Fakultät für Physik und Astronomie

Ruprecht-Karls-Universität Heidelberg

Diplomarbeit

im Studiengang Physik

vorgelegt von

Max Christoph Urban

aus Würzburg

Mai 2000

Ein schneller Trigger
für H1
bei HERA

Die Diplomarbeit wurde von Max Christoph Urban ausgeführt am
Physikalischen Institut
unter der Betreuung von
Herrn Professor Dr. Ulrich Straumann

Übersicht

Im Rahmen des Umbaus des HERA-Speicherrings wird die bisherige zentrale innere Spurkammer (**C**entral **I**nner **P**roportional-, CIP-Kammer) des H1-Detektors durch eine neue mit mehr Lagen und höherer Granularität ersetzt. Die Anzahl der auszuwertenden Kanäle erhöht sich von 1000 auf ca. 9600. Infolgedessen wird ein neues Auslese- und Triggersystem entwickelt, das mit **F**ield **P**rogrammable **G**ate **A**rrays (FPGAs) der neuesten Generation arbeitet.
Die Erfassung, Zwischenspeicherung und Auswertung der Kammerdaten, sowie das Bilden einer Triggerentscheidung erfolgt im **Triggeralgorithmus**, der flexibel in FPGAs programmiert ist. Neben den FPGAs ist keine zusätzliche Schalt- und Speicherlogik notwendig.
Die für das Triggersystem notwendige Hardware ist auf **Triggerkarten** untergebracht, auf denen sich die FPGAs befinden.
Die Triggerkarten sowie der Triggeralgorithmus sind Gegenstand der vorliegenden Arbeit.

Abstract

A fast Trigger for H1 at HERA

In the consequence of the upgrade of the HERA Ring the currently used Central Inner Proportional Chamber (CIP) in the H1 Detector is replaced by a new one with more planes and higher granularity. The number of the channels to be evaluated is increased from 1000 to about 9,600. Therefore, a new readout and trigger system is needed. The object of this diploma thesis is the development of a trigger system using the latest generation of **F**ield **P**rogrammable **G**ate **A**rrays (FPGAs).
Chamber data are collected, stored and evaluated using the **trigger algorithm** allowing a simple and fast modification process to a trigger decision and is flexibly programmed in FPGAs. Besides the FPGAs there are no other additional logic components necessary.
Trigger cards carry the hardware that is necessary for the trigger system and contain the FPGAs.
Trigger cards and the trigger algorithm are the object of this project.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Seit 1992 ist am Deutschen Elektronen Synchrotron (DESY) in Hamburg die Hadronen-Elektronen-Ring-Anlage (HERA) in Betrieb. In dieser Anlage werden Elektronen und Protonen zur Kollision gebracht. Mit dem im Ring befindlichen H1-Detektor werden diese Kollisionen erkannt. Die Auswertung der Kollisionen bietet eine Möglichkeit, den Aufbau von Elementarteilchen zu verstehen. Ab September 2000 wird der HERA Speicherring modifiziert, um Luminosität um einen Faktor fünf zu erhöhen. Im Rahmen des "H1 Upgrade-2000 Projekts" wird auch der H1 Detektor neu gestaltet. Die Zentrale Innere Proportional Kammer (**C**entral **I**nner **P**roportional Chamber (CIP-Kammer)) wird durch eine neue CIP-Kammer ersetzt. Da bei dieser Kammer etwa fünfmal so viele Kanäle ausgelesen werden müssen, wird ein neues Auslese- und Triggersystem entwickelt.
Diese Diplomarbeit befasst sich mit der Entwicklung und dem Bau des Triggersystems für die CIP-Kammer. Das Triggersystem wird in Field Programmable Gate Arrays (FPGAs) programmiert, die erst seit kurzer Zeit hinreichend große Kapazitäten bieten. Die FPGAs werden vollständig in der Hardware Beschreibungssprache *Verilog* programmiert. Ziel ist es, ein (Trigger)system zu entwickeln, das eine universelle Hardware verwendet, die durch Programmierung angepasst werden kann. Software kann ohne mechanische Änderungen verbessert und optimiert werden. Nachträgliche Veränderungen sind so gut möglich. Die Entwicklung unterteilt sich in zwei Bereiche. Auf der einen Seite wird die Entstehung des Triggeralgorithmus erfolgen, der in die FPGA programmiert wird. Auf der anderen Seite soll die Entwicklung der notwendigen Hardware beschrieben werden.

Insgesamt sind sechs Kapitel entstanden. Die Schwerpunkte der einzelnen Kapitel sollen hier kurz genannt werden:

- Kapitel 1 ist diese Einleitung
- Im Kapitel 2 erfolgt eine allgemeine Beschreibung des HERA Speicherrings, des H1-Detektors und des Triggersystems im H1-Detektor. Zudem wird die CIP-Kammer und das Auslese- und Auswertungssystems der Kammer beschrieben.
- Kapitel 3 befasst sich mit der theoretischen Beschreibung des Triggeralgorithmus, der in die oben erwähnten FPGAs programmiert wird. In diesem Kapitel erfolgt die Beschreibung des Gesamtkonzepts und eine funktionale Beschreibung der Komponenten des Triggersystems.

- Im vierten Kapitel wird die Entwicklung der Software erklärt. Ein sehr wichtiger Schritt in der Entwicklung ist die Simulation, auf die auch eingegangen wird. Es werden zudem notwendige Entwicklungswerkzeuge beschrieben.
- Im fünften Kapitel werden die entstandenen Komponenten dokumentiert.
- Im sechsten Kapitel werden die Hardwaretests geschildert, die an den entstandenen Komponenten des Systems durchgeführt wurden. Hardwaretests geben Aufschluss über die Funktion des Systems in einem realistischen Umfeld.
- Im Anhang befinden sich Schaltpläne und Programmcodes der entwickelten Komponenten.

In dieser Diplomarbeit werden sehr oft Fachbegriffe aus dem Umfeld der Elektronik verwendet. In einigen Fällen ist eine ausführliche Beschreibung dieser Begriffe sehr aufwendig und erfolgt deshalb nicht an dieser Stelle. Zur Erklärung dieser Begriffe sind die folgenden Werke gut geeignet [Ti91], [Ha95] oder [St98]. Im Anhang befindet sich eine Liste verwendeter Abkürzungen.

Kapitel 2

Das H1-Experiment

Der in der vorliegenden Arbeit beschriebene Trigger[1] dient der Spurfindung von geladenen Teilchen mit der neuen zentralen Vieldrahtproportionalkammer (**C**entral **I**nner **P**roportional chamber, CIP), die im Rahmen des H1-Upgrade 2000-Projekts [HU98] gebaut wird. Die CIP-Kammer ist Teil des H1-Detektors am Speicherring HERA der Großforschungsanlage DESY in Hamburg, der durch das Upgrade 2000 neu gestaltet wird. Im folgenden werden nach einer Beschreibung des Beschleunigersystems die wichtigsten Komponenten des H1-Detektors beschrieben. Anschließend wird das H1-Triggersystem erläutert. Besondere Beachtung soll dabei der CIP-Trigger und das H1-Upgrade finden.

2.1 Der Speicherring HERA am DESY

HERA ist das weltweit einzige Beschleunigersystem, in dem Elektronen und Protonen bei hohen Energien zur Kollision gebracht werden. Bevor die Protonen– und Elektronenstrahlen an den Wechselwirkungspunkten zur Kollision kommen, durchlaufen sie verschiedene Vorbeschleunigungsstufen. Die Schwerpunktsenergie beträgt $\sqrt{s_{ep}}=$ 320 GeV. Einen Überblick über den Speicherring und das Vorbeschleunigersystem gibt Abbildung2.1.

HERA befindet sich in einem Tunnel in einer Tiefe von 10 bis 15 Metern unter der Erde und besteht aus zwei parallel verlaufenden Strahlrohren, in denen Elektronen und Protonen in entgegengesetzter Richtung umlaufen. Der Umfang der Ringe beträgt 6336 m.

An zwei Stellen, an denen sich Nachweisdetektoren befinden, werden die Strahlrohre zusammengeführt, und die gegenläufigen Strahlen werden zur Kollision gebracht. Einer der Nachweisdetektoren ist der H1-Detektor (Halle Nord), der andere ist der ZEUS-Detektor (Halle Süd). Alle 96 ns kann eine Kollision stattfinden. Die HERA-Clock ist der zentrale Taktgeber, mit der die Beschleunigungsmagneten synchronisiert werden, die die Teilchenpakete (Bunches) bilden. Sie ist auf 10,4 MHz ($= 1/(96\,ns)$) festgelegt. Sie wird an alle Detektoren geführt, um interne Systeme zu synchronisieren. Eine Teilchenkollision wird als *Bunch Crossing* (BC) bezeichnet.

[1] Trigger = Auslöser, in diesem Fall löst der Trigger die Datennahme aus

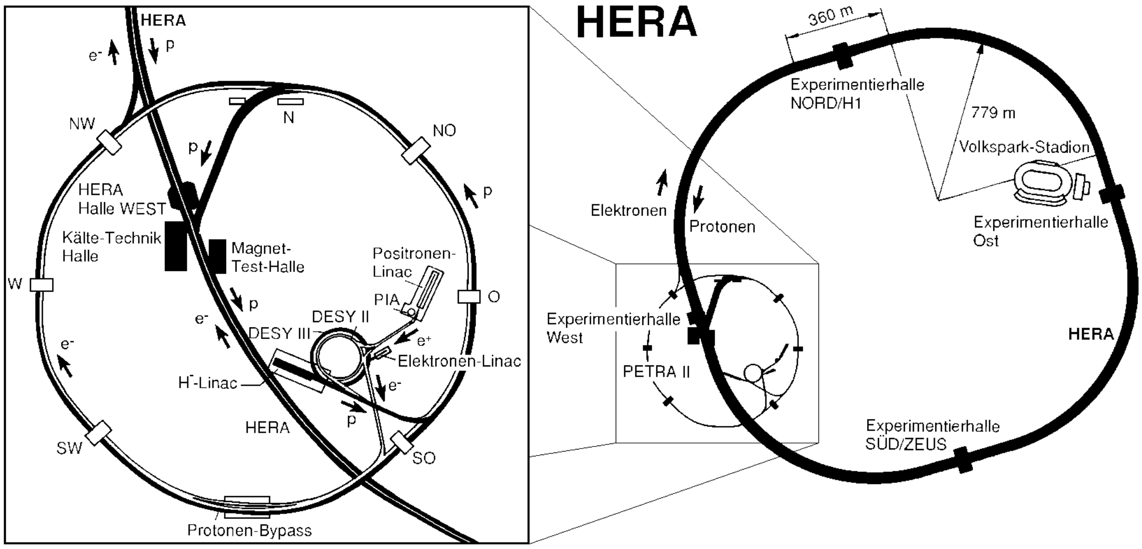


Abbildung 2.1: Der HERA–Speicherring (rechts) und die Vorbeschleuniger in der Ausschnittsvergrößerung (links).

Während der Datennahmeperiode 1994–97 lief HERA mit Positronen und Protonen, die auf eine Maximalenergie von $E_0 = 27,6\,\mathrm{GeV}$ bzw. $E_p = 820\,\mathrm{GeV}$ beschleunigt wurden.

Neben den Experimenten H1 und ZEUS, die verschiedenste Aspekte der ep–Streuung detektieren, gibt es zwei weitere Experimente am HERA–Speicherring, HERMES (Halle Ost) und HERA–B (Halle West), die jeweils nur einen Teilchenstrahl nutzen. HERMES untersucht die Streuung polarisierter Elektronen an polarisierten Atomkernen [HER98], während HERA–B, das in diesem Jahr die Datennahme aufnimmt, die CP–Verletzung im System der neutralen B–Mesonen untersucht [HE98].

Ausgewählte, typische Betriebsdaten des HERA–Speicherrings für den Zeitraum 1998/99 sind in Tabelle 2.1 zusammengefasst.

	Elektronen	Protonen
Strahlenergie [GeV]	27.6	920
Schwerpunktsenergie [GeV]	320	
Kollisionsfrequenz [MHz]	10.42	
Strahlstrom [mA]	10–30	40–100
magnetisches Ablenkfeld [T]	0.274	4.65

Tabelle 2.1: Betriebsdaten des HERA–Speicherrings während der Datennahme 1998/99.

