RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

Konstantinos Giapoutzis

LHCb Vertex-Trigger Algorithmus

Diplomarbeit

HD-KIP-02-26

KIRCHHOFF-INSTITUT FÜR PHYSIK

Fakultät für Physik und Astronomie
Ruprecht-Karls-Universität Heidelberg

Diplomarbeit
im Studiengang Physik

vorgelegt von
Konstantinos Giapoutzis
aus Mannheim
2002

LHCb Vertex Trigger Algorithmus

Die Diplomarbeit wurde von Konstantinos Giapoutzis ausgeführt am

Kirchhoff-Institut für Physik

unter der Betreuung von

Herrn Prof. Dr. Volker Lindenstruth

Inhalt

Das Experiment LHCb wird Präzisionsmessungen der CP-Verletzung anhand der Zerfälle von B-Mesonen durchführen. Das Experiment stellt hohe Anforderungen an das Triggersystem, denn nur ein kleiner Bruchteil aller erzeugten Ereignisse ist für eine solche Untersuchung geeignet. Diese Arbeit befasst sich mit dem Rekonstruktionsalgorithmus, der für die Triggerentscheidung verantwortlich ist. Es werden Wege zur Reduzierung der durchschnittlichen Berechnungszeit des Rekonstruktionsalgorithmus presentiert. Dazu gehört die Untersuchung, welche Teile für eine Hardwareimplementierung am besten geeignet sind. Eine mögliche Implementierung für die ausgewählten Teile wird im Verlauf dieser Arbeit beschrieben.

Abstract

The LHCb experiment will perform precision measurements of CP violation on the basis of B meson decays. This experiment makes high demands on the trigger system since only a small fraction of all produced events are eligible for a further analysis. This thesis is concerned with the reconstruction algorithm which is responsible for the trigger decision. Methods for reducing the processing time of the reconstruction algorithm are presented. This includes an investigation which part of the algorithm is best qualified for a hardware implementation. For the selected parts of the algorithm a possible implementation is described.

Inhaltsverzeichnis

Kapitel 1

Das Experiment LHCb

Derzeit entsteht am Europäischen Kernforschungszentrum CERN bei Genf ein neuer Teilchenbeschleuniger, LHC (**L**arge **H**adron **C**ollider) genannt. Der LHC-Beschleuniger wird in der Lage sein, bei einer maximalen Luminosität von $10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$, Proton-Proton-Kollisionen mit einer Schwerpunktsenergie von 14 TeV durchzuführen. Zusätzlich zum Proton Betrieb ist auch ein Schwerionenbetrieb (Blei Ionen) mit einer Schwerpunktsenergie von 73.8 TeV möglich [1]. Dem LHC-Beschleuniger werden vier neue Experimente angehören

ATLAS — **A** **T**oroidal **L**HC **A**pparatu**S**

CMS — **C**ompact **M**uon **S**olenoid

ALICE — **A** **L**arge **I**on **C**ollider **E**xperiment

LHCb — **L**arge **H**adron **C**ollider **b**eauty.

Die Experimente Atlas und CMS konzentrieren sich auf die Suche von Higgs-Bosonen, supersymmetrischen Teilchen und anderen neuen Effekten. Das Experiment Alice wird das Verhalten von Materie bei höchsten Energiedichten im Schwerionenbetrieb des LHC studieren. Das Experiment LHCb hat sich zur Aufgabe gestellt, die CP-Verletzung im B-Mesonen-System zu untersuchen. Eine schematische Darstellung aller am CERN befindlichen Beschleuniger, unter anderem LHC mit seinen vier Experimenten, ist in Abbildung 1.1 zu sehen. Einige Parameter des LHC-Beschleunigers im Proton Betrieb sind in Tabelle 1.1 aufgelistet.

Aufgrund der hohen Schwerpunktsenergie von 14 TeV und der hohen Luminosität ist LHC eine hervorragende Quelle für b-Quarks. So können bei LHCb, bei einer für die B-Meson Erzeugung optimierten Luminosität von nur $2 \cdot 10^{32}\mathrm{cm}^{-2}\mathrm{s}^{-1}$, etwa $5 \cdot 10^{11}\mathrm{B_d}$ und $10^{11}\mathrm{B_s}$ pro Jahr (10^7s) erzeugt werden [3].

1.1 CP-Verletzung im B-Mesonen-System

Um die CP-Verletzung im B-Mesonen-System nachweisen zu können, macht man sich die Tatsache zu Nutze, dass sowohl B^0 als auch $\overline{\mathrm{B}}^0$ Mesonen in den

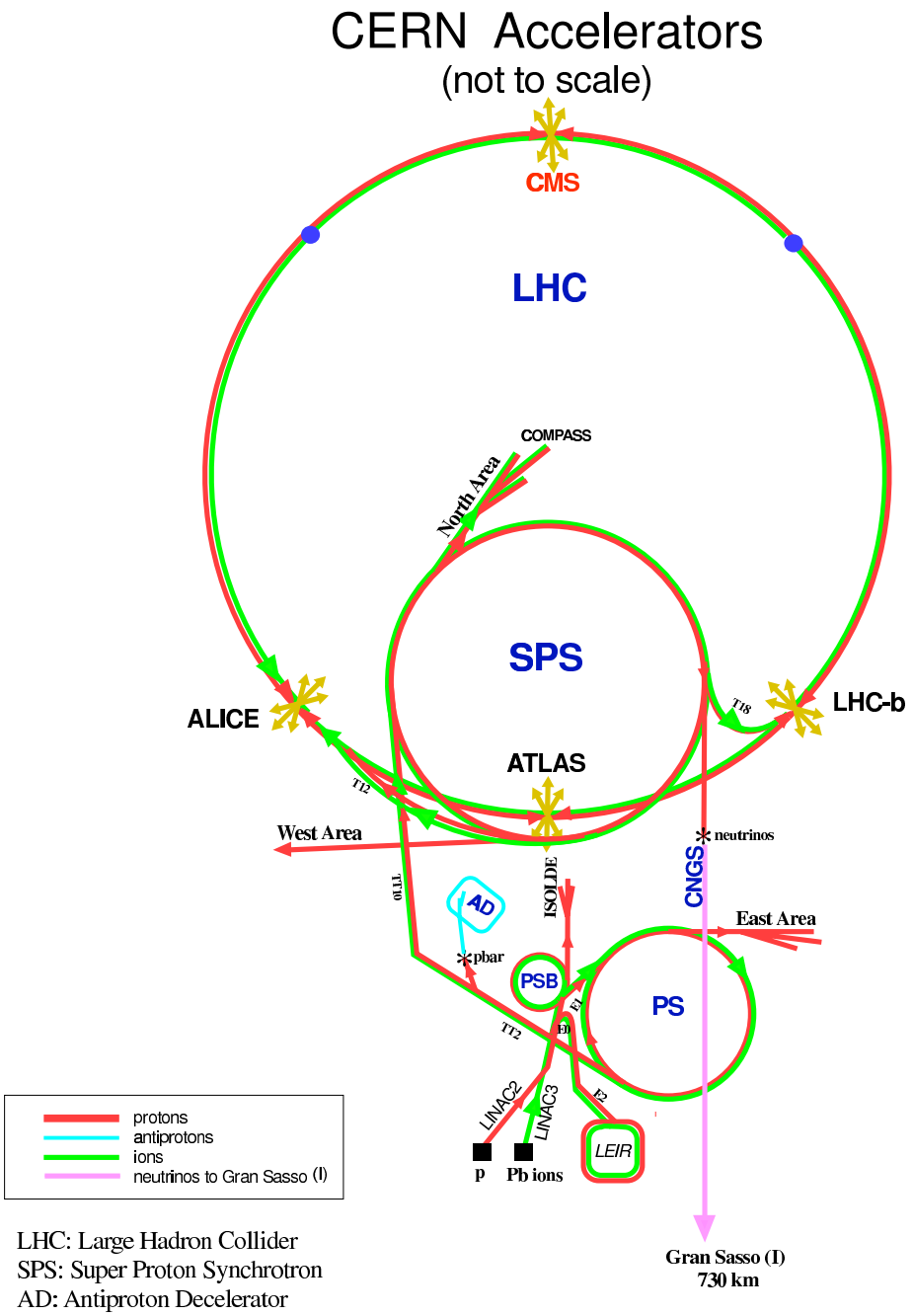

Abbildung 1.1: Skizze der Beschleuniger und der Experimente, die am CERN zu finden sind [2].

LHC Design Parameter im Proton Betrieb	
Energie bei der Kollision	7 TeV
Energie bei der Injektion	450 GeV
Dipolfeldstärke bei 7 TeV	8.33 T
Innerer Radius der Magnetspule	56 mm
Luminosität	$10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$
Stromstärke eines Strahls	0.56 A
Räumlicher Abstand zwischen den Bündeln	7.48 m
Zeitlicher Abstand zwischen den Bündeln	24.95 ns
Zahl der Teilchen pro Bündel	$1{,}1 \cdot 10^{11}$
Lebensdauer der Luminosität	10 h
Energieverlust pro Umlauf	7 keV
Abgestrahlte Leistung pro Strahl	3.8 kW
Gespeicherte Energie pro Strahl	350 MJ
Füllzeit pro Ring	4.3 min

Tabelle 1.1: Design Parameter des LHC-Beschleunigers [1].

selben Endzustand zerfallen können, wenn dieser ein CP-Eigenzustand ist. Wäre CP erhalten, so müssten Teilchen und Antiteilchen mit gleicher Häufigkeit in diese Endzustände zerfallen. Das Verzweigungsverhältnis ginge dann mit steigender Statistik gegen Null. Bei einer CP-Verletzung liegt eine Zerfallsasymmetrie vor, die durch

$$a = \frac{N(\mathrm{B}^0 \to f) - N(\overline{\mathrm{B}}^0 \to f)}{N(\mathrm{B}^0 \to f) + N(\overline{\mathrm{B}}^0 \to f)} \tag{1.1}$$

gegeben ist. Das Standardmodell sagt ein Verzweigungsverhältnis von etwa 10^{-5} voraus. Die genauen Verhältnisse für die einzelnen Zerfallskanäle sind in Tabelle 1.2 zu sehen.

Zerfälle	Verzweigungsverhältnis
$\mathrm{B}^0_\mathrm{d} \to \pi^+\pi^-$	$0{,}7 \cdot 10^{-5}$
$\mathrm{B}^0_\mathrm{d} \to \rho^+\pi^-$	$1{,}8 \cdot 10^{-5}$
$\mathrm{B}^0_\mathrm{d} \to \rho^-\pi^+$	$0{,}7 \cdot 10^{-5}$
$\mathrm{B}^0_\mathrm{d} \to \rho^0\pi^0$	$0{,}7 \cdot 10^{-5}$
$\mathrm{B}^0_\mathrm{d} \to \mathrm{D}^{*-}\pi^+$	$2{,}6 \cdot 10^{-3}$
$\mathrm{B}^0_\mathrm{d} \to \mathrm{J}/\psi\mathrm{K}_\mathrm{s}$	$4{,}4 \cdot 10^{-4}$
$\mathrm{B}^0_\mathrm{s} \to \mathrm{D}^-_\mathrm{s}\mathrm{K}^+$	$2{,}0 \cdot 10^{-4}$
$\mathrm{B}^0_\mathrm{s} \to \mathrm{D}^+_\mathrm{s}\mathrm{K}^-$	$3{,}1 \cdot 10^{-5}$
$\mathrm{B}^0_\mathrm{s} \to \mathrm{J}/\psi\phi$	$9{,}3 \cdot 10^{-4}$

Tabelle 1.2: Verzweigungsverhältnis einiger Zerfälle [4].

1.2 Der Detektor

Der LHCb Detektor ist entworfen worden, um eine große Anzahl von b-Hadronen zu registrieren, damit genaue Studien über die CP Asymmetrie und die Suche nach seltenen Zerfällen durchgeführt werden können. Der Detektor ist ein vorwärts gerichtetes Einzelarm-Spektrometer mit einer Winkelabdeckung von 10 mrad im vorderen Bereich und bis zu 300 mrad im hinteren Bereich in der horizontalen Projektion und bis zu 250 mrad in der vertikalen Projektion. In Abbildung 1.2 ist die Horizontale Projektion und in Abbildung 1.3 die vertikale Projektion des Detektors zu sehen. Der Detektor kann den Vertex aus einem B-Zerfall mit einer sehr guten Genauigkeit rekonstruieren und bietet eine hervorragende Teilchenerkennung für geladene Teilchen. Er besitzt ein Hochleistungs-Trigger, der optimiert ist effizient Ereignisse mit B-Mesonen zu selektieren, beruhend auf Teilchen mit großem transversalen Impuls und verschobenem Sekundärvertex. Der LHCb Detektor besteht aus einer Reihe von einzelnen Detektoren

- Der **Ve**rtex **Lo**cator (VELO) besitzt eine Reihe von Silizium Stationen, die entlang der Strahlachse aufgestellt sind. Sie sind dazu gedacht, genaue Messungen von Spurkoordinaten nahe der Wechselwirkungszone durchzuführen.

- Die Identifizierung von geladenen Teilchen, erfolgt durch zwei Cherenkov-Zähler, einem im vorderen Bereich (RICH1) und einem im hinteren Bereich (RICH2). RICH steht für **R**ing **I**mage **Ch**erenkov Counter. RICH1 verwendet Aerogel und C_4F_{10} als Cherenkov-Radiator[1]. RICH2 verwendet CF_4 als Cherenkov-Radiator. Es werden also insgesamt drei Cherenkov-Radiatoren verwendet, um das gesamte Impulsspektrum zu erfassen.

- Ein Magnetspektrometer wird verwendet, um Impulsmessungen durchzuführen. Er befindet sich in der Nähe der Wechselwirkungszone, damit seine Größe klein gehalten werden kann.

- Die Spurdetektoren, in Abbildung 1.2 und 1.3 als Tracker bezeichnet, sollen eine effiziente Rekonstruktion geladener Teilchen und genaue Messungen ihrer Impulse bieten. Außerdem muss die Richtungsinformation aller geladenen Teilchen, sowohl in der x- als auch in der y-Projektion, den RICH-Detektoren zur Verfügung gestellt werden, um Cherenkov-Ringe[2] rekonstruieren zu können.

- Das Kalorimeter-System umfasst mehrere Detektoren deren Aufgabe es ist, Elektronen und Hadronen zu Identifizieren und deren Energie und Position zu messen.

[1] Das Medium, in dem die Cherenkov-Strahlung erzeugt wird, wird als Cherenkov-Radiator bezeichnet. Für eine ausführliche Beschreibung siehe [5](Kap. 5.3 Cherenkov-Zähler).

[2] Die Cherenkov-Strahlung breitet sich als Kegel aus, dessen Öffnungswinkel von der Geschwindigkeit des Teilchens abhängt. Trifft dieser Lichtkegel eine flache Oberfläche, kann man einen charakteristischen Ring erkennen.

