlagerung der Einheit an den Ausgang wird nur eine benötigt. Außerdem lassen sich durch diese Architektur beliebige Logikbausteine anschließen, wie z. B. das CPLD der Transmitterplatine (Kapitel 5.1), und die Signalverzögerungen daraufhin optimieren.

Ein Verzögerungselement der Tiefe 1 (Abb. 6.5) besteht aus zwei Verzögerungszellen, zwei

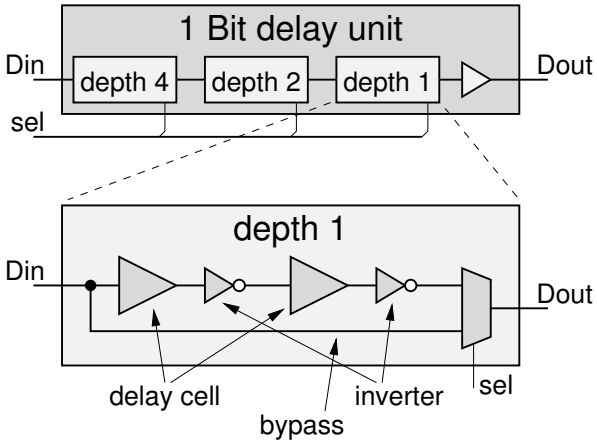

Abbildung 6.5: Eine Verzögerungseinheit und die Kaskadierung von mehreren Einheiten verschiedener Tiefe.

Invertern und einem Multiplexer, um zwischen dem verzögerten und dem unverzögerten Signal zu wählen. Zellen, wie z. B. Inverter, sind im Allgemeinen nicht ausbalanciert, d. h. sie besitzen verschiedene Verzögerungen für die beiden Flanken. Die doppelte Ausführung der Zellen ist gewählt um den Unterschied der Verzögerung von steigender und fallender Flanke möglichst gering zu halten. Kaskadiert man die Elemente und verdoppelt jeweils die Anzahl der Verzögerungszellen, lässt sich über das Signal sel binär kodiert die Tiefe und damit die Verzögerung einstellen. Am Ausgang der Kette ist ein einfacher Treiber implementiert, um die nachfolgende Logik anzusteuern.

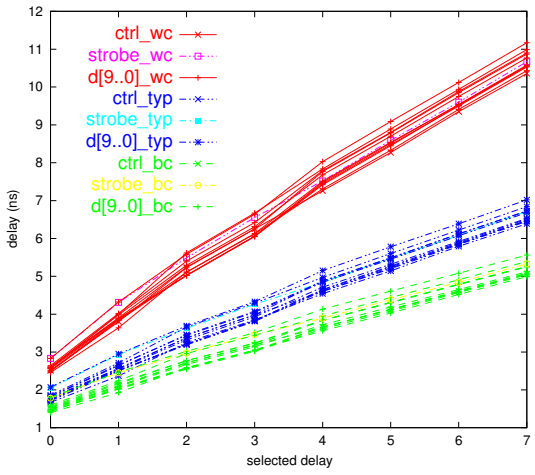

Abbildung 6.6: Programmierbare Ausgangsverzögerungen für verschiedene Umgebungsbedingungen (TRAP3 Timing Analyse).

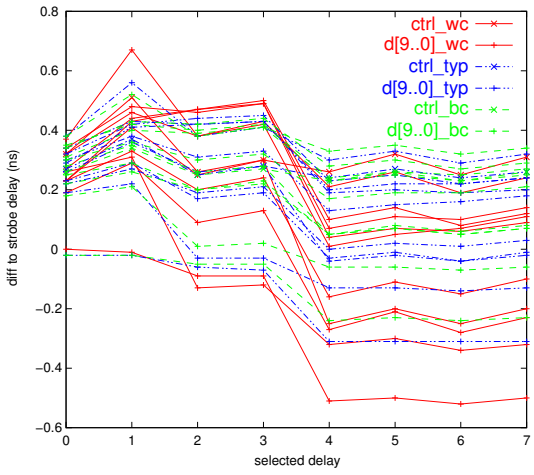

Abbildung 6.7: Programmierbare Verzögerungen relativ zum Strobe-Ausgangssignal (TRAP3 Timing Analyse).

Für den benutzten Prozess [46] ergibt sich die minimale Latenz für alle Signale aus den Verzögerungszeiten der drei Multiplexer und des Ausgangstreibers. In diesem Fall (Abb. 6.5) ist die Latenz $3 \cdot 0,120\,\text{ns} + 0,066\,\text{ns} = 0,426\,\text{ns}$. Die eigentliche Verzögerung kann in Schrit-

ten von $2 \cdot 0,281\,\text{ns} + 2 \cdot 0,029\,\text{ns} = 0{,}620\,\text{ns}$ eingestellt werden, was der Laufzeit durch zwei Verzögerungszellen und zwei Inverter entspricht.

Die Verzögerungen der Datenleitungen des Ausgangsports lassen sich durch das Register **NDLY** einstellen. Die Grundeinstellung ist auf den Wert 4 gesetzt, womit die Datensignale etwa 2,4 ns relativ zum Strobe-Signal verzögert sind. Das Strobe- und Kontrollsignal können mittels der Register **NED.s** und **NED.c** verzögert werden. Idealerweise liegt die Mitte der stabilen Datenphase genau am Strobe-Übergang. Um eine größtmögliche Flexibilität zu erhalten, ist die maximale Verzögerung etwa einen Taktzyklus lang. Wenn also sowohl das Strobe-Signal als auch die Datensignale um einen halben Takt verzögert sind, lassen sich die Datensignale um bis zu einen halben Takt vor bzw. hinter die Strobe-Flanke schieben.

6.2.4 Datenresynchronisation

Als globales Referenztaktsignal wird im gesamten ALICE-Experiment der LHC-Takt mit einer Frequenz von $\sim 40\,\text{MHz}$ benutzt. Dieses Signal wird auf dem TRD verteilt und durch PLLs[2] auf den Kammern auf 120 MHz hochmultipliziert. Das bedeutet, man kann als grundlegende Eigenschaft der internen Takte verschiedener MCMs davon ausgehen, dass

- alle internen Takte dieselbe Frequenz haben,
- die Takte beliebig gegeneinander verschoben sind,
- die Phase zur Laufzeit um weniger als einen halben Takt variiert.

Diese Eigenschaften kann man sich für eine optimierte Methode der Resynchronisation der Daten zum internen Takt zu Nutze machen. Die Gültigkeit und der Zeitpunkt der Gültigkeit der Daten (DDR[3] kodiert) werden durch ein Strobe-Signal an den Empfänger mit übertragen. Über die programmierbare Verzögerungseinheit am Datenausgang werden die Daten zueinander in Phase geschoben und die Flanken des Strobe-Signals in die Mitte des Augenmusters der Datensignale gelegt.

Die eingehenden Daten werden in vier doppelte (fallende und steigende Flanke) Register übernommen. Die Auswahl des aktuellen Registers übernimmt ein Zähler, der zur fallenden Taktflanke aktualisiert wird. Die eigentliche Synchronisation geschieht mit der Übernahme des Zählerstandes in ein Register zum internen Takt. Um Fehler bei der Übernahme durch Laufzeitbedingungen zu beseitigen, ist dieser Zähler Gray-kodiert. Mithilfe der Zählerstände und eines Signals, das angibt, ob die Daten im vorherigen Takt gültig waren, kann das Register, das die aktuell gültigen Daten enthält, ausgewählt werden. Das Datum wird zusammen mit einem Signal, das die Gültigkeit anzeigt (`we_n_i`), ausgegeben.

Abbildung 6.8 veranschaulicht die Funktionsweise der Resynchronisationseinheit. Das eingehende Strobe-Signal `strobe` (ab 20 ns) ist zu dem internen Takt etwas zurückversetzt. Die Zeit von der ersten Strobe-Flanke bis zur möglichen Übernahme der synchronisierten Daten ist hierbei etwas über 12 ns. Bei etwa 50 ns verschiebt sich das `strobe` Signal hinter

[2] Phase-Locked Loop
[3] Double-Data-Rate

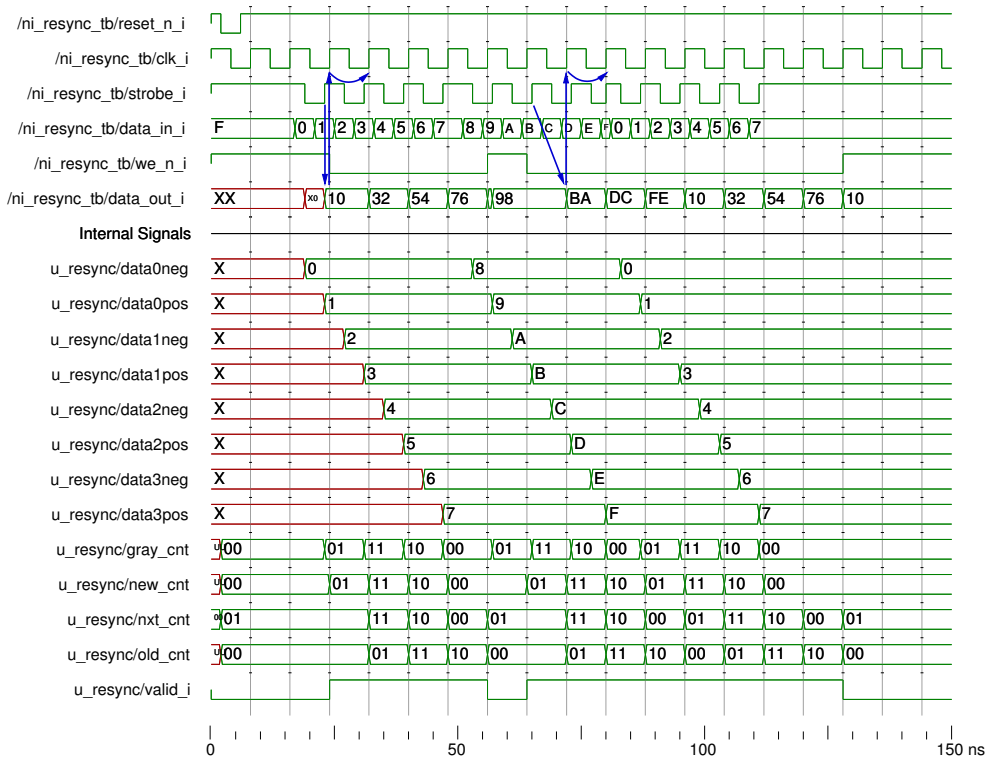

Abbildung 6.8: Timing Diagramm des Resynchronisationsmoduls aus einer funktionalen Simulation. An den Stellen 50 ns und 80 ns verschiebt sich das eingehende Strobe-Signal zum internen Takt. Es wird an diesen Stellen (blau) die minimale und maximale Latenz der Einheit deutlich.

den internen Takt. Dies kann selbst durch kleinste Störungen geschehen, wenn die beiden Flanken extrem nahe beieinander liegen. Die Latenz beträgt nun fast einen Takt mehr (maximale Latenz). Bei etwa 80 ns liegt wieder das ursprüngliche Strobe-Signal an. Der Datenfluss setzt sich dann ununterbrochen fort.

6.2.5 FIFOs

Je nach Auslesemodus (Kapitel 5.2.3) sind die Eingänge der FIFOs an die Eingangsports oder die lokalen Schnittstellen verbunden. Während der Triggerauslese werden die Daten anderer MCMs, in der Rohdatenauslese die des internen Ereignisspeichers gepuffert. Ein FIFO ist mittels eines synchronen Dual-Port-SRAMs implementiert. Um die Latenz für die zeitkritische Triggerauslese möglichst gering zu halten, ist dem FIFO ein Register vorgeschaltet, mit dessen Hilfe und einer geeigneten Auswahllogik das FIFO selbst keine Latenz erzeugt.

Werden von einem FIFO Daten angefordert, wenn noch keine eingetragen wurden, wird der Eingang direkt auf den Ausgang geschaltet. Das zusätzliche Register ist für den Fall notwendig, dass eine Anforderung genau einen Takt nach dem Eintragen erfolgt. Hierbei werden dann die Daten aus dem Register genommen und nicht in das FIFO eingetragen. Ist

der Unterschied zwischen Eintrag und Abfrage größer als einen Takt, erfolgt eine Pufferung durch das FIFO.

Über das Register **NBND** können Schwellen eingestellt werden, in Abhängigkeit derer dann Signale erzeugt werden. Die FIFOs produzieren eine ganze Reihe von Statussignalen, die über die lokalen Register **NLF** und **NLE** abgefragt werden können. Ob ein Fehler aufgetreten ist, lässt sich außerdem über das globale Register **NFIFO** ermitteln.

Im Netzwerkmodus (Kapitel 6.3.3) werden die eingehenden Daten von den FIFOs gepuffert und können dann von den lokalen Schnittstellen gelesen werden. Um diese Funktionalität zu unterstützen, ist es möglich, durch das Bit **NPn.c** das Kontroll-Ausgangssignal des jeweiligen Eingangsports auf das Statussignal **NLF.H** zu legen. Dieses zeigt an, wenn die Anzahl der Einträge über der programmierbaren Schwelle liegt. Das sendende MCM kann dann z. B. durch eine Interrupt Routine darauf reagieren.

6.2.6 Ausgangsmultiplexer

Wie in Abbildung 6.1 erkennbar, lässt sich durch den Multiplexer direkt vor dem Ausgangsport (MUXOUT) die Quelle für den Ausgang wählen, also z. B. zwischen Port 0–3 oder I/O 0–3, je nachdem wie die Multiplexer MUX 0–3 geschaltet sind. Diese Multiplexer wählen entsprechend des Zustands der Auslese zwischen dem Port, dem FIFO oder der lokalen Schnittstelle die Datenquelle aus. Durch diese Kombination lässt sich eine frei programmierbare Auslesereihenfolge ermöglichen, so dass auch Szenarien wie in Abbildung 5.5 möglich sind, bei denen die MCMs z. B. erst die Daten von Port 0 weiterleiten, dann die eigenen senden und danach die gepufferten von FIFO 3 ausgeben.

6.3 Auslesemodi

Die Auslese sowohl der Trigger- als auch der Rohdaten soll in großen Teilen autonom ablaufen, d. h. während der Prozessor abgeschaltet ist. Dies bedeutet, dass die Netzwerkschnittstelle durch einen entsprechenden Zustandsautomaten (FSM) gesteuert wird. In den folgenden Kapiteln wird detailliert auf die Abläufe während der unterschiedlichen Auslesemodi eingegangen. Durch die Funktionalität dieser Modi werden im Wesentlichen die Methoden der in Kapitel 5.2.3 beschriebenen Auslese implementiert.

Auch wenn die Steuerung der Netzwerkschnittstelle durch nur einen Zustandsautomaten (Abbildung 6.9) implementiert ist, so lassen sich die einzelnen Auslesemodi separat betrachten. Die Reduktion der gesamten Funktionalität auf nur einen Automaten rührt von der Minimierung der Latenz bei gleichzeitiger möglichst synchroner Abarbeitung her. Die Latenz von Steuersignalen wirkt sich direkt auf die Latenz der Daten aus.

Nach einem asynchronen RESET geht die FSM in einen CLEAR Zyklus. Durch diesen Zustand gesteuert, werden alle Register der Netzwerkschnittstelle zurückgesetzt. Entweder wird dieser Zustand wieder nach einer erfolgreichen Auslese, durch ein Software-seitiges Zurücksetzen mittels des Registers **NMOD.c** oder einen globalen Reset erreicht. Danach

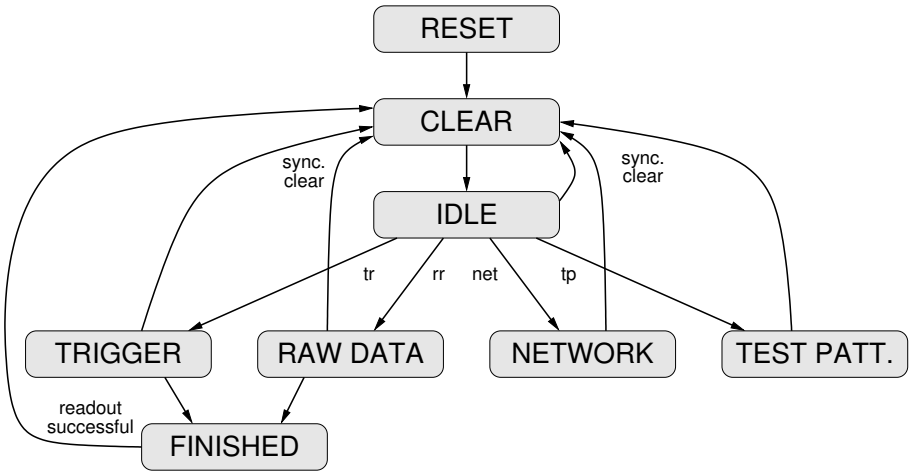

Abbildung 6.9: Zustandsdiagramm der Netzwerkschnittstelle.

wird im IDLE Zustand auf die Auswahl des Auslesemodus gewartet. Dieser kann entweder über zwei spezielle Signale durch die Master-State-Machine erfolgen, die den gesamten Ablauf nach einem Trigger steuert, oder wiederum durch die Register **NMOD.m** und **NMOD.i** über den Prozessor oder das SCSN beeinflusst werden.

Nach erfolgreicher Trigger- oder Rohdatenauslese wird der Zustand FINISHED durchlaufen, innerhalb dessen durch ein spezielles Signal der MSM dem Prozessor das Ende der entsprechenden Auslese mitgeteilt wird. Der Prozessor kann dann z. B. durch einen Interrupt wieder aufgeweckt und mit einem Takt versorgt werden. Der Netzwerkmodus NETWORK und der Testmustergenerator TEST PATT. sind lediglich durch ein Hardware- oder Softwareseitiges Zurücksetzen zu beenden.

6.3.1 Triggerauslese

Die Hauptaufgabe für den Teil des Zustandsautomaten, der die Auslesen kontrolliert, ist es, die programmierbare Auslesereihenfolge zu steuern. Die Reihenfolge für die Triggerauslese kann durch das Register **NTRO** definiert werden. Im Zustand TRIGGER wird mittels **NTRO.f** die erste Quelle für die Daten ermittelt.

Um den Ablauf der Auslese anhand des Zustandsdiagramms (Abbildung 6.10) zu erläutern, sollen als Beispiel zunächst die Daten von Port 0, dann die eigenen, und zuletzt die von Port 2 gesendet werden[4].

Im Zustand PORT/FIFO 0 werden zunächst die einkommenden Daten direkt an den Ausgang weitergeleitet. Dies geschieht so lang, bis entweder eine Endsignatur erkannt wird oder die Anzahl der Datenwörter eine einstellbare maximale Anzahl überschritten hat. Die Endsignatur ist ein durch das Register **NES.t** definierter 16-Bit-Kode. Hierbei werden nur die unteren zwei Bytes eines 32-Bit-Wortes (Spursegments) betrachtet. Dieses sind auch die

[4]Dies entspricht einer Konfiguration von **NTRO**: 0x00017FE0

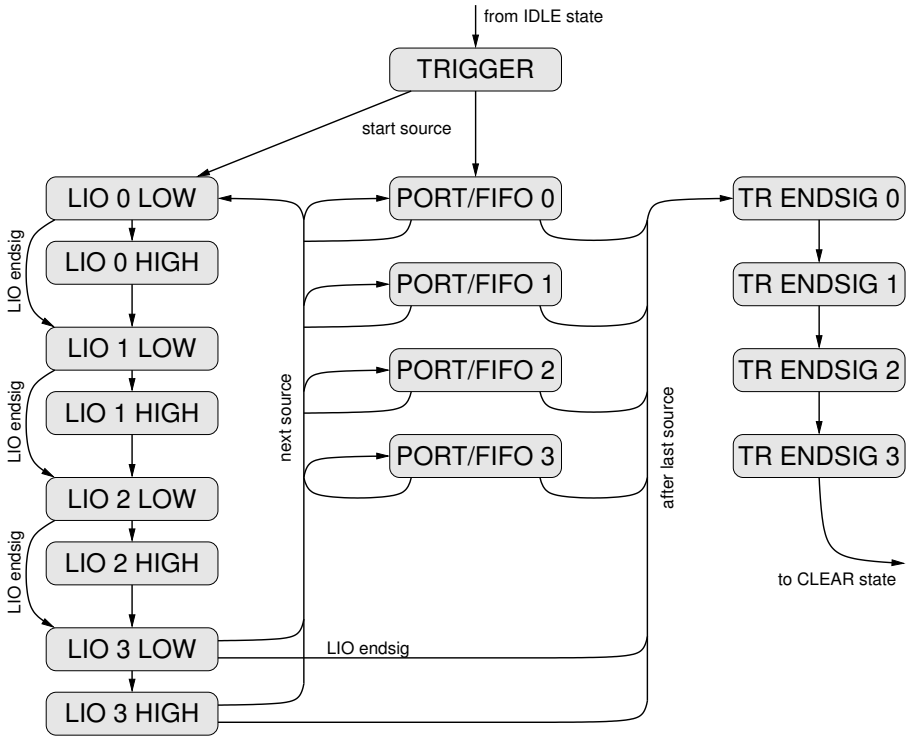

Abbildung 6.10: Zustandsdiagramm für die Triggerauslese

beiden Bytes, die als erstes übertragen werden. Die maximal zulässige Anzahl an akzeptierten Wörtern kann für jeden Port separat durch das Register **NCUT** festgelegt werden.

Danach wird erkannt, dass die eigenen Daten gesendet werden sollen. Diese bis zu vier Spursegmente legt der Prozessor, bevor er abgeschaltet wird, in die vier Datenregister der lokalen Schnittstellen. Wenn eine der CPUs kein Spursegment gefunden hat, kennzeichnet sie dies, indem sie eine Endsignatur in das Datenregister schreibt. Die Netzwerkschnittstelle wartet jeweils so lange, bis alle CPUs ihre Daten geschrieben haben. Da in diesem Beispiel zuerst die Daten von Port 0 gesendet werden, ist es wahrscheinlich, dass alle CPUs die Datenregister gefüllt haben und der Prozessor schon ausgeschaltet ist. Der Zustandsautomat geht nun in einer festgelegten Reihenfolge alle lokalen Schnittstellen durch und sendet die Daten, zuerst die unteren zwei Bytes, dann die oberen. Wird bei den unteren zwei Bytes die Endsignatur erkannt, wird sofort zur nächsten Schnittstelle gewechselt. Diese Übergänge sind in der Abbildung durch LIO endsig gekennzeichnet.