Luminosität Da die Ereignisrate dN/dt für einen bestimmten Prozess bei einem ruhenden Ziel (fixed target) bedeutend größer ist als bei einem Speicherring, ist es bei letzterem notwendig, eine hohe Strahlintensität zu erreichen. Eine entscheidend von der Beschleunigeranlage abhängige Größe ist die Luminosität $\mathcal{L}$, sie ist definiert als der Proportionalitätsfaktor zwischen der Reaktionsrate dN/dt und dem Wirkungsquerschnitt σ:

$$\frac{dN}{dt} = \mathcal{L}\,\sigma. \tag{2.1}$$

Die Gesamtzahl der Ereignisse einer Reaktion N ist also mit der über die Zeit integrierten Luminosität $L = \int \mathcal{L}\,dt$ verknüpft.
Die seit Inbetriebnahme von HERA gelieferte, sowie die von H1 genutzte integrierte

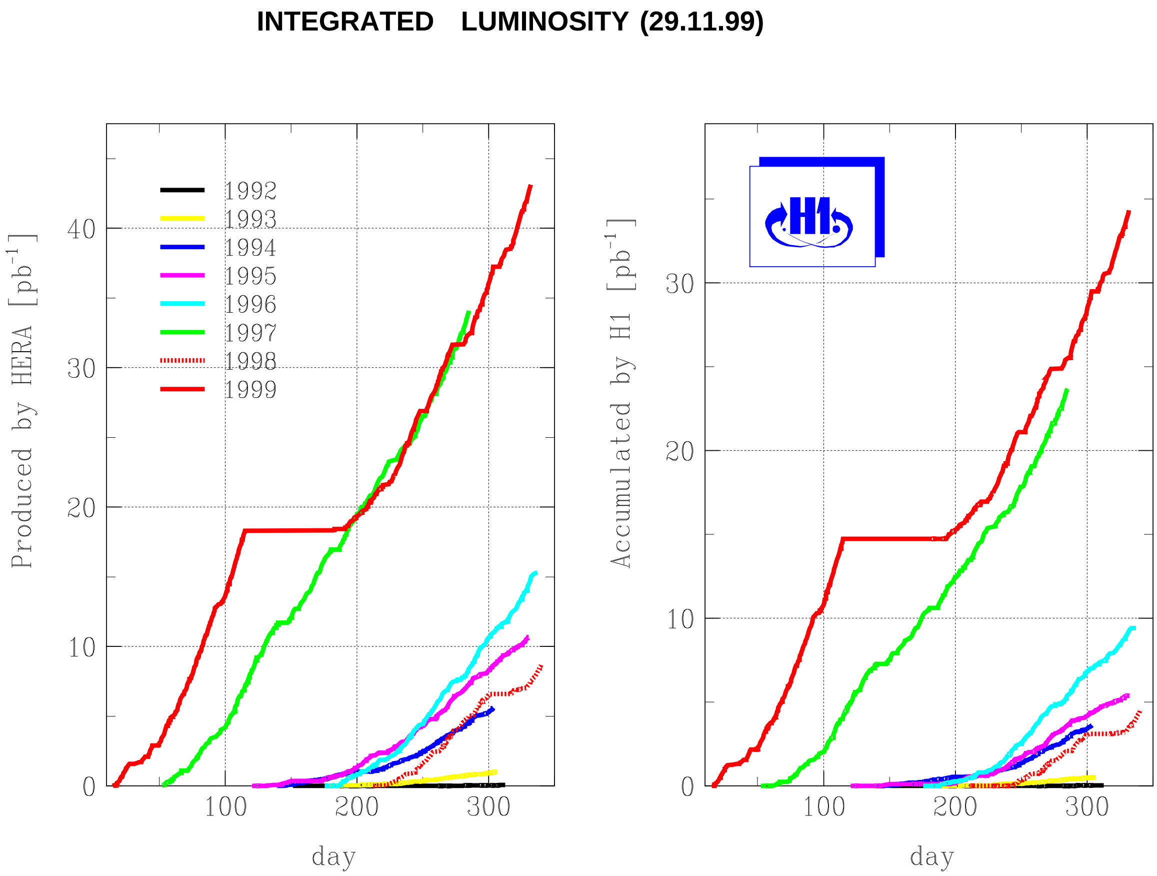


Abbildung 2.2: Die jährlich integrierte Luminosität als Funktion der Zeit.

Luminosität sind in Abbildung 2.2 dargestellt.
Zum Ende dieses Kapitels seien noch kurz die physikalischen Fragestellungen erwähnt, mit denen sich die Experimente an H1 und ZEUS beschäftigen. Eine ausführliche Darstellung kann in [DESY99] oder auch [H100] gefunden werden. Hier erfolgt nur eine Auflistung wichtiger Ziele:

- Bestimmung der Protonenstrukturfunktion $F_2(x, Q^2)$ in Abhängigkeit von x und Q^2. Unter x versteht man im Quark-Parton-Modell den relativen Anteil des Impulses des Partons am Gesamtimpuls des Protons, Q^2 ist das Quadrat des Viererimpulsübertrags.

- Untersuchung der hadronischen Struktur des Photons.
- Präzisionstests der Theorien der starken und elektroschwachen Wechselwirkung.
- Suche nach Physik jenseits des Standardmodells (z.B. Supersymmetrie, SUSY).

2.2 Der H1-Detektor

Der H1–Detektor soll eine genaue und vollständige Beobachtung der Elektron–Proton–Streuungen ermöglichen, wobei der Teilchenidentifikation besondere Beachtung zukommt. Dazu muss er eine gute Impuls– und Energieauflösung gewährleisten und die Wechselwirkungszone möglichst vollständig umschließen. Eine ausführliche Beschreibung ist in [H100] zu finden.

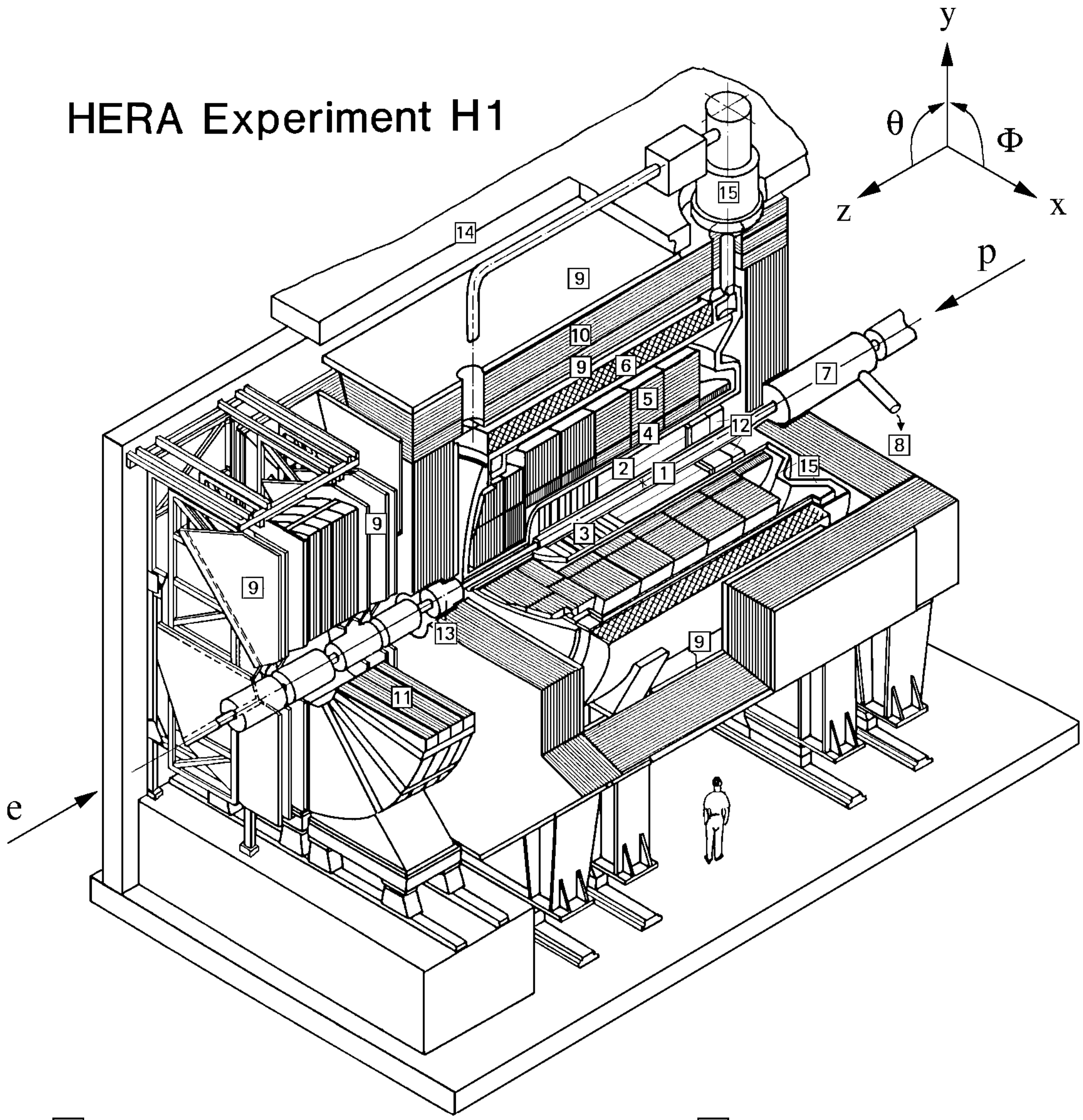
In der Darstellung des Detektors von Abbildung 2.3 kommen die Elektronen von links, die Protonen von rechts. Dadurch wird das H1-Koordinatensystem definiert: Die z–Achse zeigt in Richtung des einlaufenden Protonstrahls, die y–Achse nach oben und die x–Achse auf den Mittelpunkt des Beschleunigerrings, so dass sich ein rechtshändiges Koordinatensystem ergibt. Der Ursprung liegt im nominalen Wechselwirkungspunkt. Die geläufig verwendeten Richtungsangaben vorwärts ($z > 0$) und rückwärts ($z < 0$) beziehen sich entsprechend auf die Flugrichtung der Protonen. Vorteilhaft ist die Verwendung von Kugelkoordinaten. Der Polarwinkel ϑ wird relativ zur positiven z–Achse, der azimutale Winkel ϕ in Bezug auf die positive x–Achse angegeben. Aus Sicht des Elektrons folgt daraus, dass es umso stärker am Proton gestreut wird, je kleiner der Betrag des Winkels ϑ ist. In diesem Koordinatensystem ist der Streuwinkel des Elektrons durch $\vartheta_e^{streu} = \pi - \theta_e$ gegeben. Häufig wird statt ϑ auch die Pseudorapidität $\eta = -\ln \tan \frac{\vartheta}{2}$ benutzt.

Aufgrund der unterschiedlichen Energie der Elektronen und Protonen ist der Detektor stark asymmetrisch aufgebaut.

In der Wechselwirkungszone umschließt ein Siliziumstreifendetektor das Strahlrohr, der wiederum von zylindrischen Spurkammern umgeben ist. Sie dienen dem Nachweis von Teilchenspuren und werden in Richtung des Protonstrahls durch die Vorwärtsspurkammern ergänzt. Zur Messung der Energie und der Position geladener und neutraler Teilchen ist das gesamte Spurkammersystem von einem hufeisenförmigen Flüssig–Argon–Kalorimeter (LAr) umgeben, das aus einer inneren elektromagnetischen und einer äußeren hadronischen Komponente besteht. Der rückwärtige Bereich wird durch das Spaghetti–Kalorimeter (SPACAL) geschlossen, während der Vorwärtsbereich im wesentlichen durch das Flüssig–Argon–Kalorimeter abgedeckt wird und lediglich in der Nähe des Strahlrohrs zur Ergänzung ein kleines hadronisches Kalorimeter (Plug) steht.

Eine supraleitende Spule erzeugt im Kalorimeter– und Spurkammerbereich ein homogenes Magnetfeld parallel zur z–Achse von 1.15 T, welches geladene Teilchen auf gekrümmte Bahnen zwingt (Lorentzkraft) und dadurch mit Hilfe der Spurkammern die Messung ihres Transversalimpulses ermöglicht.

Die Myonenkammern zum Nachweis von Myonen und das mit Streamerkammern instrumentierte Eisenjoch, das neben der Rückführung des magnetischen Flusses als zusätzliches hadronisches Kalorimeter dient, fassen den Detektor ein. Ferner befinden



1 Strahlrohr
2 Zentrale Spurkammern
3 Vorwärtsspurkammersystem
4 Elektromagnetisches LAr–Kalorimeter (Blei)
5 Hadronisches LAr–Kalorimeter (Edelstahl)
6 Supraleitende Spule
7 Kompensationsmagnet
8 Helium–Kälteanlage
9 Myon–Kammern
10 Instrumentiertes Eisen
11 Myon–Toroid–Magnet
12 Kalorimeter (SPACAL)
13 Vorwärts–Kalorimeter
14 Betonabschirmung
15 Flüssig–Argon–Kryostat

Abbildung 2.3: Der H1–Detektor.

sich außerhalb des eigentlichen Detektors in Vorwärtsrichtung ein Myonspektrometer mit Myonkammern und –toroidmagnet, in Rückwärtsrichtung das Luminositätssystem.