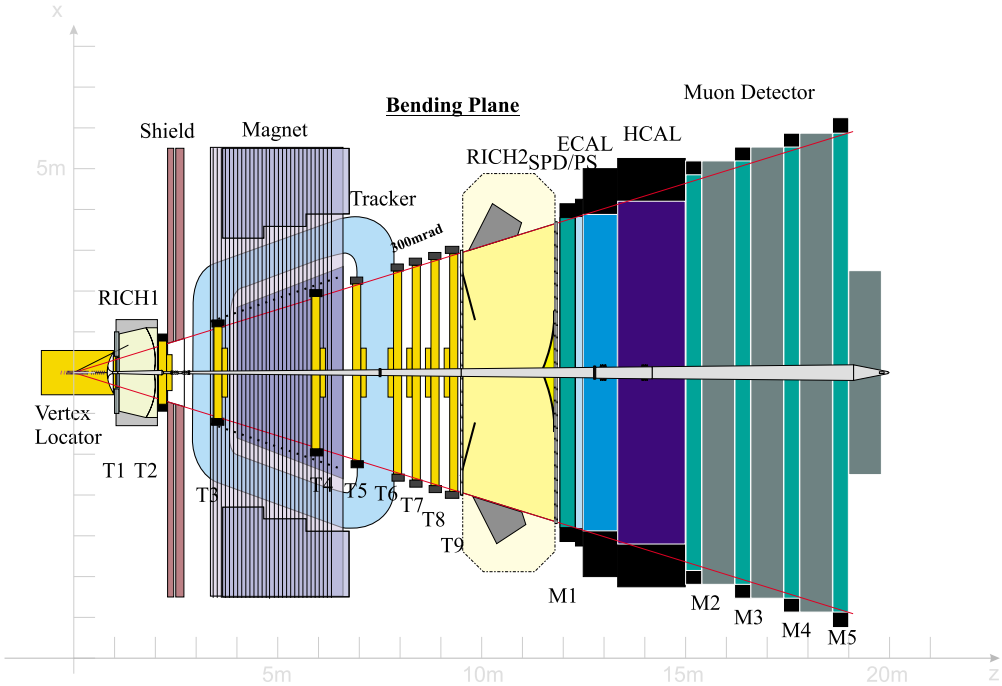

Abbildung 1.2: Draufsicht des LHCb Detektors, (z, x)-Projektion [2].

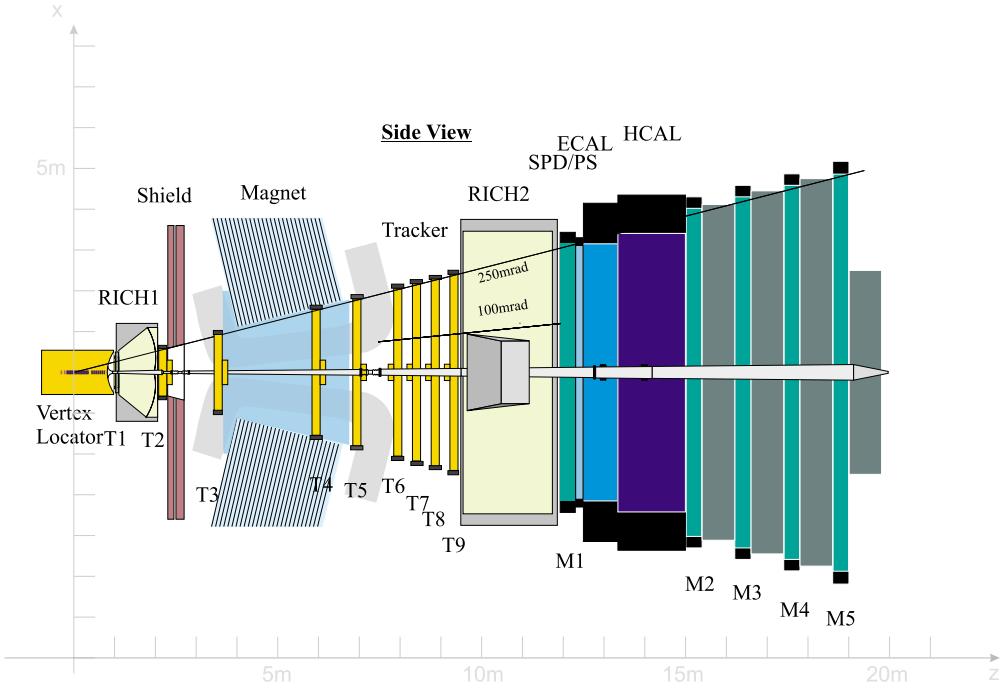

Abbildung 1.3: Seitenansicht des LHCb Detektors, (z, y)-Projektion [2].

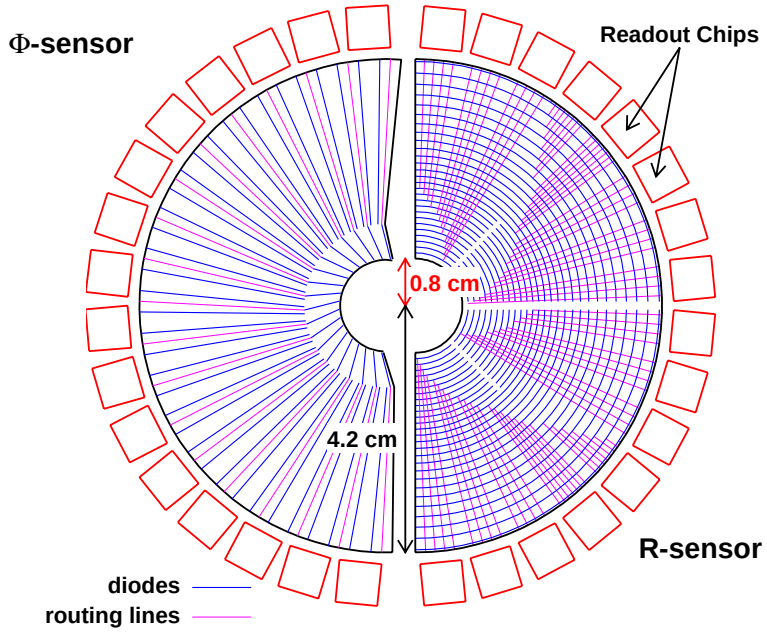

Abbildung 1.4: Geometrie der φ- und r-Sensoren. Beide Sensortypen besitzt jeweils 2048 Streifen.

  - **S**cintillator **P**ad **D**etector (SPD)
  - **P**re**s**hower Detector (PS)
  - **E**lectromagnetic **Cal**orimeter (ECAL)
  - **H**adron **Cal**orimeter (HCAL)

- Der Muon-Detektor (MUON) bewältigt zwei Funktionen. Zum einen ist er für die Erkennung von Muonen verantwortlich, zum anderen dient er dem Level-0 Trigger als Informationsquelle.

Eine besondere Aufmerksamkeit verdient der VELO, da der Level-1 Trigger aus seinen Daten eine Rekonstruktion der Ereignisse durchführt. Daher wird er im Folgenden genauer beschrieben.

1.2.1 Der Vertex Locator

Die Aufgaben des VELO sind die Rekonstruktion der Position des Primärvertex, die Erkennung von Spuren, die nicht aus dem Primärvertex stammen und die Rekonstruktion von b-Hadronen Zerfallsvertices. Um dies zu gewährleisten, muss der Vertex Locator präzise Spurkoordinaten in der Nähe des Primärvertex liefern. Außerdem umgibt der VELO die Wechselwirkungszone der beiden Protonenstrahlen und kann deshalb als einziger Detektor Informationen sowohl für die Vorwärts- als auch für die Rückwärtsrichtung liefern. Dies hilft Ereignisse mit mehreren Primärvertices zu erkennen.

	r-Sensor	φ-Sensor
Anzahl an Sensoren	50 + 4(VETO)	50
Auslesekanäle pro Sensor	2048	2048
Streifenabstand (Minimum)	40 μm	37 μm
Streifenabstand (Maximum)	92 μm	98 μm
Kürzester Streifen	6.4 mm	9.2 mm
Längster Streifen	66.6 mm	24.4 mm
Innerer Radius	8 mm	8 mm
Äußerer Radius	42 mm	42 mm
Winkelbereich	182°	$\approx 182°$
Stereo Winkel	-	10°- 20°

Tabelle 1.3: Parameter der r- und φ-Sensoren [6].

Der Aufbau

Der VELO besteht aus mehreren Silizium Detektoren. Diese Detektormodule haben die Form von Halbkreisen, wobei je zwei Detektormodule einen Kreis bilden und dann als *Station* bezeichnet werden. Ein Detektormodul besteht aus zwei Sensoren, der eine Sensor (r-Sensor) ist für die Bestimmung der Radialkoordinate und der andere Sensor (φ-Sensor) für die Bestimmung der Winkelkoordinate zuständig. In Abbildung 1.4 sind beide Sensortypen zu sehen. Der r-Sensor besitzt 2048 konzentrisch angeordnete Streifen (*engl.* strips), die für die Positionsbestimmung von geladenen Teilchen zuständig sind. Im inneren Bereich des Sensors ist der Abstand zwischen zwei Streifen (*engl.* strip pitch) 40 μm, mit zunehmendem Radius vergrößert sich dieser Abstand, bis er schließlich im äußeren Bereich 92 μm beträgt. Da die Anzahl an Streifen durch die Ausleseelektronik begrenzt ist, garantiert diese Anordnung eine bestmögliche Auflösung im inneren Bereich, wo sie für die Rekonstruktion des Primärvertex benötigt wird, und eine gute Auflösung im äußeren Bereich. Der r-Sensors ist im inneren Bereich in vier Sektoren unterteilt, der äußere Bereich ist in zwei Sektoren unterteilt. Diese Aufteilung des Sensors liefert eine optimale offline Rekonstruktion. Zur Zeit ist geplant, dass der VELO aus 25 + 2 Stationen bestehen wird. Die zwei zusätzlichen Stationen befinden sich in der Rückwärtsrichtung des Detektors und liefern die Information für ein Pileup Veto [6]. Die Anordnung der VELO-Stationen ist in Abbildung 1.5 zu sehen.

1.3 Das Trigger-System

Der Trigger ist für LHCb einer der wichtigsten Komponenten neben den Detektoren, da er die Datenrate stufenweise reduziert und nur Ereignisse, die von Interesse sind für die weitere Analyse übrig lässt. Die Datenrate, die von allen Detektoren zusammen erzeugt wird, liegt bei etwa 40 TByte/s. Die Datenmenge, die bei dieser Datenrate erzeugt wird ist immens. Es ist dementsprechend unerlässlich eine Datenreduktion während der Laufzeit durchzuführen. LHCb plant mit einer Luminosität von durchschnittlich $2 \cdot 10^{32}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ zu arbeiten.

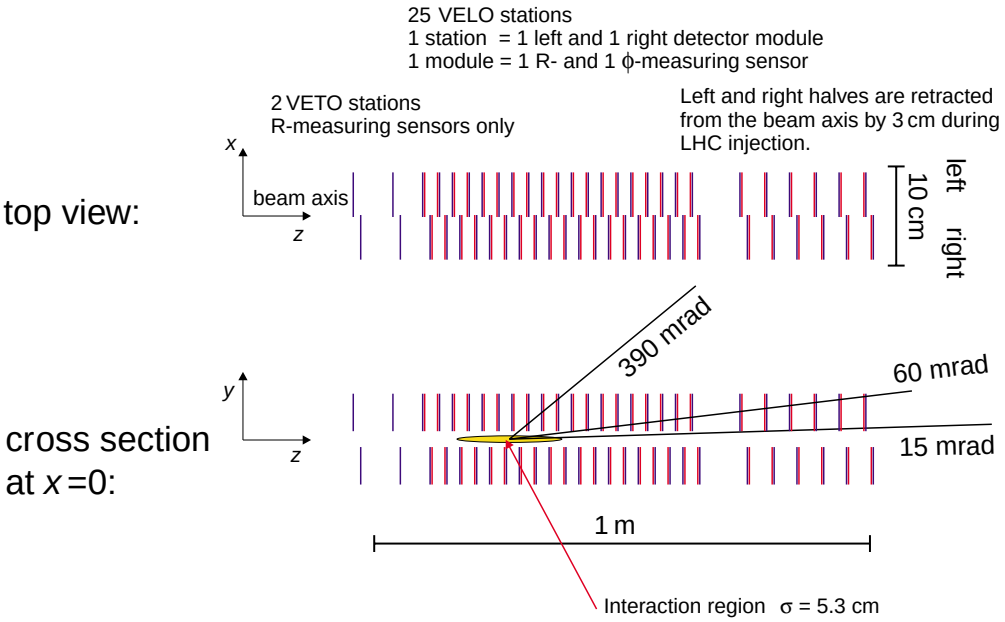

Abbildung 1.5: Anordnung der Detektorstationen entlang der Strahlachse. Zusätzlich ist die Wechselwirkungszone markiert [6].

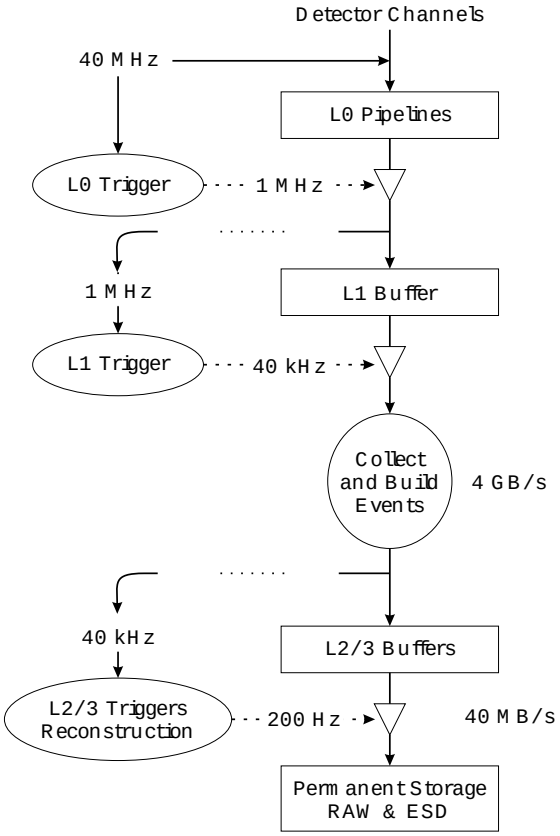

Abbildung 1.6: Diagramm des LHCb-Triggers und des Datenaquisitionssystems.

Bei dieser Luminosität haben etwa 75% aller Ereignisse genau einen Primärvertex, was die Rekonstruktion erleichtert. Es werden ungefähr 10^{12} $b\bar{b}$-Paare pro Jahr erwartet, was einer Rate von ∼100 kHz entspricht. Jedoch ist aufgrund des kleinen Verzweigungsverhältnis und der begrenzten Detektorempfindlichkeit nur ein kleiner Bruchteil dieser $b\bar{b}$-Ereignisse für eine vollständige Rekonstruktion interessant. Das Trigger-System hat die Aufgabe, aus einer großen Anzahl von $b\bar{b}$- und anderen inelastischen Proton-Proton-Ereignissen, diesen kleinen Bruchteil auszuwählen. Um dies zu bewerkstelligen, ist das Trigger-System von LHCb in vier Stufen unterteil. Jede Triggerstufe reduziert die Datenrate schrittweise(siehe Abbildung 1.6).

Triggerstufe	Eingangsrate	Ausgangsrate	Latenz
Level-0	40 MHz	1 Mhz	4 μs
Level-1	1 Mhz	40 kHz	1.68 ms
Level-2	40 kHz	5 kHz	10 ms
Level-3	5 kHz	200 Hz	200 ms

Tabelle 1.4: Aufgelistet sind die Triggerstufen mit den jeweiligen Ereignisraten. Außerdem ist die Dauer angegeben, die ein Ereignis höchstens in einer Triggerstufe verweilen darf (Latenz).