Nach der dritten lokalen Schnittstelle wird wieder das Register der Auslesereihenfolge überprüft und es werden nun die Daten von Port 2 gesendet. Diese sind im FIFO 2 als 16 Bit breite Wörter inklusive der Endsignatur zwischengespeichert und werden von dort direkt auf den Ausgang gegeben.

Die Netzwerkschnittstelle sendet danach selbst die Endsignatur, signalisiert über ein spezielles Signal das Ende der Auslese und kehrt in den CLEAR Zustand zurück. Da es möglich

ist, durch ein mangelndes Erkennen der Endsignatur, die gesamte Auslese zum Erliegen zu bringen, wird diese sicherheitshalber viermal gesendet. Damit ist ausgeschlossen, dass durch ein empfangsseitiges Vertauschen der höherwertigen und niederwertigen Bytes die Signatur übersehen wird. Auf die Geschwindigkeit der Auslese hat dies keinen Einfluss, da sofort nach dem Erkennen einer Endsignatur der Datenstrom abgebrochen wird.

6.3.2 Rohdatenauslese

Wie auch in Abbildung 5.6 zu sehen ist, werden bei der Rohdatenauslese im Gegensatz zur Triggerauslese die eigenen Daten in den FIFOs abgelegt. Dies bedeutet, dass die Daten der Ports nur direkt an den Ausgang gelegt werden können. Das sendende MCM muss also wissen, ob das empfangende MCM bereit ist, die Daten weiterzuleiten. Über ein Kontrollsignal wird dem vorhergehenden MCM mitgeteilt, dass es seine Daten senden kann. Abgesehen davon verläuft die Auslese analog zu der Triggerauslese.

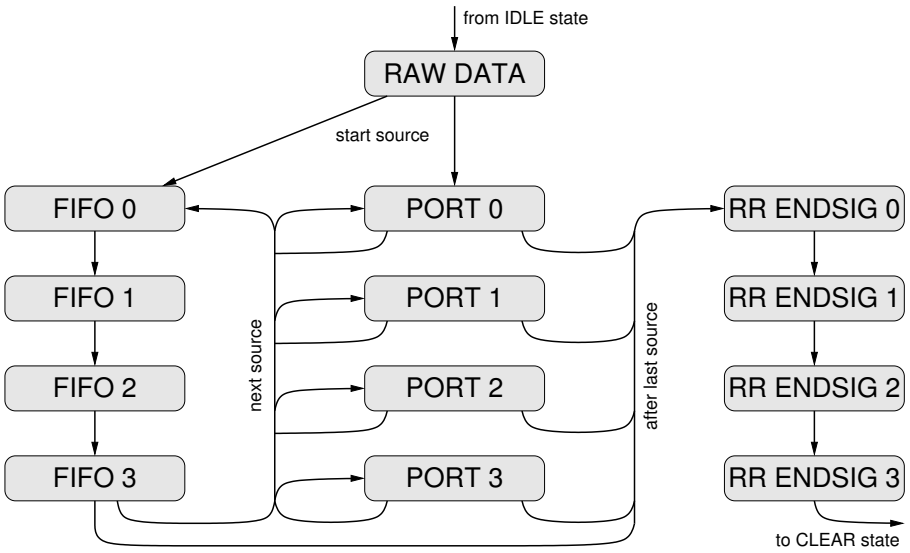

Abbildung 6.11: Zustandsdiagramm für die Rohdatenauslese

Zunächst wartet der Automat im Zustand RAW DATA bis über das eingehende Kontrollsignal am Ausgangsport mitgeteilt wird, dass das MCM jetzt Daten senden darf. Dieses Kontrollsignal wird von der Master-State-Machine empfangen, verarbeitet und an die Netzwerkschnittstelle geleitet. Die Master-State-Machine schaltet vorher die LVDS-Zellen für die Datensignale ein und versorgt die NI mit einem Takt. Natürlich muss das MCM, das die Wurzel des Auslesebaums bildet, selbständig mit der Abarbeitung beginnen. Dies erfolgt durch das Setzen des Bits **NED.r**.

Die Reihenfolge für die Rohdatenauslese wird über das Register **NRRO** festgelegt. Ist das Register auf denselben Wert gesetzt wie im Beispiel von Kapitel 6.3.1, wird wiederum über den Kontrollausgang dem MCM, das mit Port 0 verbunden ist, mitgeteilt, dass es Daten

senden kann. Jetzt werden alle Daten von Port 0 direkt an den Ausgang gelegt, bis die Endsignatur für die Rohdatenauslese (Register **NES**.**r**) erkannt wird.

Darauf werden wieder in einer festen Reihenfolge die eigenen Daten gesendet, diesmal aus den FIFOs. Hier wird das Ende wieder mit der Signatur gekennzeichnet. Das ermöglicht, dass Daten auch während des Sendens noch von dem Prozessor nachgeladen werden können. Wären mehr Daten zu senden als die FIFOs aufnehmen können, so könnten die CPUs die FIFOs füllen, sich danach abschalten und durch einen Interrupt wieder aktiv werden. Das Interruptsignal ist unter Anderem mit dem Statussignal für einen FIFO Fehler (**NLF**.**E**) und dem Signal, das anzeigt, dass die Anzahl der Einträge unter der programmierbaren Schwelle (**NLF**.**L**) liegt, gekoppelt.

Nach dem Weiterleiten der Daten von Port 2 wird auch hier mehrmals die Endsignatur gesendet und das Ende der Auslese der MSM mitgeteilt.

6.3.3 Netzwerkmodus

Über das Register **NMOD**.**m** kann die Schnittstelle in den Netzwerkmodus geschaltet werden. Dieser Modus wird für den normalen Betrieb der Auslesen nicht benötigt. Er ist jedoch auch für den Betrieb im Experiment von großem Nutzen, da hiermit beliebige Testmuster gesendet und ausgewertet und somit fehlerhafte Verbindungen aufgespürt werden können. Die Prozessoren können dann selbständig die Netzwerkschnittstelle so programmieren, dass die mangelhaften Bitleitungen ausgeblendet werden.

Im Netzwerkmodus werden die einkommenden Daten der Ports 0–3 in den FIFOs gespeichert und der Ausgang der FIFOs mit den lokalen Schnittstellen verbunden. Der Kontrollausgang eines Eingangsports kann entweder von der entsprechenden CPU gesetzt werden oder es wird das Signal **NLF**.**H** benutzt, das anzeigt, dass die Anzahl der Wörter im FIFO über der programmierbaren Schwelle liegt. Das sendende MCM kann dann darauf reagieren und die Übertragung unterbrechen.

Die CPUs lesen von ihrem entsprechenden FIFO indem sie das Datenwort von der lokalen Adresse 0x1 lesen. Das Schreiben erfolgt weiterhin über Adressen 0x0. Da die FIFOs im Gegensatz zu den lokalen Schnittstellen nur 16 Bit breit sind und erst nach dem Lesen einer CPU zwei neue Wörter aus dem FIFO genommen und in das lokale Datenregister übertragen werden, ist nur jeden dritten Takt ein Lesevorgang möglich. Die Gültigkeit eines gelesenen Datums kann mittels des FIFO Fehlersignals, über das auch Interrupts ausgelöst werden können, überprüft werden.

Schreibt eine CPU auf die lokale Schnittstelle, werden die Daten direkt an den Ausgang übertragen. Hierbei entstehen keine weiteren Verzögerungen und es kann jeden zweiten Takt ein 32-Bit-Wort geschrieben werden. Welche CPU mit dem Ausgang verbunden ist, wird über das Register **NTRO**.**f** ausgewählt. Im Gegensatz zu der Bedeutung dieses Registers für die Triggerauslesereihenfolge entspricht hier der Wert der CPU Nummer.

6.3.4 Testmustermodus

Über das Register **NTP** kann ein 32-Bit-Wort eingestellt werden, das im Testmustermodus kontinuierlich ausgegeben wird. Hiermit kann auf einfache Weise nach der Fertigung der Chips, dem Bonden oder Verlöten getestet werden (Kapitel 7.1), ob alle Verbindungen funktionieren. Da die Signale auch hierbei die programmierbare Verzögerungseinheit durchlaufen, lassen sich auch sehr gut Tests bezüglich der Signallaufzeiten machen. Der Testmustermodus wird durch das Setzen eines einzigen globalen Registers (**NMOD.m**) aktiviert und bedarf keiner weiteren Konfiguration.

7 Test & Ergebnisse

Zur Inbetriebnahme und zum Testen der TRAP Prototypen wurde zunächst eine Prototypplatine mit einem speziell für das BGA des MCMs angefertigten Nullkraftsockel entwickelt. Diese Platine besteht im Wesentlichen aus diesem Sockel, Spannungsregulatoren und Konnektoren für das SCSN, die PASA-Eingänge und die Auslese. Die Ein- und Ausgänge werden dabei von einer PCI-FPGA-Karte bedient. Mit diesem Aufbau ist es möglich, über einen PC bzw. das SCSN die gesamte Funktionalität auch unter Echtzeitbedingungen zu testen. Einer der vier Netzwerkausgänge ist darüber hinaus mit einem speziellen Konnektor für einen hochauflösenden Logikanalysator ausgestattet. Hiermit wurden die in Kapitel 7.3 vorgestellten Messergebnisse aufgenommen.

Für umfangreiche Tests unter praktisch realen Bedingungen wird eine Testplatine benutzt, die im folgenden Kapitel vorgestellt wird. Diese Platine ist so konzipiert, dass sie sowohl ein einzelnes MCM zum Testen aufnehmen als auch für Tests in der Produktion eingesetzt werden kann.

Für die Validierung der Netzwerkschnittstelle, die den Auslesebaum bildet, lassen sich zwar Fehlfunktionen oder Produktionsfehler mit der Testplatine finden, aber es ersetzt nicht die Erfahrungen mit Prototypen der Ausleseplatinen bzw. mit bestückten Modulen in einer Strahlzeit wie sie in Kapitel 7.3 beschrieben sind. Hierbei wurde auch der vollständige Auslesebaum getestet.

7.1 MCM-Tester

Der MCM-Tester [27] besteht aus einer Platine, die die Testumgebung bildet und einer Konstruktion, die automatisierte Tests für die Produktion erlaubt. Die Platine ist so konzipiert, dass sie über einen Nullkraftsockel einzelne MCMs aufnehmen kann oder mithilfe eines Adapters die Möglichkeit besteht, eine Reihe von MCMs sequentiell zu testen.

7.1.1 Testplatine

Um ein MCM in einer möglichst realistischen Umgebung zu testen, wurde eine spezielle Testplatine entwickelt. Wie in Abbildung 7.1 zu sehen, kann das zu testende MCM durch einen dafür entwickelten Nullkraftsockel eingefügt werden. Über einen FPGA (rechts) findet die Kommunikation mit einem PC statt. Mit diesem lassen sich die Tests überwachen und die Ergebnisse abfragen. Die Tests selbst können autonom ablaufen. Der FPGA übernimmt außerdem die digitale Ansteuerung der Baugruppen, d. h. es werden über separate

SCSNs das zu testende MCM, die vier fest installierten MCMs und das andere FPGA konfiguriert.

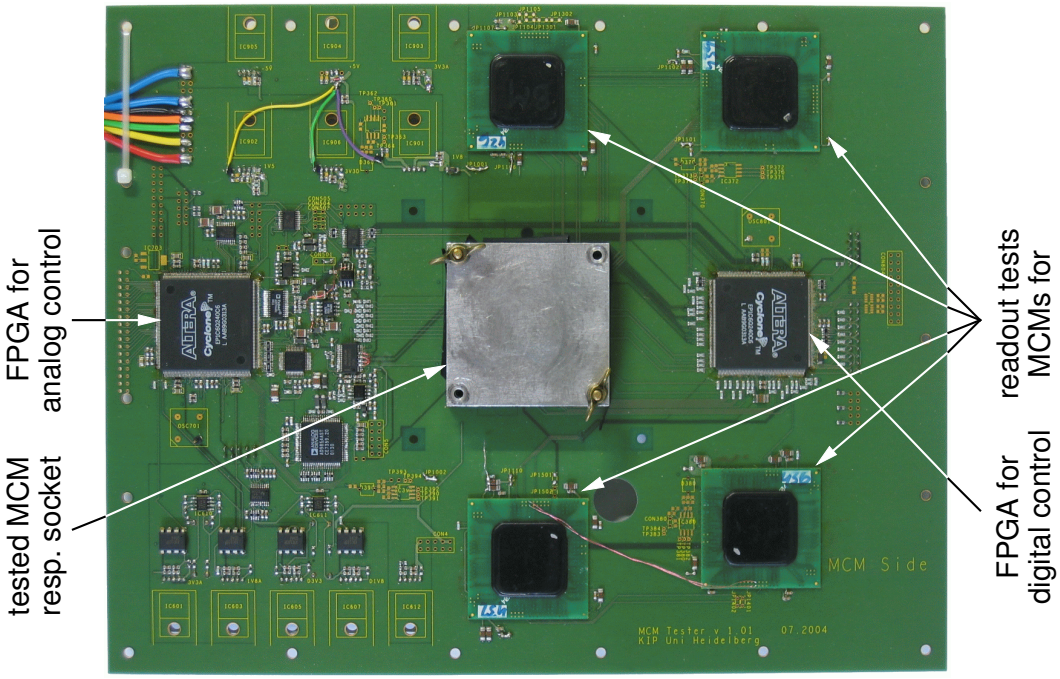

Abbildung 7.1: Der Nullkraftsockel in der Mitte der Testplatine kann entweder einen MCM oder einen Adapter für die automatisierten Tests aufnehmen. Über die beiden FPGAs werden die Tests initiiert und überwacht. Die vier zusätzlichen MCM fungieren als Datenquelle für die Auslesetests.

Der FPGA auf der linken Seite kontrolliert die analogen Signale. Sowohl die Referenzspannungen als auch die Versorgungsspannungen sind regelbar und können auch gemessen werden. Mittels der ADCs lassen sich darüber hinaus die aufgenommenen Ströme des MCMs überwachen. Über in Serie geschaltete Kapazitäten lassen sich bis zu 32 verschieden starke Ladungspulse erzeugen, mit denen der Eingang des PASAs des MCMs getestet wird.

Um die Netzwerkschnittstelle bzw. die Auslese zu testen ist die Testplatine mit vier MCMs bestückt, die allein als Datenquelle dienen. Hiermit ist es möglich das MCM mit Echtzeittests in einer Umgebung zu prüfen, die der Einsatzumgebung sehr nahe kommt. Der Ausgangsport ist verbunden mit dem rechten FPGA, das dann die empfangenen Werte mit den erwarteten vergleicht.

7.1.2 Automatisierte Tests

Am Institut für Prozessdatenverarbeitung und Elektronik (IPE) am Forschungszentrum Karlsruhe[1] wurde eine Apparatur entwickelt, mit der es möglich ist, eine größere Zahl an MCMs automatisiert zu testen. Der Tester (Abbildung 7.2, links) kann eine Platine aus 16

[1] http://www.ipe.fzk.de

MCMs (rechts) aufnehmen und die MCMs sequentiell testen. Diese Platine wird nach den Tests geschnitten, um die einzelnen MCMs zu erhalten.

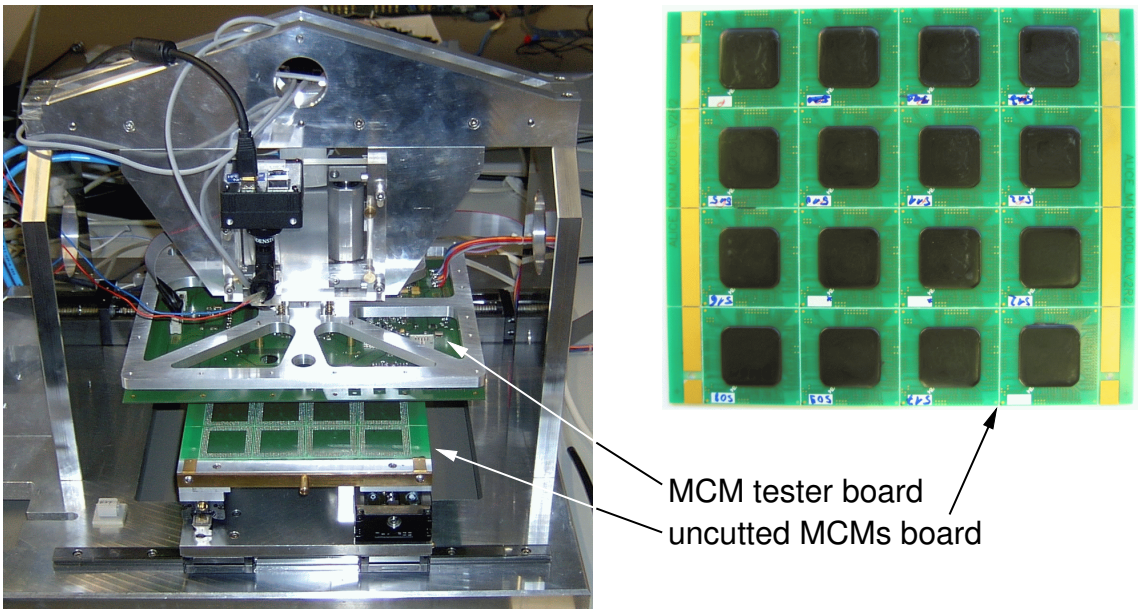

Abbildung 7.2: Der MCM-Tester trägt die Testplatine (Abbildung 7.1), ausgestattet mit einer Nadelkarte als Adapter. Die noch ungeschnittene Platine mit 16 MCMs wird mit den Anschlüssen nach oben eingelegt. Kameragesteuert werden die MCMs unter der Nadelkarte positioniert und sequentiell getestet.

In den Tester integriert ist die zuvor beschriebene Testplatine. Anstatt des Nullkraftsockels wird hierbei ein Adapter mit einer Nadelkarte benutzt. Der Tester positioniert, von einer Kamera gesteuert, die Nadelkarte über der Platine mit den MCMs, kontaktiert dann einen der MCMs und kann ein Testprogramm ablaufen lassen. Die Testabläufe und die Umgebung dafür wurden im Rahmen einer Diplomarbeit [27] entwickelt.

Zunächst werden externe Tests an dem MCM vorgenommen. Mit ADCs werden die Stromaufnahmen gemessen, wodurch Kurzschlüsse in der Versorgungsspannung aufgespürt werden können. Werden hierbei z. B. nur Störungen in der analogen Versorgung festgestellt, können die entsprechenden Spannungsregulatoren abgeschaltet und die Tests fortgesetzt werden, da das MCM durchaus noch genutzt werden kann, z. B. als eines derer, die ausschließlich Daten sammeln und weiterleiten. Das SCSN kann einfach getestet werden, indem der SCSN Master, also eines der FPGAs, Kommandopakete verschickt. Im SCSN kommen alle Pakete, wenn auch durch die Targets modifiziert, wieder beim Master an und können analysiert werden. Bevor die Auslese durch die vier sendenden MCMs getestet wird, können über JTAG[2] die LVDS-Zellen und die Bond-Verbindungen aller Ein- und Ausgänge überprüft werden. Genau wie der Ausgangsport der Netzwerkschnittstelle sind auch die übrigen Kontrollsignale, wie z. B. die Taktausgänge oder das Pretriggersignal, an einen FPGA angeschlossen, der die Funktionalität testet.

[2] Joint Test Action Group, IEEE 1149.1

Zum Testen der internen Funktionalität des TRAPs werden über SCSN verschiedene Programme[3] in den Instruktionsspeicher des Prozessors geschrieben und dann ausgeführt. Um den Instruktionsspeicher selbst zu überprüfen werden die Programme über SCSN wieder ausgelesen und verglichen. Außerdem führt jede CPU Schreib- und Lesetests auf dem Speicher einer anderen CPU durch. Den gemeinsamen Datenspeicher überprüft jede CPU ebenfalls durch Schreiben und Lesen von Testmustern (mit und ohne Hamming Korrektur). Hierbei ist vor allem das simultane Lesen aller vier CPUs kritisch. Durch diese Art von Tests können auch die Konstantenregister, die Ereignisspeicher, die globalen und lokalen Register und die Look-up Tabellen geprüft werden. Die ALU-Operationen, die Interrupts und die Filter werden durch dafür optimierte Programme möglichst abdeckend getestet. Die ADCs und der PASA können durch das Auswerten der ADC-Werte bei Anlegen von Referenzspannungen und Ladungspulsen überprüft werden.

Die Tests der Netzwerkschnittstelle und der Auslese mithilfe der Testplatine und des MCM-Testers sind in Kapitel 7.3.2 näher beschrieben.

7.2 Wafer-Tester

Bei einer Produktion von etwa 100 000 TRAPs liegt die Herausforderung darin, möglichst frühzeitig die Chips zu testen und defekte Chips auszusortieren. Die Herstellung der MCMs ist relativ aufwendig und teuer, und da die Ausbeute an verwertbaren Chips nur bei etwa 73 % liegt, sind diese Wafer-Tests unverzichtbar.