2.2.1 Das Spurkammersystem

Das Spurkammersystem ist in den Abbildungen 2.4 und 2.5 im Längs– bzw. Radial-

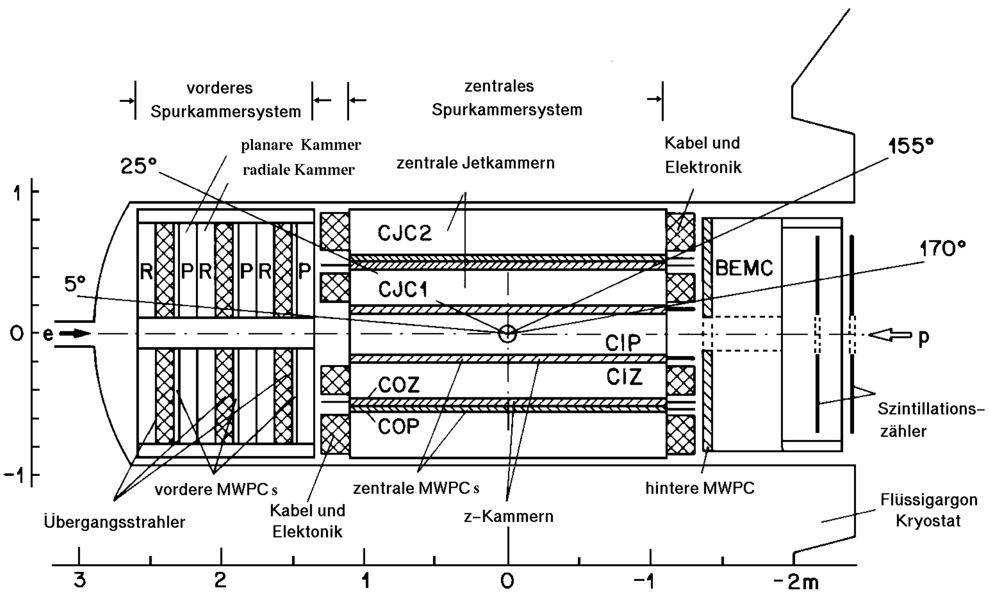


Abbildung 2.4: Längsschnitt durch das Spurkammersystem (1995 wurde das elektromagnetische Rückwärtskalorimeter (BEMC) durch das SPACAL ersetzt).

schnitt dargestellt. Es werden zwei Bereiche unterschieden, die Vorwärtsspurkammern (Forward–Tracker) mit einer Winkelabdeckung von $7^\circ < \theta < 25^\circ$ und die zentralen Spurkammern (Central–Tracker) in einem Winkelbereich von $25^\circ < \theta < 155^\circ$. Im Rückwärtsbereich komplettiert die rückwärtige Driftkammer (Backward Drift Chamber, BDC, $153^\circ < \theta < 177^\circ$) die Spurmessungen.

Die Proportionalkammern werden vom Triggersystem verwendet, welches die Datennahme startet, und dienen zur Bestimmung des Wechselwirkungszeitpunktes.

Das zentrale Spurkammersystem weist zwei Driftkammern auf, die beiden zentralen Jet–Kammern CJC1 und CJC2. Beide Kammern sind zylindrisch um das Strahlrohr angeordnet. Über eine Ladungsvergleichsmessung an beiden Drahtenden ist eine Messung der z–Koordinate mit begrenzter Auflösung möglich.

Neben den Jet–Kammern befinden sich im CTD (Central Trigger Device) des weiteren eine innere und eine äußere z–Kammer (CIZ und COZ), deren Signaldrähte in einer Ebene senkrecht zum Strahlrohr ringförmig verlaufen, wodurch sie eine präzisere Messung der z–Koordinate und des Polarwinkels gestatten. Schließlich sind noch zwei Vieldrahtproportionalkammern vorhanden, die zentrale innere (CIP) und äußere (COP) Proportionalkammer, die als schnelle Auslöser zur Unterscheidung aufeinanderfolgender Bunch Crossings verwendet werden. Dieses Kammersystem sowie das Triggersystem wird im Rahmen des Upgrade–Projekts neu gestaltet (Abschnitt 2.4).

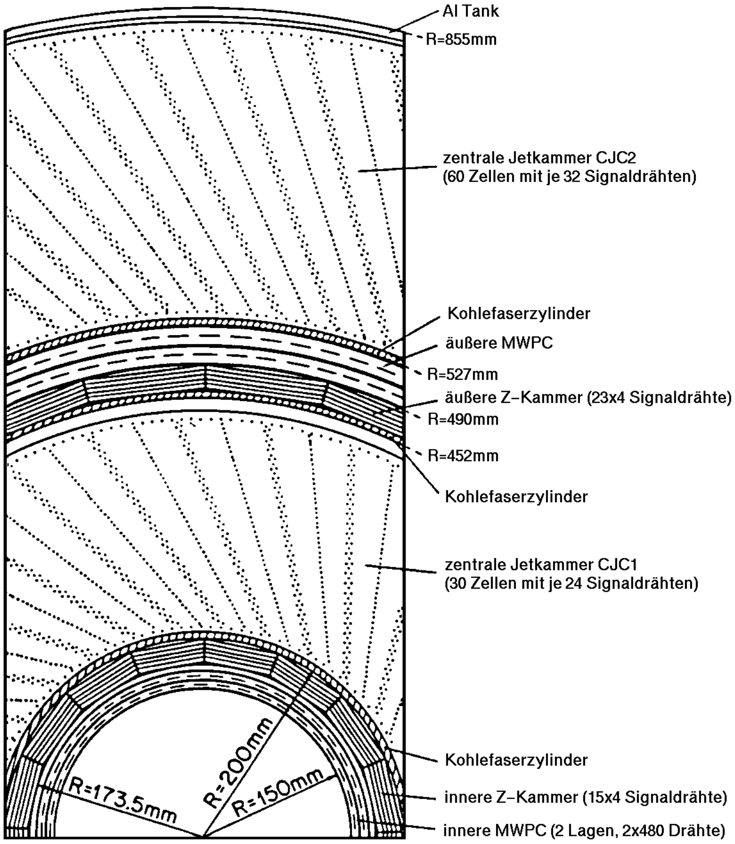


Abbildung 2.5: Querschnitt durch das Spurkammersystem.

2.3 Das H1-Triggersystem

Das Triggersystem hat die Aufgabe, physikalisch interessante *ep*-Streuprozesse von zahlenmäßig dominanten Untergrundereignissen zu unterscheiden. Bei ungefähr jeder tausendsten Strahlkreuzung kommt es zu einer Teilchenkollision. Kollisionen mit dem Strahlrohr oder dem Restgas sollen von solchen unterschieden werden, die etwa im Mittelpunkt des Detektors, also am nominalen Vertex, zwischen Elektronen und Protonen stattfinden. Um zu entscheiden, welche Teilchenkollisionen gespeichert werden sollen, bedarf es eines sehr komplexen Triggersystems.
Trigger, die nur die Information einer einzigen Detektorkomponente verwenden, nennt man Teil– oder Subtrigger. Das gesamte H1–Triggersystem ist aus vielen solcher Subtrigger zusammengesetzt. Die Triggerentscheidung dieser Teil-Trigger wird einem zentralen Trigger Kontrollsystem[2] mitgeteilt. Das Kontrollsystem stellt jedem Subtrigger mehrere Datenkanäle zur Verfügung, in die Informationen über die untersuchte Kollision geschrieben werden. Diese Kanäle, anhand derer die Kontrolleinheit eine Gesamttriggerentscheidung trifft, werden auch Triggerelemente genannt.

Das Triggerergebnis wird von einer mehrstufigen Entscheidungslogik gebildet. Die

[2]Central Trigger Control (CTC)

unterschiedlichen Stufen, sogenannte Triggerlevel, können unterschiedlich schnell eine Triggerentscheidung treffen. Schnelle Systeme, wie dem Level 1–Trigger, werten ein Ereignis "online", also gleichzeitig mit der Kollisionsrate von 10,4 MHz aus. Diese Systeme sind zwar sehr schnell aber dafür nicht genau. Systeme in höheren Stufen überprüfen die Entscheidung vorangehender Triggerlevel. Somit werden die Stufen oder Level nacheinander durchlaufen, bis eine Stufe ein Ereignis verwirft oder alle Stufen durchlaufen wurden und das Ereignis bestätigt wird. Mit steigendem Level werden mehr Informationen für die Triggerentscheidungen ausgewertet.
Das H1-Triggersystem besteht aus mehreren Triggersubsystemen, es gibt vier Triggerstufen. Zur Zeit gibt es allerdings keine Komponenten, die auf Level 3 eine Triggerentscheidung treffen. Jede Triggerstufe reduziert die Daten etwa um einen Faktor 10^4, so dass aus dem anfänglichen Datenstrom (10,4 Mhz) am Ende einige Events pro Sekunde auf Band gespeichert werden (Abb. 2.6).

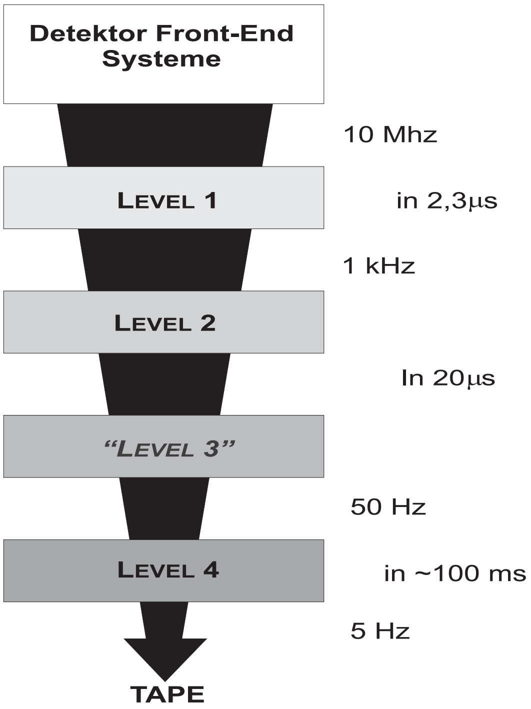


Abbildung 2.6: Datenreduktion durch das Triggersystem.

Level1: Level 1-Trigger arbeiten mit dem HERA-Referenztakt. Sie müssen für jedes Bunch Crossing eine Triggerentscheidung liefern. Insgesamt gibt es 256 Triggerelemente für Level 1, von denen momentan 200 verwendet werden.
Die Level 1 Subtrigger sind:

- Vorwärtsspurtrigger
- CIP-Trigger
- CJC-Trigger

- LAr-Trigger
- SpaCal-Trigger

Die letzten vier setzen sich aus einzelnen Triggerelementen verschiedener Subsysteme zusammen:

- z-Vertex-Trigger
- Cosmics-Trigger
- Flugzeitzähler
- Myonen-Trigger

In Abbildung 2.7 wird der funktionale Ablauf einer L1-Triggerentscheidung anhand eines L1-Triggersubsystems erläutert.