1.3.1 Level-0 Trigger

Der Level-0 Trigger ist ein Hardware-Trigger, der das Abarbeiten von Ereignissen mit einer Rate von bis zu 40 MHz durchführen muss. Er besteht aus drei Triggern, die jeweils auf der Basis von Muon-, Elektron- und Hadronerkennung arbeiten. Außerdem erkennt er, mit Hilfe zweier zusätzlicher r-Sensoren im VELO, Ereignisse, die mehr als einen Primärvertex haben. Für solche Ereignisse wird ein sog. Pileup Veto erzeugt. Es werden 80% der Ereignisse mit mehr als einer Proton-Proton-Interaktion erkannt, während 95% der Ereignisse mit genau einem Primärvertex erhalten bleiben.

- **Der Elektronen-Trigger** verwendet Informationen aus dem Muon-Detektor (MUON), Preshower-Detektor (PS) und dem ECAL. Der Trigger erkennt Elektronen mit hoher Energie (E_{T}), indem er nach Clustern mit hoher Energie im ECAL sucht. Außerdem wird der Entstehungspunkt mit Hilfe des Preshower- und Muon-Detektors ermittelt. Ereignisse mit einer Energie von $E_{\mathrm{T}} > 2.34$ GeV werden akzeptiert.

- **Der Hadronen-Trigger** verwendet Daten aus dem HCAL, dem Preshower-Detektor und dem Muon-Detektor. Er sucht nach hochenergetischen Hadronen. Ein Ereignis wird akzeptiert, wenn die Energie $E_{\mathrm{T}} > 2.34$ GeV ist.

- **Der Muon-Trigger** sucht nach Muonen, die aus der Wechselwirkungszone stammen und über einen hohen transversalen Impuls (p_{T}) verfügen. Ein Ereignis wird akzeptiert, wenn $p_{\mathrm{T}} > 1$ GeV ist.

1.3.2 Level-1 Trigger

Die Aufgabe des Level-1 Triggers ist es Ereignisse mit einem oder mehreren Sekundärvertices zu erkennen. Außerdem soll er Ereignisse ablehnen, die keinen hohen transversalen Impuls besitzen aber von Level-0 fälschlicherweise akzeptiert wurden. Das Herzstück des Triggers wird eine Rechnerfarm sein, die aus mehreren hundert handelsüblichen Personal Computern bestehen. Die Rechner werden über ein Hochgeschwindigkeits-Netzwerk miteinander verbunden sein. Der Level-1 Trigger ist in Kapitel 2 im Detail beschrieben.

1.3.3 Level-2 und Level-3 Trigger

Nach Level-1werden die Daten im sog. Data Acquisition System (DAQ) gespeichert. Die Level-2 und Level-3 Trigger führen eine vorläufige Spurrekonstruktion der DAQ Daten durch und die Ereignisse werden nach vordefinierten Zerfallskanälen eingeteilt. Die Grenze zwischen Level-2 und Level-3 steht momentan noch nicht fest und bietet Raum für Optimierung. Die gegenwärtige Einteilung sieht wie folgt aus

- **Der Level-2 Trigger** verfolgt die von Level-1gefundenen Spuren bis hin zu Station T5 und überprüft deren Impuls. Aus der zusätzlich gewonnenen Impulsinformation, können Ereignisse mit fälschlich angenommenem Sekundärvertex eliminiert werden.

- **Der Level-3 Trigger** verwendet die Information aller Detektoren zur Rekonstruktion. Er sucht nach speziellen Zerfallskanälen und behält nur solche Ereignisse, die diese besitzen.

Nach der letzten Triggerstufe ist die Ereignisrate von einst 40 MHz auf 200 Hz reduziert und die Ereignisse können dann mit einer Datentransferrate von etwa 40 MB/s abgespeichert werden.

Kapitel 2

Die Hardware Architektur des Level-1 Triggers

Die Aufgabe des Level-1 Triggers ist es, Ereignisse mit einem möglichen B-Zerfall zu identifizieren. Dazu wird eine Spurrekonstruktion mit Hilfe der Daten des VELO durchgeführt. Die Rate, mit der dies geschehen muss, liegt bei 1 MHz. Bei einer durchschnittlichen Größe eines Ereignisses von 4 kByte, ergibt dies eine Datenrate von etwa 4 GByte/s, die verarbeitet werden muss. Um diese Datenmenge verarbeiten zu können, ist eine Rechnerfarm notwendig, deren Knoten jeweils ein Ereignis abarbeiten (bei Doppelprozessor-Rechner können zwei Ereignisse pro Knoten abgearbeitet werden). Eine Kommunikation zwischen den Knoten während der Rekonstruktion, ist aufgrund von strengen Zeitvorgaben nicht sinnvoll.

Der Level-1 Trigger kann in drei logische Bereiche aufgeteilt werden. Am Anfang steht die Ausleseelektronik und die sog. Readout-Units (etwa 34 Stück). Die Aufgabe der Ausleseelektronik ist die Digitalisierung und die Vorverarbeitung der Daten. Die Readout-Units empfangen diese Daten und senden sie an einen freien Rechner. Sie bilden die Schnittstelle zwischen Level-0 und der Rechnerfarm. Der zweite Bereich ist die Rechnerfarm. Sie besteht aus handelsüblichen Personal Computern. Die PCs sind über ein Hochgeschwindigkeits-Netzwerk (SCI) miteinander verbunden. Hier wird die Ereignisrekonstruktion durchgeführt. Der dritte Bereich ist die sog. Level-1-Decision-Unit. Sie liefert das eigentliche Triggersignal mit einer Rate von 40 kHz. Die Latenz des gesamten Level-1 Triggers liegt bei etwa 1.6 ms. In dieser Zeit muss ein Ereignis abgearbeitet sein, ansonsten ist es verloren.

Um mit einer Eingangsrate von 1 MHz Ereignisse rekonstruieren zu können, sind zwei Faktoren von Bedeutung. Zum einen ist die Anzahl der Rechner entscheidend und zum anderen ist die durchschnittliche Zeit für die Rekonstruktion eines Ereignisses wichtig. Nimmt man eine Anzahl von 300 Rechnern an, so darf die Rekonstruktion eines Ereignisses im Durchschnitt nicht länger als 300 μs dauern. Zieht man noch die Dauer für das Verschicken eines Ereignisses ab ($\sim$10 μs), bleiben durchschnittlich 290 μs Zeit für die Rekonstruktion.

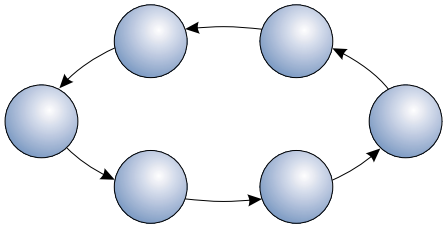

Abbildung 2.1: Ringtopologie

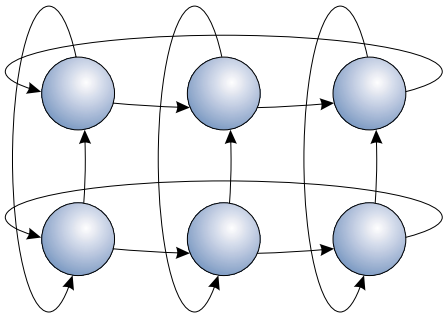

Abbildung 2.2: 2D-Torus

2.1 Die Readout-Unit

Die digitalisierten Daten des VELO gelangen über S-Link, ein am CERN entwickeltes Datentransferprotokoll [7], zu den Readout-Units (RU). Jede RU erhält einen fest vorgegebenen Teil der Daten des VELO. Ein Ereignis ist somit auf alle RUs verteilt. Ein Scheduler entscheidet, zu welchem Rechner die Daten geschickt werden sollen. Jede RU verschickt dann zeitlich verschoben mit einer Taktrate von 1 MHz ihr Teilereignis an diesen Rechner. Die Teilereignisse kommen dann in unterschiedlicher Reihenfolge bei dem Rechner an. Wenn alle Teilereignisse angekommen sind, beginnt der Rekonstruktionsalgorithmus seine Arbeit.

2.2 Rechnerfarm und Netzwerk

Wie zu Beginn erwähnt, besteht die Rechnerfarm aus mehreren Hundert PCs. Sie sind über ein SCI-Netzwerk miteinander verbunden. SCI unterstützt im Prinzip nur die Ringtopologie als Netzwerktopologie. Es ist jedoch möglich, dass jeder Knoten zwei oder mehr Ringen angehört. Gehören alle Knoten genau einem Ring an, so bezeichnet man diese Topologie auch als 1D-Torus. Gehört jeder Knoten genau zwei Ringen an, so bezeichnet man dies als 2D-Torus (vgl. Abb. 2.1 und 2.2). Mit wachsender Anzahl der Knoten, sind höhere Dimensionen durchaus denkbar.

Ein entscheidender Grund für die Verwendung von SCI ist, dass das Shared Memory Konzept Bestandteil des Netzwerkprotokolls ist. Bei Shared Memory kann ein globaler virtueller Adressraum angelegt werden. Zwei Knoten, die beide einen Teil ihres lokalen Adressraumes in den globalen Adressraum abgebildet haben, können durch einfaches schreiben in diesen globalen Adressbereich Daten miteinander austauschen. Dabei ist es für die Anwendung nicht sichtbar, ob die Daten lokal oder auf einem anderen Knoten gespeichert sind. Bei einer Schreiboperation, in einen Adressbereich der nicht lokal ist, gelangen die Daten über das Netzwerkprotokoll an die richtige Zieladresse. Bei einer Leseoperation, wird eine Leseanforderung über das Netzwerk geschickt, der zuständige Knoten übermittelt anschließend die Daten über das Netzwerk. Der entscheidende Vorteil am Shared Memory Konzept ist, dass für das Übertragen von Daten

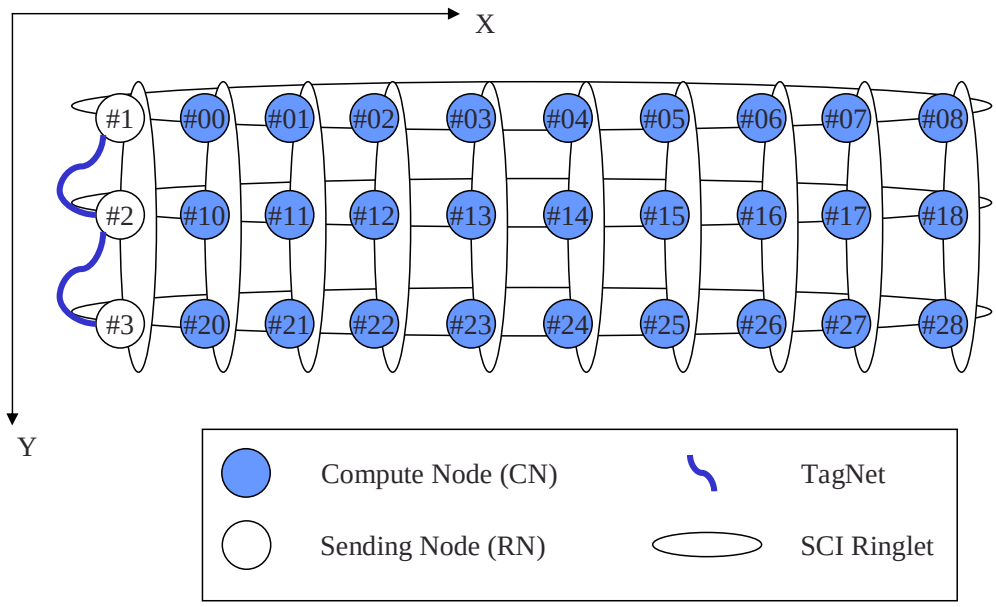

Abbildung 2.3: Aufbau des Level-1Triggers [8].

der zusätzliche Anteil an Daten, der für das Netzwerkprotokoll verwendet wird, minimal gehalten werden kann. Dies bedeutet, dass beinahe die gesamte Bandbreite des Netzwerks für Anwendungsdaten verwendet werden kann, während nur ein kleiner Teil der Bandbreite für das Protokoll verschwendet wird.

Für den Level-1Trigger wird ein 2D-Torus als Netzwertopologie verwendet. Das Netzwerk ist in zwei Gruppen aufgeteilt. Die eine Gruppe besteht aus den Readout-Units (RU). Diese Knoten, verschicken die Daten innerhalb des Netzwerks. Die andere Gruppe sind Knoten, die Daten empfangen und verarbeiten. Alle RUs sind in einer Spalte zu finden und sind untereinander zusätzlich mit einem weiteren Netzwerk verbunden, das als Tag-Net bezeichnet wird. Innerhalb dieses Netzes werden kleine Datensätze mit einer Rate von 1 MHz von Readout-Unit zu Readout-Unit verschickt. Mithilfe dieser sog. Tags, wird den RUs mitgeteilt zu welchem Rechenknoten die Ereignisdaten geschickt werden. Dabei verschickt jede RU nur einen Teil der Ereignisdaten. Durch das Prinzip der Tags wird ausgeschlossen, das zwei RUs gleichzeitig Daten zum gleichen Rechenknoten verschicken. Dies ist wichtig um überflüssigen Datenverkehr zu vermeiden. In Abbildung 2.3 ist der Aufbau des Netzwerks schematisch dargestellt.

Kapitel 3

Der Tracking Algorithmus

Die Aufgabe des Tracking Algorithmus ist es, Ereignisse mit speziellen Zerfällen, wie zum Beispiel $B_d^0 \to \pi^+\pi^-$, zu erkennen. Solche Ereignisse haben Sekundärzerfälle nahe am Primärvertex, so dass sie dadurch erkannt werden können. Der Algorithmus muss also in der Lage sein, aus den vom Detektor gelieferten Daten Spuren zu rekonstruieren. Die Spurensuche beschränkt sich auf Geraden. Diese Einschränkung kann gemacht werden, da im Bereich des VELO das Magnetfeld kaum mehr vorhanden ist, so dass die Flugbahn geladener Teilchen auf einer Strecke von etwa einem Meter (Größe des VELO) einer Geraden entspricht. Effekte, wie elastische Streuprozesse mit Elektronen im Inneren des Detektormaterials, *multiple scattering* genannt, überwiegen hier bei weitem.

Die Zeit, die dem Algorithmus für die Rekonstruktion, und der daraufhin resultierenden Entscheidung zur Verfügung steht, liegt im Moment bei etwa 1.6 ms. Dieser Wert wird hauptsächlich durch die Größe des Level-1 Buffers bzw. der Anzahl der Ereignisse, die im Level-1 Buffer Platz finden, bestimmt. Um dieser strengen Zeitanforderung gerecht zu werden, rekonstruiert der Algorithmus nicht wie allgemein üblich sofort 3-dimensonale Spuren, sondern sucht zunächst einmal 2-dimensionale Spuren in der (z, r)-Ebene. Aus diesen Spuren werden dann nach einem bestimmten Kriterium einzelne Spuren ausgewählt, die dann zu vollständigen 3-dimensionalen Spuren ergänzt werden. Möglich wird dieses Verfahren durch den speziellen Detektor, der Raumpunkte als Zylinderkoordinaten (r, φ, z) liefert.

Der gesamte Algorithmus kann in mehrere Teilalgorithmen aufgegliedert werden.