Bei dieser großen Anzahl an TRAPs müssen die Tests vollautomatisiert vorgenommen werden. Abbildung 7.3 zeigt die eingesetzte Platine für den Wafer-Tester. Über Nadeln werden nacheinander Verbindungen zu den TRAP Chips auf dem Wafer hergestellt. Die Chips werden mit einer Reihe von Testroutinen untersucht und entsprechend markiert. Wie in Kapitel 7.3.2 beschrieben, führt ein gefundenes Fehlverhalten nicht zwangläufig dazu, dass der Chip nicht eingesetzt werden kann. Alle interne Funktionalität, also z. B. die Speicher, der Prozessor oder der Präprozessor, kann über die serielle Schnittstelle SCSN getestet werden. Außerdem können Teststimuli an die Eingänge der ADCs gelegt und die Ergebnisse ausgelesen werden. Da ein MCM nicht eingesetzt werden kann, wenn von der Netzwerkschnittstelle nicht mindestens der Ausgangsport funktioniert, wird auch dieser schon hier getestet. Im Folgenden ist die automatisierte Testprozedur aufgelistet.

- Kontaktierung des TRAP Chips.
- Einschalten der Stromversorgung, Auslösen eines Resets und Messung des Stromverbrauchs. Übersteigt der Strom eine bestimmte Grenze, wird der Chip als defekt markiert.
- Messung der Eingangswiderstände der vier LVDS-Zellen für die serielle Schnittstelle SCSN, Test der seriellen Kommunikation und Markierung als defekt, falls keine Verbindung möglich ist.

[3] entwickelt am KIP von Pejo Popov und Markus Gutfleisch.

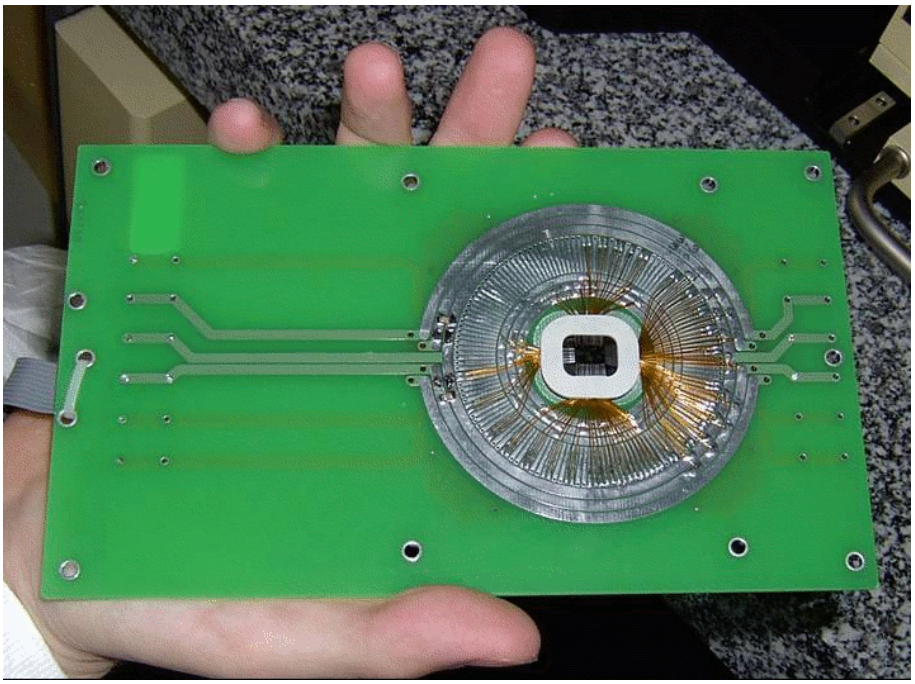

Abbildung 7.3: Das Foto zeigt die Platine für den Wafertester. Über Nadeln werden die TRAP Chips auf dem Wafer kontaktiert und Tests vorgenommen.

- Test des Reset-Pins und des Software-seitigen Resets für das SCSN.
- Überprüfung des Instruktionsspeichers IMEM für die erste CPU durch Schreiben und Zurücklesen eines Programms. Bei einem Fehler wird der Chip jeweils dementsprechend markiert.
- Start des geladenen Programms. Dieses überprüft den Instruktionsspeicher für die nächste CPU und so weiter.
- Laden, überprüfen und starten weiterer Testprogramme. Diese Programme testen alle Module des TRAPs. Funktionieren der Datenspeicher DMEM, die Filter, der Präprozessor oder die ADCs nicht, werden die Tests dennoch fortgeführt, da der Chip für entsprechende andere Aufgaben noch verwendbar ist. Außerdem werden die programmierbaren Konstanten, die Ereignisspeicher und verschiedene Konfigurationsregister getestet.
- Die Speicher können optional mit einer Hamming-Kodierung mit Bitfehlerkorrektur versehen werden. Alle Tests werden dementsprechend mit und ohne Hamming-Korrektur durchgeführt.
- Über den Wafer-Tester werden als Stimuli für die ADCs sinusförmige Signale angelegt, 63 Messungen gemacht, ein Fit an die Wellenform vorgenommen und die errechneten Amplituden überprüft.

Produktion	Wafer	getestet	verwertbar	relativ
Präproduktion	25	14324	5056	35 %
Februar 2006	25	14400	8143	57 %
Juni 2006	49	28006	24018	86 %
September 2006	49	28212	24600	87 %
Januar 2007	3	1728	1568	91 %
Insgesamt	151	86670	63385	73 %

Tabelle 7.1: Ausbeute bei der Waferproduktion.

Die Tabelle 7.1 zeigt für die verschiedenen Produktionsdaten die Ausbeute an verwendbaren TRAP Chips, wie sie mit dem Wafer-Tester ermittelt wurden. Gut zu erkennen ist die Steigerung der relativen Anzahl verwertbarer Chips von der Präproduktion bis zur Produktion im Januar 2007.

7.3 Netzwerkschnittstelle und Auslesenetzwerk

Viele Eigenschaften der Netzwerkschnittstelle lassen sich am besten mit der Prototypplatine testen. Mit dieser hat man direkten Zugriff auf die Ein- und Ausgangsports. Die Eingänge können entweder durch die Ausgänge weiterer Prototypplatinen, FPGAs mit dafür entwickelten Designs oder einen Funktionsgenerator stimuliert werden. Hiermit lassen sich die Resynchronisationseinheiten, die Ausmaskierungen von Bitleitungen und die FIFOs testen. Werden mehrere Prototypplatinen verwendet, kann damit auch ein minimaler Auslesebaum aufgebaut werden und das Zusammenspiel mit der Master-State-Machine getestet werden.

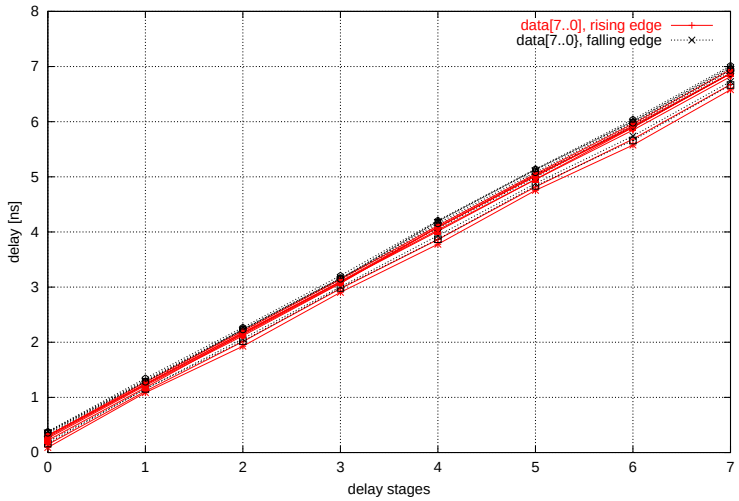

Abbildung 7.4: Das Diagramm zeigt die gemessenen Verzögerungen der Datenleitungen relativ zu der unverzögerten Strobe-Leitung für beide Flanken.

Gerade für die Untersuchung der Verzögerungseinheit des Ausgangsports ist es wichtig, mit einem Logikanalysator und Oszilloskop direkten Zugriff auf diesen zu haben. Für das in Abbildung 7.4 gezeigte Diagramm wurde ein 0x00FF Muster auf den Ausgang gelegt, d. h. der Strobe und die Datenleitungen haben eine Frequenz von 120 MHz. Mit einem Logikanalysator mit einer Auflösung von 125 ps wurden die Verzögerungen der Datenleitungen relativ zum unverzögerten Strobe-Signal gemessen und gemittelt. Es ist zu erkennen, dass sowohl für die steigenden als auch für die fallenden Flanken identische Verzögerungswerte erreicht werden.

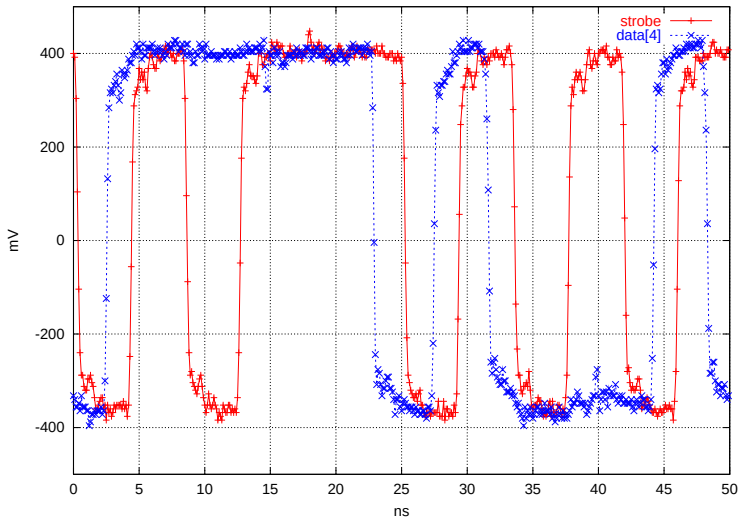

Abbildung 7.5: Gezeigt ist ein Ausschnitt aus dem Datenstrom einer Rohdatenauslese mit 120 MHz. Mithilfe der programmierbaren Verzögerungseinheit wurde die stabile Phase des Datensignals über die Strobe-Flanken positioniert.

Wird die relative Verzögerung so eingestellt, dass das Strobe-Signal seinen Übergang genau in der Mitte des Auges eines Datensignals hat, ergibt sich für ein einzelnes Datensignal eine Wellenform, wie sie in Abbildung 7.5 dargestellt ist. Diese Messung wurde mit demselben Testaufbau wie die von Abbildung 7.4 gemacht.

7.3.1 Simulation

Die Simulation der Netzwerkschnittstelle erfolgt in vier Stufen, bei denen verschieden umfangreich die Umgebungen mitsimuliert werden. Wie in Kapitel C.4.1 beschrieben, existieren Umgebungen, die allein die Schnittstelle, den TRAP inklusive der Schnittstelle, oder eine gesamte Ausleseplatine simulieren. Für die Netzwerkschnittstelle separat existieren keine Informationen zum zeitlichen Verhalten. Es stehen also lediglich die funktionale und eine Standardzellen-Simulation zu Verfügung. Mit diesem Modell lässt sich auch das Verhalten des Auslesebaums bzw. einer Ausleseplatine untersuchen. Die Simulation des TRAPs hingegen kann neben der funktionalen Simulation auch mit den Zeitinformationen aus dem Layout des Chips erfolgen. Diese Art der Simulation ist allerdings sehr rechenaufwendig

und kann durchaus mehrere Stunden dauern. Wie auch in Kapitel 4.2.4 erläutert, wird aus diesem Grund bei der Beschreibung einer Ausleseplatine mit 17 TRAPs entweder rein funktional simuliert oder es werden nur wenige TRAPs durch ein Modell mit zeitlicher Verhaltensbeschreibung ersetzt. Im Folgenden werden Beispiele der funktionalen Simulation der Netzwerkschnittstelle und einer Ausleseplatine durch Zusammenschließen von 17 solcher Schnittstellen vorgestellt.

Anhand der Simulation lässt sich die Funktionsweise der Netzwerkschnittstelle nochmals erkennen. In Abbildung 7.6 ist eine einfache Triggerauslese ohne Protokoll mit der Master-State-Machine gezeigt.

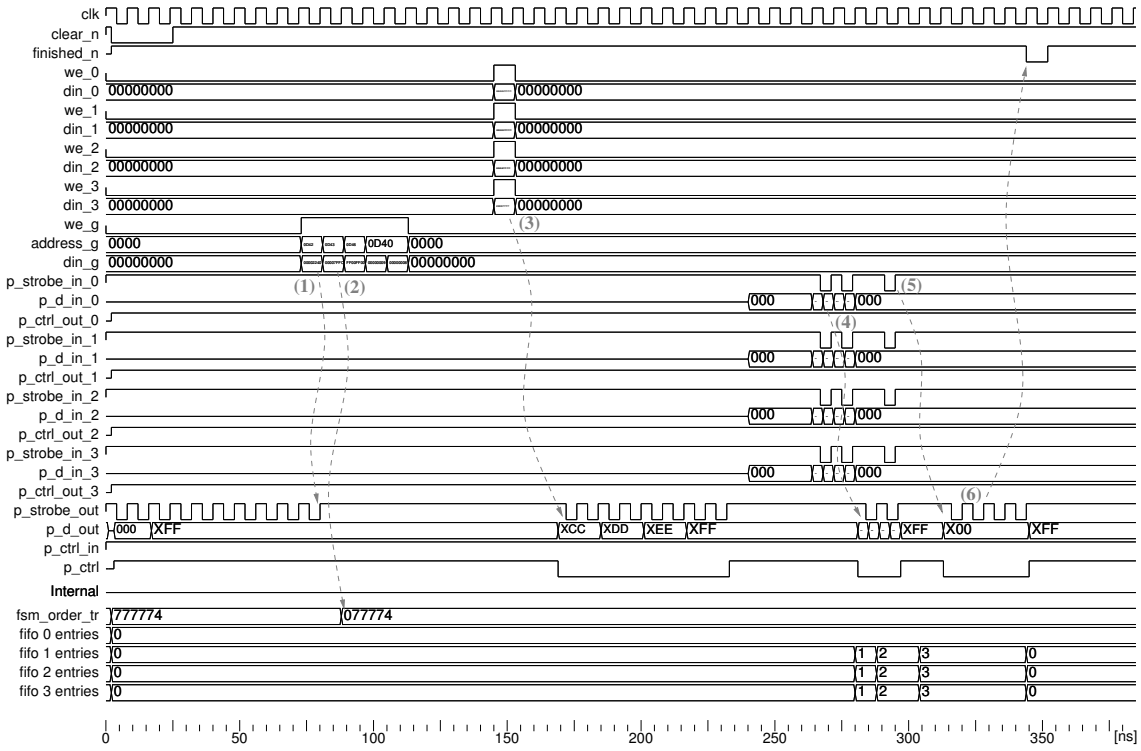

Abbildung 7.6: In diesem Beispiel ist eine minimale Triggerauslese funktional simuliert. Jede CPU sendet ihr Spursegment, danach werden die Daten von Port 0 weitergeleitet.

Zunächst wird über die globale Schnittstelle die Netzwerkschnittstelle konfiguriert. In der Grundeinstellung wird das Ausgangs-Strobe-Signal `p_strobe_out` als Taktsignal genutzt. Über die Konfiguration (1) wird auf den Strobe-Betrieb umgeschaltet. In diesem Beispiel soll die Netzwerkschnittstelle zunächst die eigenen Spursegmente an den Ausgang geben, danach die vom Eingangsport 0 und dann die Endsignatur senden. Durch den Konfigurationszyklus (2) wird diese Auslesereihenfolge `fsm_order_tr` auf den Wert `077774` oktal gesetzt (siehe Register **NTRO**). Danach wird der Trigger-Auslesemodus gestartet. Sobald die CPUs ihre Daten in lokalen Schnittstellen schreiben (3), werden diese in der Reihenfolge CPU 0–3 an den Ausgang gelegt. Jetzt wartet die Netzwerkschnittstelle auf die Daten von Port 0, die dann sofort an den Ausgang weitergeleitet werden. Das erkennt man an dem

Konstanten Wert `0` der `fifo 0 entries`, was bedeutet, dass keine Zwischenspeicherung im FIFO stattfindet. Nach dem Erkennen der Endsignatur (5) von Port 0 sendet die Schnittstelle ebenfalls diese Signatur und meldet der Master-State-Machine, dass die Übertragung abgeschlossen ist (6).

Bei dem Beispiel der Rohdatenauslese (Abbildung 7.7) sollen wiederum zunächst die eigenen Daten, die über die CPUs geschrieben werden, ausgegeben werden. Danach werden von den MCMs, die an den Eingangsports 0–3 angeschlossen sind, ihre Daten angefordert und weitergeleitet.

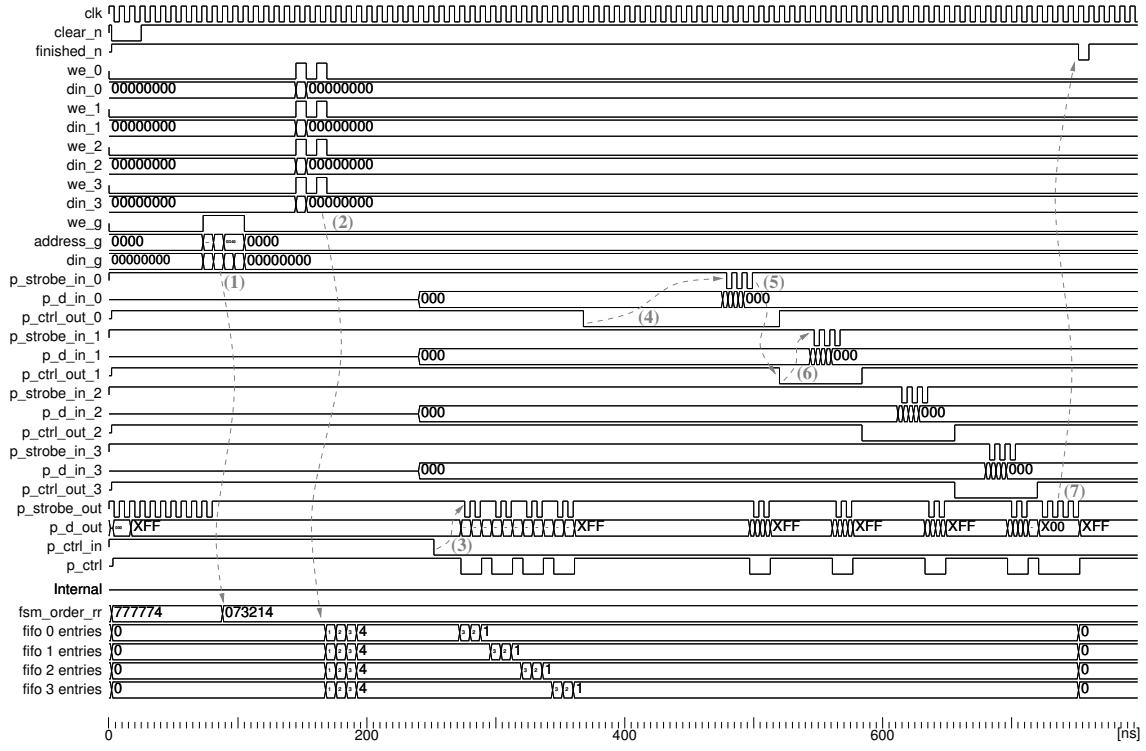

Abbildung 7.7: Nachdem die internen Rohdaten über die CPUs geschrieben sind, werden über die `p_ctrl_out` Signale die Daten der vorangeschalteten MCMs angefordert.

Durch die Konfigurationszyklen (1) wird die Rohdaten-Auslesereihenfolge festgelegt und der Auslesemodus gestartet. Während der Rohdatenauslese schreiben die CPUs ihre Daten in die internen FIFOs der Netzwerkschnittstelle (2) und signalisieren das Ende des Datenstroms mit einer Endsignatur. Ein MCM, das sowohl eigene Daten hat als auch welche weiterleitet, liegt inmitten des Auslesebaums. Entsprechend wartet das MCM auf die Aufforderung über das Signal `p_ctrl_in` (3), bis es beginnt, Daten zu senden. Nach dem Senden der eigenen Daten aus den FIFOs – zu erkennen an der Anzahl der FIFO Einträge `fifo 0-3 entries` – wird dem an am Eingangsport 0 angeschlossenen MCM über das Signal `p_ctrl_out_0` mitgeteilt(4), dass es jetzt seine Daten senden kann. Dieser Datenstrom wird direkt an den Ausgang weitergegeben. Nach dem Erkennen der Endsignatur (5), erfolgt das gleiche Vorgehen mit Port 1 (6). Sind alle Ports abgearbeitet, sendet die

Netzwerkschnittstelle selbst die Endsignatur für die Rohdatenauslese und signalisiert (7) der Master-State-Machine das Ende der Übertragung.

Mit dieser Art Simulationen wurde die Netzwerkschnittstelle entwickelt und separat getestet. Hierbei wurden die Eingänge der Schnittstellen und Ports mithilfe Textdateien stimuliert. In der entsprechenden Simulationsumgebung sind Beispiele für verschiedene Trigger- und Rohdatenauslesen, den Netzwerkmodus und den Testmustermodus zu finden.

Für die folgenden Simulationen wurde durch das Zusammenschalten von 17 Netzwerkschnittstellen eine Ausleseplatine modelliert, so dass sie der in Abbildung 5.3 dargestellten entspricht. Die MCMs fangen hierbei nicht gleichzeitig an zu arbeiten und ihre Daten zu senden, sondern der Startpunkt ist zufällig zwischen 0 und 13 Taktzyklen verzögert. Hiermit wird die unterschiedliche Verarbeitungszeit der Prozessoren modelliert.