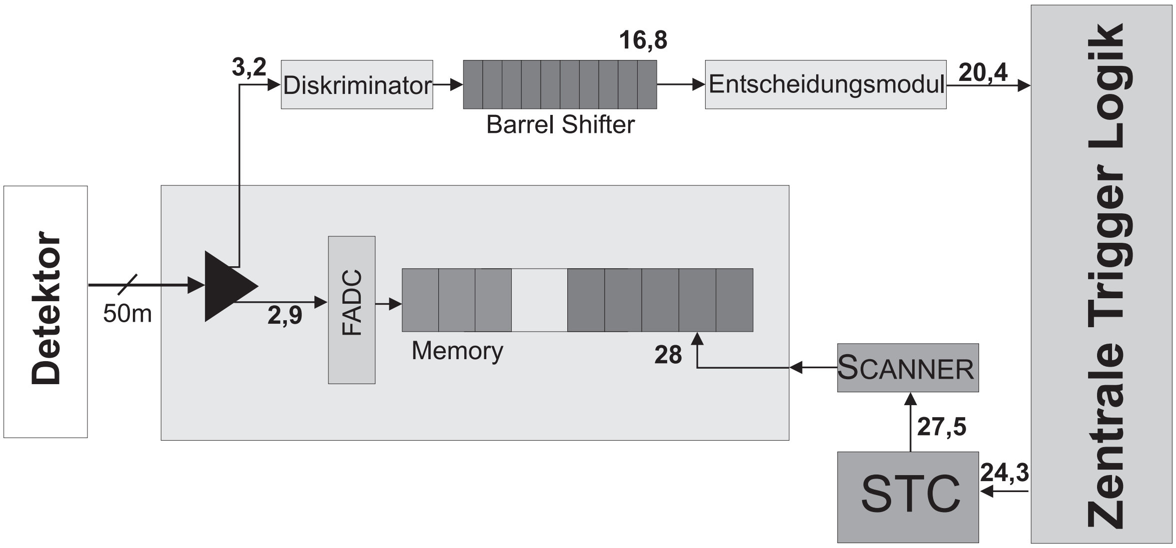


Abbildung 2.7: Ein Level 1 Triggerzyklus am Beispiel des Vorwärts Myon Detektors. Die angegebenen Zahlen bedeuten die Anzahl der BCs, die vergangen sind, bis die Daten an der entsprechenden Stelle sind.

Die Daten einer ep-Kollision aus dem Detektor durchlaufen zwei Wege. Zum einen gelangen die Daten nach 2,9 BCs in einen Analog Digital Wandler (ADC) und werden dann in einem Ringspeicher gespeichert. Gleichzeitig werden sie nach einem Kippschwelle (”ja” oder ”nein”) und einer Verzögerung von 12 BCs im Entscheidungsmodul ausgewertet. Die Verzögerung erreicht man durch einen Barrel Shifter. Ein Barrel Shifter ist ein Schieberegister, das die Daten pro Taktzyklus ein Register weitertransportiert. Die Anzahl der hintereinandergeschalteten Register legt die Verzögerung fest. Nach 20,4 BCs liefert die Triggerelektronik eine Entscheidung an die zentrale Triggerlogik. Diese wertet die Information aller Subtrigger aus und liefert nach 24 BCs die Gesamtentscheidung an alle Subsysteme. Ist die Kollision interessant (Triggerentscheidung positiv), werden die Daten des entsprechenden Ereignisses aus dem Speicher ausgelesen, und das Triggersystem wird gestoppt. Für andere Subtriggersysteme ist der Verlauf ähnlich, auf den Verlauf des CIP-Triggers wird in Kapitel 3.3.2 noch genauer eingegangen.

Level 2: Wird ein Ereignis vom Level 1-Trigger gefunden, gelangt es in den Level 2-Trigger. Dort wird es entweder verworfen (L2-Reject) oder an den nächsten Triggerlevel weitergegeben.
Sobald der L1-Trigger ein Ereignis auswählt, wird die Datennahme unterbrochen. Es entsteht also immer eine Totzeit, wenn der Level 1-Trigger ein Ereignis auswählt. Die Level 1-Systeme beginnen nun die Daten des betreffenden Ereignisses auszulesen, während die Level 2-Subtrigger das Ereignis analysieren. Eine Entscheidung des Level 2-Triggers liegt nach $20\mu s$ vor. Der Level 2-Trigger benutzt topologische Informationen und neuronale Netzwerke zur Entscheidungsfindung. Eine genauere Beschreibung des Level 2-Triggersystems ist in [HU99] zu finden.
Verwirft der Level 2 Trigger das Ereignis, wird das Triggersystem wieder auf die Datennahme vorbereitet.

Level 4: Vom Level 2-Trigger gelangen die Daten in den Level 4-Trigger. Eine positive Entscheidung vom Level 4-Trigger führt zur Aufzeichnung der Detektordaten. Zusätzlich wird 1% der vom Level 4-Trigger verworfenen Ereignisse aufgezeichnet.

höhere Trigger-Level: Die aufgezeichneten Detektordaten, auch Rohdaten genannt, werden von einem als Level 5 bezeichneten Computersystem (Silicon Graphics Challenge) rekonstruiert. Die vom Level 5-Trigger durchgeführte Rekonstruktion wird unabhängig von der Datennahme durchgeführt (Offline).

2.4 H1 Upgrade 2000

Im September 2000 wird der HERA-Speicherring abgeschaltet und im Rahmen des *Upgrade 2000 Projekts* [HU98] umgebaut. Ziel des Umbaus ist es, die Luminosität $\mathcal{L}$ zu erhöhen. Gegenüber den Luminositätswerten, die man für den bisherigen Speichering erwartet hat soll die Luminosität um das fünffache steigen, die integrierte Luminosität wird $\int \mathcal{L} dt = 1,5 \cdot 10^{31} cm^{-2}/s$ betragen.

Man erhofft sich durch ein Upgrade eine sensiblere Beobachtung möglicher Physik außerhalb des Standardmodells. Durch Veränderungen am HERA–Speicherring erwartet man mehr Ereignisse mit hohem Q^2. Die Luminositätssteigerung erreicht man zum einen durch eine bessere Strahlfokussierung von Elektronen- und Protonenstrahl, zum anderen wird der Protonenstrom, also die Anzahl der gespeicherten Protonen, erhöht. Eine bessere Fokussierung erreicht man durch die Neugestaltung der Fokussierungsmagneten. Die Magneten müssen näher an den Kollisionspunkt beider Strahlen gebracht werden, dazu werden sie bis in momentan aktive Bereiche des Detektors installiert, entsprechend muss der Detektor verändert werden. Betroffen sind vor allem die Vorwärtsspurkammern, die sich am vorderen Ende des Detektors befinden, und die elektromagnetischen Kalorimeter im hinteren Teil des Detektors (Spaghetti Kalorimeter, SpaCal).

Die aus der Luminositätssteigerung resultierende Teilchenzahlerhöhung macht es notwendig, die vorhandenen Detektoren sensibler auf die Trennung zwischen den deutlich ansteigenden Untergrundstreuungen und erwünschten Kollisionen auszurichten. Aus diesem Grund werden viele Detektorkomponenten modifiziert. Auf die Änderungen am Spurkammersystem wird nun näher eingegangen:

- Die beiden zentralen Jet Kammern CJC1 und CJC2 (Central Jet Chamber) erhalten eine schnellere Ausleseelektronik. Dadurch wird die Berechnung der invarianten Masse der detektierten und rekonstruierten Teilchen schon nach 20μs möglich und steht so schon dem Level 2-Trigger zur Verfügung. Bisher konnte diese Information erst im Level 4-Trigger verwendet werden.

- Triggerentscheidungen des zentralen Trackingsystems werden momentan noch durch den z-Vertex Trigger getroffen, der eine Komponente des Level 1-Triggersystems ist. Dieser bestimmt den z-Vertex, also den Entstehungspunkt ionisierter Teilchen bei ep-Kollisionen. Das System bildet eine Triggerentscheidung aus den Informationen von drei Detektorkomponenten, der zentralen inneren Spurkammer (CIP), der zentralen äußeren Spurkammer (COZ) und der Vorwärts-Spurkammer (FTD). Von diesen drei Kammern wird nur noch die äußere Spurkammer in unveränderter Form übrigbleiben.

- Die bisherige CIP-Kammer wird durch eine neue CIP-Kammer ersetzt, die durch eine deutlich verbesserte Auflösung in z-Richtung der Luminositätserhöhung gerecht wird [CIP00]. Aufgrund der Nähe zum Strahlrohr ist der Austausch der CIP-Kammer zudem notwendig, da starke Alterungserscheinungen zu erkennen sind. Die neue Kammer wird in Abschnitt 2.5.1 beschrieben.

In Abbildung 2.8 ist das innere Spurkammersystem vor und nach dem Upgrade dargestellt.

- Der Central Silicon Tracker (CST) und der Backward Silicon Tracker (BST) werden im wesentlichen der geometrischen Veränderung des Strahlrohrs angepasst. Beim CST werden durch das neue ovale Strahlrohr die alten Detektormodule neu angeordnet. Die verbesserte Auflösung der neuen CIP soll es ermöglichen, die dreifache Ambiguität des vorhandenen CSTs aufzulösen, da eine genaue Ortsauflösung durch Spurerkennung in der CIP auf den CST interpoliert werden kann. Der BST wird mit neuen ϕ-Detektoren ausgerüstet.

- Die Vieldrahtporportionalkammern des Vorwärts Trackers (FTD) werden durch planare Driftkammern ersetzt, die nur Teilchen erkennen können, die unter großen Streuwinkeln gestreut werden. Triggerinformationen bei kleinem Streuwinkel θ werden von der neuen CIP Kammer erfasst.

- Die neue CIP wird in der Lage sein, auch in Vorwärtsrichtung Spuren zu erkennen, die zuvor von der nicht mehr vorhandenen Forward Multi Wire Proportional Chamber(FMWPC) gefunden wurden. Der z-Vertex Trigger wird weiterhin in Betrieb bleiben. Informationen, die die alte CIP bisher geliefert hat, werden entsprechend von der neuen CIP-Kammer geliefert. Ein System, das die Daten

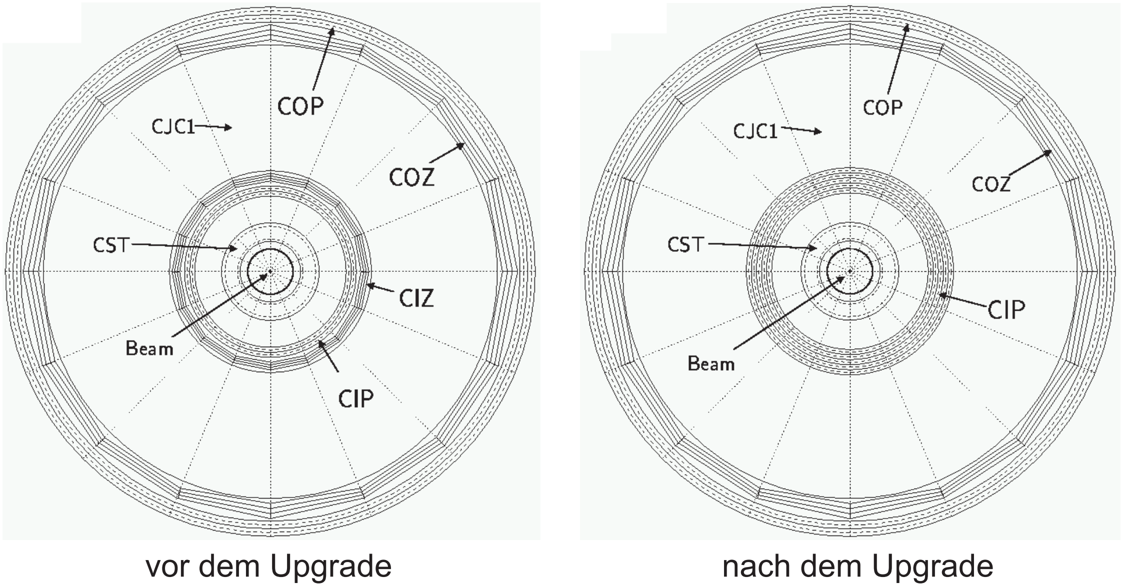


Abbildung 2.8: Innere Spurkammern vor und nach dem Upgrade. Abkürzungen: *CIZ Central Inner Z*-Kammer, *COZ Central Outer Z*-Kammer, *CIP Central Inner Proportional*-Kammer, *CST Central Silicon Tracker*, *CJC1 Central Jet Chamber* [Be00].

der neuen CIP-Kammer zur Verfügung stellt, ist im Rahmen dieser Diplomarbeit entstanden und wird in Kapitel ?? beschrieben.

- Zusätzlich zum z-Vertextrigger entsteht ein neues Triggersystem, das an Hand der Daten der neuen CIP Kammer ein z-Vertex Histogramm bildet, dass über einen größeren z-Bereich Spuren erkennen kann allerdings eine gröbere Granularität[3] hat. Das neue CIP-Triggersystem wird in Kapitel 3 beschrieben.

2.5 Das CIP-Kammer 2000-System

In Abbildung 2.9 ist das gesamte CIP 2000-System bestehend aus CIP-Kammer, Frontend-Elektronik[4] und Auswertungssystem dargestellt [Be00].

Links sind das Kammersystem und die CIPix-Ausleseelektronikmodule dargestellt. Die Ausleseelektronik befindet sich direkt an der $(-z)$ Seite der Kammer. Über ein optisches Übertragungssystem gelangen die Daten in das Triggercrate. Insgesamt gibt es vier Triggercrates[5]. Auf die in der Abbildung dargestellten Beschriftungen wird in Kapitel 3.3.1 eingegangen.