- 2D-Spurensuche
- Bestimmung der Position des Primärvertex
- 3D-Spurensuche
- Bestimmung von Sekundärvertices

Zu Beginn wird eine 2D-Spurensuche durchgeführt. Dann folgt die Bestimmung der Position des Primärvertex. Als nächstes wird jeder 2D-Spur ein Stoßparameter (*engl.* impact parameter) zugewiesen, der sich aus der relativen Posi-

tion zum Primärvertex ergibt. Der Stoßparameter dient als Kennzeichen, ob eine Spur ihren Ursprung im Primärvertex hat oder aus Sekundärzerfällen stammt. Hat eine Spur einen großen Stoßparameter, so ist die Wahrscheinlichkeit hoch, dass diese Spur aus einem Sekundärzerfall stammt. Es werden 2D-Spuren mit großem Stoßparameter ausgewählt und zu 3D-Spuren rekonstruiert. Mit Hilfe dieser 3D-Spuren, werden die Positionen der Sekundärvertices bestimmt, um schließlich zu entscheiden, ob das Ereignis der nächsten Triggerstufe zugänglich gemacht werden kann.

3.1 Die Datenstruktur eines Ereignisses

Wie bereits oben erwähnt sind die Daten als Zylinderkoordinaten gegeben. Ein Ereignis besteht aus r- und φ-Daten. Im Folgenden wird die Bezeichnung *Cluster* für einen Datenpunkt verwendet. Jeder VELO-Station ist eine gewisse Menge an r- und φ-Clustern zugeordnet. Die r-Cluster sind zusätzlich nach den einzelnen Sektoren einer Station aufgeteilt und sind innerhalb des zugehörigen Sektors aufsteigend sortiert. Im weiteren Verlauf wird die Notation $r_i^{(n,m)}$ für ein r-Cluster benutzt. Dabei ist n die Stationsnummer und m die Sektornummer der Station. Die Anzahl der Stationen beträgt $N = 25$. Der Index i gibt den Cluster innerhalb des Sektors an. Die Position einer Station in z-Richtung wird mit z_n bezeichnet.

Ein Ereignis hat durchschnittlich 650 r-Cluster und etwa dieselbe Anzahl φ-Cluster. Bei diesem Wert ist nicht die Anzahl an Clustern berücksichtigt, die durch Rauschen entstanden sind. Es ist noch zu erwähnen, dass es keine Information über die Beziehung eines r-Clusters zu einem φ-Cluster gibt. Es ist die Aufgabe des Rekonstruktionsalgorithmus festzustellen, welcher φ-Cluster zu einem gegebenen r-Cluster gehört.

3.2 2D-Spurensuche

Die Suche nach 2D-Spuren ist im gesamten Algorithmus, trotz gewisser Vereinfachungen, der wohl zeitaufwendigste Teil. Man beschränkt sich bei der Suche nach Geraden auf gleiche Sektoren im Detektor. Das heißt, dass Spuren die mehrere Sektoren kreuzen grundsätzlich nicht erkannt werden. Außerdem beschränkt sich die Suche im ersten Schritt auf jeweils drei aufeinander folgende Detektorstationen, um den Rechenaufwand gering zuhalten. Innerhalb dieser drei Stationen werden r-Cluster gesucht die eine Geraden bilden. Sie werden als *Triplett* bezeichnet. In einem zweiten Schritt werden diese Tripletts zu ganzen Spuren zusammengefügt. Eine weiter Tatsache, die dem Rechenaufwand zugute kommt ist, dass die r-Daten aufsteigend sortiert sind.

3.2.1 2D-Triplettsuche

Liegen drei Punkte aus drei benachbarten Stationen in der (z, r)-Projektion auf einer Geraden, so wird dies im Folgenden als Triplett bezeichnet (Abb. 3.1). Die

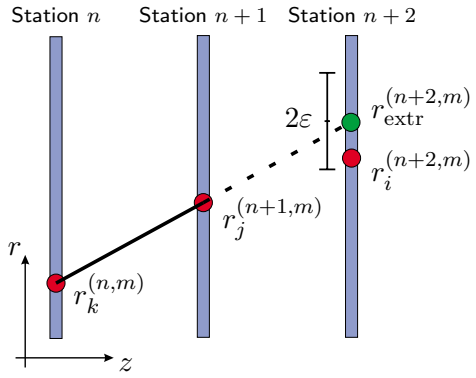

Abbildung 3.1: Bedingung für ein Triplett. Die drei blauen Linien stellen einen Sektor eines Sensors dar. Die roten Punkte sind r-Cluster. Der grüne Punkt stellt den extrapolierten Wert dar. n ist die Stationsnummer und m die Sektornummer.

Bedingung für ein Triplett lautet

$$\left| r_i^{(n+2,m)} - r_{\text{extr}}^{(n+2,m)} \right| < \varepsilon,$$

mit dem extrapolierten Wert

$$r_{\text{extr}}^{(n+2,m)} = (r_j^{(n+1,m)} - r_k^{(n,m)}) \frac{z_{n+2} - z_n}{z_{n+1} - z_n} + r_k^{(n,m)}$$

Um möglichst schnell Tripletts innerhalb der Daten dreier aufeinander folgender Stationen bzw. Sektoren zu finden, muss auf intelligente Weise danach gesucht werden. Das Suchverfahren muss die Anzahl der zu testenden Kombinationen möglichst gering halten.

Das Verfahren, das in der Softwareimplementierung zum Auffinden von Tripletts implementiert ist hat folgende Struktur. Es werden drei ineinander verschachtelte Schleifen verwendet. Die äußerste und damit langsamste Schleife durchläuft die Daten eines Sektors der *linken* Station, die mittlere Schleife durchläuft die Daten des gleichen Sektors der *mittleren* Station und die innerste Schleife durchläuft die Daten des gleichen Sektors der *rechten* Station. Außerdem wird in der innersten Schleife überprüft, ob die drei Datenpunkte (Datenpunkt der linken, mittleren und rechten Station) ein Triplett ergeben (vgl. 3.2.1). Zusätzlich dazu wird nun ausgenützt, dass die Daten sortiert sind. Findet man einen Datenpunkt der rechten Station, der über dem Toleranzbereich liegt, so kann die innerste Schleife abgebrochen und auf den ersten Datenwert zurückgesetzt werden, denn alle folgenden Werte liegen ebenfalls außerhalb des Toleranzbereichs. Durch diese Abbruchbedingung wird vermieden Datenkombinationen zu Prüfen, die ohnehin kein Triplett bilden können. Dies verringert die Anzahl an Dreierkombinationen, die überprüft werden ohne dabei Kombinationen zu übersehen. In Abbildung 3.2 ist die Suche nach Tripletts in Schritten graphisch dargestellt.

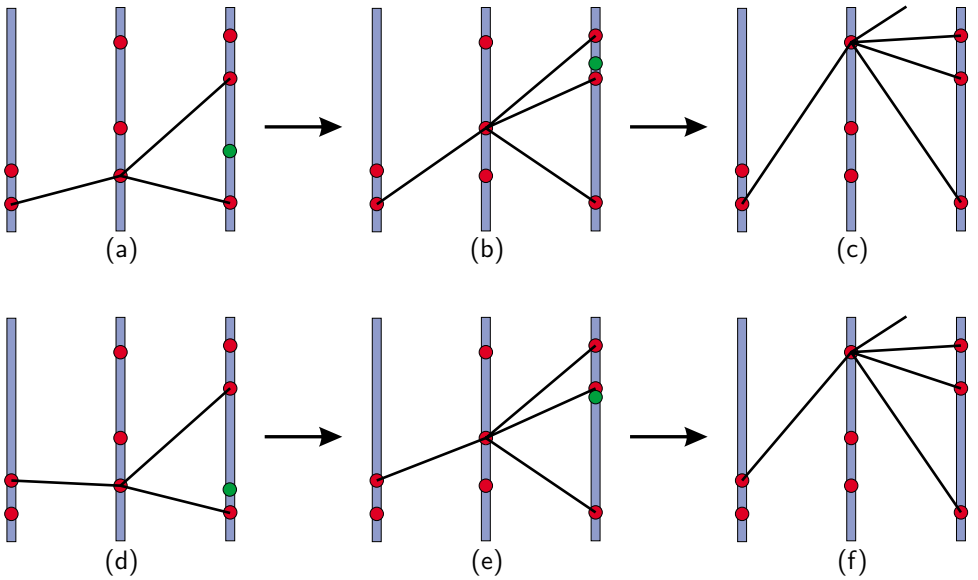

Abbildung 3.2: Triplettsuche in einzelnen Schritten . Die roten Punkte stellen r-Cluster dar. Der grüne Punkt steht für den extrapolierten Wert. Der Toleranzbereich ε ist durch die Größe des grünen Punktes gegeben. Die Bedingung für ein Triplett ist also dann erfüllt, wenn ein roter und grüner Punkt sich überschneiden. (a) Die ersten beiden Werte aus Station 1 und 2 werden verwendet, um den extrapolierten Wert zu ermitteln. Dann werden die Datenpunkte der 3. Station von unten nach oben durchlaufen. Nachdem der zweite Datenpunkt überprüft wurde, wird die Schleife abgebrochen, denn der weite Datenpunkt liegt über dem Toleranzbereich ε. (b) Der erste Datenpunkt aus Station 1 und der zweite Datenpunkt aus Station 2 werden verwendet, um den extrapolierten Wert zu ermitteln. Die unteren drei Datenpunkte aus Station 3 müssen durchlaufen werden. (c) Die Suche wird fortgesetzt bis der letzte Datenpunkt der 2. Station durchlaufen wurde. (d) Der zweite Datenpunkt aus Station 1 und der erste Datenpunkt aus Station 2 werden verwendet, um den extrapolierten Wert zu ermitteln. Dann werden erneut die Datenpunkte der Station 3 bis zur Abbruchbedingung durchlaufen. (e) Es wird der nächste Datenpunkt aus Station 2 verwendet und mit dem zweiten Datenpunkt aus Station 1 der extrapolierte Wert bestimmt. Dann wird Station 3 erneut durchlaufen. (f) Die Suche endet, wenn der letzte Datenpunkt der aus Station 1 durchlaufen wurde.

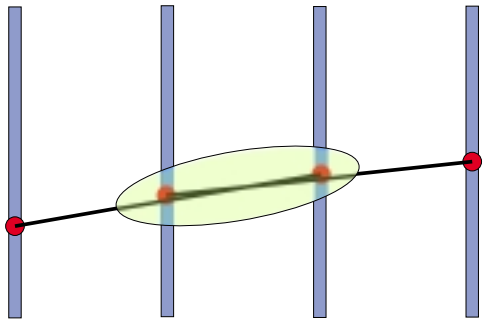

Abbildung 3.3: Zusammenfügen von Tripletts zu Spuren. Gehören die gleichen zwei Cluster zu zwei Tripletts, wie oben zu sehen, dann werden alle zugehörigen Cluster der selben Spur zugeordnet.

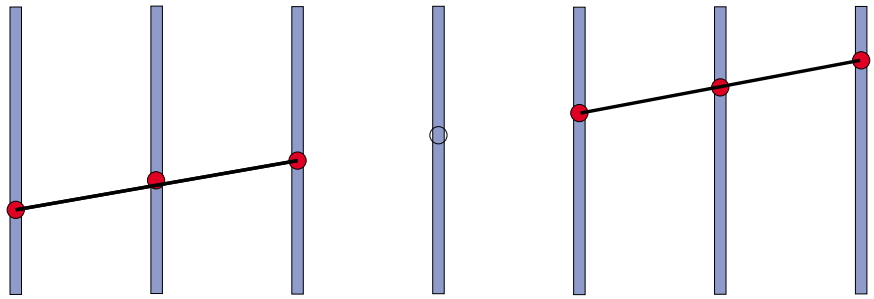

Abbildung 3.4: Es sind zwei Tripletts zu sehen, die beide zur gleichen Spur gehören. Die Station in der Mitte des Bildes müsste einen Cluster aufweisen, hat dies jedoch nicht detektiert. Deshalb fehlen insgesamt drei zusätzliche Tripletts. Um beide Tripletts der selben Spur zuzuordnen, können hier die Spurparameter der beiden Tripletts miteinander verglichen werden.

3.2.2 Zusammenfügen von Tripletts

Das Zusammenfügen von Tripletts zu 2D-Spuren kann direkt nach dem Finden eines Tripletts stattfinden. Es gibt mehrere Varianten Tripletts zu verbinden.

Die einfachste Möglichkeit ist zu überprüfen, ob die ersten zwei Cluster des Tripletts mit den letzten zwei Clustern eines der Tripletts aus der vorangegangenen Suche übereinstimmen(siehe Abb. 3.3). Diese Methode ist jedoch anfällig gegenüber Detektorineffizienz. Vergleiche dazu Abbildung 3.4

Eine andere Methode, die nicht so anfällig gegenüber Detektorineffizienz ist, vergleicht die Steigung und den Achsenabschnitt des gefundenen Tripletts mit den vorangegangenen Tripletts. Findet sich eine Übereinstimmung innerhalb einer gewissen Toleranz, so gehören diese beiden Tripletts zu der selben Spur.

3.2.3 Resultat der 2D-Spurensuche

Um die Qualität der 2D-Spurensuche zu überprüfen, sind Messungen durchgeführt worden. Diese zeigen die Effizienz der Spurensuche. Folgende Abkürzungen werden im weiteren Verlauf verwendet.

RefB — Teilchen stammt aus einem B-Zerfall und hat eine Energie größer 1 GeV.

RefPrim — Teilchen stammt aus dem Primärvertex und hat eine Energie größer 1 GeV.

RefSet — Alle Teilchen mit einer Energie größer 1 GeV.

AllSet — Alle Teilchen, die drei oder mehr Stationen durchqueren.

Clone — Spuren, die mehrfach erkannt wurden.

Ghost — Spuren, die keinem Teilchen entsprechen.

Der Rekonstruktionsalgorithmus verwendet eine Datenbank mit 500 Ereignissen. Alle Ereignisse besitzen $B_d^0 \rightarrow \pi^+\pi^-$ Zerfälle und wurden mit einer Monte-Carlo-Simulation erzeugt.

RefB	96.2%
RefPrim	95.1%
RefSet	93.8%
AllSet	88.7%
Clone	2.9%
Ghost	5.5%

Tabelle 3.1: Effizienz für bestimmte Teilmengen von Spuren.

3.3 Berechnung des Primärvertex

Nachdem die 2D-Spurrekonstruktion abgeschlossen ist, kann mit der Rekonstruktion des Primärvertex begonnen werden. Die Rekonstruktion ist in zwei Teile aufgeteilt. Zunächst wird die z-Position ermittelt. Im zweiten Schritt, wird die x- und y-Position bestimmt.