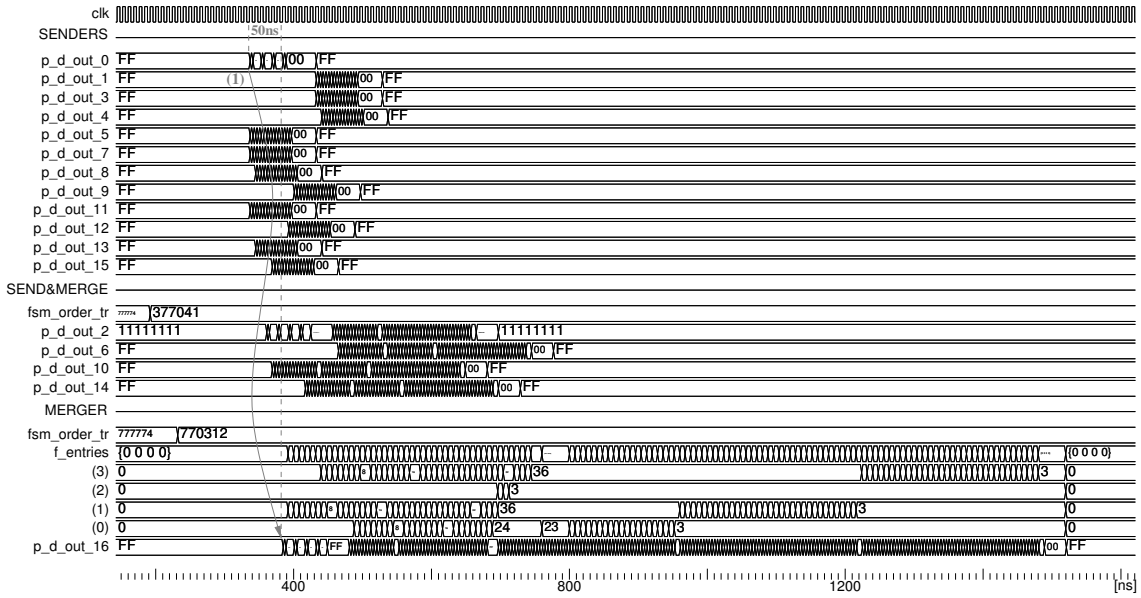

Abbildung 7.8: Alle nur sendenden MCMs übertragen sofort ihre Daten. Vier MCMs sammeln die Daten von jeweils drei anderen und geben sie zusammen mit den eigenen Daten an MCM 16 weiter. Dieses MCM bildet somit einen fast kontinuierlichen Datenstrom.

In Abbildung 7.8 ist die Simulation einer Triggerauslese für eine Ausleseplatine gezeigt. Die nur sendenden MCM beginnen, zufällig verteilt, ihre Daten zu senden. Die MCMs 2, 6, 10 und 14 sind Module, die sowohl eigene Daten senden als auch die Daten von jeweils drei MCMs sammeln und weiterleiten. Die programmierte Auslesereihenfolge `fsm_order_tr` ist hierbei für diese gleich. Damit werden z. B. von MCM 2 die Daten in der Reihenfolge MCM 0→1→2→3 ausgegeben.

Die Auslesereihenfolge des MCM 16, das ausschließlich Daten sammelt und weiterleitet, ist so programmiert, dass die Daten aller MCMs in Reihenfolge ihrer Nummerierung ausgegeben werden. Die Auslesereihenfolge `fsm_order_tr` ist also so, dass die Eingangsports in der Folge 2→0→1→3 abgearbeitet werden, was einer Reihenfolge der MCMs 2→6→10→14 entspricht. Auch anhand der Anzahl der FIFO Einträge `f_entries` lässt sich erkennen, dass

die Daten des Eingangsports 2 zuerst bearbeitet werden, da der Datenstrom nicht zwischengespeichert, sondern direkt an den Ausgang weitergeleitet wird. Die Restanzahl von drei Einträgen in den FIFOs rührt daher, dass jedes MCM die Endsignatur viermal sendet (Kapitel 6.3.1), bei Auslesen der FIFOs jedoch nach dem Erkennen der ersten abgebrochen wird. Jedes MCM sendet in diesem Beispiel die maximale Anzahl von vier Spursegmenten. Das bedeutet, dass dies inklusive der Endsignaturen $18 \cdot 32$-Bit-Wörter sind, die an jedem Eingang von MCM 16 eintreffen. Dies entspricht 36 FIFO Einträgen. Beim Bearbeiten des Datenstroms von Eingangsport 0 sieht man, dass ab etwa 700 ns die Daten gleichzeitig in das FIFO eingetragen und ausgelesen werden.

Die Latenz für die Ausleseplatine bestimmt sich durch die Zeit vom Beginn des Sendens des MCMs, das zuerst ausgelesen werden soll, bis zum Erscheinen der Daten am Ausgang der Platine. Dies ist die durch (1) markierte Zeitspanne von etwa 50 ns, was einer Latenz von zwei Taktzyklen pro MCM entspricht. Am Ausgang von MCM 16 kann man erkennen, dass schon kurz nach dem Beginn der Auslese ein nahezu kontinuierlicher Datenstrom erzeugt wird, der die Bandbreite optimal nutzt.

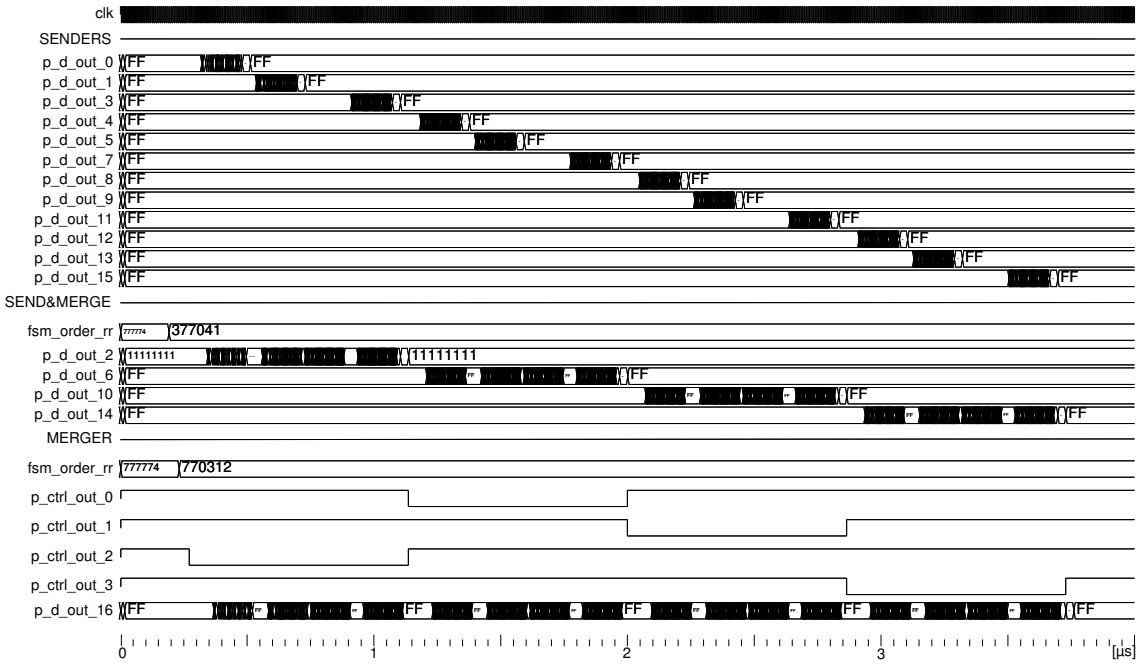

Abbildung 7.9: Die Wurzel des Auslesebaums MCM 16 fordert über das Signal `p_ctrl_out_2` zunächst die Daten von MCM 2 an. Dieses wiederum macht eine Anforderung an MCM 0, das die Daten sendet, die dann als erstes am Ausgang von MCM 16 erscheinen.

Abbildung 7.9 zeigt eine Rohdatenauslese einer Ausleseplatine, ebenfalls entsprechend Abbildung 5.3, mit der gleichen Auslesereihenfolge `fsm_order_tr` wie die der Triggerauslese. Auch senden die Prozessoren der MCMs mit derselben zeitlichen Verteilung ihre Daten. Außer bei MCM 0 werden diese Daten zunächst in den eigenen FIFOs der Netzwerkschnittstelle zwischengespeichert, so dass sich die Prozessoren schon nach etwa 600 ns ausschalten können und die Auslese autonom abläuft. Anhand der Anforderungssignale `p_ctrl_out_0`

bis 3 des MCM 16 kann man sehen, dass nach und nach die Daten der MCMs 2→6→10→14 abgeholt werden. Diese verfahren in gleicher Weise mit jeweils weiteren drei angeschlossenen MCMs. Auch hier lässt sich die relativ kurze Latenz der Auslese erkennen, jedoch wird durch das Handshake-Protokoll und die lokale Zwischenspeicherung die Bandbreite nicht voll ausgenutzt. Es entstehen Sendepausen zwischen den einzelnen Datenpaketen.

Wie in Kapitel 4.2.4 beschrieben, ist es nicht möglich eine mit 17 MCMs vollständig bestückte Ausleseplatine mit zurückannotierter Gatternetzliste zu simulieren. Um dennoch das Zusammenspiel des Prozessors, der Master-State-Machine und der Netzwerkschnittstelle mit zeitlichem Verhalten zu untersuchen, sind hierfür drei MCMs in unterschiedlichen Betriebsmodi zusammengeschaltet.

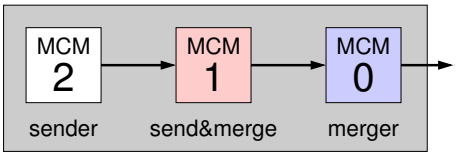

Abbildung 7.10: Für eine Simulation der Gatternetzliste mit zeitlichem Verhalten ist die Anzahl der aktiven MCMs auf drei reduziert. Zur Untersuchung der Netzwerkschnittstelle arbeiten sie in den drei unterschiedlichen Positionen, wie es sie auf einer vollständig bestückten Ausleseplatine gibt.

Die in Abbildung 7.10 gezeigten MCMs übernehmen jeweils die Position innerhalb einer Ausleseplatine, wie sie auch in Abbildung 5.3 farblich identisch markiert ist. MCM 2 sendet ausschließlich seine eigenen Daten. MCM 1 sammelt die Daten von MCM 2 und sendet sie zusammen mit seinen eigenen Daten. MCM 0 leitet lediglich den Datenstrom weiter. In dieser Simulation wird – gesteuert von der Master-State-Machine – ein vollständiger Trigger-Zyklus durchlaufen. Abbildung 7.11 zeigt die Wellenform eines solchen Zyklus, der im Folgenden näher beschrieben wird.

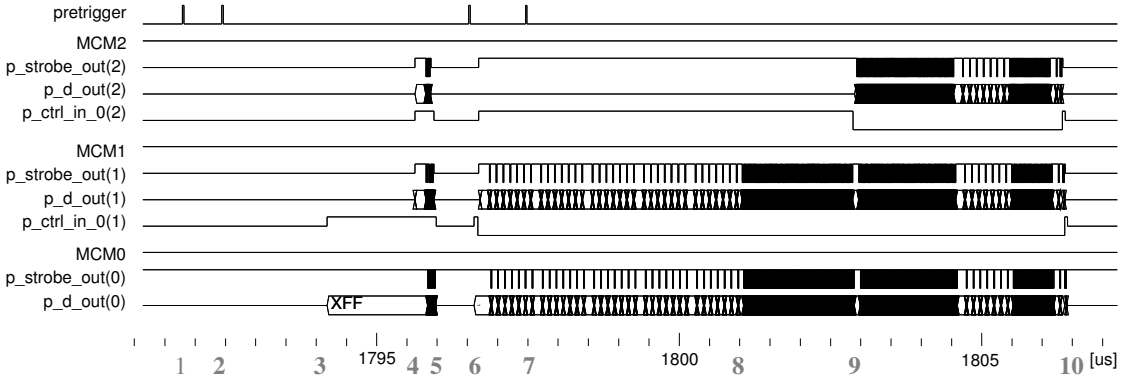

Abbildung 7.11: Es ist die Wellenform einer Simulation mit zeitlichem Verhalten von den in Abbildung 7.10 gezeigten MCMs dargestellt. Ausgelöst durch den Prätrigger und bestätigt durch den Level-0-Trigger erfolgt bei 1796 µs die Triggerauslese. Nach erfolgreichem Level-1-Trigger werden die Rohdaten von MCM 1, dann von MCM 2 ausgegeben.

Da zu Beginn der Simulation zunächst die MCMs über das SCSN konfiguriert werden, beginnt in dieser Simulation die aktive Phase erst nach etwa 1790 µs. Durch die grauen

Zahlen unterhalb der Zeitskala sind die interessanten Zeitpunkte für den Ablauf der Verarbeitung markiert. Nach dem Prätrigger zum Zeitpunkt (1) beginnt der Präprozessor die Daten aufzunehmen. Durch den Level-0-Trigger (2) innerhalb von 1,2 µs wird signalisiert, dass der Ablauf nicht abgebrochen werden soll. Nach dem Ende der Driftzeit (3), 2 µs nach dem Prätrigger, beginnen die Prozessoren die Spursegmente zu berechnen. Da MCM 0 so konfiguriert ist, dass es keine eigenen Daten verarbeitet, sind schon kurz darauf die Ein- und Ausgänge aktiv. Zum Zeitpunkt (4) sind die anderen beiden MCMs mit den Berechnungen fertig und die Ports der Netzwerkschnittstelle werden eingeschaltet. Nach der Triggerauslese (5) wird durch den Level-1-Trigger (6) die Rohdatenauslese eingeleitet. Der Level-2-Trigger (7) bestätigt den vorherigen Trigger und verhindert den Abbruch. Nach der Anforderung von MCM 0 über das Signal p_ctrl_in_0(1), sendet MCM 1 zunächst seine eigenen Daten. Es ist gut zu erkennen, dass bis zum Zeitpunkt (8) eine der CPUs direkt auf den Ausgang schreibt, während die Daten der übrigen drei CPUs in den entsprechenden FIFOs gepuffert werden. Diese Daten werden danach (8) mit voller Bandbreite übertragen. MCM 1 fordert darauf über p_ctrl_in_0(2) die Daten von MCM 2 an und leitet diese weiter. Zum Zeitpunkt (10) ist die Rohdatenauslese beendet und die Ports werden abgeschaltet.

7.3.2 MCM-Tester

Bei den Tests mit der Testplatine werden die vom MCM gesendeten Daten mit dem FPGA (Abbildung 7.1, rechts) aufgenommen und zwischengespeichert. Entweder können diese Werte mit Sollwerten verglichen oder über das SCSN ausgelesen werden. Die Synchronisierung der Daten erfolgt dabei in gleicher Weise, wie es bei einem Eingangsport der Netzwerkschnittstelle der Fall ist. Damit kann auch das zeitliche Verhalten, wie z. B. bei Änderung der Ausgangsverzögerungen, untersucht werden.

Wie in Kapitel 5.2.2 beschrieben, übernimmt ein MCM für die Auslese verschiedene Aufgaben. Es gibt MCMs die nur ihre eigenen Daten senden, solche, die Daten sammeln und weiterleiten, und solche, die beide Aufgaben erfüllen. Im Gegensatz zu den MCMs, die allein zum Aufbau des Auslesebaums fungieren, werden bei den übrigen entweder nur drei Eingangsports oder gar keiner genutzt. Es ist also möglich auch fehlerhafte MCMs für das Experiment einzusetzen. Die Tests sind deshalb auch daraufhin ausgelegt, die Ausbeute möglichst hoch zu halten.

Die Netzwerkschnittstelle besitzt einen Testmodus, in dem ein kontinuierlicher Datenstrom mit programmierbaren Wort erzeugt wird. Mit diesem ersten Test können zunächst die Verbindungen und die Funktion der LVDS-Zellen geprüft werden. Ein Durchfahren aller einstellbaren Verzögerungen der Ausgangssignale ergibt ein Fenster, in dem die Daten noch gültig sind. Dieses Fenster wird für jedes MCM gespeichert und kann später bei der Anpassung an die Umgebung im Experiment herangezogen werden. Im Testmodus werden außerdem die Einstellmöglichkeiten für die Bitpositionen des Reservebits und des Paritätbits getestet. Ist z. B. nur eine Verbindung fehlerhaft und diese Leitung kann ausgeblendet werden, kann dieses MCM durchaus noch genutzt werden. Es ist wichtig diese Ergebnisse zu speichern, da natürlich darauf zu achten ist, dass bei einem empfangenden MCM an dem entsprechenden Eingangsport entweder keine oder die gleiche Leitung defekt ist.

Im nächsten Test arbeitet das MCM allein als Datenquelle. Hierzu werden Testdaten in der Triggerauslese gesendet und durch das FPGA kontrolliert. Noch wichtiger ist die Überprüfung der Rohdatenauslese. Hierbei werden die internen FIFOs benutzt, deren Funktion dabei getestet wird. Außerdem spielt bei der Rohdatenauslese der Kontrolleingang des Ausgangsports für das Netzwerkprotokoll eine Rolle. Hierbei wird dieses Kontrollsignal wiederum vom FPGA erzeugt. Arbeitet in diesem Test ein MCM fehlerfrei, kann es zumindest als eines derer genutzt werden, die allein ihre eigenen Daten senden.

Für die Tests der datensammelnden MCMs werden die vier weiteren MCMs auf der Testplatine verwendet. Die Eingangsports werden wiederum daraufhin untersucht, ob die Ausmaskierung der Leitungen funktioniert. Außerdem enthalten die Eingänge die zeitkritische Datenresynchronisation. Bei diesen MCM ist besonders darauf zu achten, dass alle Datenleitungen einwandfrei funktionieren, damit ein eventueller Fehler im Betrieb des Experiments ausgeglichen werden kann. Bei einem Ausfall des Ausgangsports eines datensammelnden MCMs ist z. B. der gesamte Baum oberhalb des MCMs abgeschnitten. Mit diesem Test können außerdem ein vollständiger Triggerzyklus getestet werden – inklusive Prätrigger, Triggerauslese und Rohdatenauslese – und die entsprechenden Triggersignale für einen Abbruch eines Zyklus. Dies ist wichtig um das Zusammenspiel der Global-State-Machine mit der Netzwerkschnittstelle zu überprüfen.

Für den Typ von MCM der sowohl als Datenquelle fungiert als auch Daten sammelt und weiterleitet, werden speziell angepasste Tests gefahren. Diese MCMs haben die Besonderheit, dass allein der Eingangsport 2 nicht benutzt wird. Alle anderen Funktionen müssen gewährleistet sein.

Während der Produktion der TRAPs werden zunächst alle grundlegenden Funktionen des Chips mittels des Wafer-Testers (Kapitel 7.2) überprüft. Wenn diese Test bestanden sind, werden die Chips auf die MCMs gebondet und vergossen. Erst dann kann Verifikation der Netzwerkschnittstelle und der LVDS-Zellen vorgenommen werden, da ein großer Anteil der Fehler auf das Bonden und die Kontakte der MCMs zurückzuführen ist. Zu beachten ist dabei allerdings, dass ein MCM nicht zwangsläufig unbrauchbar ist, wenn diese Schnittstellen Fehler aufweisen. So wird z. B. bei den meisten der MCMs – denen, die Daten aufnehmen und senden – nur der Ausgangsport verwendet. Diejenigen MCMs mit einem Fehlverhalten bei den Eingangsports können hier jedoch weiterhin eingesetzt werden.

Funktion	Anzahl	Anteil
voll funktionsfähig	19088	89,1 %
alle Eingangsports	1165	5,4 %
Eingangsport 0, 1 und 3	84	0,4 %
keine Eingangsports	698	3,3 %
nicht funktionsfähig	393	1,8 %

Tabelle 7.2: Gelistet sind die Anteile, für die die mit dem MCM-Tester überprüften Module eingesetzt werden können. Da nicht bei allen MCMs die volle Funktionalität genutzt wird, können für bestimmte Aufgaben auch solche verwendet werden, bei denen z. B. nicht alle Eingangsports funktionieren.

Bei den Untersuchungen mit dem MCM-Tester sind bisher 21428 MCMs getestet worden. Tabelle 7.2 zeigt die Ergebnisse bezüglich der Netzwerkschnittstelle und der Einsetzbarkeit der MCMs. Als „voll funktionsfähig“ werden dabei die MCMs bezeichnet, bei denen sowohl alle Ein- und Ausgangports als auch die übrige Logik funktionieren. Diejenigen MCMs, bei denen zwar alle Ports funktionsfähig sind, die aber Fehler bei der analogen Datenakquisition oder beim Verarbeiten dieser Daten aufweisen, können weiterhin zum Sammeln und Weiterleiten von Datenströmen genutzt werden. Dies betrifft die MCMs 16 (Abbildung 5.3) der Ausleseplatinen und die MCMs (HMM), die die Daten der vier Ausleseplatinen sammeln. Wie ebenfalls in Abbildung 5.3 zu erkennen ist, werden bei den MCMs 2, 6, 10 und 14 zwar Daten gesammelt und weitergeleitet, jedoch nicht vom Eingangsport 2. Am häufigsten werden die MCMs eingesetzt, die ausschließlich Daten senden, bei denen also kein Eingangport verwendet wird, die übrige Elektronik jedoch ohne Fehler ist. Aus diesem Grund ist der relativ hohe Anteil von 3,3 % MCMs, die nur hier eingesetzt werden können, ohne Bedeutung für eine gute Ausbeute. Diejenigen MCMs, die „nicht funktionsfähig“ sind, können an keiner Stelle des TRDs eingesetzt werden.