In den Triggercrates werden die Daten der Kammer ausgewertet. Die Daten werden zwischengespeichert, es wird eine Triggerentscheidung getroffen und den nachfolgenden Systemen an zwei Stellen zur Verfügung gestellt: Es gibt eine CPU, die gespeicherte

[3]Die Granularität ist ein Maß für die Auflösung pro Fläche

[4]Elektronik direkt an der Kammer

[5]ein Crate ist ein Überrahmen, in den Steckkarten mit elektronischen Schaltungen eingesteckt werden können

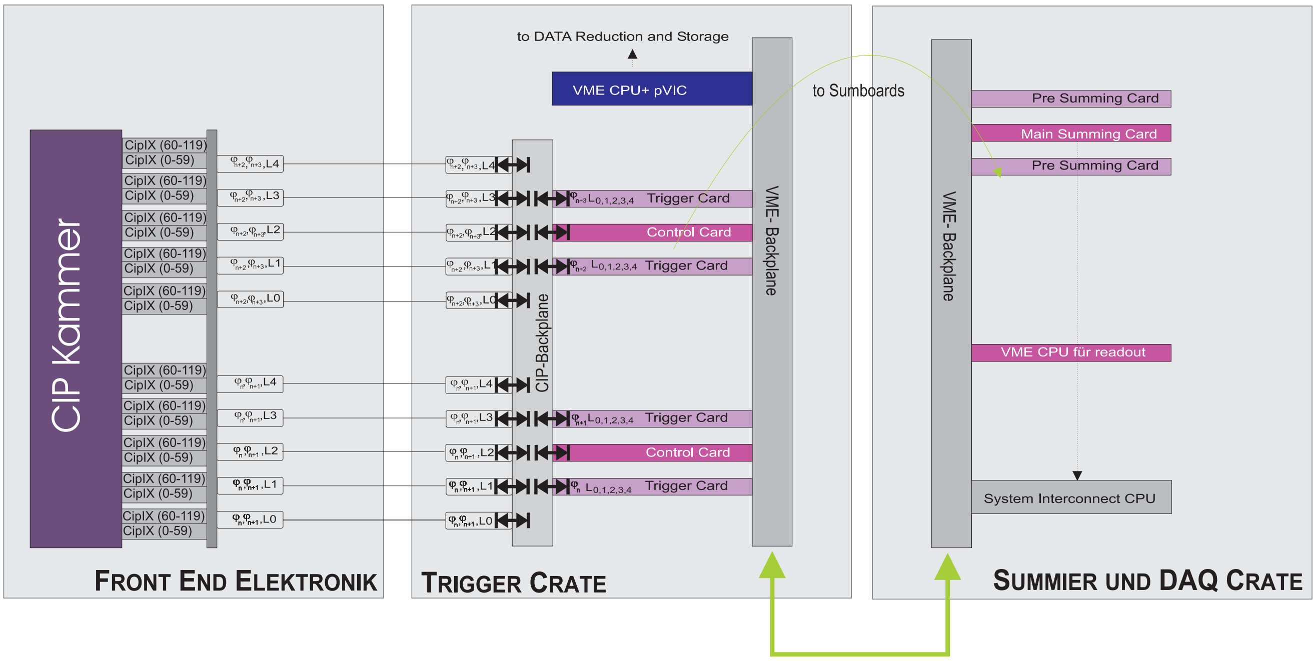


Abbildung 2.9: Das CIP-Kammer Gesamtsystem (siehe Text).

Daten ausließt (dunkler Block in der mittleren Einheit), wenn auf ein Ereignis getriggert wurde. Zudem gelangen die Daten der Triggerentscheidung aus dem Summierer und DAQ[6]-Crate in die zentralen Trigger Kontrolle, wo die Daten aller Subtrigger zusammenlaufen (vgl. Abschnitt 4.3).

2.5.1 Die Kammer

Die neue CIP-Kammer ist eine Vieldrahtproportionalkammer. Ein Teilchenstrahl ionisiert Gas in der Nähe eines Pads, auf dem dann eine Ladung induziert wird. Die Ladung wird durch eine Auslese Elektronik ausgewertet. Im Gegensatz zur alten hat die neue CIP-Kammer nicht nur zwei sondern fünf übereinanderliegende Ebenen mit jeweils 120 Pads in z-Richtung, die Pads liegen nicht wie bei der alten Kammer exakt übereinander, sondern sind versetzt zueinander angeordnet (siehe Abschnitt 3.2.2). Damit hat sich die Anzahl der Pads pro Ebene verdoppelt. Durch fünf übereinanderliegende Ebenen werden vor allem flache Spuren besser erkannt, so das die neue CIP-Kammer Untergrundereignisse bei denen seitlich in die Kammer einfallende Spuren auftreten, besser trennen kann.

Die neue Kammer hat zudem 16 statt der früher verwendeten 8 ϕ-Sektoren. Insgesamt müssen etwa 9600 Pad-Kanäle ausgewertet werden (früher waren es nur ca. 1000 Kanäle).

Die Kammer muss im Hinblick auf das Upgrade folgende Spezifikationen erfüllen:

- bessere Akzeptanz in Vorwärtsrichtung um das Wegfallen der PMWPC zu kompensieren

[6]**D**aten **AQ**uisation

- Auflösung der 3-fachen Ambiguität des CST in Rückwärtsrichtung
- genaue Vertexinformation über großen z-Bereich
- Trennung von Untergrund zu interessanten Ereignissen

In Abbildung 2.10 ist eine simulierte z-Vertexverteilung dargestellt, die das Verhältnis zwischen Untergrund und interessanten Ereignissen wiedergibt. Die Verteilung wurde aus Daten rekonstruiert, die mit der alten CIP-Kammer gemacht und auf die neue CIP-Kammer interpoliert wurden. Anhand der Abbildung wird verdeutlicht, dass die neue CIP-Kammer zwischen ep-Streuprozessen und Untergrundereignisse unterscheiden kann.

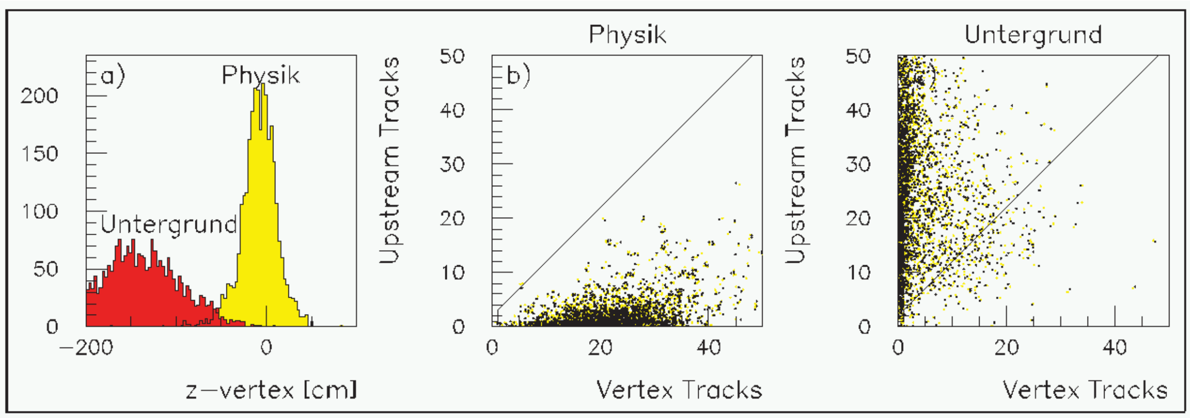


Abbildung 2.10: (a)Die z-Vertex Verteilung, rekonstruiert aus der 5-lagigen CIP, für Physik (hell) und Untergrund (dunkel), und die Verteilung von Upstream-Tracks ($z \leq 70cm$) gegen Vertex-Tracks($z =$ +-50cm) für Physik- (b) und Untergrundereignisse (c). Die Linie zeigt den Schnitt, der zur Untergrundunterdrückung verwendet wurde [H198].

Eine genaue Untersuchung des Verhaltens der neuen CIP-Kammer ist anhand alter Daten von der gegenwärtigen CIP-Kammer simuliert worden [Be98].

Ebene	Radius [mm]	Padlänge [mm]	Anzahl der Pads
Lage 1	157	18,250	120
Lage 2	166	19,323	112
Lage 3	175	20,531	106
Lage 4	184	21,900	100
Lage 5	193	23,464	96

Tabelle 2.2: Eckdaten der neuen CIP-Kammer

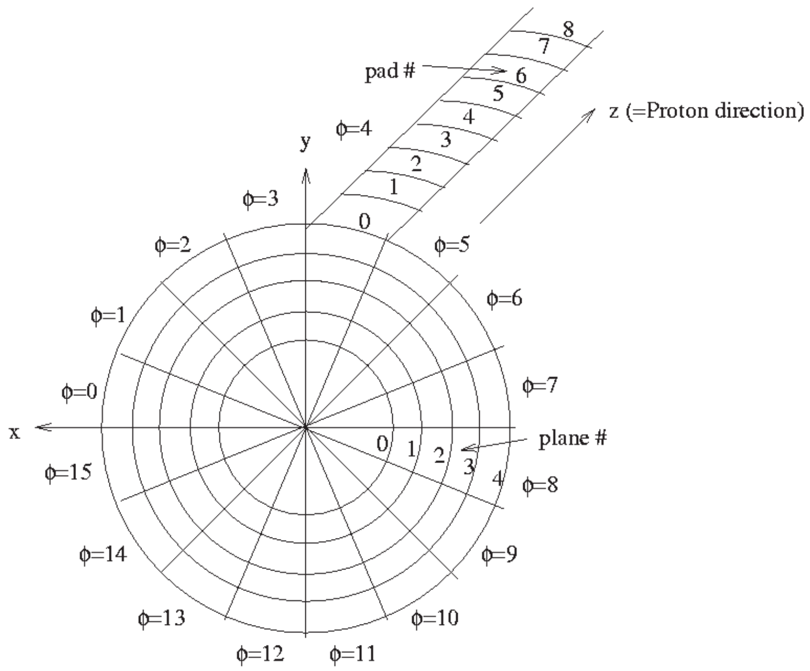


Abbildung 2.11: Anordnung der Ebenen und Sektoren in der neuen CIP-Kammer: Es gibt 16 ϕ-Sektoren mit jeweils 5 Ebenen und 120 Pads in z-Richtung [St98].

2.5.2 Die Kammerelektronik

Die Informationen der einzelnen Kammerpads müssen von einem sehr sensiblen Vorverstärker verstärkt werden. Diese Vorverstärker werden an die Padgeometrie der CIP-Kammer angepasst. Insbesondere das Verhältnis zwischen Eingangskapazität und Auslesegeschwindigkeit sowie das Signalverhalten wurden genau untersucht [Ko98]. Mit diesem Wissen wurde ein Auslesechip entwickelt, der auf Basis der Bestandteile des Helix-Auslesechip von HERA-B entwickelt worden ist [Hx98].

Neben den Vorverstärkern ist auf diesem Chip die analoge Signalverarbeitung und ein 1 bit ADC[7] integriert. Da insgesamt fast 10000 Kanäle ausgelesen werden müssen, ist die Verwendung von konventionellen Systemen nicht mehr möglich, so dass die geforderten Funktionen auf einem ASIC[8] untergebracht werden, der im ASIC Labor Heidelberg [**?**] entworfen und getestet wurde. Dieser Auslese-ASIC ist der CIPix [Ba98], [Lo98] und [St00].

Der CIPix Auslesechip befindet sich auf einer Auslesekarte, die direkt an die Kammer montiert wird. Auf dieser Karte sind sämtliche Systeme untergebracht, die die Signale zum Auswertungssystem transportieren. Auch hier zwingt die Menge der Signale die Entwickler zu neuen Wegen. Zum Transport der Daten wird ein optisches System aus optischem Sender, Lichtwellenleiterübertragungsstrecke und optischen Empfänger bereitgestellt.

[7] Analog–Digitalwandler, hier wird zum Wandeln der Daten ein Komparator verwendet
[8] ASIC = Application Specific Integratet Circuit

Die Datenübertragungsrate dieses Systems beträgt 0,8 Gigabyte/s. Das System wird am Paul Scherer Institut (PSI) und der ETH Zürich gebaut und getestet [Lu99]. Über die Übertragungsstrecke gelangen die Daten in das Triggercrate. Dort werden sie vom Triggersystem ausgewertet.