Um die z-Position des Primärvertex zu bestimmen, wird als erstes ein Histogramm erstellt, welches die Schnittpunkte der rekonstruierten 2D-Spuren mit der z-Achse enthält. Das Maximum dieses Histogramms ergibt eine erste gute Position des Primärvertex. In einem zweiten Schritt, wird ein neues Histogramm mit gewichteten Schnittpunkten erstellt. Die Wichtungsfunktion ist eine Gauß-Verteilung, deren Maximum der ermittelten Position des Primärvertex entspricht. Die Größe des Wichtungsfaktors hängt von der Position des Ursprungs der Spur ab. Liegt der Ursprung der Spur weit entfernt vom Primärvertex, so wird der Schnittpunkt mit einem kleinen Wert gewichtet, auch wenn der

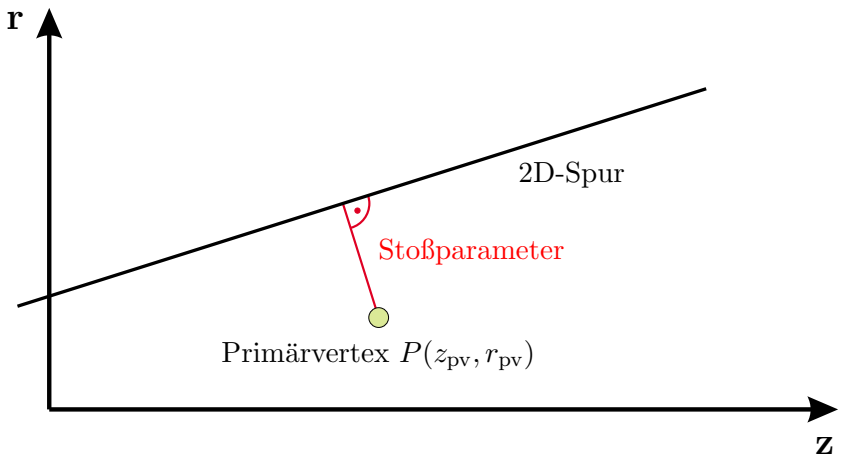

Abbildung 3.5: Graphische Darstellung der hier als Stoßparameter bezeichneten geometrischen Entfernung.

Schnittpunkt mit der z-Achse sehr nahe am Primärvertex liegen sollte. Durch diese Selektion wird erreicht, dass Spuren die aus Sekundärvertices stammen, bei der Berechnung des Primärvertex unterdrückt werden. Mit der neu erhaltenen Position des Primärvertex, wird ein letzter Durchlauf mit einer schmaleren und dem Maximum angepassten Gauß-Verteilung durchgeführt. Durch dieses iterative Verfahren ist es Möglich eine sehr gute Auflösung der z-Position zu erreichen. Sie liegt bei etwa $50\mu m$. Dieser Wert entspricht in etwa dem Wert, der bei der Offline-Rekonstruktion erreicht wird.

3.4 3D-Spurensuche

Der nächste Schritt im Ablauf des Rekonstruktionsalgorithmus ist die vollständige 3D-Rekonstruktion der zuvor gefundenen 2D-Spuren. Man beschränkt sich dabei auf Spuren, die nicht vom Primärvertex stammen, denn das Ziel des Rekonstruktionsalgorithmus ist es Sekundärvertices zu finden und deren vollständige 3-dimensionale Position zu bestimmen. Um 2D-Spuren zu finden, die gute Kandidaten für eine 3D-Rekonstruktion sind, wird sowohl der Achsenabschnitt als auch die Steigung der 2D-Spuren in der (z, r)-Projektion als Hinweis verwendet. Es ist z.B. nicht sinnvoll eine 2D-Spur auszuwählen, deren Steigung so groß ist, dass sie denn Akzeptanzbereich des Detektors überschreitet. Das Hauptkriterium für die Selektion ist jedoch der Stoßparameter einer Spur. Dieser gibt den kürzesten geometrischen Abstand einer 2D-Spur in der (z, r)-Projektion zum Primärvertex an (siehe Abb. 3.5). Die Gleichung lautet

$$s = \frac{r_{\mathrm{pv}} - a z_{\mathrm{pv}} - b}{\sqrt{a^2 + 1}} \tag{3.1}$$

mit a als Steigung und b als Achsenabschnitt der 2D-Spur.

Spuren mit signifikantem Stoßparameter werden für die 3D-Rekonstruktion ausgewählt. Die Aufgabe besteht nun darin zusammengehörige r- und φ-Cluster zu finden. Das korrekte zuordnen eines gegebenen r-Clusters zu einem entsprechenden φ-Cluster ist nicht trivial, da es keine direkte Beziehung zwischen den Daten gibt. Eine Überprüfung, die man durchführen kann ist, zu vergleichen ob r- und φ-Cluster aus dem selben Sektor stammen. Zusätzlich wird angenommen, dass Spuren aus Primär- und Sekundärvertices in den meisten Fällen ihre φ-Koordinate entlang ihres Weges kaum ändern. Es kann also nach ungefähr gleichen φ-Koordinaten in den unterschiedlichen Stationen gesucht werden. Das Verfahren mit dem man φ-Cluster der gleichen Spur findet, ist im Prinzip das gleiche wie für r-Cluster. Auch hier wird nach Tripletts gesucht, dabei werden je zwei φ-Cluster ausgewählt und ein extrapolierter Wert berechnet. Dieser wird dann auf Übereinstimmung getestet.

3.5 Sekundärvertex Bestimmung

Nach der vollständigen 3D-Rekonstruktion einzelner ausgewählter Spuren, wird nach Sekundärvertices gesucht, die möglicherweise einem B-Zerfall entsprechen. Dazu muss im Prinzip nach dem Schnittpunkt von zwei 3D-Spuren gesucht werden. Aufgrund von Streueffekten und der endlichen Auflösung des Detektors, werden zwei Geraden immer windschief zueinander liegen, das heißt es gibt keinen Schnittpunkt. Deshalb wird der geringste räumliche Abstand zwischen zwei 3D-Spuren bestimmt. Liegt der Abstand unter einem bestimmten Schwellwert, dann wird die Mitte dieses Abstands als Vertexposition angenommen. Ist die Entfernung zum Primärvertex ausreichend klein und liegt die Vertexposition in der Vorwärtsrichtung des Detektors, also rechts vom Primärvertex, kann davon ausgegangen werden, dass es sich um einen Sekundärvertex handelt.

Die Laufzeit dieser Routine hängt stark von der Anzahl der 3D-Spuren ab, denn es müssen alle Zweierkombinationen überprüft werden. Es ist deshalb wichtig nur wenige *gute* 3D-Spuren zu verwenden. Der Stoßparameter, als einziges Auswahlkriterium für *gute* Spuren, ist nur bedingt ausreichend. Zusätzliche Informationen aus anderen Detektoren, wie z.B. der Impuls der Teilchen, würde die Auswahl erleichtern und damit die Rekonstruktion von Sekundärvertices hinsichtlich der Geschwindigkeit und der Genauigkeit verbessern. Die Präzision mit der man zum gegenwärtigen Zeitpunkt Sekundärvertices bestimmen kann liegt bei etwa 225 μm.

3.6 Zeitverhalten des Algorithmus

Zeitmessungen ergaben, dass der Algorithmus fast 70% der Gesamtlaufzeit benötigt, um 2D-Spuren zu finden (siehe Tabelle 3.2). Grund hierfür ist vor allem die große Anzahl an Kombinationen, und damit auch eine Vielzahl an Berechnungen, die durchgeführt werden müssen. Dieses Verhalten ist gut in Abbildung 3.6 zu sehen. Die Berechnungszeit steigt mit wachsender Ereignisgröße quadratisch. Der gesamte Rekonstruktionsalgorithmus hat eine durchschnittli-

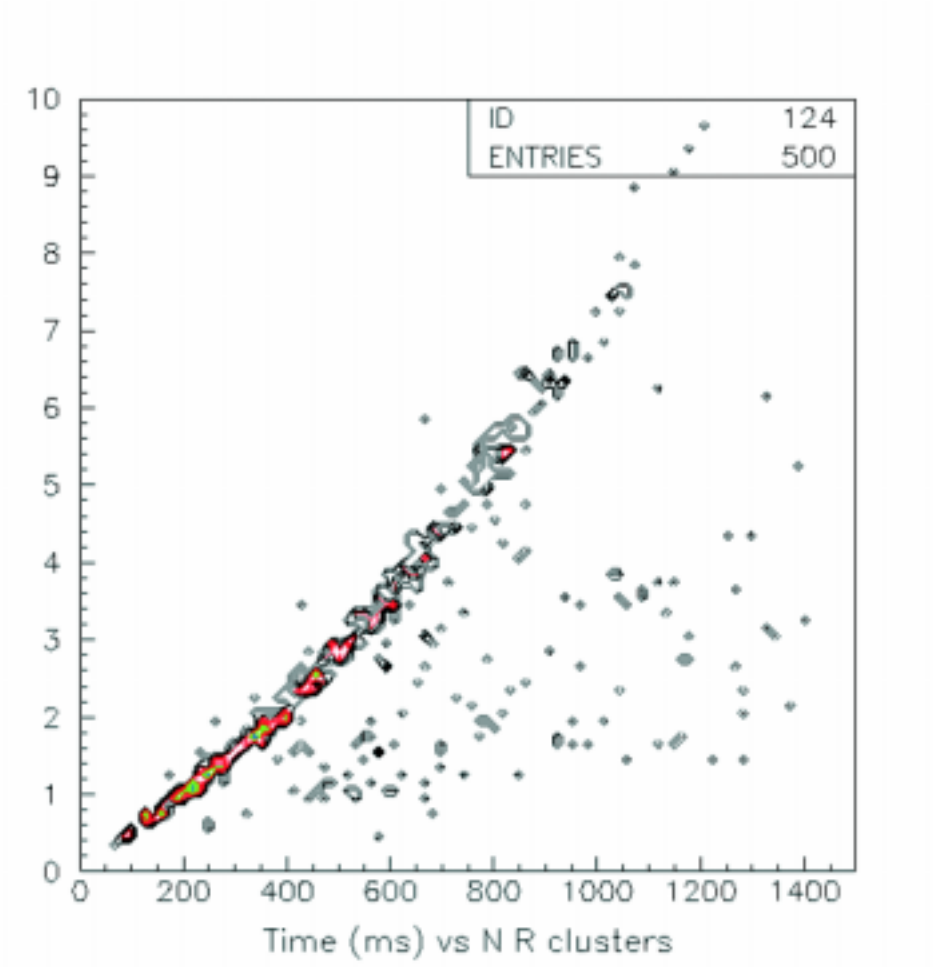

Abbildung 3.6: Zeitverhalten in Abhängigkeit von der Ereignisgröße. Aufgetragen ist die Zeit in ms über die Anzahl der r-Cluster pro Ereignis.

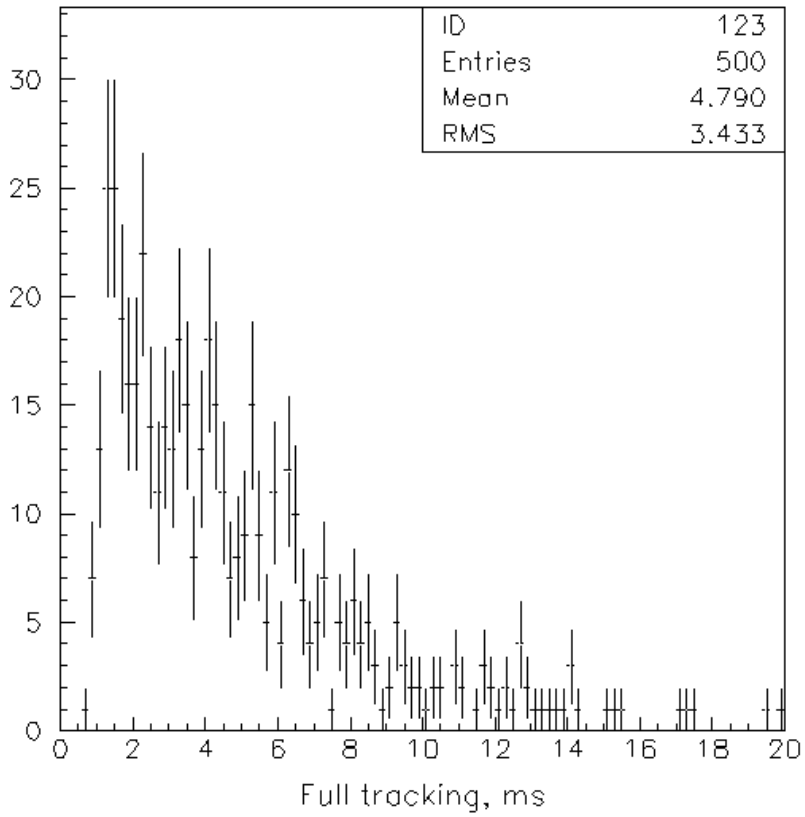

Abbildung 3.7: Zeitverhalten des gesamten Algorithmus. Die Zeit ist in ms angegeben. Die durchschnittliche Laufzeit liegt bei 4.8 ms.

che Laufzeit von etwa 4.8 ms. Die Messung wurde mit einem 1 GHz Pentium Rechner durchgeführt. Es muss erwähnt werden, dass der Rekonstruktionsalgorithmus in der derzeitigen Version nicht auf Geschwindigkeit optimiert ist. Der Wert kann deshalb bestenfalls als obere Schranke angesehen werden. Abbildung 3.7 zeigt das Zeitverhalten des Algorithmus für 500 Ereignisse.

2D-Spurensuche	$\sim$70%
Bestimmung der Position des Primärvertex	$\sim$15%
3D-Spurensuche	$\sim$15%
Bestimmung von Sekundärvertices	$< 1\%$

Tabelle 3.2: Zeitverhalten der Teilalgorithmen.

Kapitel 4

Verwendung eines FPGA Coprozessors

In Anbetracht der strengen Zeitvorgaben an den Rekonstruktionsalgorithmus, wurde über eine mögliche Unterstützung durch einen FPGA Coprozessor nachgedacht. Dieser Coprozessor soll als PCI-Karte in jedem PC der Rechnerfarm zu finden sein. FPGAs haben besonders eine Eigenschaft, die sie für diesen Anwendungszweck zur idealen Wahl macht. Sie sind rekonfigurierbar. Dies ist besonders wichtig, da sich der Rekonstruktionsalgorithmus während der Laufzeit des Experimentes sicherlich mehrmals ändern wird, um sich den jeweiligen Bedingungen des Experimentes besser anzupassen. Denkbar sind Änderungen im Algorithmus, um die Akzeptanz für bestimmte Zerfallskanäle zu erhöhen.

Aufgabe des Coprozessors soll es sein, die durchschnittliche Berechnungszeit für die Rekonstruktion zu verringern. Dies kann auf zwei Wegen geschehen. Ist der Coprozessor in der Lage den zeitintensivsten Teil des Algorithmus schneller als die CPU zu bearbeiten, so kann dieser vom Coprozessor übernommen werden. Eine weitere Möglichkeit ist, dass Ereignisse schon vom Coprozessor herausgefiltert werden, wenn gewisse Anforderungen für das Ereignis nicht erfüllt sind.

Aus dem gemessenen Zeitverhalten des Algorithmus (Abschnitt 3.6) ist ersichtlich, dass der Algorithmus fast 70% seiner Zeit für die 2D-Spurensuche verwendet. Es ist also vernünftig das Augenmerk auf diesen Teil zu richten. Vorteilhaft ist, dass die 2D-Spurensuche am Anfang des Rekonstruktionsalgorithmus steht, so dass ausschließlich die VELO Detektordaten verwendet werden können. Somit muss der Algorithmus (CPU) keine Daten an den Coprozessor schicken.