7.3.3 Strahlzeit

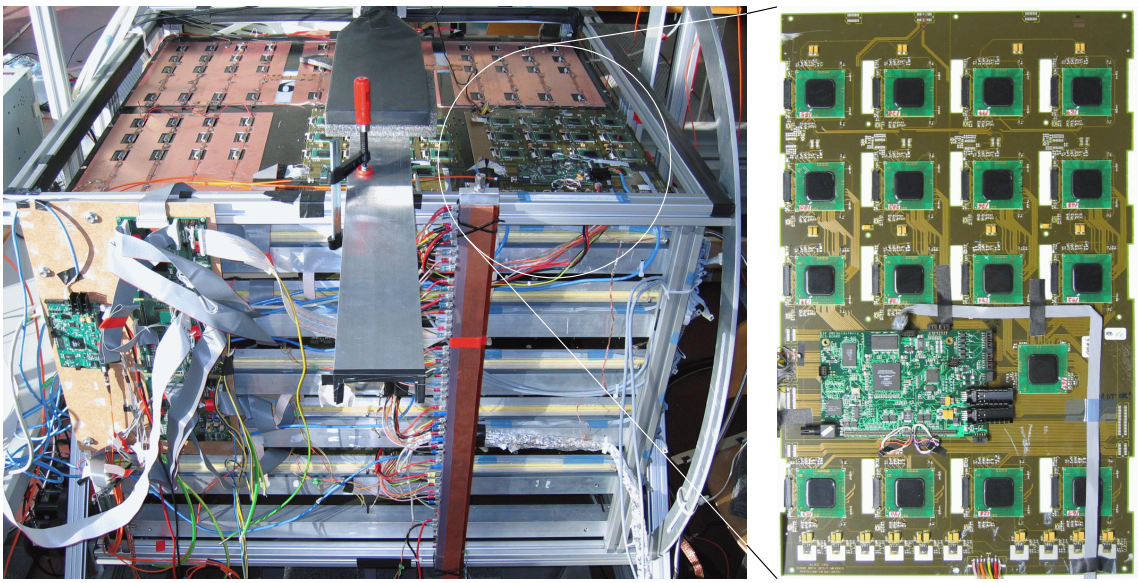

Abbildung 7.12: Das Foto zeigt einen Prototypen eines Moduls mit teilweiser Bestückung mit Ausleseplatinen. Dieses Modul wurde für die Strahlzeit im Herbst 2004 am CERN eingesetzt.

Im Herbst 2004 wurde am CERN eine Strahlzeit durchgeführt, bei der ein Prototyp eines Moduls (Abbildung 7.12) zum Einsatz kam. Hierbei werden acht vollbestückte Ausleseplatinen benutzt, eine in jeder Lage und jeweils eine weitere in der innersten und äußersten Lage. Jede Platine wird durch eine DCS-Karte angesprochen und per SCSN konfiguriert.

Zur Auslese wurde ein erweiterter Auslesebaum benutzt. Im Normalfall werden die Daten von vier Ausleseplatinen durch ein zusätzliches, auf einem der Ausleseplatinen befindliches,

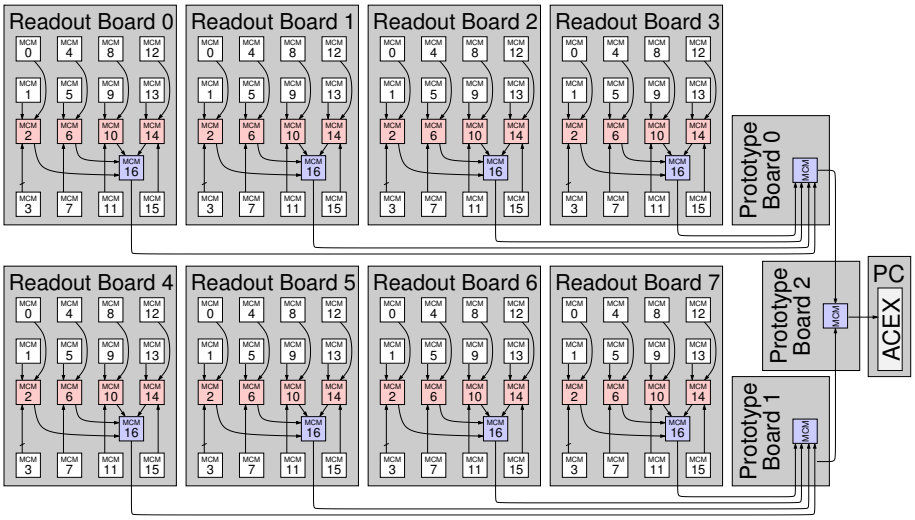

Abbildung 7.13: Während der Strahlzeit 2004 wurde eine erweiterte Hierarchie des Auslesebaums benutzt. Hier wurden anstatt der Transmitter-Platinen die Prototypplatinen 0 und 1 eingesetzt, die die Datenströme an eine weitere Prototypplatine 2 weiterleiten. Über eine FPGA-PCI-Karte (ACEX) wird der Datenstrom dekodiert und in einem PC weiterverarbeitet.

MCM gesammelt und an die Transmitter-Platine weitergegeben. In dem Aufbau sind diese MCMs durch Prototypplatinen ersetzt. Über diese beiden Platinen werden die Daten wiederum gesammelt und an eine weitere Prototypplatine gegeben. Eine FPGA-PCI-Karte dekodiert den Datenstrom und speichert die Daten zwischen. Damit ist eine Auslese in voller Übertragungsgeschwindigkeit möglich. Der PC kann dann über die PCI-Schnittstelle diesen Speicher auslesen und die Daten weiterverarbeiten. Auf diese Weise wurden während der Strahlzeit etwa 60 GByte Rohdaten ausgelesen. Nach einigen Tagen fiel eine der Ausleseplatinen vollständig aus. Ansonsten funktionierte der gesamte Ablauf der Datenübertragung reibungslos. Hierbei wurde auch die Master-State-Machine für die verschiedenen Triggerstufen eingesetzt.

Die Integration der Supermodule (fünf Stacks in z-Richtung) begann Mitte 2006 und Ende 2006 konnten am CERN die ersten Tests mit voll bestückten Supermodulen durchgeführt werden. Ein Supermodul besteht aus 30 Modulen wobei die Auslese über zwei Auslesebäume pro Modul implementiert ist. Insgesamt wurden hierbei also 228 Ausleseplatinen mit 3648 Daten erzeugenden Multi-Chip-Modulen und zusätzlichen 258 MCMs zum Aufbau der Auslesebäume parallel betrieben. Abbildung 7.14 zeigt die Integration eines solchen Supermoduls. Die innerste der sechs Lagen ist hier schon integriert und mit Ausleseplatinen bestückt.

Zunächst wurden Messungen zum Rauschverhalten angestellt. Über Software-seitig ausgelöste Trigger wurden jeweils 100 Ereignisse erzeugt, ausgelesen und ausgewertet. Zur Aufnahme der Daten kamen, wie in der Strahlzeit von 2004, DCS-Karten zum Einsatz. Bei diesen Untersuchungen stellte sich vor allem heraus, dass auf eine saubere Kabelführung und gute Masseanbindung der Ausleseplatinen zu achten ist. Die Auslese selbst funktionierte jedoch ohne nennenswerte Probleme.

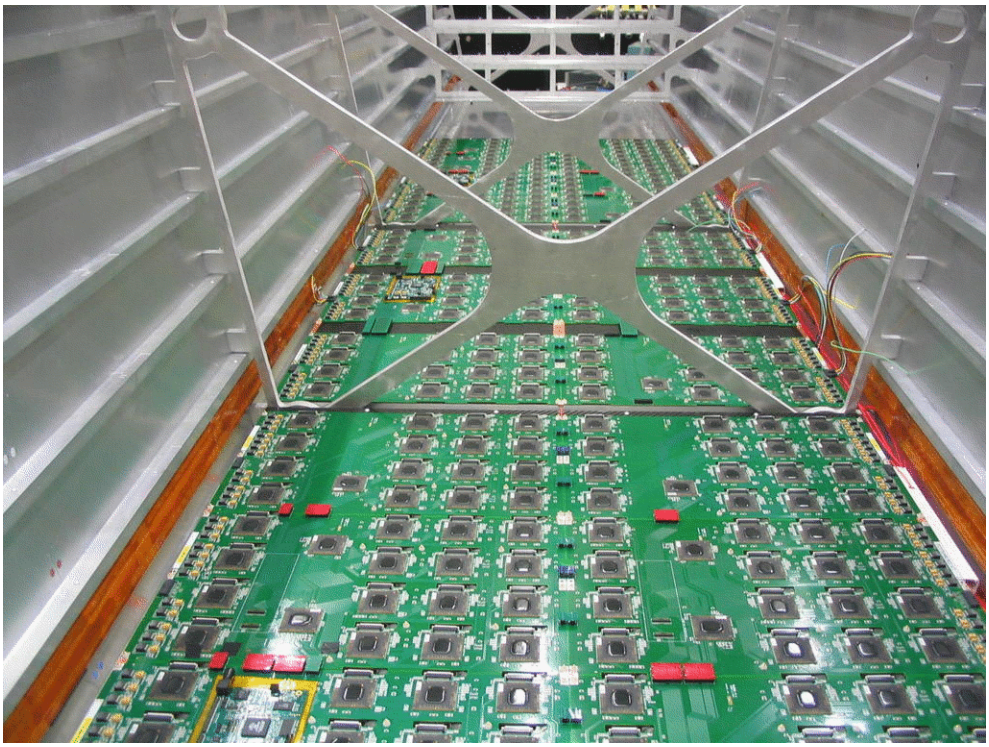

Abbildung 7.14: Das Foto zeigt den Aufbau eines Supermoduls am Kirchhoff-Institut für Physik in Heidelberg. Die innerste der sechs Lagen ist schon integriert und mit Ausleseplatinen bestückt. Quelle: Physikalisches Institut der Universität Heidelberg

Um die gesamte Kette des Triggers innerhalb des TRDs, die Datenverarbeitung und die Auslese zu testen, wurde das Supermodul mit Höhenstrahlung stimuliert. Mithilfe von Szintillatoren wurde nach Ereignissen mit möglichst vielen Spuren gesucht, um die volle Funktionalität des TRAPs zu testen. Hierbei kamen die verschiedenen Auslesemodi zum Einsatz und es konnten die korrekten Abläufe während einer Auslese verifiziert werden.

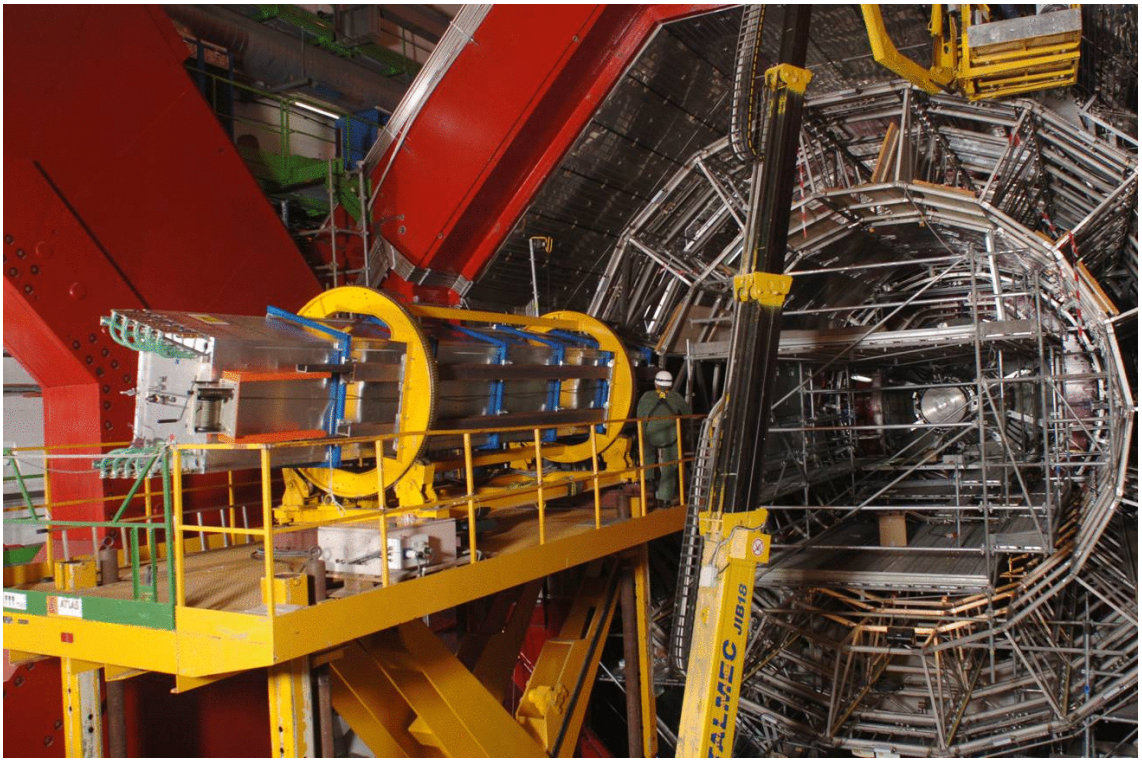

Abbildung 7.15: Installation des ersten Supermoduls in das ALICE Experiment am CERN Ende 2006. Quelle: Venelin Angelov

8 Zusammenfassung und Ausblick

Im ALICE-Experiment werden mit einer Vielzahl von Detektoren Spuren erkannt und rekonstruiert. Mit deren Hilfe können Rückschlüsse auf die Abläufe bei einer Kollision der aufeinander geschossenen Teilchen gezogen und damit z. B. der Zustand des Quark-Gluonen-Plasmas untersucht werden. Der Übergangsstrahlungsdetektor (TRD) dient hierbei insbesondere als Trigger-Detektor für die langsamere und hochauflösendere Zeit-Projektions-Kammer (TPC). Durch eine geeignete Triggerhierarchie lässt sich die Effizienz des Experiments deutlich erhöhen. Der TRD ist in mehreren zylinderförmigen Lagen aufgebaut und enthält Elektronik, die die Signale der mehr als 1,2 Millionen Kanäle digital wandelt und noch auf dem Detektor vorverarbeitet. Dies geschieht durch 65 664 Multi-Chip-Module, die die Verstärker, ADCs, Filter und jeweils vier CPUs beinhalten. Diese frühe und massiv parallele Verarbeitung der Daten bietet enorme Potentiale für eine schnelle Vorverarbeitung und Reduktion der Daten und damit die Möglichkeit zum Entwurf eines effizienten und neuartigen Triggersystems. Die Aufgabe des TRDs als Trigger ist es, die über die Lagen verteilten Spursegmente zu bestimmen, zusammenzuführen, die Spuren zu rekonstruieren und innerhalb von nur 6,5 µs eine Teilchenidentifizierung und Impulsbestimmung durchzuführen. Anhand dieser Ergebnisse wird entschieden, ob die Rohdaten des TRDs und der TPC ausgelesen und der nächsten Triggerstufe übergeben werden.

In dieser Arbeit wurden die Leistungsfähigkeit des TRDs untersucht, Parameter optimiert und ein entsprechendes Triggerkonzept entwickelt, sowie die damit in engem Zusammenhang stehende Auslese des Detektors entworfen und implementiert. Die Simulationen des TRDs, der Elektronik und der Algorithmen zur Bestimmung der Spursegmente geben Aufschluss über die Genauigkeit der Spursegmentparameter und die Möglichkeiten der Reduktion der Daten auf dem Detektor. Diese Ergebnisse sind die Voraussetzung zur Entwicklung einer Auslesestruktur, die den harten zeitlichen Anforderungen entspricht. Außerdem wirkt sich die Funktionalität der lokalen Tracking-Einheiten (LTUs) und die Architektur der Auslese direkt auf die GTU aus, ein performantes System, das die Spursegmente zusammenführt und eine genaue Impulsbestimmung durchführt.

Die besondere Herausforderung lag in der Gestaltung eines Gesamtkonzeptes, das das Zusammenspiel der LTUs, der Auslese und der GTU berücksichtigt. Aus diesem Grund wurde im Rahmen dieser Arbeit neben der Auslese des Detektors auch ein grundlegendes Konzept für die Funktionsweise der GTU entwickelt. Besonders die zeitlichen Anforderungen an das Triggersystem verlangen dabei Rücksichtnahme auf eine effiziente Implementierbarkeit in Hardware. Hierzu wurde eine Hardware-nahe Beschreibung des Multi-Chip-Moduls entwickelt und in die Simulationen eingebunden. Einschränkungen in der Funktionalität der Algorithmen, hervorgerufen durch z. B. Verarbeitungsgeschwindigkeit, Platzbedarf oder

Mechanik, können somit untersucht und die Wirkung auf die Gesamtfunktionalität überprüft werden. Zur Simulation des ALICE-Experiments wird das Softwarepaket AliRoot verwendet. Zur Beschreibung der Triggerelektronik wurde die Klassenbibliothek des TRDs um Modelle der LTUs, der GTU und um Klassen zur Analyse des Verhaltens erweitert.

Durch die angestellten Simulationen konnte gezeigt werden, dass es durch Optimierungen möglich ist, ein Triggersystem zu entwickeln, das die hohen Anforderungen erfüllen kann. So wurde z. B. hergeleitet, dass erst durch eine Anpassung des dynamischen Bereichs und der Auflösung der ADCs eine hinreichend gute Ortsauflösung erzielt werden kann. Diese ist für eine genaue Impulsbestimmung auf Modulebene notwendig, durch die eine Reduktion der Datenmenge ermöglicht wird. In direktem Zusammenhang mit der Ortsauflösung steht außerdem die Ablenkungsgenauigkeit der Spursegmente, die für die Funktionsweise der GTU notwendig ist. Ein weiteres Beispiel eines Simulationsergebnisses ist der Zusammenhang der Zeilen eines Moduls. Bedingt durch die Architektur des Detektors und der Funktionsweise der LTUs ist eine Berücksichtigung benachbarter Zeilen nicht notwendig. Durch dieses Ergebnis konnte die Komplexität der Datenverarbeitung und die Struktur der Vernetzung erheblich vereinfacht werden. Außerdem haben sowohl die Reduktion der Datenmenge als auch die Unabhängigkeit der Zeilen entscheidenden Einfluss auf die Architektur der Auslese des Detektors.

Bei der Detektorauslese werden die Daten der mehr als 65 000 MCMs, die über die Module des Detektors – einer Fläche von mehr als $700\,\mathrm{m}^2$ – verteilt sind, gesammelt und an die GTU übertragen. Es wird zwischen der Triggerauslese und der Rohdatenauslese unterschieden. Besondere Anforderungen bestehen hierbei an die Auslese für die Funktion des Detektors als Trigger. Durch die Simulation konnte die Menge der zu übertragenden Daten pro MCM minimiert und die zur Verfügung stehende Zeit hergeleitet werden, woraus sich das Datenvolumen, die maximale Latenz und der benötigte Datendurchsatz ergeben. Unter Berücksichtigung weiterer Parameter, wie z. B. einer Fehlertoleranz, der Mechanik, der Modularität und des Stromverbrauchs, wurde ein speziell auf diese Anforderungen optimiertes Netzwerk in Form eines Auslesebaums entwickelt. Diese Struktur verbindet die Vorteile weniger Verbindungen zwischen den MCMs mit einer geringen Tiefe der Auslese, was die Latenz gering hält. Die Daten werden dabei von vier Ausleseplatinen mittels eines Baums der Tiefe vier gesammelt und über einen optischen Link an die GTU übertragen. Insgesamt werden 1080 solcher Links aufgebaut. Durch eine optimierte Reihenfolge der Auslese der MCMs kann außerdem die Verarbeitungszeit der GTU minimiert werden. Zum Aufbau des Auslesenetzwerks werden dieselben MCMs genutzt, die auch die Daten verarbeiten. Hierdurch ist eine effiziente und stromsparende Implementierung und ein sehr modularer Aufbau der Auslese möglich geworden.

Die Netzwerkschnittstelle wurde als ein Modul innerhalb des Tracklet-Prozessors entworfen und bildet die Verbindung von den CPUs zum Auslesenetzwerk. Durch die direkte Anbindung an die CPUs des Prozessors kann die Schnittstelle sehr flexibel eingesetzt werden. Mit den vier Eingangsports und einem Ausgangsport der Netzwerkschnittstelle wird der Auslesebaum aufgebaut. Vor allem durch die Triggerauslese werden an das Auslesenetzwerk und damit die Schnittstelle hohe Anforderungen an die Latenz und die Übertragungsrate gestellt. Die Latenz ergibt sich in erster Linie daraus, wie lange es dauert, ein Datum durch

ein Modul hindurch weiterzuleiten. Durch geschickte Ausnutzung der Gegebenheit, dass alle MCMs mit einem Takt derselben Frequenz versorgt werden, auch wenn sich die Phasen unterscheiden, konnte diese Latenz auf maximal zwei Takte reduziert werden. Bei gegebener Übertragungsfrequenz und Wortbreite ist es zum Erreichen einer möglichst hohen Übertragungsrate von Nutzwörtern vor allem wichtig, so wenig Lücken wie möglich im Datenstrom zu erzeugen. Das Auslesenetzwerk wurde so entworfen, dass alle MCMs gleichzeitig ihre Daten senden und ein empfangendes MCM alle Daten zwischenspeichern kann. Dieser Verzicht auf ein Handshake-Protokoll in der Triggerauslese erfordert zwar entsprechend große Datenpuffer in den MCMs, jedoch kann somit ein nahezu ununterbrochener Datenstrom erzeugt werden. Erst durch die Simulationen des Gesamtsystems konnte z. B. die nötige Größe dieser Speicher bestimmt und diese Architektur der Auslese ermöglicht werden. Ein weiteres wichtiges Entwurfskriterium für die Netzwerkschnittstelle ist der Stromverbrauch. Während des Triggers arbeiten auf dem gesamten Detektor alle Prozessoren parallel, was eine sehr hohe Leistungsaufnahme bedeutet. Die Schnittstelle wurde so konzipiert, dass sich die CPUs möglichst früh ausschalten können und die Auslese der Daten autonom erfolgen kann. Auch die Rohdatenauslese ist optimiert auf einen möglichst geringen Stromverbrauch. Zwar weist die Leistungsaufnahme der Prozessoren hierbei nicht solche Spitzen auf, jedoch sind die Datenmenge und damit die Verarbeitungszeit deutlich höher. Durch ein effizientes Handshake-Protokoll konnte erreicht werden, dass auch hier der gerade nicht beanspruchte Teil des Auslesebaums vollständig abgeschaltet werden kann.

Der Tracklet-Prozessor und damit auch die Netzwerkschnittstelle sind in der Hardwarebeschreibungssprache VHDL entworfen. Gefertigt wurde der Chip in einem UMC-0,18-µm-Prozess. Im Rahmen dieser Arbeit wurde ein Design-Fluss erarbeitet, der sowohl die Synthese und das Layout von TRAP als auch verschiedene Simulationsmethoden und Analysen des zeitlichen Verhaltens beinhaltet. Hierbei wurde besonderer Wert darauf gelegt, dass all diese Design-Schritte Skript-basiert sind, also ohne interaktives Eingreifen erfolgen können. Zusammen mit dem ersten erfolgreich submittierten Chip FaRo wurden insgesamt vier Prototypen entwickelt und gefertigt.