2.5.3 Das Triggersystem

Das CIP-Triggersystem bestimmt vom z-Vertex ausgehende Spuren über einen Bereich von 2,2 m. Anhand der Spurverteilung wird eine Triggerentscheidung gefunden. Das Triggersystem erstreckt sich über das Triggercrate und die Summiereinheiten im Summier- und DAQ Crate (mittlerer und rechter Block in Abbildung 2.9).

Dieses System kann Ereignisse, die im Zentrum der Kammer gefunden werden, von solchen unterscheiden, die in Randbereichen Spuren erzeugen. Dadurch wird eine genaue Unterscheidung zwischen Untergrundstrahlung und physikalisch interessanten Ereignissen möglich.

Das CIP-Triggersystem, Tests und Realisation werden in späteren Kapiteln ausführlich beschrieben.

2.5.4 DAQ und Systemintegration

Nachdem das Triggersystem die Triggerentscheidung an die CTC weitergegeben hat, werden dort die Informationen aller Triggersysteme ausgewertet. Es wird eine zentrale Triggerentscheidung gebildet.

Hat die zentrale Triggerkontrolle ein Ereignis bestätigt, untersuchen die L2-Systeme das Event weiter. Gleichzeitig werden die Daten der CIP-Kammer des entsprechenden Ereignisses an das übergeordnete Triggersystem übertragen. Ein eigens dazu entwickeltes Speichersystem wird in Kapitel 4.3 erläutert.

Kapitel 3

Das neue z-Vertex-Triggersystem

Nach einer Beschreibung des CIP-Kammer-Systems im zweiten Kapitel befasst sich dieses Kapitel genauer mit dem Triggersystem dieser Kammer. Das Triggersystem wertet die Padinformation der CIP-Kammer aus und stellt sie anderen Systemen zur Verfügung. Es liefert nach 1,8 μs eine Triggerentscheidung an die zentrale Triggerkontrolleinheit und gibt zwischengespeicherte Padinformation zur Auswertung weiter.

3.1 Anforderungen

In Abbildung 3.1 sind die Ein- und Ausgangssignale des Triggersystems dargestellt.

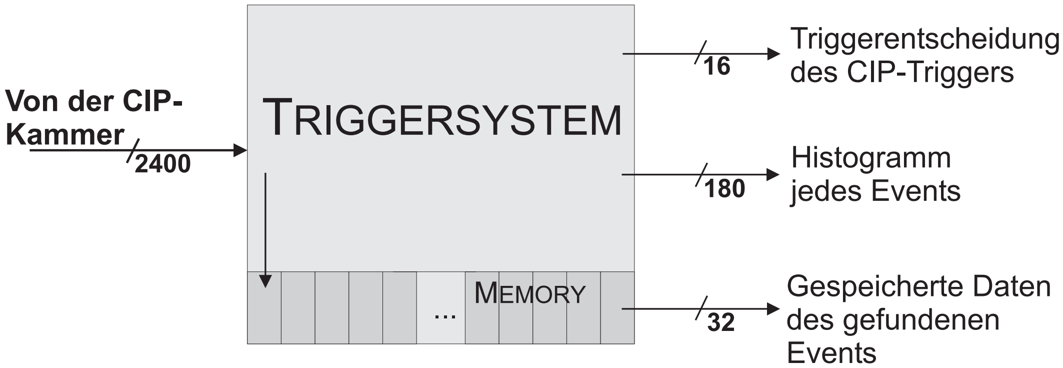


Abbildung 3.1: Das Triggersystem wertet insgesamt 2400 vierfach gemultiplexte[2] Daten von der CIP-Kammer aus, die sich wie folgt zusammensetzen: $2400 = (120_{Pads} \cdot 5_{Ebenen} \cdot 16_{Sektoren})/4_{multiplext}$. Das System soll aus diesen für jedes BC eine Triggerentscheidung liefern, die in einem 16 Bit Wort verschlüsselt werden. Zudem wird ein Histogramm ausgegeben, das eine Verteilung der z-Vertices der Spuren eines BCs darstellt. Für gefundene Events können über einen 32 Bit breiten Datenbus die gespeicherten Daten ausgelesen werden.

Ein Triggeralgorithmus muss zur Bestimmung einer Triggerentscheidung abgearbeitet werden. Für diese Algorithmus haben sich folgende Anforderungen im Verlauf der Entwicklungsarbeit als wichtig herausgestellt:

- Da es sich um einen Level1 Trigger handelt, muss alle 96 ns eine Triggerentscheidung geliefert werden. Die Entscheidung selbst kann allerdings nicht in 96 ns getroffen werden, so dass die Daten eines Events in einem Zwischenspeicher abgelegt werden müssen. Mit jedem BC kommen neue Daten, also darf jeder Speicher nur für die Dauer eines BCs belegt werden. Die Daten bewegen sich von Speicher zu Speicher. Dazwischen werden sie entsprechend dem Algorithmus verändert. Der Triggeralgorithmus, der ein Ergebnis nicht in einer Rechenstufe bestimmen kann, muss somit mit Zwischenspeichern arbeiten. Solche Zwischenspeicher werden Register genannt. Ein Konzept, das über mehrere Rechenschritte ein Ergebnis bestimmt, nennt man Pipelinekonzept.

- Das Triggersystem muss eine Triggerentscheidung spätestens 2,3 μs nach der Kollision an die CTC weitergeben. Diese Zeit ergibt sich aus Laufzeiten verschiedener Systeme.

- Das Triggersystem muss mit einem Algorithmus programmiert werden, der alle Pads in die Triggerentscheidung einbezieht und möglichst effizient arbeitet. Der Algorithmus muss tatsächlich ausgelöste Spuren erkennen und sie von Spurfragmenten oder zufällig ausgelösten Pads unterscheiden können. Die erkannten Spuren müssen zudem einem z-Vertex zugeordnet werden können.

- Das Triggersystem muss dem Datenformat der Kammer angepasst sein. Die Daten erreichen das System als vierfach gemultiplexte Datenpakete mit einer Frequenz von 40 MHz, so dass die Daten noch demultiplext werden müssen, bevor sie analysiert werden können.

- Da das gesamte H1-Triggersystem eine vorgegebene Zeit braucht, bis es eine Entscheidung über die Brauchbarkeit der Daten einer Kollision gefunden hat werden die Daten zurückliegender BCs gespeichert. Dadurch kann man die Daten eines Events später auslesen und auswerten, wenn es sich um ein brauchbares Event handelt. Das System muss die Daten der letzten 32 BCs zur Verfügung stellen. Einen Speicher, der eine feste Anzahl an Elementen speichert und dann zyklisch überschreibt, nennt man einen Ringspeicher.

- Bei eventuellen Defekten an der Kammer oder im Übertragungssystem muss das Triggersystem robust weiterarbeiten und auf Veränderungen flexibel eingehen können. Denkbare Fehler wären das Ausfallen einzelner Pad-Ebenen oder ϕ-Sektoren. Es wäre sinnvoll, in so einem Fall die Padinformation anders gewichten und im Triggeralgorithmus entsprechende Veränderungen vornehmen zu können.

- Das Testen des Triggersystems muss unabhängig von den anderen Systemkomponenten möglich sein. Das Triggersystem muss somit als "stand alone"-System funktionieren. Zur Systemoptimierung sollte es einen Debugmode geben, in dem das System schnell analysiert, verändert und verbessert werden kann.

- Das Triggersystem muss so gestaltet werden, dass dessen Steuerung und Kontrolle über ein externes System erfolgen kann, das sich nicht in dessen unmittelbarer Nähe befindet. Das bedeutet, dass es eine Möglichkeit geben muss, Kontroll- und

Steuersignale über ein Übertragungssystem auszugeben. Für solche Systeme gibt es vorgegebene Standards, die das Triggersystem erfüllen muss.

Im wesentlichen können die genannte Forderungen in zwei Kategorien unterteilt werden:

Zum einen in Forderungen, die eine logische Erfassung und Optimierung benötigen. Insbesondere die Gestaltung des Triggeralgorithmus und des Speichersystems sowie die Überlegung der Erfassung von Fehlfunktionen an Kammer und Auslesesystem sind dieser Kategorie zuzuordnen.

Zum anderen in Forderungen, die sich auf die technische Realisierung und auf Problemlösungen mechanischer und elektronischer Art beziehen. Hier sei die Notwendigkeit einer Elektronik genannt, die im Rahmen gegebener Zeitvorgaben schnell und zuverlässig arbeitet. Auf die Wahl von elektronischen Komponenten, Bussystemen oder Versorgungskomponenten wird in Kapitel 4 eingegangen.

3.2 Der Triggeralgorithmus

Der Triggeralgorithmus wertet die Padinformation der CIP-Kammer über mehrere Rechenschritte hinweg aus. Die Daten durchlaufen eine fest vorgegebene Sequenz von der Ankunft der Rohdaten am Triggeralgorithmus bis hin zur Triggerentscheidung am Ausgang. Sie befinden sich in einer mehrstufigen Pipeline. In jeder Pipelinestufe erfahren die Daten eine eindeutig zugeordnete Bearbeitung. Es sind immer so viele Datensätze des jeweiligen BCs in der Pipeline wie sie Stufen hat.

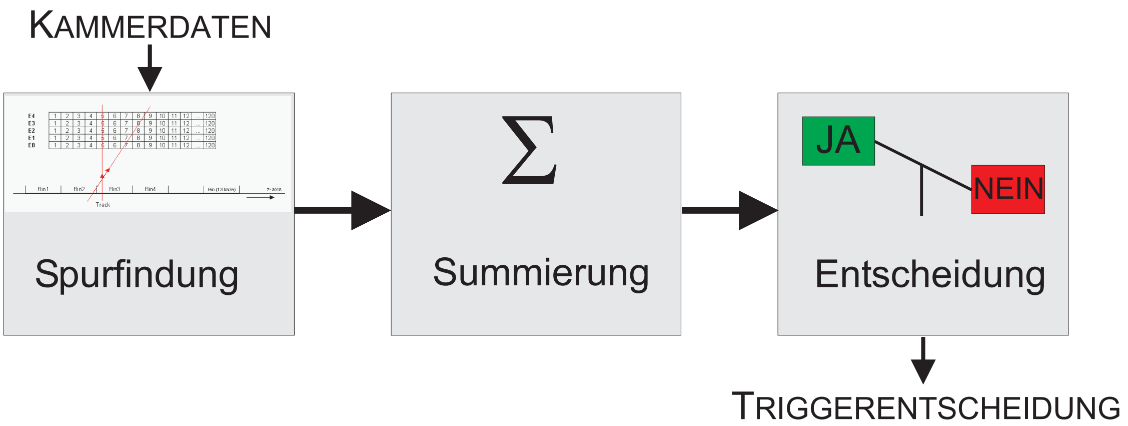


Abbildung 3.2: Schematische Darstellung der Hauptschritte des Triggeralgorithmus (siehe Text).

Der Triggeralgorithmus kann im wesentlichen in drei Stufen unterteilt werden (Abbildung 3.2).

- In der ersten Stufe werden die Rohdaten aus der Kammer bestimmten Spurbereichen nach eindeutig vorgegebenen Spurmustern zugeordnet.
- In der zweiten Stufe werden diese Spurmuster verschiedenen Bereichen der z-Achse zugeordnet. Alle Spuren eines Bereichs werden summiert.

- In der dritten Stufe wird das aus der Zuordnung der Daten in z-Bereiche entstandene Histogramm ausgewertet und eine Triggerentscheidung getroffen.

3.2.1 Spurfindung

Der Triggeralgorithmus ist so ausgelegt, dass er die 2400 vierfach gemultiplexten Signale der ca. 9600 Pads der Kammer zunächst in separate Informationen der 16 ϕ-Sektoren aufteilt. Damit werden die Daten der einzelnen ϕ-Sektoren unabhängig voneinander ausgewertet und erst für die Triggerentscheidung wieder zusammengeführt.

Jeder ϕ-Sektor hat etwa 600 Pads, die sich auf fünf Ebenen verteilen. Diese Verteilung ist nicht gleichmäßig, die unterste Ebene hat noch 120 Pads, die darüber liegenden haben dann jeweils weniger Pads (siehe Tabelle 2.2). Es wird später auf die besondere Geometrie der Kammer eingegangen. An dieser Stelle sei zunächst angenommen, dass jede Ebene 120 Pads hat, dann kommt man auf die angegebenen $120 \cdot 5 \cdot 16 = 9600$ Pads (symmetrische Geometrie). Unter dieser Annahme kann man sich die Anordnung der Pads wie ein Gitter vorstellen, das in die von der Strahlachse weglaufende Richtung 5 und in Strahlrichtung 120 Knotenpunkte hat.