4.1 Implementierung der 2D-Triplettsuche

4.1.1 Suchverfahren

Das Suchverfahren der 2D-Triplettsuche wurde in einer leicht abgewandelten Form umgesetzt. Wie auch in der Softwareimplementierung werden drei Datenmengen verwendet, die jeweils für drei benachbarte Stationen die r-Positionen

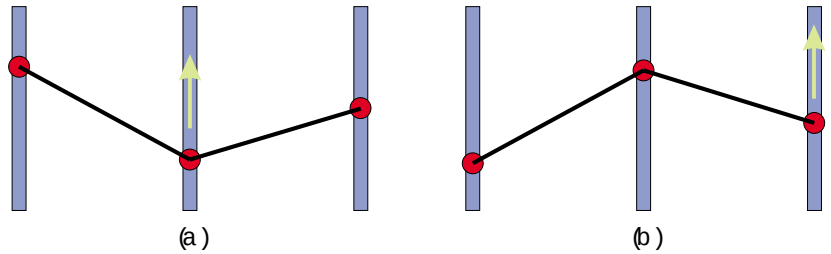

Abbildung 4.1: (a) Bei konvexer Anordnung, wird der nächste Wert der mittleren Station genommen. (b) Bei konkaver Anordnung, wird der nächste Wert der rechten Station genommen.

aller Cluster in einem Sektor beinhalten. Ein Cluster wird mit $r_i^{(n)}$ bezeichnet, wobei dies der i-te Cluster aus Station n ist. Die Anzahl der Stationen liegt momentan bei $N = 25$. Die Datenmengen sind noch zusätzlich nach Sektoren aufgeteilt, jedoch wird die Triplettsuche nur innerhalb von gleichen Sektoren durchgeführt. Deshalb ist der Index für den Sektor nicht aufgeführt.

Es werden Dreierkombinationen gesucht, welche die Bedingung

$$\left| r_i^{(3)} - r_{\text{extr}}^{(3)} \right| < \varepsilon, \tag{4.1}$$

mit dem extrapolierten Wert

$$r_{\text{extr}}^{(3)} = (r_j^{(2)} - r_k^{(1)}) \frac{z_3 - z_1}{z_2 - z_1} + r_k^{(1)} \tag{4.2}$$

erfüllen. Um die Anzahl der zu überprüfenden Dreierkombinationen gering zu halten, wird eine zusätzliche Information über die geometrische Anordnung der Cluster verwendet, die dann Aufschluss darüber gibt, aus welcher Datenmenge der nächste Cluster genommen wird. Die Anordnung der drei Cluster kann konvex oder konkav sein. Um dies festzustellen sind keine zusätzlichen aufwendigen Berechnungen nötig. Lediglich ein Vergleich ist notwendig. Ist eine konvexe Anordnung gegeben, so gilt

$$r_i^{(3)} - r_{\text{extr}}^{(3)} > 0. \tag{4.3}$$

Haben die drei Cluster eine konkave Anordnung, so gilt

$$r_i^{(3)} - r_{\text{extr}}^{(3)} < 0. \tag{4.4}$$

In Abbildung 4.1 sind beide Anordnungen graphisch dargestellt. Liegt eine konvexe Anordnung vor, so wird $r_j^{(2)}$ durch $r_{j+1}^{(2)}$ ersetzt. Die Werte für Station 1 und 3 ändern sich nicht. Liegt eine konkave Anordnung vor, so wird $r_i^{(3)}$ durch $r_{i+1}^{(3)}$ ersetzt. Die Werte für Station 1 und 2 bleiben gleich. Ist kein weiterer Cluster in Station 2 oder 3 vorhanden, wird $r_i^{(1)}$ durch $r_{i+1}^{(1)}$ ersetzt und jeweils

die ersten Werte für Station 2 und 3 genommen ($r_1^{(2)}$, $r_1^{(3)}$). Beendet wird die Suche wenn das letzte Element aus Station 1 vollständig überprüft wurde.

Durch dieses Suchverfahren wird die Anzahl von nicht sinnvollen Kombinationen, die überprüft werden müssen, auf ein Minimum reduziert. Die Komplexität liegt bei $\mathrm{O}\left(N_1 \cdot (N_2 + N_3)\right)$, wobei N_i die Anzahl der Cluster in Station i ist. Aus der Komplexität ist ersichtlich, dass durch dieses Suchverfahren, das Problem von einem Problem mit drei Mengen, auf ein Problem mit zwei Mengen reduziert wurde. Die eine Menge besitzt N_1 Elemente, die zweite besitzt $N_2 + N_3$ Elemente. Im ungünstigsten Fall müssen nun alle Zweierkombinationen dieser beiden Mengen durchlaufen werden.

Für dieses Suchverfahren besteht die Möglichkeit, gewisse Kombinationen, die ein Triplett bilden zu übersehen. Dies kann genau dann auftreten, wenn mehrere Cluster innerhalb des Akzeptanzbereiches ε liegen. Jedoch ist dies nicht kritisch, da aufgrund der hohen Detektorgenauigkeit eine gute Spurseparation gegeben ist. Dennoch muss dieser Fall behandelt werden. In Abbildung 4.2 und 4.3 ist eine mögliche Anordnung gezeigt, bei der zwei Kombinationen übersehen werden. Dabei wird angenommen, dass der Akzeptanzbereich für ein Triplett dem vertikalen Bildausschnitt entspricht. Denn dann genügen alle 6 Kombinationen der Bedingung für ein Triplett. Ein solcher Fall tritt in der Realität nur

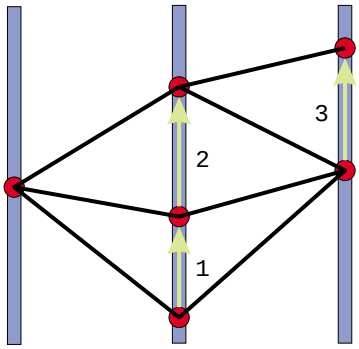

Abbildung 4.2: Hier sind alle Dreierkombinationen gekennzeichnet, die vom Suchalgorithmus durchlaufen werden. Die Pfeile zeigen den jeweils nächsten Schritt an.

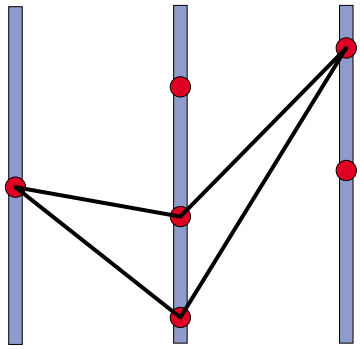

Abbildung 4.3: Diese beiden Dreierkombinationen werden vom Suchalgorithmus übersehen.

dann auf, wenn zwei Spuren sehr dicht beieinander liegen, so dass sie beide im Akzeptanzbereich für ein Triplett liegen. Für einen kleinen Akzeptanzbereich ist dieser Fall sehr unwahrscheinlich. Tritt er dennoch ein, so wären die Spurparameter fast identisch und lägen innerhalb desselben Fehlers, so dass es kaum einen Unterschied für die Analyse macht, ob eine oder zwei Spuren vorhanden sind. Die Messung in Abschnitt 4.1.3 gibt Aufschluss über die Effizienz des Suchalgorithmus für das Auffinden von speziellen Spuren.

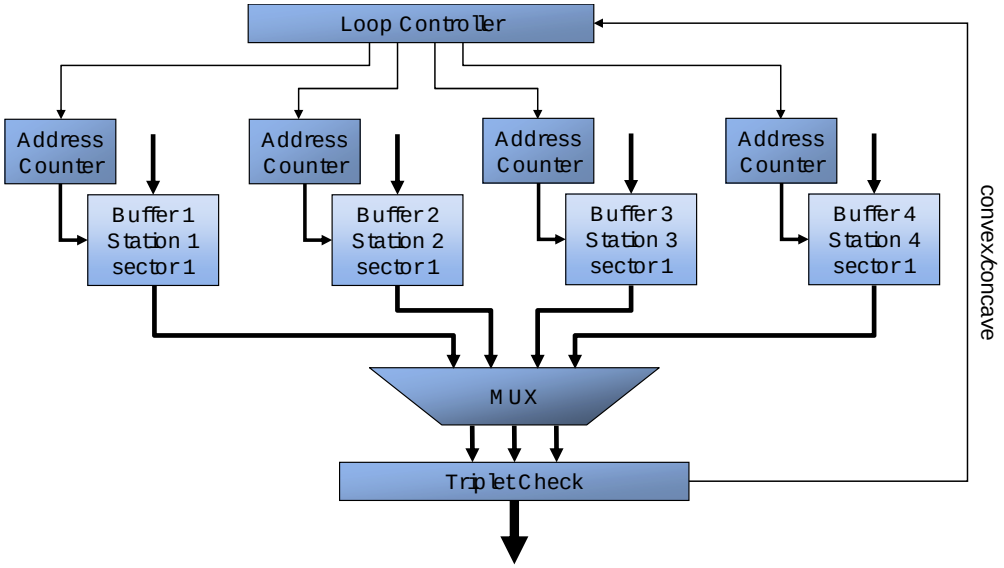

Abbildung 4.4: Blockdiagram für die Triplettsuche.

4.1.2 Implementierung

Die Implementierung wurde so gewählt, dass in jedem Takt eine Entscheidung für ein Triplett produziert wird. Dies setzt voraus, dass pro Takt eine neue Dreierkombination zur Überprüfung an die *Triplet-Check*-Einheit geliefert werden muss. Um dies auf einfachem Wege zu erreichen sind drei Speicherblöcke vorhanden, die jeweils die r-Daten aus demselben Sektor von jeweils drei aufeinander folgenden Stationen enthalten. In Abbildung 4.4 ist ein vierter Speicherblock zu sehen. Der Grund warum dieser benötigt wird ist weiter unten beschrieben. Jedem Speicherblock ist ein eigener Adresszähler zugeordnet. Dieser Adresszähler ist auf Null initialisiert und wird über einen *Enable*-Eingang gesteuert. Zusätzlich ist die Möglichkeit gegeben ihn auf Null zurückzusetzen. Die Adresszähler werden über die *Loop-Controller*-Einheit angesprochen. Die *Loop-Controller*-Einheit ist für den korrekten Ablauf des Suchalgorithmus verantwortlich. Hierzu verwendet die Einheit die Information aus der *Triplet-Check*-Einheit über die Anordnung der vorherigen Dreierkombination, um zu entscheiden, welcher Adresszähler inkrementiert wird.

Der Ablauf

Im Folgenden wird der gesamte Ablauf der Triplettsuche beschrieben (Abb. 4.5 und Abb.4.6). Zu Beginn wird eine Initialisierungsphase durchlaufen. In Buffer 1 werden die r-Daten aus Station-1/Sektor-1 geschrieben. Der zugehörige Adresszähler wird über die Anzahl der r-Cluster informiert. Außerdem wird der Zähler auf Null gesetzt, so dass er den ersten r-Cluster referenziert. Dasselbe wird mit Buffer 2 und 3 durchgeführt, wobei jeweils die Daten aus Station-2/Sektor-1 und Station-3/Sektor-1 in die jeweiligen Buffer geschrieben werden. Jetzt kann die Suche nach Tripletts beginnen. Während die Triplettsuche läuft,

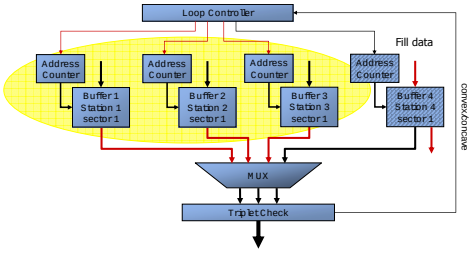

Abbildung 4.5: Erster Schritt der Triplettsuche. Buffer 1, 2 und 3 werden verwendet. Buffer 4 erhält die Daten der nächsten Station.

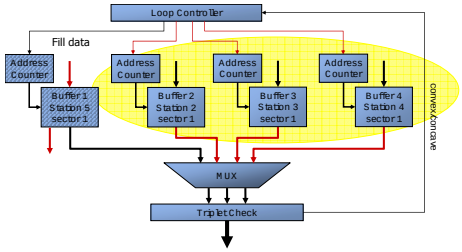

Abbildung 4.6: Zweiter Schritt der Triplettsuche. Buffer 2, 3 und 4 werden verwendet. Buffer 1 erhält die Daten der nächsten Station.

werden die Daten aus Station-4/Sektor-1 in Buffer 4 geschrieben. Ist die Triplettsuche für Station 1, 2, 3 beendet, kann jetzt sofort mit der nächsten Triplettsuche für Station 2, 3, 4 begonnen werden, ohne die Initialisierungsphase für Station-4 abzuwarten. Dies kann in gewisser Weise als eine zweistufige Pipeline angesehen werden, in der die Initialisierungsphase Parallel zur Berechnungsphase abläuft. Möglich wird dies, da in den meisten Fällen, die Berechnungsphase wesentlich länger dauert als die Initialisierungsphase.[1] Im weiteren Verlauf wird der Buffer, der bei der Triplettsuche nicht verwendet wird, mit den Daten der jeweils nächsten Station neu initialisiert. Dies führt zu einem Round-Robin-Verfahren (Die Reihenfolge der verwendeten Buffer ist: 1, 2, 3 → 2, 3, 4 → 3, 4, 1 → 4, 1, 2 → ...). Das bedeutet, dass im ersten Schritt die Daten von Buffer 1, Buffer 2 und Buffer 3 zunächst jeweils zu den Daten der linken, mittleren und rechten Station korrespondieren. Im zweiten Schritt korrespondieren die Daten von Buffer 2, Buffer 3 und Buffer 4 zu den Daten der linken, mittleren und rechten Station. Alle Buffer ändern somit ihre relative Position. Deswegen ist ein 4 zu 3 Multiplexer zwischen den Buffern und der *Triplet-Check*-Einheit geschaltet, der die Fähigkeit hat seine Ausgänge zu permutieren. Im Grunde besteht dieser Multiplexer aus drei parallel geschalteten 4 zu 1 Multiplexern, die über eine einfache Kontrolllogik gesteuert werden.

Die Triplet-Check-Einheit

Die *Triplet-Check*-Einheit liefert die Entscheidung ob eine Dreierkombination ein Triplett bildet. Zusätzlich informiert sie die *Loop-Controller*-Einheit über die Anordnung der geprüften Dreierkombination (konvex/konkav). Die arithmetischen Operationen, die durchgeführt werden entsprechen bis auf einer Ausnahme der Ungleichung (4.1). Gleichung (4.2) beinhaltet eine Division, die nicht während der Laufzeit durchgeführt werden braucht. Die Quotienten

$$q_i = \frac{z_{i+2} - z_i}{z_{i+1} - z_i}, \quad i = 1, 2, ..., N-2 \tag{4.5}$$

[1] Die Zeit für die Initialisierungsphase steigt linear mit der Anzahl der r-Cluster. Während die Zeit für die Berechnungsphase im schlechtesten Fall einen quadratischen Verlauf aufweist (siehe Abschnitt 4.1.1).

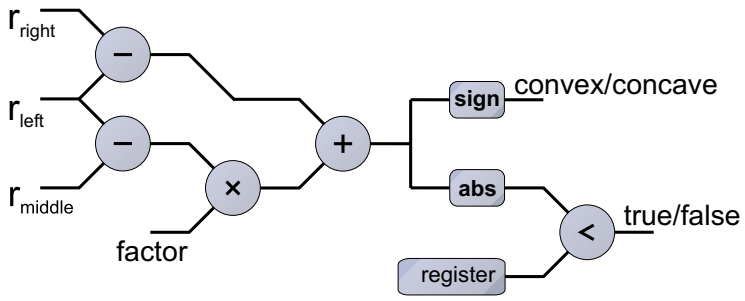

Abbildung 4.7: Blockdiagramm der *Triplet-Check*-Einheit

sind Konstanten, da die z-Positionen der einzelnen VELO-Stationen während der Laufzeit fest sind. Dies bedeutet, dass für eine Anzahl von $N = 24$ Stationen 22 Konstanten vorberechnet werden können. Durch diese Vorberechnung kann die Division eingespart werden.[2] In Abbildung 4.7 ist das Blockdiagramm der *Triplet-Check*-Einheit zu sehen. Ist eine Dreierkombination als Triplett erkannt worden, so wird diese in einen dafür vorgesehenen Speicherblock geschrieben.