Während einer Strahlzeit am CERN im Herbst 2004 kam ein Prototyp eines Moduls zum Einsatz, das mit acht vollbestückten Ausleseplatinen ausgestattet war. Hierbei wurden auch der komplette Auslesebaum erfolgreich getestet und etwa 60 GByte an Rohdaten verarbeitet. Außerdem wurden umfangreiche Tests im Kirchhoff-Institut für Physik angestellt, in denen kosmische Strahlung detektiert und ausgelesen wurde. Seit Ende 2006 werden die ersten Supermodule im ALICE-Experiment integriert und es wurden erfolgreiche Messungen mit einem voll bestücktem Supermodul mit 228 Ausleseplatinen durchgeführt. Der nächste große Schritt wird die Verknüpfung der TRD-Elektronik mit der GTU sein, was für Ende 2007 geplant ist. 2008 soll der TRD vollständig aufgebaut sein und ALICE in Betrieb gehen.

Danksagung Ich möchte mich bei allen bedanken, die zum Gelingen dieser Arbeit beigetragen haben, insbesondere bei meinem Doktorvater Prof. Dr. Volker Lindenstruth und meinen Eltern. Außerdem bedanke ich mich bei allen Kollegen für Ratschläge, Hilfestellungen und eine sehr schöne Zeit.

Glossar

ACEX	Multifunktions PCI-Karte mit einem Altera ACEX FPGA, entwickelt am Lehrstuhl für Technische Informatik an der Universität Heidelberg.
ADC	Analog to Digital Converter.
ALICE	A Large Ion Collider Experiment; Kapitel 2.1.
AMS	Austria Micro System, http://www.austriamicrosystems.com.
ATLAS	A Toroidal LHC Apparatus; LHC-Experiment am CERN.
CERN	Organisation Européenne pour la Recherche Nucléaire.
CMS	Compact Muon Solenoid.
COG	Center Of Gravity.
CPLD	Complex Programmable Logic Device.
CPU	Central Processing Unit; Kapitel 4.1.4.
CTGEN	Clock Tree Generator von Cadence Silicon Ensemble.
CTP	Central Trigger Processor; Kapitel 2.2.
CTS	Clock Tree Synthesis.
CVS	Concurrent Versions System, http://www.nongnu.org/cvs/.
DAC	Digital to Analog Converter.
DCS	Detector Control System.
DDR	Double Data Rate; Übertragung von Daten zur steigenden und fallenden Flanke eines Taktes.
DMEM	Data Memory; Datenspeicher des Tracklet-Prozessors; Kapitel 4.1.4.
DRC	Design Rule Check.
EB	Event Buffer, Ereignisspeicher des Präprozessors; Kapitel 4.1.2.
FIFO	First In First Out.
FIR	Finite Impulse Response.
FMD	Forward Multiplicity Detector.
FPGA	Field Programmable Gate Array.
GCC	GNU Compiler Collection.

GDSII	Graphical Design System II; Ein Datenformat zur Beschreibung von IC Layouts.
GIO	Global I/O; Globale Schnittstelle der Netzwerkschnittstelle zum Tracklet-Prozessor; Kapitel 6.2.
GRF	Global Register File; Kapitel 4.1.4.
GTU	Global Tracking Unit; Globale Tracking-Einheit Kapitel 3.4.
HDL	Hardware Description Language.
HIJING	Heavy Ion Jet INteraction Generator; Kapitel 3.2.1.
HLT	High Level Trigger; Kapitel 2.2.
HMM	Half Module Merger; Das MCM, das die Daten eines halben Moduls, also von vier Ausleseplatinen, sammelt und an die Transmitterplatine gibt; Kapitel 5.2.1.
HMPID	High Momentum Particle Identification.
IC	Integrated Circuit.
IEEE	Institute of Electrical and Electronics Engineers, http://www.ieee.org.
IIR	Infinite Impulse Response.
IMEC	Interuniversity Microelectronics Centre, http://www.imec.be.
IMEM	Instruction Memory; Instruktionsspeicher des Tracklet-Prozessors; Kapitel 4.1.4.
IP	Intellectual Property.
ITS	Inner Tracking System; Kapitel 2.1.
JTAG	Joint Test Action Group, IEEE 1149.1-1990.
L0-L2	Level0- bis Level2-Trigger; Kapitel 2.2.
LEF	Library Exchange Format; Beinhaltet alle zum Layout notwendigen Informationen einer Standard- oder Makrozelle.
LEP	Electron-Positron Collider.
LHC	Large Hadron Collider; Der Beschleunigerring am CERN.
LHCb	LHC Beauty Experiment; LHC-Experiment am CERN.
LIO	Local I/O; Lokale Schnittstelle der Netzwerkschnittstelle zum Tracklet-Prozessor; Kapitel 6.2.
LSB	Least Significant Bit.
LTU	Lokale Tracking-Einheit; Kapitel 3.3.
LUT	Look Up Table.
LVDS	Low Voltage Differential Signaling.
LVS	Layout Versus Schematic; Vergleich zwischen einer Netzliste und einer aus dem Chip-Layout extrahierten Schaltung.

MC	Monte Carlo; Kapitel 3.2.1
MCM	Multi Chip Module; Platine, die den Tracklet-Prozessor und den PASA-Chip trägt; Kapitel 2.3.2.
MIMD	Multiple Instruction Multiple Data.
Modul	Detektorkammer des TRDs mit Radiator und Elektronik; Kapitel 2.3.2.
MSB	Most Significant Bit.
MSM	Master State Machine; Sie verwaltet auf höherer Ebene die Ablaufsteuerung des Tracklet-Prozessors; Kapitel 4.1.
NI	Network Interface; Die Netzwerkschnittstelle; Kapitel 6.
PASA	Pre-Amplifier Shaper-Amplifier; Ladungsempfindliche Vorverstärker; Kapitel 2.3.1.
PCI	Peripheral Component Interconnect, http://www.pcisig.com.
PDU	Programmable Delay Unit; Programmierbare Verzögerungseinheit der Netzwerkschnittstelle; Kapitel 6.2.3.
PHOS	Photon Spectrometer.
PLL	Phase-Locked Loop; Phasengekoppelter Regelkreis zur Taktgenerierung.
PRF	Private Register File; Kapitel 4.1.4.
QCD	Quanten-Chromodynamik.
QGP	Quark-Guon-Plasma.
RAM	Random Access Memory.
ROB	Readout Board; Ausleseplatine; Kapitel 5.2.2.
ROOT	Ein objektorientiertes Datenanalyse-Framework, http://root.cern.ch.
RSD	Redundant Signed Digit.
SCSN	Slow Control Serial Network; Kapitel 4.1.
SDF	Standard Delay Format.
SIP	Semiconductor Intellectual Property.
SPEF	Standard Parasitic Exchange Format.
SPS	Super Proton Synchrotron.
Stack	Sechs übereinanderliegende Module im TRD; Kapitel 2.3.2.
Supermodul	Fünf Stacks in z-Richtung.
SVN	Subversion, http://subversion.tigris.org/.
TOF	Time Of Flight; Kapitel 2.1.
TPC	Time Projection Chamber; Kapitel 2.1.

TRAP	Tracklet-Prozessor; Kapitel 4.1.
TRD	Transition Radiation Detector; Übergangsstrahlungsdetektor; Kapitel 2.3.
TTC	Timing and Trigger Control; Kapitel 2.2.
VHDL	VHSIC HDL; Hardware-Beschreibungssprache; Kapitel 4.2.1.
VHSIC	Very High Speed Integrated Circuit.
VST	Virtual Silicon Technologies; Anbieter für Halbleiter-IPs, http://www.virtual-silicon.com bzw. seit Oktober 2005 http://www.mosaid.com.

Literaturverzeichnis

[1] ALICE Collaboration, *Technical Proposal for A Large Ion Collider Experiment at the CERN LHC* CERN/LHCC 95-71, 1995

[2] ALICE Collaboration, *Technical Design Report on Forward Detectors: FMD, T0 and V0*, CERN/LHCC 2004-025, 2004

[3] ALICE Collaboration, *Technical Design Report of the Inner Tracking System (ITS)*, CERN/LHCC 99-12. 18, 1999

[4] ALICE Collaboration, *Technical Design Report of the Time Projection Chamber (TPC)*, CERN/LHCC 2000-001, 2000

[5] ALICE Collaboration, *Technical Design Report of the Transition Radiation Detector*, 2001

[6] ALICE Collaboration, *Technical Design Report of the Time of Flight System (TOF)*, CERN/LHCC 2000-12, 2000

[7] ALICE Collaboration, *Technical Design Report of the Photon Spectrometer (PHOS)*

[8] ALICE Collaboration, *Technical Design Report of the High Momentum Particle Idenfification Detector (HMPID)*, CERN/LHCC 98-19, 1998

[9] Boris Dolgoshein, *Transition radiation detectors*, Nuclear Instruments and Methods in Physics Research Section A, Seite 434-469, 19 July 1992.

[10] Anton Andronic, *The ALICE Transition Radiation Detector*, An overview for students, GSI, Feb. 6th, 2007.

[11] Anton Andronic, *Space charge in drift chambers operated with the Xe,CO2(15%) mixture*, Nuclear Instruments and Methods A, Nr. 525, S. 447, 2004

[12] F. Carminati and A. Morsch, *Simulation in ALICE*, Computing in High Energy and Nuclear Physics, La Jolla, California, March 2003

[13] J. P. Wessels, *The ALICE Transition Radiation Detector - Design and Performance*, ALICE-CA-2004-07 version 1.0, 2004-05-14

[14] Miklos Gyulassy and Xin-Nian Wang, *HIJING 1.0: A Monte Carlo Program for Parton and Particle Production in High Energy Hadronic and Nuclear Collisions*, LBL-34246, January 29, 2002

[15] Torbjörn Sjöstrand, *Pythia 5.7 and Jetset 7.4 Physics and Manual*, University of Lund, LU TP 95-20, December 1998

[16] C. F. G. Delaney and E. C. Finch, *Radiation Detectors - Physical Principles and Applications*, Oxford University Press, New York, 1992

[17] Claus Grupen, *Teilchendetektoren*, Bibliographisches Institut, Wissenschaftverlag, Mannheim, 1993

[18] H.-K. Soltveit, *Final version of the Analog Front-end Electronics for the ALICE TPC detector and ALICE TRD-Detector*, GSI Scientific Report, 2003

[19] H.-K. Soltveit, *Development of a Fast TRD Pre-Amplifier Shaper*, GSI Scientific Report, 2004

[20] Bogdan Vulpescu, *Electron Trigger with the ALICE TRD*, http://www.physi.uni-heidelberg.de/∼bogdan/

[21] Falk Lesser, *Entwurf und Realisierung eines vierfach MIMD Prozessor Mikrochips für eine Anwendung in der Hochenergiephysik*, Dissertation, Heidelberg, 2002

[22] Falk Lesser, *A MIMD-Based Multi-Threaded Processor*, Hot Chips, 2001

[23] Marcus Gutfleisch, *Digitales Frontend und Präprozessor im TRAP1 Chip des TRD Triggers für das ALICE Experiment am LHC (CERN)*, Diplomarbeit, Heidelberg, 2002

[24] Jan de Cuveland, *Entwicklung der globalen Spurrekonstruktionseinheit für den Transition Radiation Detector*, Diplomarbeit, Heidelberg, 2003

[25] Jan de Cuveland, *Entwicklung einer 32 Bit ALU*, Hilfwissenschaftliche Arbeit, private Kommunikation

[26] Christian Reichling, *Entwicklung eines Quadport Memory für den Einsatz in einem Triggerprozessort am CERN*, Diplomarbeit, Heidelberg, 2001

[27] Frederik Ferner, *Development of a Test Environment for the ALICE TRD Readout Chip*, Diplomarbeit, Heidelberg, 2005

[28] Robin Gareus, *Slow Control - Serial Network and its implementation for the Transition Radiation Detector*, Diplomarbeit, Heidelberg, 2002

[29] D. Muthers and R. Tielert, *Ein 10 Bit 10 MS/s Low-Power AD-Converter in 0,11 mm²*, Advances in Radio Science, Manuscript-No. 12345, 2003

[30] Gunther Lehmann, Bernhard Wunder, Manfred Selz, *Schaltungsdesign mit VHDL*, Franzis-Verlag, Poing, 1994

[31] R. Walker, D. Thomas, *A Model of Design Representation and Synthesis*, in: 22nd Design Automation Conference, Las Vegas, 1985

[32] Dennis Sylvester and Kurt Keutzer, *Rethinking deep-submicron circuit design*, IEEE Computer, vol. 32, November 1999

[33] Dennis Sylvester and Kurt Keutzer, *Getting to the Bottom of Deep Submicron*, IEEE Conference On Computer Aided Design (ICCAD), 1998

[34] Horst Kuchling, *Taschenbuch der Physik*, Fachbuchverlag GmbH Leipzig, 13. Auflage, 1991

[35] René Brun und Fons Rademakers, *ROOT - An Object-Oriented Data Analysis Framework - Users Guide 3.02b*, March 2002

[36] The ALICE Off-line Project, *AliRoot*, http://aliweb.cern.ch/offline/

[37] William R. Leo, *Techniques for Nuclear and Particle Physics Experiments*, Springer-Verlag, Second Revised Edition, 1994

[38] Philip R. Bevington, *Data Reduction and Error Analysis for Physical Science*, McGraw-Hill Book Company, 1969

[39] Helmut Kopka, *LaTeX Einführung*, Addison-Wesley, 2. überarbeitete Auflage, 1996

[40] Jörg Knappen, Hubert Partl, Elisabeth Schlegl, Irene Hyna, *LaTeX 2ε - Kurzbeschreibung*, Version 1.2, 12. November 1995

[41] Per Cederqvist et al, *Version Management with CVS for CVS 1.9*, Signum Support AB, 1992/1993

[42] Xilinx Inc., *Virtex-4 RocketIO Multi-Gigabit Transceiver - User Guide*, UG076 (v1.2) April 7, 2005

[43] Texas Instruments, *1.5 to 2.5 GBPS Transceiver - Datasheet*, TLK2501, SLLS427D, August 2000, Revised July 2003

[44] Infineon Technologies AG, *iSFP - Intelligent Small Form-factor Pluggable 2.5 Bgit/s InfiniBand 1X-SX Multimode 850 nm Transceiver with LC Connector*, V23848-N305-C56, Preliminary Data Sheet, Revision 2004-01-28, 2004

[45] Virtual Silicon Inc., *eSi-PLL - Phase-Locked Loop Compiler*, Part Number: UMCL18U800, Rev. 2.0, September 07th, 2000

[46] Virtual Silicon Inc., *eSi-Route/11 - High Performance 0.18μ Standard Cell Library*, Part Number: UMCL18U250, Rev. 2.4, November, 2001

A Programmiermodell der Netzwerkschnittstelle

A.1 Globale Kontrollregister

NMOD	0x0D40	**NI Modus Kontrolle**
	Beschreibung	Kontrolliert den Netzwerkmodus und das synchrone Zurücksetzen.
	Register	`---- ---- ---- ---- ---- ---- ---i cmmm`
	Reset Wert	`---- ---- ---- ---- ---- ---- ---0 1000`
	Einträge	**i** Ignoriert den Auslesemodus der MSM
		c synchrones Clear (low active)
		m Auslesemodus:
		`000` : Leerlauf Zustand, wartet auf den Auslesemodus
		`001` : startet die Triggerauslese
		`010` : startet die Rohdatenauslese
		`011` : startet den Netzwerkmodus
		`100` : sendet das Testmuster
		`else` : setzt die Netzwerkschnittstelle zurück

NDLY	0x0D41	**Verzögerungen der Datenausgangbits**
	Beschreibung	Verzögerung der Datensignale (a/d(0) - j/d(9)).
	Register	`--jj jiii hhhg ggff feee dddc ccbb baaa`
	Reset Wert	`--10 0100 1001 0010 0100 1001 0010 0100`

NED	0x0D42	**Ausgangs-Bitpositionen und Verzögerungen**
	Beschreibung	Setzt die Bitpositionen des Paritäts und Reservebits und die Verzögerungen des Strobe- und Kontrollausgangs. Setzt außerdem die Position zum optischen Übertrager.
	Register	`---- ---- ---- ---- orpp ppff ffcc csss`
	Reset Wert	`---- ---- ---- ---- 0010 0010 0100 0000`
	Einträge	**o** Optischer Übertragermodus (clk/ctrl Ausgang)
		r Wurzel des Auslesebaums
		p Position des Paritätsbits (wird zuerst gesetzt)
		f Position des Reservebits (wird nach der Parität gesetzt)
		c Verzögerung des Kontroll-Ausgangssignals
		s Verzögerung des Strobe Signals

NTRO	0x0D43	**Trigger-Auslesereihenfolge**

Beschreibung Setzt die Reihenfolge zur Triggerauslese. Die Werte (oktal), die jeweils gesetzt werden können, haben folgende Bedeutung: 0–3 entspricht Port 0–3, 4 sind die internen Daten, 7 bedeutet keine weiteren Quellen und alle anderen Werte sind Fehler, die zum Zurücksetzen der NI führen.

Register
```
---- ---- ---- --ii iddd cccb bbaa afff
```

Reset Wert
```
---- ---- ---- --11 1111 1111 1111 1100
```

Einträge

i	zu senden nach den internen Daten
d	zu senden nach Port 3
c	zu senden nach Port 2
b	zu senden nach Port 1
a	zu senden nach Port 0
f	als erstes zu senden

NRRO	0x0D44	**Rohdaten Auslesereihenfolge**

Beschreibung Setzt die Reihenfolge zur Rohdatenauslese (siehe **NTRO**).

Register
```
---- ---- ---- --ii iddd cccb bbaa afff
```

Reset Wert
```
---- ---- ---- --11 1111 1111 1111 1100
```

NES	0x0D45	**Endsignaturen**

Beschreibung Endsignaturen für die Trigger- und Rohdatenauslese.

Register
```
rrrr rrrr rrrr rrrr tttt tttt tttt tttt
```

Reset Wert
```
0000 0000 0000 0000 0000 0000 0000 0000
```

Einträge

r	Endsignatur für die Rohdatenauslese
t	Endsignatur für die Triggerauslese

NTP	0x0D46	**Test Muster**

Beschreibung Testmuster das im Testmuster Modus gesendet wird.

Register
```
pppp pppp pppp pppp pppp pppp pppp pppp
```

Reset Wert
```
0000 0000 0000 0000 1111 1111 1111 1111
```

NBND	0x0D47	**FIFO Schwellen**

Beschreibung Programmierbare FIFO Schwellen, die die Statussignale **NLF.H** und **NLF.L** beeinflussen.

Register
```
---- ---- ---- ---- hhhh hhhh llll llll
```

Reset Wert
```
---- ---- ---- ---- 0110 0000 0010 0000
```

Einträge

h	Obere Schwelle (für alle FIFOs)
l	Untere Schwelle (für alle FIFOs)

NPn	0x0D48+n	**Eingangsport n∈[0..3] Kontrolle**
	Beschreibung	Bit Positionen und Kontrollsignale für die Eingangsports n∈[0..3].
	Register	`---- ---- ---- ---- ---- -ppp pfff fecs`
	Reset Wert	`---- ---- ---- ---- ---- -100 0100 1100`
	Einträge	**p** Position des Paritätsbits (nach Reservebit) **f** Position des Reservebits **e** Aktiviert den Zähler für Paritätsfehler **c** Ermöglicht das Kontrollausgangsbit über die Konfiguration zu setzten **s** Setzt das Kontrollausgangsbit wenn **c** gesetzt ist

NCUT	0x0D4C	**Abschneiden des Datenstromes**
	Beschreibung	Schneidet den eingehenden Datenstrom während der Triggerauslese von Port 0..3 bei dem Wort a..d ab.
	Register	`dddd dddd cccc cccc bbbb bbbb aaaa aaaa`
	Reset Wert	`1111 1111 1111 1111 1111 1111 1111 1111`

A.2 Globale Statusregister

NCTRL	0x0DC0	**Kontroll Eingang des Ausgangsports**
	Beschreibung	Kontrolleingang des Ausgangsports abgebildet auf das GIO.
	Register	`---- ---- ---- ---- ---- ---- ---- ---C`
	Reset Wert	`---- ---- ---- ---- ---- ---- ---- ---1`

NFIFO	0x0DC1	**FIFO Fehler Signale**
	Beschreibung	FIFO 0..3 Fehler Signale, abgebildet auf das GIO (A..D).
	Register	`---- ---- ---- ---- ---- ---- ---- DCBA`
	Reset Wert	`---- ---- ---- ---- ---- ---- ---- 0000`

NFSM	0x0DC2	Zustand der Netzwerkschnittstelle

Beschreibung Aktueller Zustand der Netzwerkschnittstelle, abgebildet auf das GIO. Die Bezeichnungen der Zustände entsprechen denen aus Kapitel 6.3.