Spurmuster: Spurmuster sind logisch miteinander verschaltete Pads. Jede Teilchenspur kann einem Spurmuster zugeordnet werden und wird dadurch erkannt. Die Anordnung der Spurmuster ist dicht, so dass jeder tatsächlichen Teilchenspur, das von einem z-Vertex durch den Detektor geht, ein Spurmuster zugeordnet werden kann.

zentrales Pad: Zur Bildung der Spurmuster nimmt man an, dass es einen Keim gibt, aus dem Spurmuster entstehen. Jedes Pad der zweiten Ebene (E2) ist ein Keim, der in einer lokalen Umgebung um das Pad Spurmuster bildet. Dieser Keim wird zentrales Pad genannt. Es kann kein Spurmuster gebildet werden, ohne dass dieses Pad enthalten ist.

lokale Umgebung: Die lokale Umgebung eines zentralen Pads ist eine Menge von Pads, die zum Bilden der Spurmuster nötig sind. Ein Spurmuster wird aus maximal 7 Pads gebildet. Ein Pad, welches nicht in der lokalen Umgebung ist, hat keinen Einfluss auf die Spurmuster.

In Abbildung 3.3 ist ein Abschnitt der Kammer schematisch dargestellt. Anhand der Abbildung soll verdeutlicht werden, wie Spuren gefunden und bestimmten Bereichen zugeordnet werden sollen. Generell gilt, dass eine Teilchenspur dann als solche erkannt wird, wenn:

a) das zentrale Pad,

b) je ein Pad aus der untersten (E0) und obersten Ebene (E4) und

c) ein oder zwei Pads der zweiten(E1) und vierten(E3) Ebene (die logisch ver*oder*t werden) ausgelöst werden.

Alle Pads müssen zur lokalen Umgebung gehören.

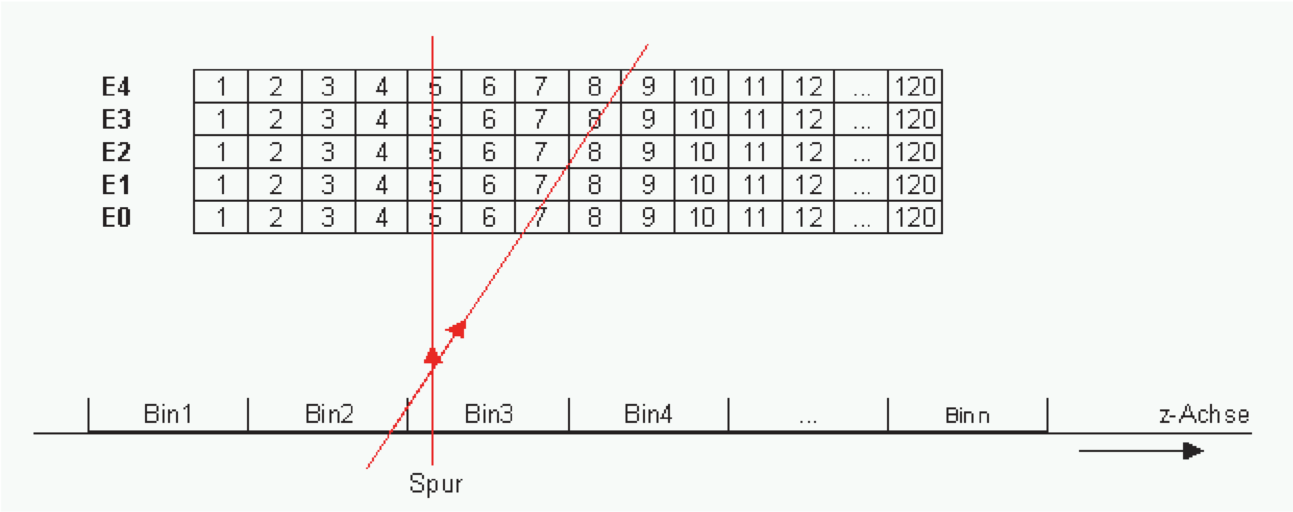


Abbildung 3.3: Schema des vereinfachten Triggeralgorithmus: Spuren aus eindeutigen Bins werden Spurmustern aus Pads zugeordnet.

Die Größe der lokalen Umgebung hängt von der Padgröße und der Anzahl n der Pads ab. Die Größe der Pads legt die maximale Auflösung in z-Richtung fest, die neben dem Abstand der CIP-Kammer zur Strahlachse und dem Abstand der Pad-Ebenen die einzige Größe ist, die man beim Bau variieren konnte[3].

Die bestmögliche Auflösung des Triggers ergibt sich, indem man ein Kreuz so durch das jeweils obere und untere Pad legt, dass die Strahlen durch die Ränder der Pads gehen. Der Abstand der zwei Kreuzungspunkte ist die maximal mögliche Auflösung (s. Abb. 3.4).

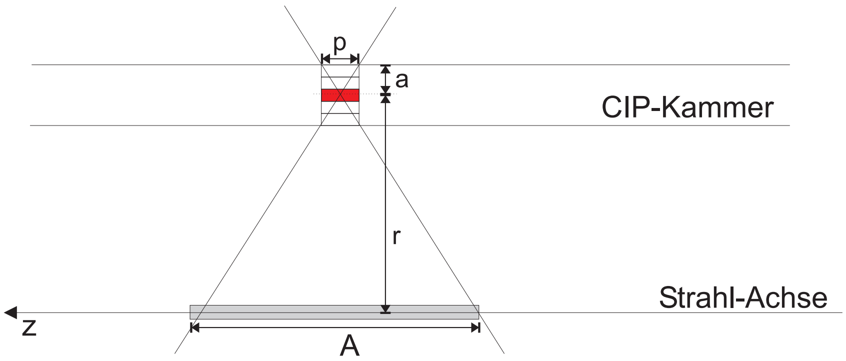


Abbildung 3.4: Darstellung der maximalen Auflösung in Abhängigkeit der Padgröße.

Durch den Strahlensatz ergibt sich:

$$A = \frac{r \cdot p}{a} = \frac{175\,mm \cdot 18,3\,mm}{22,5\,mm} = 144\,mm \tag{3.1}$$

[3]Die Padgröße in der alten CIP war zwar doppelt so groß, die Auflösung hängt hier aber sehr wesentlich vom Abstand zwischen CIP und COP ab und beträgt 44 mm

A ist die maximale Auflösung, a ist der Abstand zwischen oberstem und unterstem Pad, p ist die Padgröße und r ist der Abstand der unteren Pad-Ebene zur Strahlachse.

Im Fall der (später behandelten) projektiven Geometrie erhält man eine geringfügig schlechtere Auflösung von **164 mm** [Be00].

Binbereich: Der gesamte z-Bereich, den der Trigger auflösen kann, wird in einzelne Abschnitte unterteilt, die Binbereiche oder einfach nur Bins genannt werden. Die Größe eines solchen Abschnitts ist A , sie wurde in Gleichung 3.1 errechnet.

Bei einer Bingröße von 144 mm deckt man bei n =15 Binbereichen einen Gesamtbereich Z von

$$Z = n \cdot A = 15\, Bins \cdot 144\, mm = 2,16\, m \tag{3.2}$$

ab. Es hat sich herausgestellt, das mit 15 Binbereichen ein gutes Verhältnis zwischen Hardwareaufwand und maximal abgedeckten Triggerbereich gefunden wurde. Simulationen haben bestätigt, dass eine Aufteilung in mehr als 15 Binbereiche kein genaueres Triggerergebnis liefert [Be98]. Da die Spurfindung spiegelsymmetrisch zum zentralen Pad ist, ist die Anzahl der Bins immer ungerade.

Es besteht ein wichtiger Zusammenhang zwischen der Anzahl der Binbereiche und der Anzahl der Spurmuster einer lokalen Umgebung. Jede Spur der lokalen Umgebung zeigt in genau einen Binbereich. Da es 15 Binbereiche gibt, gibt es 15 Spurmuster pro lokaler Umgebung, daraus folgt, das jeweils 15 Pads in der oberen und unteren Padebene zu einer lokalen Umgebung gehören. Eine lokale Umgebung ist in Abbildung 3.5 abgebildet. Solange man eine symmetrische Kammer zur Spurfindung annimmt,

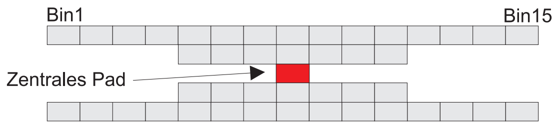


Abbildung 3.5: Die lokale Umgebung eines zentralen Pads

ist diese Zuordnung nicht möglich, wenn man aber die Anordnung der Pads zueinander verändert, wird diese Zuordnung möglich. Diese Idee wird nun diskutiert.

In Abbildung 3.6 sind verschiedene Padkombinationen dargestellt, die der Trigger als Spur erkennt. Die in der Abbildung dunkel markierten Padkombinationen führen nicht zu einer Spuridentifikation, die hell eingezeichneten Spuren hingegen schon.

Es sei bemerkt, dass jedes zentrales Pad eine eigene lokale Umgebung hat. Für 120 zentrale Pads wird eine große Anzahl an Logikelementen benötigt wird, die parallel verknüpft werden müssen.

3.2.2 Summierung

Projektive Geometrie: Wie bereits erwähnt gibt es eine Möglichkeit Pads so anzuordnen, dass sich Teilchenspuren von einem Kollisionspunkt auf jede Stelle der Kammer projizieren lassen. Dadurch wird die Programmierung des Triggeralgorithmus vereinfacht, da man für jedes zentrale Pad die gleiche lokale Umgebung verwenden kann. Diese Anordnung der Kammer-Pads nennt man *Projektive Geometrie.*

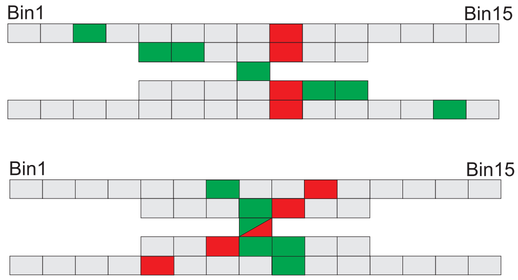


Abbildung 3.6: Spurerkennung im Triggeralgorithmus: Padkombinationen, die als Spuren identifiziert werden sind hell eingezeichnet, Untergrund ist dunkel markiert. Obwohl die Erkennung der dunkel markierten Felder nahe liegt, werden sie nicht erkannt.

Zur Verdeutlichung soll eine symmetrische Kammergeometrie mit der projektiven Kammergeometrie verglichen werden. Die Zuordnung der Spuren in Spurmuster macht den Hauptunterschied zwischen projektiver und symmetrischer Geometrie aus.

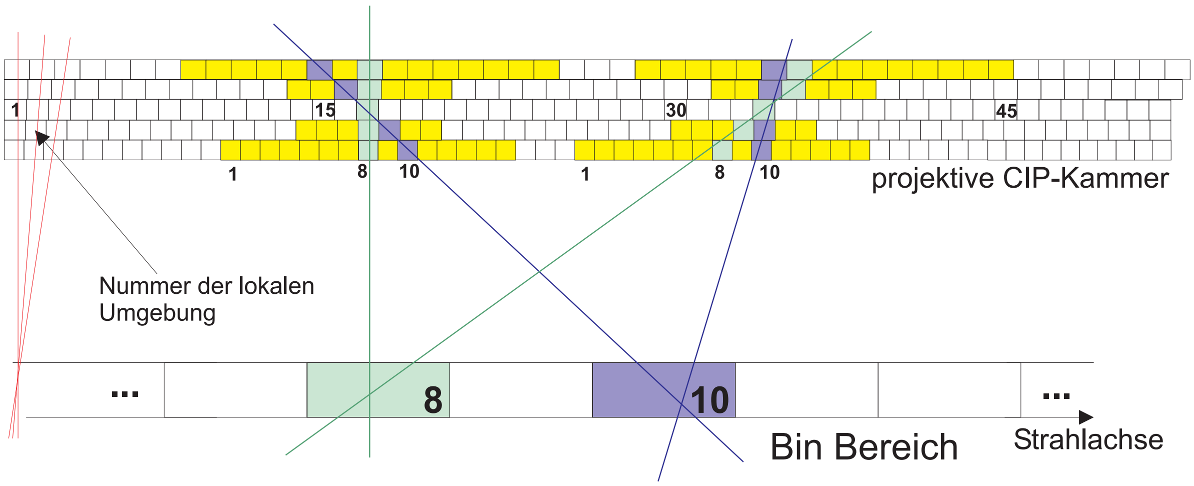


Abbildung 3.7: Projektive Geometrie im Triggeralgorithmus: Zwei lokale Umgebungen sind schraffiert eingezeichnet. Man erkennt, dass Spuren vom gleichen Vertex in verschiedenen lokalen Umgebungen gleiche Muster bilden.