4.1.3 Resultat der Simulation

Um zu Überprüfen, inwiefern die 2D-Triplettsuche nützliche Tripletts übersieht, wurde eine Simulation durchgeführt. Dabei werden Tripletts als nützlich angesehen, wenn die entsprechenden Teilchen eine Energie von mehr als 1 GeV aufweisen (RefSet). Teilchen, die aus dem Primärvertex stammen und eine Energie von mehr als 1 GeV haben werden RefPrim genannt. Teilchen, die aus einem B-Zerfall stammen und eine Energie von mehr als 1 GeV haben werden RefB genannt. Die Simulation verwendet eine Datenbank mit 500 Ereignissen. Alle Ereignisse besitzen $B_d^0 \to \pi^+\pi^-$ Zerfälle und wurden mit einer Monte-Carlo-Simulation erzeugt. In Tabelle 4.1 ist der prozentuale Anteil der gefundenen Teilchen für die jeweilige Menge angegeben. Aus diesen Zahlen ist ersichtlich,

Effizienz RefB	99.3%
Effizienz RefPrim	98.9%
Effizienz RefSet	97.8%

Tabelle 4.1: Effizienz der 2D-Triplettsuche.

dass der Suchalgorithmus fast alle Spuren auffindet. Nur ein geringer Teil der Spuren wird übersehen.

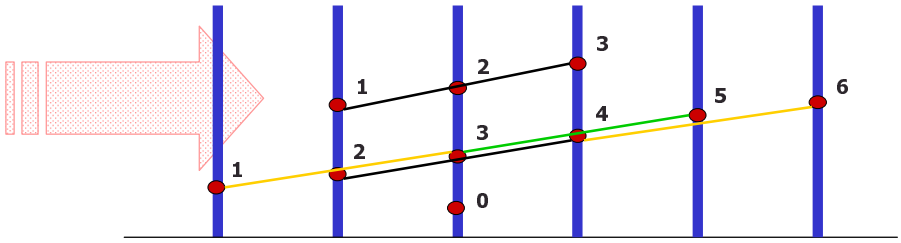

Abbildung 4.8: Graphische Darstellung des Verfahrens.

4.2 Rauschreduktion und Datenfilterung

Durch das Auffinden von Tripletts wird der Informationsgehalt für das Ereignis erhöht. Im speziellen kann jedem r-Cluster eine zusätzliche Information gegeben werden. Im einfachsten Fall ist es denkbar Cluster, die zu einem Triplett gehören, mit einem Flag zu markieren. So ist nach Beendigung der Triplettsuche ersichtlich, welche Cluster keinem Triplett angehören. Diese Cluster können dann im weiteren Verlauf der Rekonstruktion ausgelassen werden.

Ein besserer Ansatz ist, jedem Cluster statt einem Flag eine Zahl zuzuweisen. Zu Beginn der Triplettsuche sind alle Zahlen auf Null gesetzt. Wird ein Triplett gefunden, so wird die Zahl des linken Clusters mit einer Eins, des mittleren Clusters mit einer Zwei und des rechten Clusters mit einer Drei versehen. Wenn im weiteren Verlauf der Triplettsuche, Tripletts gefunden werden, deren linker und mittlerer Cluster bereits Zahlen mit einem Wert ungleich Null besitzten (z.B. 2 und 3), so wird der rechte Cluster mit dem nächsthöheren Zahlenwert versehen (z.B. 4). Durch dieses Verfahren besitzt jeder Cluster im Idealfall die Information über die Anzahl der Cluster, die zur gleichen Spur gehören und sich links von diesem Cluster befinden. Oder mit anderen Worten, jeder Cluster besitzt im Idealfall die Länge seiner Spur links von ihm. Da die Suche von links nach rechts abläuft, ist die zusätzliche Information nur für vorwärts gerichtete Spuren korrekt.

4.2.1 Resultat der Simulation

Mit den beiden oben beschriebenen Verfahren ist im Prinzip ein intelligenter Rauschfilter implementiert. Es können alle Cluster erkannt werden die nicht zu Tripletts gehören. Die Genauigkeit dieses Filters, wurde mit Hilfe einer Simulation überprüft. Dazu wurden 500 $\mathrm{B}_\mathrm{d}^0 \rightarrow \pi^+\pi^-$ Ereignisse mit zusätzlichen zufälligen r-Clustern versehen.

Die Anzahl der Rausch-Cluster hängt vom Rauschpegel ab. Dieser Rauschpegel wird üblicherweise für einen Streifen (vgl. Abschnitt 1.2.1) angegeben, bei $2 \cdot 2048$ Streifen pro r-Station und 25 Stationen, ergibt dies einen Faktor von 102400. Zusammen mit einem Rauschpegel von z.B. 0.1% sind dann

[2] In einem speziellen Fall ist es zumindest theoretisch möglich die Multiplikation einzusparen. Haben alle Stationen den gleichen Abstand zueinander, so haben alle Quotienten den Wert $q_i = 2$. Somit kann die Multiplikation durch einen Linksshift ersetzt werden.

etwa 102 Rausch-Cluster vorhanden. Ein Ereignis hat im Durchschnitt etwa 650 r-Cluster. Zählt man die Rausch-Cluster hinzu, so ergibt sich mit einem Rauschpegel von 0.1% ein Rauschanteil von etwa 13.6% r-Clustern für ein Ereignis. Dieser Wert ist relativ hoch, denn man sollte bedenken, dass die Laufzeit des Rekonstruktionsalgorithmus mit anwachsender Größe des Ereignisses einem quadratischen Verlauf folgt. Eine Reduktion dieser Rausch-Cluster würde zu einer Verbesserung der Laufzeit führen.

Im Folgenden sind drei Tabelle aufgeführt, die das Verhalten des Rauschfilters für unterschiedliche Rauschpegel zeigen. Die Bezeichnungen RefB, RefPrim und RefSet sind aus Abschnitt 4.1.3 übernommen. Mit der Bezeichnung nützliche Spuren sind Spuren gemeint, die drei oder mehr r-Cluster hinterlassen haben. Dementsprechend sind nicht nützliche Spuren solche, die weniger als drei r-Cluster hinterlassen haben. Nicht nützliche Spuren können vom Algorithmus nicht erkannt werden, da mindestens drei r-Cluster eine Spur bilden. Aus Tabelle 4.2 ist ersichtlich, dass für unterschiedliche Rauschpegel etwa 17% der r-Cluster nicht zu Tripletts zugeordnet werden können. Diese gilt vor allem für die Messung ohne Rauschen. Der überwiegende Teil dieser r-Cluster stammt aus Spuren mit nur einem oder zwei Clustern, es sind also nicht nützliche Cluster. Die Daten aus Tabelle 4.3 stützten diese Behauptung, denn fast 88% der nicht nützlichen Cluster sind erkannt und herausgefiltert worden. Der restliche Teil stammt aus Spuren, die zwei Sektoren des Detektors kreuzen. Da sich die Triplettsuche auf gleiche Sektoren beschränkt, können solche Spuren Fragmente hinterlassen, die nicht zu Tripletts verbunden werden können und dementsprechend nicht erkannt werden. Der Anteil der gefilterten nützlichen Cluster aus Tabelle 4.3 entspricht in etwa dem Anteil dieser Fragmente. Es liegt also die Vermutung nahe, dass der überwiegende Teil der gefilterten nützlichen r-Cluster von diesem Typ ist.

Mit anwachsendem Rauschpegel, reduziert sich die Menge der gefilterten r-Cluster. Dieser Effekt ist dadurch zu erklären, dass mit steigender Anzahl der Rausch-Cluster auch die Wahrscheinlichkeit steigt, dass einzelne Rausch-Cluster mit anderen Clustern *falsche* Tripletts bilden.

Rauschpegel	Gefilterte Cluster	Gefilterte Rausch-Cluster
0%	17.3%	-
0.05%	17.1%	90.4%
0.10%	17.0%	90.3%
0.15%	16.6%	89.6%
0.20%	16.5%	88.3%
0.30%	16.0%	86.5%

Tabelle 4.2: Für unterschiedliche Rauschpegel ist angegeben, um wie viel Prozent die Anzahl der r-Cluster reduziert wird. Zusätzlich ist die Menge der gefundenen und herausgefilterten Rausch-Cluster aufgelistet.

Aus Tabelle 4.4 ist ersichtlich, dass durch diese Art des Filterns von r-Clustern, die Effizienz für das Auffinden spezieller Gruppen von Spuren nicht beeinträchtigt ist. Vor allem die Effizienz der gefundenen Spuren aus einem B-

Rauschpegel	Gefilterte nützliche Cluster	Gefilterte nicht nützliche Cluster
0%	3.9%	87.8%
0.05%	3.9%	87.8%
0.10%	3.9%	87.8%
0.15%	3.6%	86.5%
0.20%	3.6%	85.1%
0.30%	3.6%	82.9%

Tabelle 4.3: Für unterschiedliche Rauschpegel ist die durchschnittliche Menge an gefilterten nützlichen und nicht nützlichen r-Clustern angegeben.

Zerfall (RefB) ist stabil gegenüber Rauschen. Dies liegt an der geringen Anzahl der Zerfallsprodukte. Für RefPrim und RefSet ist die Schwankung der Effizienz mit steigendem Rauschpegel vernachlässigbar klein. Die Zahlenwerte in Tabelle 4.4 wurden dennoch so genau angegeben, um zu zeigen, dass eine Schwankung vorhanden ist.

Rauschpegel	Effizienz RefB	Effizienz RefPrim	Effizienz RefSet
0%	99.30%	98.90%	97.81%
0.05%	99.30%	98.90%	97.81%
0.10%	99.30%	98.88%	97.81%
0.15%	99.30%	98.90%	97.81%
0.20%	99.30%	98.88%	97.80%
0.30%	99.30%	98.88%	97.78%

Tabelle 4.4: Für unterschiedliche Rauschpegel ist die Effizienz des Algorithmus für unterschiedliche Spurgruppen angegeben.

Zusammenfassend ist zu sagen, dass durch dieses Verfahren, etwa 17% der r-Daten herausgefiltert werden können. Die Effizienz ist davon nicht beeinträchtigt. Des Weiteren zeigt der Algorithmus bei steigendem Rauschpegel ein stabiles Verhalten bezüglich der Effizienz.

4.3 Implementierung der Primärvertex Bestimmung

In der Softwareimplementierung wird die z-Position des Primärvertex mit Hilfe von rekonstruierten Spuren bestimmt. Hier wurde die Möglichkeit untersucht, die z Position des Primärvertex mit Hilfe von Tripletts zu bestimmen. Außerdem wird kein iteratives Verfahren, wie für die Softwareimplementierung in Abschnitt 3.3 beschrieben, verwendet. Hier wird ausschließlich ein Histogramm erstellt, aus dem die z-Position des Primärvertex bestimmt wird. Für das Histogramm wird ein separater Speicherblock verwendet. Dieser ist zu Beginn vollständig auf Null initialisiert ist. Jede Speicheradresse entspricht einem Bereich auf der z-Achse. Wenn ein Bereich von $\pm$100mm um den Ursprung abgedeckt werden soll und das Histogramm 200 Einträge besitzt, dann entspricht ein Eintrag einem Bereich von 1 mm. Liegt der Schnittpunkt eines Tripletts

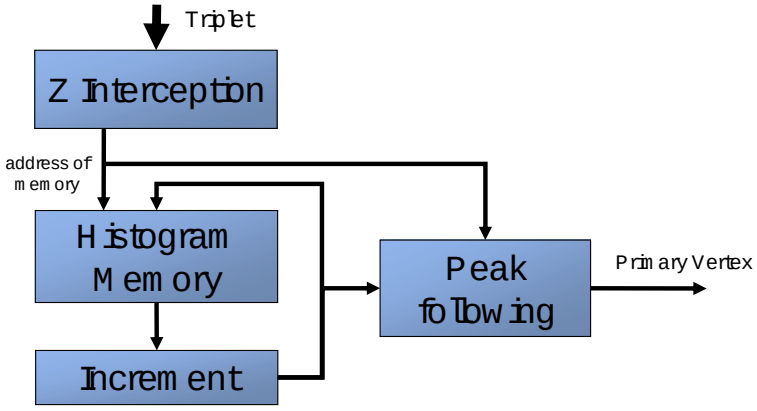

Abbildung 4.9: Blockdiagramm zur Bestimmung der z-Position des Primärvertex

mit der z-Achse in einem dieser Bereiche, so wird der Wert der entsprechenden Speicheradresse ausgelesen, um Eins erhöht und auf die gleiche Speicheradresse zurückgeschrieben. Gleichzeitig wird überprüft, ob der inkrementierte Wert das Maximum im Histogramm ist. Wenn dies der Fall ist, wird sowohl der Wert als auch die Adresse separat gespeichert. Hat der Wert eine gewisse Größe erreicht wird die gesamte Berechnung angehalten. Zusammen mit den Werten der linken und rechten Nachbaradressen wird ein gewichteter Mittelwert bestimmt. Dieser entspricht der z-Position des Primärvertex. In Abschnitt 4.3.1 sind mehrere Messungen aufgeführt, die zeigen mit welcher Genauigkeit die z-Position des Primärvertex bestimmt werden kann. Hinsichtlich der Implementierung wurde ein Dual-Port-Memory verwendet, da gleichzeitig vom Speicher gelesen und in den Speicher geschrieben werden muss. Für die Schnittpunktberechnung werden die ersten zwei Cluster eines Tripletts verwendet. Die Formel lautet

$$z_{intercept} = z^{(i)} - \frac{r^{(i)}}{r^{(i+1)} - r^{(i)}} \left(z^{(i+1)} - z^{(i)} \right). \tag{4.6}$$

Es spricht nichts dagegen, die Bestimmung der z-Position parallel zur Triplettsuche durchzuführen. Falls die Verarbeitung nicht schnell genug erfolgt, ist es denkbar, den Speicherblock für die Tripletts als Puffer zu verwenden. Dies setzt voraus, dass dieser Speicherblock als Dual-Port-Memory implementiert ist.

Der Level-0 Trigger ist nicht in der Lage alle Ereignisse mit mehr als einem Primärvertex herauszufiltern. Es ist denkbar, dass diese Einheit zusätzlich für die Erkennung von Ereignissen mit mehreren Primärvertices verwendet werden kann. Dieses Ereignis könnte dann sofort zurückgewiesen werden, ohne Prozessorzeit für dieses Ereignis zu verschwenden.