Register `---- ---- ---- ---- ---- ---- ---S SSSS`

Reset Wert `---- ---- ---- ---- ---- ---- ---0 0000`

Einträge

S Kodierter Zustand:

- `00000` : CLEAR
- `00001` : IDLE
- `00010` : TRIGGER - IDLE
- `00011` : RAW DATA - IDLE
- `00100` : NETWORK - IDLE
- `00101` : TRIGGER - LIO 0 LOW
- `00110` : TRIGGER - LIO 0 HIGH
- `00111` : TRIGGER - LIO 1 LOW
- `01000` : TRIGGER - LIO 1 HIGH
- `01001` : TRIGGER - LIO 2 LOW
- `01010` : TRIGGER - LIO 2 HIGH
- `01011` : TRIGGER - LIO 3 LOW
- `01100` : TRIGGER - LIO 4 HIGH
- `01101` : TRIGGER - PORT/FIFO 0
- `01110` : TRIGGER - PORT/FIFO 1
- `01111` : TRIGGER - PORT/FIFO 2
- `10000` : TRIGGER - PORT/FIFO 3
- `10001` : RAW DATA - PORT 0
- `10010` : RAW DATA - PORT 1
- `10011` : RAW DATA - PORT 2
- `10100` : RAW DATA - PORT 3
- `10101` : RAW DATA - FIFO 0
- `10110` : RAW DATA - FIFO 1
- `10111` : RAW DATA - FIFO 2
- `11000` : RAW DATA - FIFO 3
- `11001` : TRIGGER/RAW DATA - FINISHED
- `11010` : TEST PATTERN LOW
- `11011` : TEST PATTERN HIGH

A.3 Lokale Statusregister

NLF	0x00C0	**FIFO Zustände (LIO)**
	Beschreibung	Status der FIFOs.
	Register	---- ---- ---- ---- ---- -DSS EHLZ YXWV
	Reset Wert	---- ---- ---- ---- ---- -000 0000 0001
	Einträge	**D** Daten sind gültig. Ein 32-Bit-Wort kann von der CPU gelesen werden
		S Aktelle Paritätsstatus
		E FIFO Fehler Signal
		H Anzahl der Einträge liegt über der oberen Schwelle **NBND.h**
		L Anzahl der Einträge liegt unter der unteren Schwelle **NBND.l**
		Z FIFO ist voll (128·16 Bit)
		Y FIFO ist fast voll (127·16 Bit)
		X FIFO ist halbvoll (64·16 Bit)
		W FIFO ist fast leer (1·16 Bit)
		V FIFO ist leer

NLP	0x00C1	**Paritätsfehler- und Wortzähler (LIO)**
	Beschreibung	Status des Paritätsfehler- und Wortzählers. Siehe auch **NPn.e**.
	Register	---- ---- HHHH HHHH LLLL LLLL CCCC CCCC
	Reset Wert	---- ---- 0000 0000 0000 0000 0000 0000
	Einträge	**H** Paritätszähler für das höherwertige Byte
		L Paritätszähler für das niederwertige Byte
		C Wortzählerstand (hoch- und niederwertige Bytes)

NLE	0x00C2	**Anzahl der FIFO Einträge (LIO)**
	Beschreibung	Anzahl der 16-Bit-Wörter FIFO Einträge.
	Register	---- ---- ---- ---- ---- ---- EEEE EEEE
	Reset Wert	---- ---- ---- ---- ---- ---- 0000 0000

B AliRoot Simulation

B.1 Quellkode-Übersicht

Die Simulationsumgebung ist in drei Module aufgeteilt (Abbildung 3.2) und wird als Bibliotheken zu der AliRoot-Umgebung eingebunden. Auf dem beigelegten Datenträger befinden sich unter dem Verzeichnis simulation die in Tabelle B.1 aufgelisteten Unterverzeichnisse.

AliRoot	**Sktiptdateien zur Ereigniserzeugung mit AliRoot**
Config.C	Konfigurationsdatei für AliRoot mit z. B. der Anzahl der zu erzeugenden Spuren.
grun.C	Erzeugung eines Ereignisses
slowDigitsCreate.C	Fügt der Root-Datei mit den Ereignisdaten die ADC-Werte des TRDs hinzu.
AliTRDmacros	**Skripte zur Analyse und Auswertung der Monte-Carlo Daten, des lokalen Trackings und der Globalen Tracking-Einheit. Eine Übersicht der Ergebnisse ist in Kapitel B.2 zu finden.**
global.h	Globale Definitionen für die Skripte.
AliTRDnh_<*mc*>.C	Generiert die unter *mc* in Kapitel B.2 gelisteten Histogramme.
AliTRDfit	**Repräsentation des linearen Fits für eine Padzeile**
AliTRDfitConst.h	Globale Konstanten, wie z. B. Anzahl der Zeitabschnitte.
AliTRDfitPadrow	Darstellung einer Padzeile. Da die Algorithmen über benachbarte Chips arbeiten, werden in der Simulation die TRAPs nicht separat abgebildet.
AliTRDfitTracklet	Klasse zur Darstellung eines durch TRAP gefundenen Spursegments
AliTRDfitType	Definition zur in Hardware implementierten Arithmetik
AliTRDgtu	**Abbildung eines Moduls, Aufbereitung der Monte-Carlo Daten**
AliTRDgtu	Klasse zur Instantiierung des gesamten Detektors aus Blöcken (Modulen).
AliTRDgtuBlock	Darstellung eines Blocks (Moduls) aus Link-Verbindungen und TMUs und Visualisierung eines Moduls
AliTRDgtuCandidate	Spursegmentdarstellung, so wie sie über die Links übertragen werden.
AliTRDgtuConst.h	Globale Konstanten, wie z. B. Bitbreiten oder Anzahl der maximal übertragbaren Spursegmente.
AliTRDgtuGlobal	Globale Funktionen
AliTRDgtuLink	Darstellung eines Optischen Links aus Kandidaten (Spursegmenten).
AliTRDgtuReal	Berechnung und Zusammenfassung der Monte-Carlo Spursegmente zu einem Ereignis.
AliTRDgtuRealTrack	Darstellung eines Spursegments anhand der Monte-Carlo Daten.

AliTRDgtuStatistic	Analytische Funktionen, die von den Makros aufgerufen werden. Diese Funktionen sind zum Teil sehr rechenintensiv und liegen daher in kompilierter Form vor.
AliTRDgtuTMU	Repräsentiert eine Track-Matching Einheit der GTU, die auf einer Padzeile über alle Lagen arbeitet.
AliTRDgtuTrack	Darstellung einer Spur, die durch die GTU aus Kandidaten (Spursegmenten) zusammengesetzt wurde.
AliTRDgtuVisual	2D und 3D Visualisierung von Kandidaten und Spuren.
AliTRDtrap	**Nachbildung des im Hardware implementierten Filters**
AliTRDtrap	Instantiiert die Filter für die 21 Kanäle eines TRAPs.
AliTRDtrapConst.h	Globale Konstanten.
AliTRDtrapFilter	Modell des implementierten digitalen Filters für einen Kanal.

Tabelle B.1: Übersicht der Verzeichnisse in der Simulationsumgebung.

B.2 Übersicht der Simulationsergebnisse

chi2	**distribution of chi2 vs. threshold**
h1_1	distribution of chi2 vs. threshold (all, primary, high pt primary particles)
comp	**efficiency vs. pt**
h1_1	efficiency vs. pt
defl	**number of found tracklets vs. pt (absolute, relative)**
h1_1_1	absolute number of found tracklets
h2_1_1	relative number of found tracklets
h3_1_1	number of tracklets vs. detector number for different deflection cuts
h4_1_1	number of tracklets per chamber distribution for different deflection cuts
h5_1_1	number of high pt primary MC track segments not found vs. detector number for different deflection cuts
digit	**digit values vs. pt (all particles, primary particles).**
h1_1	digit values vs. pt
eff	**efficiency vs. threshold and pt**
h1_1_1	efficiency vs. threshold and pt
h1_2_1	efficiency vs. threshold and pt (primary particles)
eff2	**efficiency vs. HitsMin and pt**
h1_1_1	efficiency vs. HitsMin and pt
h1_2_1	efficiency vs. HitsMin and pt (primary particles)
eff3	**efficiencies (vs. pt and theta)**
h1_1_1	efficiency vs. pt
h1_2_1	efficiency vs. theta
eff4	**pretrigger efficiency vs. pt**
h1_1_1	pretrigger efficiency vs. pt

gen	**some general results like deflections, number of timebins, ...**
h1_1	number of hits (clusters) for tracklet
h1_4	number of planes where tracklets with same track id were found
h1_7	number of plane where track was found vs. pt
h1_8	number of planes where track was found vs. pt
h1_10	number of track ids involved in single tracklet
h1_13	number of columns where track was found in the same plane
h2_1	tracklet deflection vs. pt
h2_2	tracklet deflection vs. pt (primary particles)
h2_3	mean deflection calculated by pt and position
h2_4	mean deflection calculated by pt and position (primary particles)
h2_5	difference of deflection
h2_6	difference of deflection (primary particles)
h2_9	distribution of chi2 (standard deviation)
h2_11	variance of difference of deflection vs. pt
h2_13	ABS(difference of deflection) vs. chi2
lay	**projection of tracklets to reference plane**
h1_1	projection of fitted tracklets to the reference plane
h1_2	projection of MC track segments to the reference plane
pt	**pt distribution and fit resolution in pt**
h1_1	pt distribution of fitted tracklets
h1_2	pt distribution of MC track segments
h1_3	distribution of difference of fit pt and MC pt
h1_4	normalized difference of pt vs. pt
h1_5	difference of deflection vs. pt
h1_6	normalized difference of deflection vs. normalized difference of pt
h1_7	standard deviation of normalized difference of deflection vs. pt
h1_8	variance of normalized difference of deflection vs. pt
realhist	**statistics of MC data by using AliTRDgtuReal class**
h1_1	average energy loss vs. pt
h1_5	pt distribution
h1_7	mean deflection vs. pt
h1_8	deflection of MC tracklets in AliTRDgtuReal class
h2_1	number of track segments vs. theta
h2_2	number of planes for track
res	**residuae for different thresholds**
h1_1	residuae vs. threshold
h2_1	residuae for different thresholds
res3	**residuae, quantisation errors, slope errors, amplitudes distribution, ...**
h1_1	fit residuae
h1_2	cluster position error by quantisation
h1_3	ampR - ampL
h1_4	ampL + ampC + ampR
h1_5	clusters per tracklet vs. threshold

h1_6	ampC
h1_7	average (ampR + ampL) / (ampL + ampC + ampR)
h1_8	slope error by linear regression using residuae
h1_9	slope error by linear regression using cluster position error
h1_10	cluster position error by quantisation vs. clusters sum
h1_11	number of ghost tracks vs. theta
h1_12	sigma of residuae for single tracklets
res4	**correlations of the min. distance to the next track segment**
h1_1	residuae vs. distance to next track segment
h1_2	slope error vs. distance to next track segment
res5	**residuae and slope difference also to MC tracks**
h1_1	residuae to fit line (all, primary, high pt particles)
h1_2	residuae to MC track segment (all, primary, high pt primary particles)
h1_3	position error by quantisation distribution
h1_4	slope difference (all, primary, high pt primary particles)
row	**some dependencies to theta**
h1_1	number of tracks vs. theta
h1_2	number of tracks in more than one padrow vs. theta
h1_3	relative number of tracks in more than one padrow vs. theta
h1_4	relative energy loss in single padrow vs. theta
h1_5	relative energy loss in single padrow vs. theta for cluster threshold=15
h1_6	relative number of hits in single padrow vs. theta, th=15
h1_7	relative timebin difference for main padrow vs. theta, th=15
tr	**energy deposition distribution within drift region**
h1_1	cluster sum vs. time for isolated tracklets of electrons and pions

Tabelle B.2: Übersicht der Simulationsergebnisse.

C Prototypen

In diesem Kapitel werden die im Laufe dieser Arbeit entstandenen Prototypen-Chips vorgestellt. Während mit dem ersten Prototyp FaRo, der in einem AMS-0,35-µm-CMOS-Prozess gefertigt wurde, in erster Linie Erfahrungen mit digitalem Chip-Design gesammelt wurden, stellen TRAP1 - TRAP3 die im UMC-0,18-µm-Prozess gefertigten Entwicklungsversionen des später produzierten Tracklet-Prozessors dar. Für die beiden Chips FaRo und TRAP3 sind außerdem Übersichten zu den Quellen gegeben, die auf dem beigelegten Datenträger zu finden sind. Hierbei sind ausschließlich diejenigen Quellen vorhanden, die auch im Rahmen dieser Arbeit entwickelt wurden.

C.1 FaRo

Der erste entwickelte Chip am Lehrstuhl für Technische Informatik ist FaRo. Dieser diente vorrangig dazu, um Erfahrungen mit dem Entwurf digitaler Chips zu sammeln.

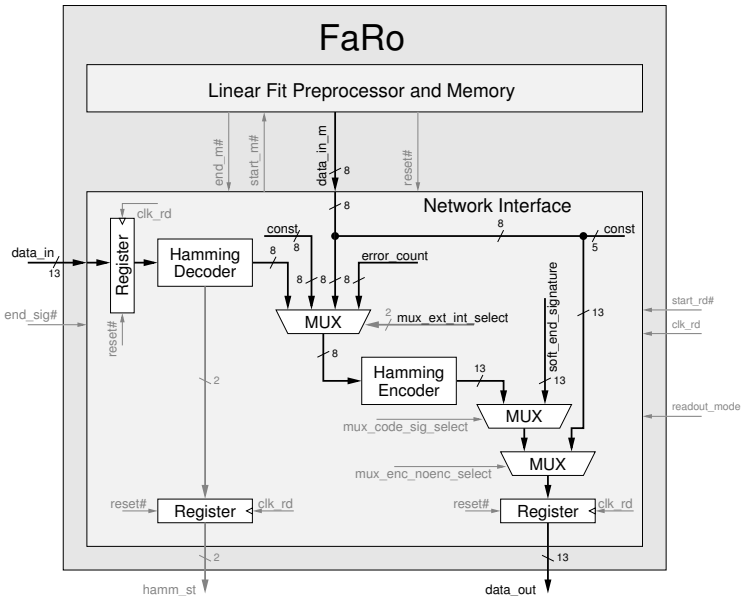

Abbildung C.1: Die Netzwerkschnittstelle von FaRo verarbeitet einen externen und einen internen Datenstrom. Zur Fehlertoleranz werden die Daten Hamming-kodiert. Auch FaRo arbeitet ohne Handshake-Protokoll mit einer Signatur zum Erkennen des Datenstromendes.

In FaRo ist eine erste Variante des Präprozessors [21] (Kapitel 4.1.3) und eine einfache Variante der Netzwerkschnittstelle implementiert. Der Präprozessor (Abbildung C.1 oben) verarbeitet acht Kanäle und zwei Nachbarkanäle. Anhand der ADC-Werte werden die so genannten „Hits“ gebildet und die Summen für einen linearen Regressionsfit bestimmt. Die Netzwerkschnittstelle kann auf diese Daten über einen Adress-/Datenbus direkt zugreifen.

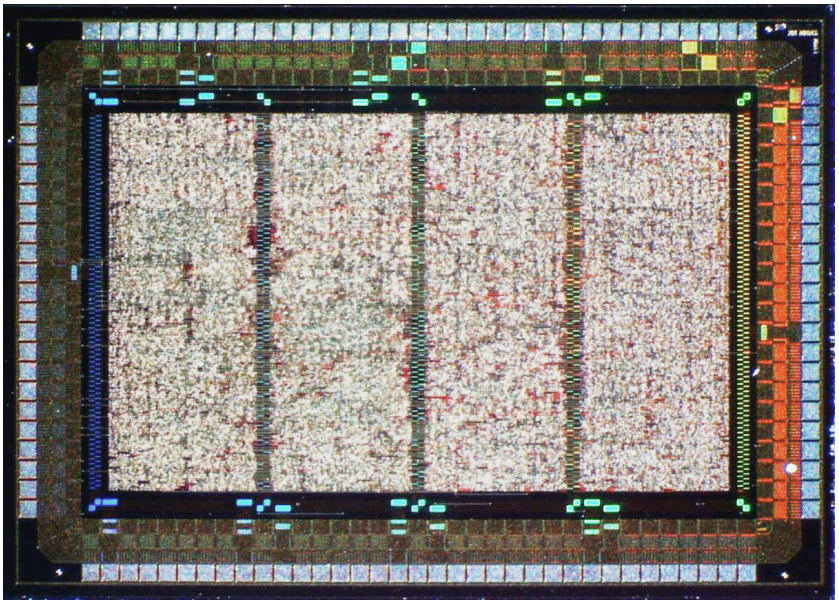

Abbildung C.2: Foto des ersten Prototypen FaRo, submittiert im März 2000 im AMS-0,35-µm-CMOS-Prozess.

Neben den internen Daten vom Präprozessor verfügt die Netzwerkschnittstelle über einen Eingangport, dessen Datenstrom zur Fehlerkorrektur Hamming-kodiert ist. Die Schnittstelle dekodiert und korrigiert gegebenenfalls die Daten. Nach einer erneuten Hamming-Enkodierung werden die Daten weitergeleitet. Danach werden die internen Daten abgerufen, enkodiert und ebenfalls ausgegeben. Das Ende der Übertragung wird durch eine Signatur gekennzeichnet. Über den externen Pin end_sig# wird definiert, dass das entsprechende Modul anfängt, die eigenen Daten zu senden, ohne auf eine Signatur zu warten.

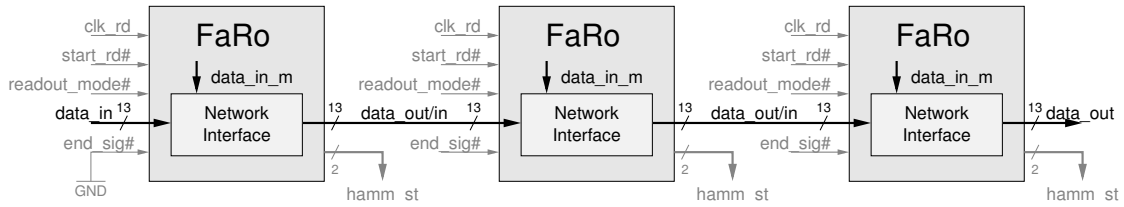

Abbildung C.3: Die Netzwerkschnittstelle von FaRo ermöglicht den Aufbau einer linearen Auslesekette. Der zuerst sendende Chip wird durch eine Masseverbindung von end_sig gekennzeichnet. Die übrigen FaRos reagieren auf eine Signatur im Datenstrom.

Mit dieser Funktionalität ist es möglich, eine Auslesekette aus FaRos aufzubauen. Wie in

Abbildung C.3 dargestellt, ist das end_sig#-Signal des linken Moduls auf Masse gezogen. Dieses bildet damit den Beginn der Auslese. Das mittlere Modul leitet diese Daten weiter, schickt dann die internen Daten und schließt die Übertragung mit dem Senden der Signatur ab.

FaRo wurde im März 2000 in einem AMS-0,35-µm-CMOS-Prozess submittiert. Für das Design wurde der Compiler von Synopsys 1999.10 verwendet. FaRo besteht ausschließlich aus Standardzellen, so dass sich das Layout auf eine automatische Platzierung (Sea-Of-Gates) und die Infrastruktur beschränkt. Das Layout ist mit Silicon Ensemble 5.2.137 von Cadence erstellt.

C.1.1 Quellen

Auf dem zu dieser Arbeit gehörigen Datenträger befinden sich die VHDL-Quellen zu der Netzwerkschnittstelle von FaRo und die für das Layout relevanten Dateien.

faro/	Quellen zum Prototypen FaRo.
layout/	Silicon Ensemble Design-Fluss.
CTGEN/	Definitionen für den Taktbaumgenerator.
DEF/	Definition der Spannungsversorgungen.
csd3.3V.gcf	Referenz auf die Timing-Bibliothek.
ctgen.cmd	Kommandos für den Taktbaumgenerator.
faro_io.ioc	I/O-Zellen.
netcomp_v.cmd	Kommando zum Netzlistenvergleich.
*.mac	Makros für Silicon Ensemble.
src	VHDL-Quellen für die Netzwerkschnittstelle.
Global.vhd	Globale Definitionen.
Hamming_Decoder.vhd	Hamming-Dekoder für die Eingänge.
Hamming_Encoder.vhd	Hamming-Enkoder zur Ausgabe.
MainStage.vhd	Datenpfade des Toplevels.
TopControl.vhd	Kontrolleinheit für den Toplevel.
TopTree.vhd	Toplevel.
*.vhd	VHDL-Quellen, wie z. B. Multiplexer oder Register.

Tabelle C.1: Quellenübersicht zur Netzwerkschnittstelle von FaRo und dem Layout.

C.2 TRAP1

Im ersten Prototypen des Tracklet-Prozessors TRAP1 sind bis auf die ADCs schon alle benötigten Module enthalten. TRAP1 beinhaltet drei ADCs, die mit unterschiedlichen Parametern entworfen und mit eigenen Pads zum Abgreifen interner Signale versehen sind. Dies ermöglicht genaue Untersuchungen von Prozessparametern und Optimierungen für die spätere Integration. Die Eingänge des Präprozessors sind digital ausgeführt und können

durch Multiplexen den gesamten Filter bedienen.

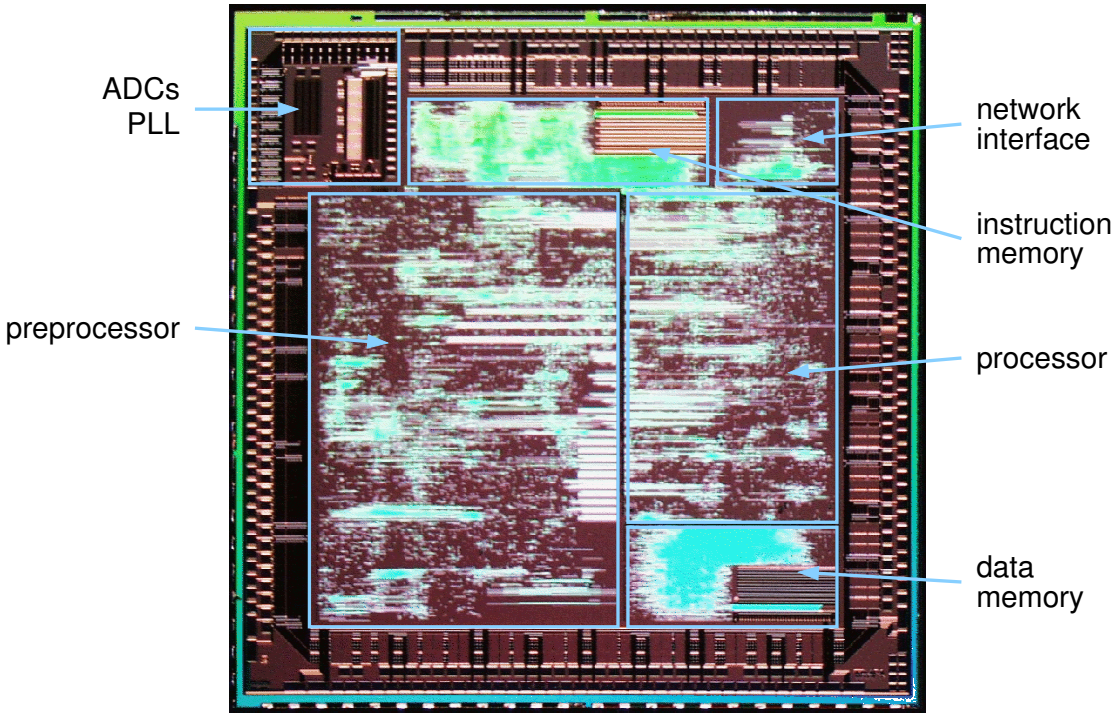

Abbildung C.4: Foto des ersten Prototypen von TRAP, submittiert im April 2002 im UMC-0,18-µm-Prozess.