4.3.1 Resultat

Im Folgenden wird dargelegt, mit welcher Genauigkeit das oben beschriebene Verfahren den Primärvertex rekonstruiert. Es wurde ein Programm erstellt,

welches das obige verfahren simuliert. Dieses Programm verwendet eine Datenbank, bestehend aus 500 Ereignissen. Alle Ereignisse besitzen $B_d^0 \rightarrow \pi^+\pi^-$ Zerfälle und wurden durch eine Monte-Carlo-Simulation erzeugt. Für jedes Ereignis wurde die z-Position des Primärvertex rekonstruiert und anschließend mit der wirklichen Position des Primärvertex verglichen. Die Differenz wurde in ein Histogramm eingetragen(siehe Abb. 4.10). Da für die arithmetischen Operationen ausschließlich Integer-Arithmetik verwendet wird, sind mehrere Messungen mit unterschiedlichen Genauigkeiten der r-Cluster durchgeführt worden. Die Abweichung vom tatsächlichen Wert, entspricht einer Gauß-Verteilung.

$$f(x) \propto e^{\frac{(x-x_0)^2}{2\sigma^2}} \tag{4.7}$$

In Tabelle 4.5 ist die Standardabweichung σ der Gauß-Verteilung für unterschiedliche Genauigkeiten der r-Cluster angegeben. Die Genauigkeit mit dem

Genauigkeit	x_0	σ
1 μm	-0.03 mm	0.23 mm
5 μm	0.01 mm	0.21 mm
10 μm	0.005 mm	0.23 mm
20 μm	-0.01 mm	0.27 mm
50 μm	-0.01 mm	0.31 mm

Tabelle 4.5: Mittelwert und Standardabweichung der Gauß-Verteilung für unterschiedliche Genauigkeiten der r-Cluster

der Primärvertex bestimmt werden kann, liegt bei etwa 300 μm. Es ist zu sehen, dass bis zu einer Genauigkeit von 10 μm der Primärvertex auf etwa 0.23 mm genau bestimmt werden kann, dann steigt der Wert an.

4.3.2 Laufzeitabschätzung

Der dominierende Teil der Hardwareimplementierung ist die 2D-Triplettsuche. Die Primärvertex Bestimmung mit Tripletts und die Rauschreduktion können parallel zur Triplettsuche durchgeführt werden. Die Laufzeit der Triplettsuche hängt von der Anzahl der überprüften Dreierkombinationen und der Taktfrequenz, mit der eine Überprüfung durchgeführt wird ab. Mit den oben erwähnten Ereignisdaten wurde die Anzahl der Triplettüberprüfungen bei 500 Ereignissen bestimmt und in ein Histogramm eingetragen. Da die Triplettsuche auf gleiche Sektoren beschränkt ist, können mehrere Triplettsucheinheiten parallel laufen. Für eine Detektorgeometrie, die zukünftig 8 statt wie bisher 6 Sektoren aufweist, ergeben sich 8 Triplettsucheinheiten, die parallel ausgeführt werden. Bei einer angenommenen Taktfrequenz von 50 MHz, führt dies zu der in Abbildung 4.11 gezeigten Zeitverteilung. Die durchschnittliche Laufzeit liegt bei 15 μs.

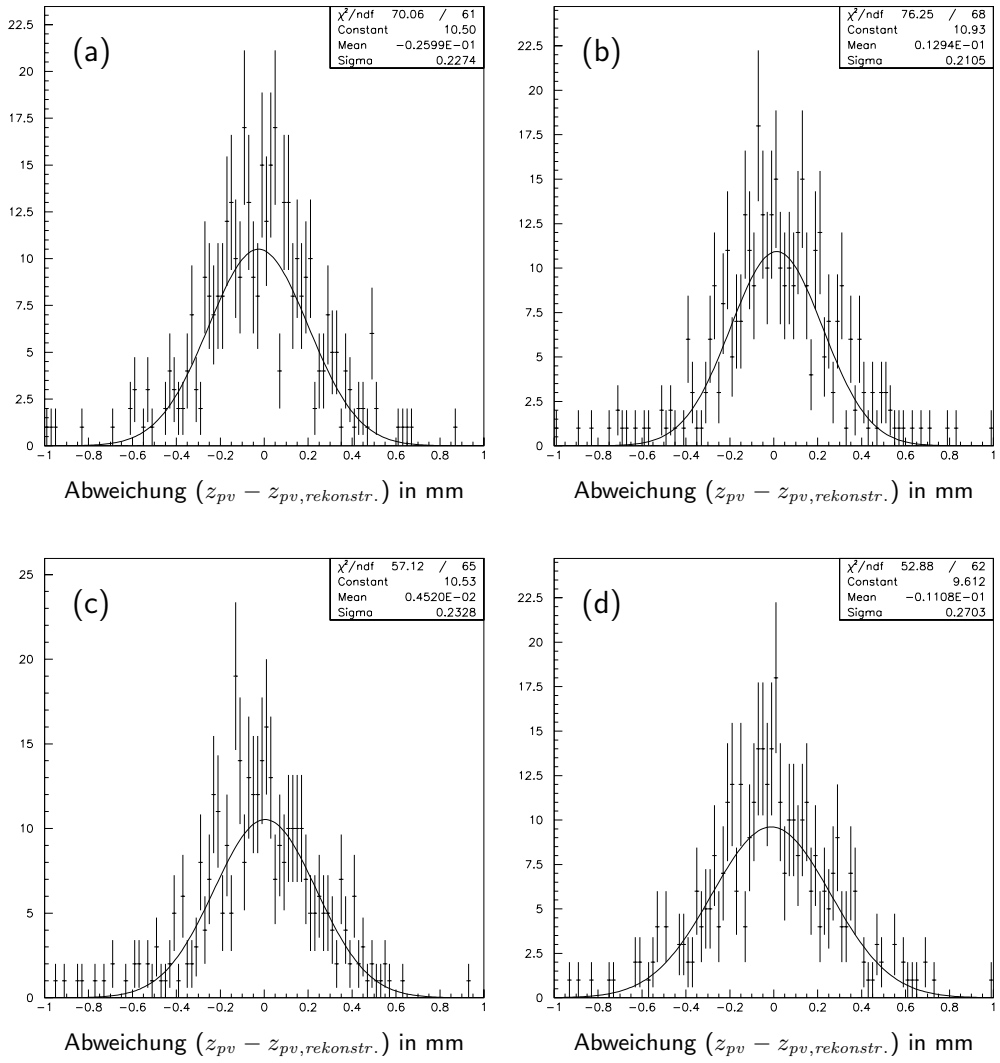

Abbildung 4.10: Die Histogramme zeigen die Abweichung in mm zwischen der rekonstruierten und der wahren z-Position des Primärvertex für unterschidliche Genauigkeiten der r-Cluster an. Für jedes Histogramm sind die gleichen 500 Ereignisse verwendet worden. (a) Die r-Cluster haben eine Genauigkeit von 1 μm. (b) Die r-Cluster haben eine Genauigkeit von 5 μm. (c) Die r-Cluster haben eine Genauigkeit von 10 μm. (d) Die r-Cluster haben eine Genauigkeit von 20 μm.

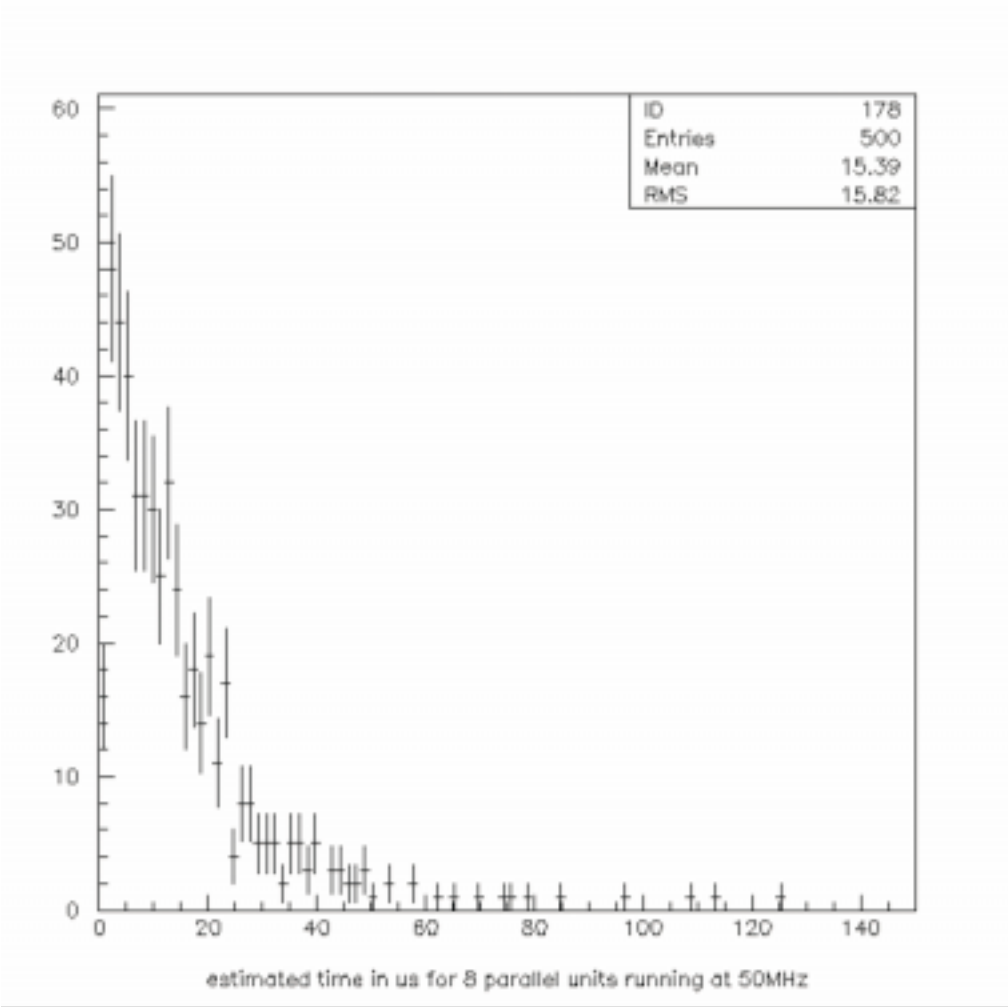

Abbildung 4.11: Zeitverteilung der 2D-Triplettsuche. Aufgetragen ist die Anzahl der Ereignisse über die Zeit in μs.

Kapitel 5

Zusammenfassung

Mit dieser Arbeit wurde gezeigt, dass eine Auslagerung eines Teils des Rekonstruktionsalgorithmus auf einen Coprozessor möglich ist. Es wurde dargelegt welche Teile für eine Hardwareimplementierung sinnvoll sind. Zusätzlich wurden Verbesserungsvorschläge hinsichtlich des Suchverfahrens und der Datenreduktion gemacht.

Eine Untersuchung zur Bestimmung des Primärvertex mit Hilfe von Tripletts anstelle von Spuren wurde durchgeführt. Dieser Teil kann parallel zur Triplettsuche erfolgen. Die Genauigkeit mit dem der Primärvertex bestimmt wird, ist nicht mit der Softwareimplementierung zu vergleichen. In der Softwareimplementierung wird eine Genauigkeit von etwa 50 μm erreich, während in der Hardwareimplementierung eine Genauigkeit von etwa 300 μm erreicht wird. Dies liegt hauptsächlich daran, dass Tripletts anstelle von vollständigen Spuren zur Bestimmung verwendet werden. Die Genauigkeit reicht jedoch aus, um sie an den Softwareteil weiterzugeben, der dann durch ein iteratives Verfahren die Genauigkeit auf 50 μm erhöhen kann.

Des Weiteren ist eine Methode aufgezeigt worden, wie die Datenmenge reduziert werden kann, ohne den Rekonstruktionsalgorithmus zu beeinflussen. Der Softwareteil kann dadurch schneller ausgeführt werden, wenn nur die für die Rekonstruktion wichtigen Daten verwendet werden. Außerdem ist mit diesem Verfahren ein guter Filter entwickelt, der Rauschen gut unterdrücken kann.

Die Berechnungszeit für die 2D-Triplettsuche liegt bei geschätzten 15 mus. Dies ist ungefähr $100 - 150$ mal schneller als es zum gegenwärtigen Zeitpunkt mit der Softwareimplementierung möglich ist.

Abschließend lässt sich sagen, dass es durchaus möglich ist Teile des Rekonstruktionsalgorithmus auf ein FPGA auszulagern, um die Spurrekonstruktion schneller durchzuführen. Dies kann zu einer Einsparung von Rechenknoten führen.

Literaturverzeichnis

[1] LHC Homepage.
`http://lhc-new-homepage.web.cern.ch`.

[2] LHCb Homepage.
`http://lhcb.cern.ch`.

[3] Homepage von F. Eisele.
`http://www1.physi.uni-heidelberg.de/~eisele`.

[4] LHCb Technical Proposal — A Large Hadron Collider Beauty Experiment for Precision Measurements of CP Violation and Rare Decays. CERN LHCC 98-4, LHCC/P4, 20. Februar 1998.

[5] K. Kleinknecht. *Detektoren für Teilchenstrahlung*. Teubner, 1992.

[6] LHCb VELO TDR. CERN/LHCC 2001-0011, LHCb TDR 5, 31. May 2001.

[7] CERN S-Link Homepage.
`http://hsi.web.cern.ch/HSI/s-link`.

[8] A. Walsch. *Architecture and Prototype of a Real-Time Processor Farm Running at 1 MHz*. Dissertation, Universität Mannheim, 2002.

[9] B. Pohv. *Teilchen und Kerne*. Springer, 1999.

[10] B. Stroustrup. *Die C++-Programmiersprache*. Addison-Wesley, 2000.

[11] K. Skahill. *VHDL for Programmable Logic*. Addison-Wesley, 1996.

[12] J. Bhasker. *A VHDL Primer*. Prentice Hall PTR, 1999.

[13] I. Kisel I. Abt, D. Emeliyanov and S. Masciocchi. CATS: a Cellular Automaton for Tracking in Silicon for the HERA-B vertex detector. *NIM*, 2002. to be published.

[14] R. M. Richter. Implementierung eines Algorithmus für den LHCb Level-1 Vertex-Trigger. Diplomarbeit, Universität Heidelberg, 2000.

[15] ALICE Homepage.
`http://alice.web.cern.ch/Alice/`.

[16] ATLAS Homepage.
http://atlasinfo.cern.ch.

[17] CMS Homepage.
http://cmsdoc.cern.ch.

[18] Altera Corporation.
http://www.altera.com.

[19] Homepage von H. Müller.
http://home.cern.ch/hmuller.

[20] LHCb Online System — Data Acquisition & Experiment Control. CERN/LHCC 2001-40, LHCB TDR 7, 19. Dezember 2001.

[21] M. Schulz. Präsentation zum L1 Status, 26. September 2000.
http://lhcb-trig.web.cern.ch/lhcb-trig/pdf/2000_09_26_schultz.pdf.

[22] P. Binko. C++ Coding Conventions. LHCb Computing Note, LHCb 98-049 COMP, May 1998.

[23] Dolphin Interconnect Solutions AS. *SISCI API User Guide*, 2001.

[24] *GAUDI - User Guide Version 7*, 2001.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich beim erstellen dieser Diplomarbeit unterstützt haben.

An erster Stelle danke ich Prof. Volker Lindenstruth, der mich, wo immer es ging, unterstützt hat. Vor allem für seine Ratschläge bin ich ihm sehr dankbar.

Besonderen Dank gilt Dr. Ivan Kisel, der mir seit seiner Einstellung immer mit Rat und Tat zur Seite stand. Für seine Hilfe bei der Zusammenstellung meines Vortrags für das L1-Review in Lausanne bin ich ihm äußerst dankbar. Außerdem wurde er nie müde meine unzähligen Fragen, die ich hatte zu beantworten. Danke dafür.

Zum Schluss möchte ich meinen Eltern und meinen beiden Brüdern danken. Besonders meiner Mutter bin ich für ihre moralische Unterstützung dankbar, die sie mir all die Jahre gegeben hat, und ohne die ich nie so weit gekommen wäre.

Erklärung:

Ich versichere, daß ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den 20. Oktober 2002

.....

Konstantinos Giapoutzis