Als Instruktions- und Datenspeicher wird das eigens entwickelte Quad-Port-Memory (Kapitel 4.1.4) eingesetzt. Um bei auftretenden Problemen mit den Speichern dennoch die volle Funktionalität des TRAPs testen zu können, sind kleine Bereiche der Speicher zusätzlich Register-basiert implementiert und in den Speicherbereich abgebildet.

TRAP1 ist im April 2002 im UMC-0,18-µm-Prozess submittiert worden. Für die Synthese wurde Synopsys 2001.08-SP1 und für das Layout Silicon Ensemble 5.3.116 von Cadence benutzt. Der Design-Fluss für das Layout wurde vollständig im Rahmen dieser Doktorarbeit entwickelt.

C.3 TRAP2

Im zweiten Prototypen TRAP2 sind einige Erweiterungen und Korrekturen gegenüber TRAP1 vorgenommen worden. Zwar sind alle 21 ADCs integriert, jedoch noch mit unterschiedlichen Parametern, da die Anbindung an den Vorverstärker (PASA) (Kapitel 4.1.1) sehr genau angepasst sein muss. Die übrige Funktionalität des TRAPs lässt sich damit dennoch ohne Einschränkungen testen. Zwischen TRAP2 und der dritten Version ist außerdem noch ein Prototyp (TRAPADC) entwickelt worden, um separat die ADCs und auch das Quad-Port-Memory weiter zu optimieren. Am Datenspeicher wurde das Timing für den

Zugriff verändert, um kritische Pfade im Prozessor-Design zu entspannen. Außerdem ist in TRAP2 der Instruktionsspeicher (IMEM), der bislang ebenfalls als Quad-Port-Memory ausgelegt war, durch vier Speicher von jeweils 4 KByte (Kapitel 4.1.4) von VST ersetzt worden. Dies ermöglicht neben einer flexibleren Programmierung der einzelnen CPUs einen schnelleren Zugriff auf den Speicher. Dafür muss als Nachteil ein etwas höherer Platzbedarf in Kauf genommen werden.

Die Ergebnisse des Präprozessors werden dem Prozessor über eine spezielle Registerbank (Abbildung 4.9 und 4.10) zur Verfügung gestellt. Diese Register sind überarbeitet worden, da es zu Problemen beim zeitlichen Verhalten kam. Außerdem sind die Ereignisspeicher des Filters (Abbildung 4.6) anstatt über den global Bus jetzt über die lokalen Schnittstellen erreichbar. Damit können die CPUs alle gleichzeitig und auch schneller auf diese Daten zugreifen. Dies beschleunigt in erster Linie die Rohdatenauslese und die Prozessoren können sich durch das schnellere Ablegen der Daten in die Netzwerkschnittstelle früher abschalten und damit Strom sparen.

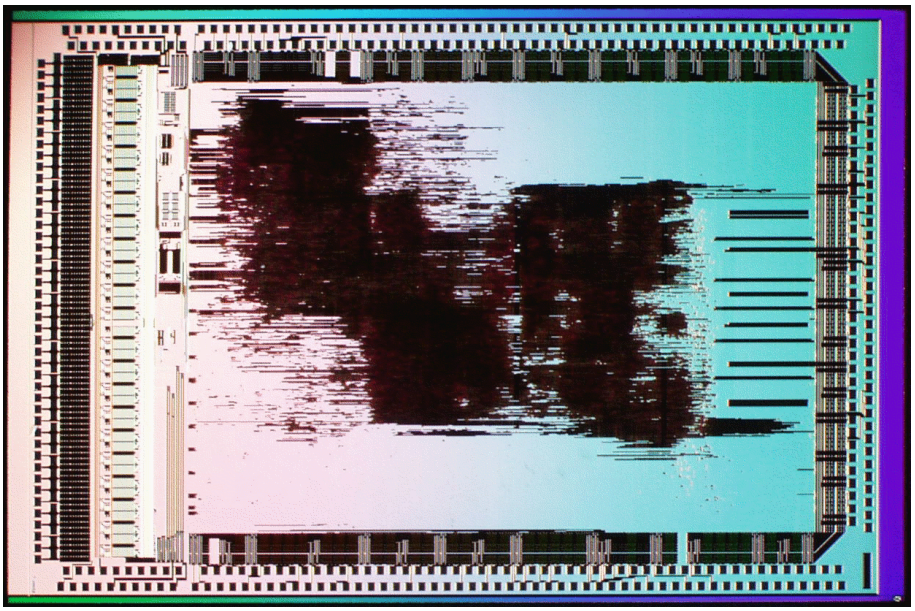

Abbildung C.5: Foto des zweiten Prototypen von TRAP, submittiert im April 2003 im UMC-0,18-µm-Prozess.

Am Prozessor wurde ein Bug im Anschluss des Dividierers behoben, die Konstantenregister breiter ausgelegt und um Zähler (Counter und Timer) ergänzt, mit denen Interrupts ausgelöst werden können. Da mit diesen Interrupts auch eine CPU aufgeweckt werden kann, die gerade abgeschaltet ist, ist es möglich den Strombedarf weiter zu senken. Die CPUs können damit sowohl durch ein Ereignis als auch nach einer bestimmten Zeit wieder eingeschaltet werden. Außerdem wurde der Arbiter für den Zugriff auf den globalen Bus modifiziert, so dass er jetzt synchron arbeitet. Damit wurden zeitlich kritische Pfade eliminiert. Ein Schreibzugriff dauert damit 2 Taktzyklen, während ein Lesezugriff nach 3 Zyklen abgeschlossen ist.

Die für den TRAP speziell entwickelten LVDS-Zellen (Kapitel 5.1) wurden überarbeitet (SUPER-LVDS) und um eine integrierte JTAG-Logik erweitert. Dies ist vor allem deshalb besonders elegant, da die JTAG-Ringe dem Ring der Ein-/Ausgabezellen entspricht und bei geeigneter Verdrahtung auch das zeitliche Verhalten optimal ist. Außerdem war bei der bisherigen Implementierung die zu den Ein-/Ausgabezellen zugehörige JTAG-Logik über den gesamten Chip verstreut.

In der Netzwerkschnittstelle ist ein Fehler im Zustandsautomaten für die Abarbeitung der Rohdatenauslese entfernt worden. Dieser Fehler wurde durch unglückliche Umstände in der Simulation nicht erkannt. Zwar hätte die Auslese auch mit diesem Fehler funktioniert, wäre aber als Einschränkung deutlich langsamer gewesen und hätte erheblich mehr Stromverbrauch mit sich gebracht, da dann Teile des Ablaufs durch eine CPU hätten gesteuert werden müssen.

TRAP2 wurde im April 2003 submittiert. Im Gegensatz zu TRAP1 ist TRAP2 mit dem neuen Programm von Cadence für das Layout – FirstEncounter Version v02.30-s045_1 – entwickelt worden. Für die Synthese wurden weiterhin die Programme von Synopsys in der Version 2003.03 benutzt. Außerdem fand für TRAP2 eine vollständige Analyse des zeitlichen Verhaltens nach dem Layout mittels Primetime statt. Dies war erst durch die entsprechenden Ausgabedateien (Abbildung 4.12) durch FirstEncounter möglich.

C.4 TRAP3

Für TRAP3 wurden die ADCs nochmals verbessert, indem die Aufteilung der Zyklen zur Erzeugung der digitalisierten 10-Bit-Werte (Kapitel 4.1.1) optimiert wurden. Durch den Einsatz einer 240-MHz-PLL erhält man eine höhere Granularität der Zyklen und kann so den kritischen Phasen des Wandelns entsprechend mehr Zeit geben.

Die Filter wurden um Indikatoren ergänzt, die noch während der Driftzeit bestimmt werden und interessante ADC-Werte markieren. Damit wird der Prozessor bei der späteren Bestimmung der Spurparameter entlastet. Um entstandene kritische Pfade zu vermeiden, sind die Schnittstellen zum Präprozessor angepasst worden. Außerdem werden nun alle Parameter des Filters und Präprozessors bei einem Reset mit vorgegebenen Werten versehen. Damit kann die Konfiguration deutlich beschleunigt werden.

Die Verdrahtung der mit den SUPER-LVDS-Zellen aufgebauten JTAG-Ring musste korrigiert werden. Außerdem war ein Fehler im SCSN-Ring 1 (Kapitel 4.1). Dieser funktionierte zwar, war jedoch in der Geschwindigkeit stark eingeschränkt. Durch die doppelte Auslegung der Konfigurationsschnittstelle konnte jedoch der Chip auch in voller Geschwindigkeit (z. B. während der Strahlzeit, Kapitel 7.3.3) in Betrieb genommen werden

Die CPUs wurden um einige Befehle ergänzt, wie z. B. mehrere Sprungbefehle, durch die auf Statusflags direkt reagiert werden kann und somit Zeit in der Abarbeitung sparen. Das zeitliche Verhalten des Multiplizierers ist durch das Einfügen einer Pipelinestufe entschärft worden. Während in den vorhergehenden Versionen das Ergebnis einer Division genau nach 18 Taktzyklen zur Verfügung stand (Kapitel 4.1.4), wird jetzt das Ergebnis gehalten und

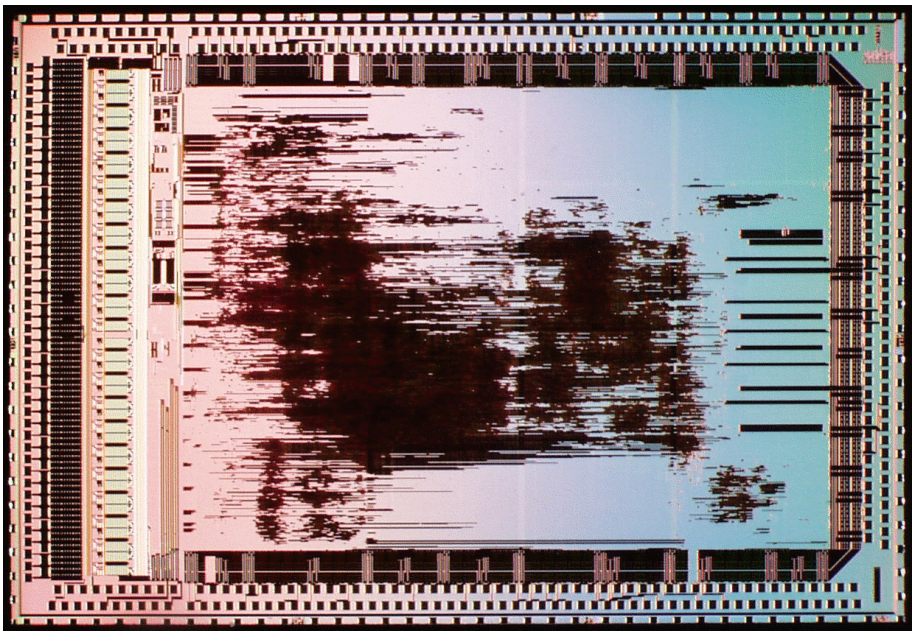

Abbildung C.6: Foto des dritten Prototypen von TRAP, submittiert im April 2004 im UMC-0,18-µm-Prozess.

dies der entsprechenden CPU mitgeteilt. Als sehr nützlich erweist sich ein weiterer Speicherblock (VST), auf den über die globale Schnittstelle von allen CPUs aus zugegriffen werden kann. Damit lassen sich Parametersätze, Wertetabellen usw. effizient implementieren. Der Datenspeicher und der Befehlssatz des Prozessors sind für TRAP3 außerdem um einen Byte-weisen Zugriff erweitert worden.

TRAP3 wurde im April 2004 submittiert. Wie für TRAP2 wurde für die Synthese und Analyse des Zeitverhaltens Synopsys 2003.03 eingesetzt. Für das Layout wurde FirstEncounter in der Version v03.20-s014_1 benutzt.

C.4.1 Quellen

Im Folgenden wird ein Überblick über die auf dem beigelegten Datenträger zusammengestellten Quellen zu TRAP3 und im Speziellen zur Netzwerkschnittstelle gegeben.

Netzwerkschnittstelle Im Unterverzeichnis trap3/TOP_TM/ befinden sich die Quellen zur Beschreibung der Netzwerkschnittstelle und die Skripte zur Synthese und zur zeitlichen Verifikation. Tabelle C.2 gibt eine vollständige Auflistung der VHDL-Quellen. Die Speicher der FIFOs (Kapitel 6.2.5) sind mithilfe des Speichergenerators von VST erzeugt. Eine genaue Schnittstellenbeschreibung dieser Speicher und die Skripte zur Erstellung der Bibliotheken, die das zeitliche Verhalten beschreiben, sind im Verzeichnis trap3/TOP_TM/BLOCK/ zu finden.

SRC/	VHDL-Quellen zur Netzwerkschnittstelle.
ni_delay_bit.vhd	Verzögerung einer Bit-Leitung.
ni_delay.vhd	Einzelne Verzögerungseinheit.
ni_delay_word.vhd	Verzögerungseinheit mit einstellbarer Breite und Tiefe.
ni_exclude_in.vhd	Verarbeitung des Reserve- und Paritätsbits am Eingangsport.
ni_exclude_out.vhd	Erzeugung der Reserve- und Paritätsbits am Ausgangsport.
ni_fifo_mb.vhd	Kontrolllogik für das aus dem VST-Speicher gebildeten FIFO.
ni_fsm.vhd	Zentraler Zustandsautomat zur Steuerung der gesamten Netzwerkschnittstelle.
ni_global.vhd	Globale Variablen, Typdeklarationen, Funktionen und die Reset-Werte aller Kontrollregister.
ni_io_interface_g.vhd	Globale Schnittstelle zum Prozessor.
ni_io_interface.vhd	Lokale Schnittstelle zu einer CPU.
ni_port_in.vhd	Eingangsport.
ni_port_out.vhd	Ausgangsport.
ni_register.vhd	Einfaches synchrones Register.
ni_register_we.vhd	Synchrones Register mit synchronem Schreiben.
ni_resync.vhd	Resynchronisation der Eingangsdaten zum internen Takt.
ni_sync.vhd	Erzeugung des synchronen DDR-Datenausgangsstroms.
ni_top_blkbx.vhd	Leeres VHDL-Modell des Toplevels, das für das Layout benötigt wird.
ni_top.vhd	Toplevel der Netzwerkschnittstelle.

Tabelle C.2: Quellenübersicht zur VHDL-Beschreibung der Netzwerkschnittstelle.

Das im Verzeichnis trap3/TOP_TM/SYN/SCRIPTS/ befindliche Synthese-Skript lässt sich entweder direkt in Synopsys ausführen oder mithilfe des Makefile im übergeordneten Verzeichnis. In beiden Fällen sind jedoch für TRAP3 global angelegte Skripte aus dem Verzeichnis trap3/scripts/ notwendig.

Gerade die Verzögerungseinheit der Netzwerkschnittstelle ist im Bezug auf das zeitliche Verhalten sehr kritisch. Aus diesem Grund wurden das zeitliche Verhalten der Eingangsports und des Ausgangsports genauer betrachtet. Im Verzeichnis trap3/TOP_TM/PT/ befinden sich die Skripte für diese Analysen. Um exakte Aussagen über das Verhalten machen zu können, sind hierfür allerdings die Netzliste und die Beschreibung des zeitlichen Verhaltens des gesamten Tracklet-Prozessors Voraussetzung. Diese Auswertungen können ebenfalls mittels eines Makefile erzeugt werden. Außerdem werden dabei gleich grafische Darstellungen wie z. B. Abbildung 6.6 erzeugt.

Simulation Bei der Simulation wird zwischen einer Betrachtung der Netzwerkschnittstelle allein und einer Zusammenschaltung von Netzwerkschnittstellen, um das Verhalten einer Ausleseplatine zu beschreiben, unterschieden. Außerdem wurde auch der gesamte Tracklet-Prozessor und eine entsprechende Zusammenschaltung von TRAPs simuliert. Da hierzu

allerdings die Quellen aller übrigen Module und die Simulationsmodelle von Drittherstellern Voraussetzungen sind, werden diese Umgebungen hier nicht behandelt. Im Verzeichnis trap3/SIM/ findet sich folgende Verzeichnisstruktur:

PROJECTS/	Modul-spezifische Simulationen.
ni/	Netzwerkschnittstelle.
EXPECTED/	Textdateien der erwarteten Ausgaben der Schnittstelle zum Vergleich mit den simulierten Ausgaben.
REPORTS/	Log-Dateien vom Compiler und den Simulationen.
RESULTS/	Ausgabe der Simulationsergebnisse als Textdateien.
SCRIPTS/	Skripte zur übersichtlicheren Darstellung der Ausgaben und der Wellenformen in ModelSim.
SRC/	Testbenches für die Netzwerkschnittstelle, Beschreibung einer Ausleseplatine und Hilfsfunktionen.
STIMULI/	Eingangsvektoren für die Schnittstellen zu den CPUs, den Eingangsports und dem Ausgangsport.
Makefile	Zur Ausführung aller Simulationsvarianten.
SHARE/	Modul-übergreifende Simulationen.
SRC/	Testbench-Quellen.
ni_sniffer.vhd	Greif die Vektoren am Ausgangsport ab und speichert sie als Textdatei. Sie dient zum Vergleich mit den erwarteten Daten.

Tabelle C.3: Quellenübersicht zur Simulation der Netzwerkschnittstelle.

Für eine Simulation der Netzwerkschnittstelle als Gatternetzliste sind außerdem die Modelle der Standardzellen von VST notwendig.

Layout Im Verzeichnis trap3/TOP/LAYOUT/FE/ sind die Skripte für das mit FirstEncounter erstellte Layout des TRAP Chips zu finden. Hierbei wird jedoch ebenfalls sowohl auf die Standardzellen von VST als auch auf die nicht im Rahmen dieser Arbeit entwickelten Module verzichtet. Der Ablauf zum Erstellen des Layouts lässt sich am besten durch das Makefile erkennen. In Kapitel 4.2.6 ist dieser Ablauf mit den im Makefile benutzten Bezeichnungen näher beschrieben.

CFG/	Konfigurationsdateien.
CTS/	Definitionen der Taktbäume und der Abhängigkeiten.
FLOORPLAN/	Alle von Hand platzierten Module und Leitungen.
POWER/	Definition der Netze für die Versorgungsspannungen.
ImportDesign.conf	Import des Designs und globale Einstellungen von FirstEncounter wie z. B. Referenzen auf die Bibliotheken.
streamOut.map	Lagendefinitionen für den GDSII-Export.
top_pad.io	Anordnung der I/O-Zellen.
*.tc	Vorgaben zum zeitlichen Verhalten (timing constraints).
GDS/	Verweise auf die Full-Custom-Designs im GDSII-Format.

LEF_BLOCKS/	Verweise auf die leeren Full-Custom-Blöcke. Sie enthalten lediglich die Größe, die Pins und die benutzten Metalllagen.
LEF_VST/	Verweise auf die Standard- und I/O-Zellenbibliothek von VST.
LIB/	Verweise auf Modelle zur Beschreibung des zeitlichen Verhaltens der Makro-Blöcke für verschiedene Umgebungen.
NETLIST/	Mittels eines Makefile werden hier alle Netzlisten in eine einzige zusammengefasst. Diese bildet die Eingangsnetzliste für FirstEncounter.
SCRIPTS/	TCL-Skripte für FirstEncounter. Diese Skripte werden durch das Makefile gesteuert, in definierter Reihenfolge abgearbeitet.
Makefile	Ablaufsteuerung des gesamten Design-Flusses für das Layout des TRAPs.

Tabelle C.4: Quellenübersicht zu den Skripten und Konfigurationsdateien zum Erstellen des Layouts des TRAP Chips mit FirstEncounter.

Wie in Kapitel 4.2.7 beschrieben, kann mithilfe eines „Layout Versus Schematic“ sichergestellt werden, dass das fertige Layout auch nach allen Optimierungsschritten noch logisch der Eingangsnetzliste entspricht. Hierzu wird das Programm Formality von Synopsys verwendet. Im Verzeichnis trap3/TOP/LAYOUT/FM/ befinden sich die hierfür notwendigen Dateien. Dieser Verifikationsschritt kann ebenfalls durch ein Makefile ausgeführt werden.

Zwar wird innerhalb von FirstEncounter auch eine Analyse des zeitlichen Verhalten unter Berücksichtigung der Leitungskapazitäten durchgeführt, jedoch hat FirstEncounter hierbei eine Schwäche. Werden keine Vorgaben zum zeitlichen Verhalten gemacht oder übersehen – und sei es z. B. nur für ein einziges Register – werden alle Pfade, die davon direkt oder indirekt abhängen, nicht berücksichtigt und auch keine Warnung ausgegeben. Aus diesem Grund und auch, weil das Timing-Analyse-Programm Primetime von Synopsys eine komfortable Skript-Schnittstelle zur Verfügung stellt, werden nach dem Layout hiermit noch detaillierte Untersuchungen der kritischen Bereiche des Design für alle Umgebungen (worst case, typical und best case) durchgeführt. Die Skripte hierzu und auch das übergeordnete Makefile sind im Verzeichnis trap3/TOP/LAYOUT/PT/ zu finden.