Bei einer *symmetrischen Kammer* hätte jede Ebene 120 Pads, insgesamt gäbe es fünf Ebenen der Trigger Algorithmus müsste somit $120 \cdot 5$ Pads in den Triggerprozess einbeziehen.

Bei der *projektiven Kammer* sind die Pads verschiedener Ebenen nicht gleich groß, während Pads gleicher Ebenen die gleiche Größe haben. Die ersten Pads liegen übereinander, wobei jedes Pad einer höheren Ebene etwas größer ist als das der darunter liegenden. Diese Verschiebung nimmt mit jedem weiteren Pad zu, welches in z-Richtung angeordnet wird. Zeigt die erste Spur aus fünf übereinanderliegenden Spuren senkrecht nach unten, ist die zweite, die zur zweiten lokalen Umgebung gehört, geneigt. Jede weitere Spur ist ein wenig mehr geneigt, alle Spuren treffen sich aber in einem Punkt. (siehe Abbildung 3.7 links).

Die Kammer wurde so konstruiert, dass der Punkt, in dem sich die Spuren kreuzen, die Strahlenachse ist. Teilchenspuren von einem Vertex können also jedes der genannten Spurenmuster erzeugen. Allerdings ist der Vertex auf die Größe jeweils eines Binbereichs verschmiert. In der Abbildung sind Teilchenspuren eingezeichnet, die im 8. Binbereich entstehen und in der 17. als auch 34. lokalen Umgebung das selbe Spurmuster (und zwar das 8.) erzeugen.

Auch Teilchenspuren, die im 10. Binbereich entstehen, zeigen auf das 10. Spurmuster in den lokalen Umgebungen 17 und 34. Das gilt für Teilchenspuren aller Binbereiche. Insgesamt zeigen also maximal 106 Spuren aus dem selben Binbereich in 106 lokale Umgebungen von 106 zentralen Pads.

In einer Kammer mit projektiver Geometrie lösen Spuren, die im i-ten Binbereich entstehen, in jeder lokalen Umgebung das i-te Spurmuster aus.

In Abbildung 3.7 ist beim 17. zentralen Pad Spurmuster 8 aus Pads zusammengesetzt, die senkrecht übereinanderstehen. Spurmuster zentraler Pads mit "größerer Nummer" sind nach *rechts* geneigt, welche mit "kleinerer Nummer" nach links. Je nach dem, welchem zentralen Pad das senkrechte 8.Spurmuster zukommt, ändert sich der Bereich der Strahlachse, der untersucht werden soll, die Eigenschaften des Algorithmus ändern sich dadurch nicht.

Bei symmetrischer Geometrie gibt es keinen Zusammenhang zwischen Spurmuster und Binbereich. Spuren aus verschiedenen Kollisionspunkten erzeugen in verschiedenen lokalen Umgebungen verschiedene Muster. Hier muss die Zuordnung der Binbereiche entsprechend der Muster für jedes der 15 Spurmuster bestimmt werden. Für jede der 106 lokalen Umgebungen muss dieser Vorgang wiederholt werden. Insgesamt müssen also 106*15 = 1590 Zuordnungen abgewägt und programmiert werden.

Aus diesen Gründen wurde entschieden, eine Kammer mit projektiver Geometrie zu bauen.

Nach der Beschreibung der eindeutigen Zuordnung der Spuren in Binbereiche wird jetzt erklärt, wie die Anzahl der Spuren pro Binbereich bestimmt wird. Das geschieht mit der Hitlist.

Die Hitlist: Die *Hitlist* ist eine Konstruktion, die es ermöglichen soll, eine große Anzahl an gefundenen Spuren zu erfassen und schnell weiterzuverarbeiten.

Sie ist notwendig, da es nicht möglich ist, die Spuren in dem Moment zu zählen, in dem sie entstehen, wie bei einem sequentiell arbeitenden Programm. In diesem würden die Spuren, die gefunden werden, hintereinander abgefragt. Die Treffervariable eines Binbereichs kann dann nacheinander inkrementiert werden. Für den Triggeralgorithmus ist es aber wichtig, dass alle Suren gleichzeitig gefunden werden. Die Trefferinformation wird deshalb zunächst zwischengespeichert und anschließend addiert. Die *Hitlist* ist also ein Zwischenspeicher, in den parallel geschrieben werden kann. Da auf jeden Binbereich 106 Treffer, also erkannte Spuren aus den zentralen Pads fallen, und es 15 Binbereiche gibt, muss die *Hitlist* $106 \cdot 15$ also 1590 Speicherplätze haben.

Jeder Spur eines zentralen Pads ist ein Platz in der Hitlist zugeordnet. Wird eine Spur gefunden, wird an der Stelle der Hitlist, die genau dieser Spur zugeordnet ist, eine

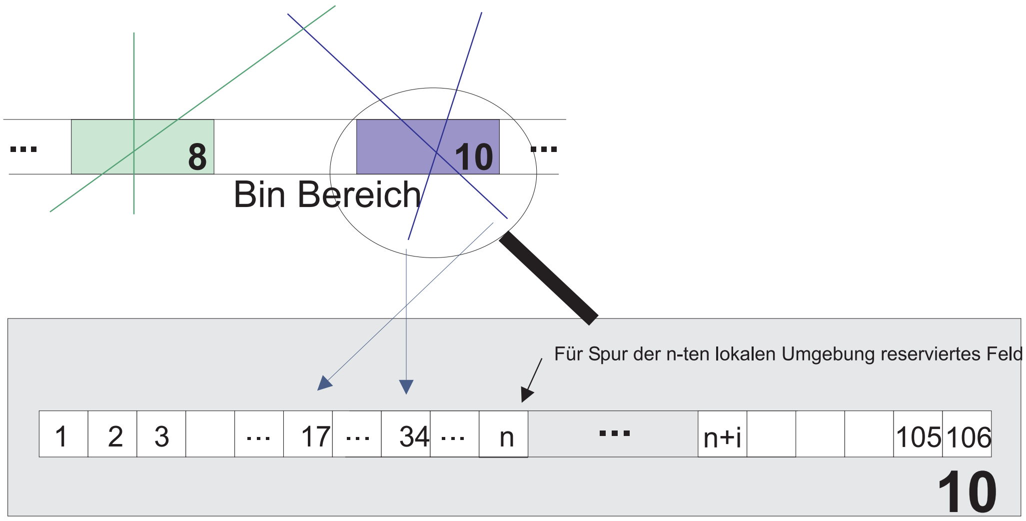


Abbildung 3.8: Verdeutlichung der Idee der Hitlist: Jeder Binbereich (hier der 10.) hat für jede lokale Umgebung einen eindeutigen Speicherbereich reserviert. Wird eine Spur erkannt, wird an den Speicherplatz der entsprechenden lokalen Umgebung eine "1" geschrieben.

"1" geschrieben, ansonsten bleibt der Wert "0".

In Abbildung 3.8 ist diese Idee anhand eines Beispiels erläutert. Dabei wurde die aus Abbildung 3.7 dargestellte projektive Pad-Anordnung verwendet. Es sind die Hitlistensegmente 17 und 34 des 10. Binbereichs für exakt diese Spuren reserviert. Da diese Spuren erkannt wurden, wird an die reservierte Stelle eine "1" geschrieben.

Analog verfährt man für die anderen 14 Binbereiche. Die *Hitlist* wird ausgewertet, indem man nun die Quersumme eines Binbereichs bildet. So errechnet sich die Anzahl der Spuren, die aus diesem Binbereich stammen. In Abbildung 3.9 ist die Hitlist als zweidimensionales Array dargestellt. In gleichen Spalten stehen Spuren gleicher lokaler Umgebungen und in gleichen Zeilen Spuren gleicher Vertices (Binbereiche).

Es sei erwähnt, dass für das Auswerten der Hitlist ein schneller Algorithmus notwendig ist: Da der Triggerprozess sehr schnell erfolgen muss, soll der Trigger-Algorithmus in möglichst wenig Takten ein Ergebnis finden. Dazu ist ein hohes Maß an Parallelisierung notwendig. Das Finden von Spuren und das Ablegen in der Hitlist ist in einem Zyklus möglich.

Auswerten der Hitlist: Die Hitlist wird mit einen schnellen Addierer ausgewertet, der die Quersumme der 106 Bit breiten Zahl bestimmt. Diese Auswertung ist in $log_2(n)$ Schritten möglich, wobei n die Anzahl der zu addierenden Bits (hier also 106) ist. Laufzeiten, die logarithmisch proportional zu n sind, erreicht man durch kaskadiertes Hintereinanderschalten von 2 Bit Addierern. In jeder Kaskade verringert sich die Bittiefe um einen Faktor 2.

Für den abgebildeten Addierer (Abb.3.10) gilt:

$$T = log_2(n) = log_2(8) = 3 \tag{3.3}$$

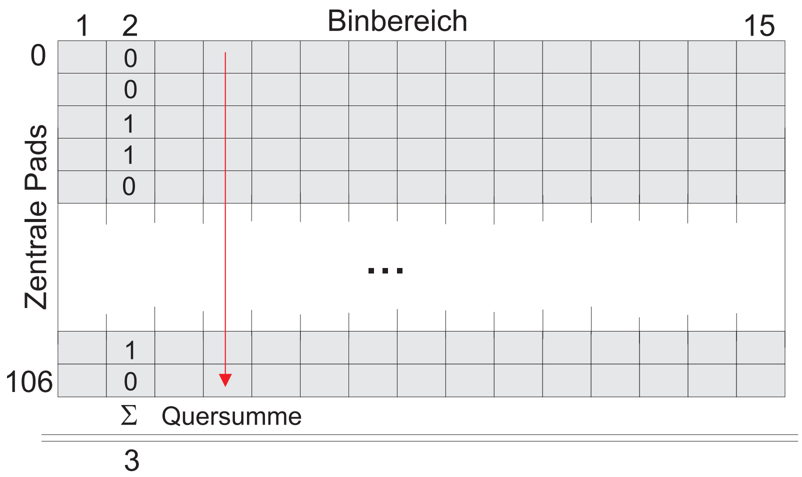


Abbildung 3.9: Die Bildung der Quersumme in der Hitlist: Es werden Bits gleicher Spalten addiert, Bits der gleichen Zeile gehören zu gleichen lokalen Umgebungen

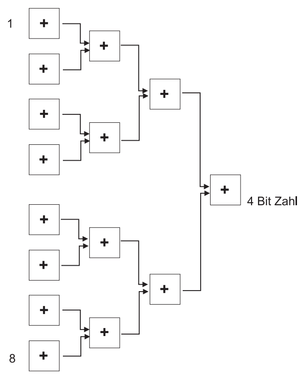


Abbildung 3.10: Kaskadischer Addieren mit logarithmischen Laufzeiten.

T ist die Anzahl Rechenschritte oder Taktzyklen, die notwendig sind, um die Quersumme der n Bit breiten Zahl zu bestimmen. Die eingesetzten Zahlen sind Abbildung 3.10 entnommen.

Da jeder Binbereich maximal 106 Treffer haben kann, kann die Hitlist in $log_2 106 \cong 6,72$, also maximal 7 Schritten ausgewertet werden. Eine noch schnellere Implementierung erreicht man durch *Look Up Tables (LUTs)* , die einen direkten funktionalen Zusammenhang zwischen Eingangsinformation und zugeordnetem Wert herstellen.

Allerdings werden hier auch 2^7 Speicherzellen benötigt, so dass der Aufwand nicht in einem sinnvollen Verhältnis zum Ergebnis steht. Eine gute Alternative wäre eine Kombination aus LUT und Addierer. Würde man LUTs verwenden, könnte der Addierer ein Ergebnis in zwei Takten liefern, für die geforderte Performance ist das nicht notwendig, so dass keine LUTs verwendet werden[4]. Nach der Auswertung der Hitlist

[4]Es hat sich herausgestellt, das die Verwendung von LUTs doch sinnvoll gewesen wäre, da Hersteller von digitalen Schaltungen von der Verwendung von LUTs ausgehen und deswegen die Technologie

liegt die Anzahl der Treffer pro Binbereich als 7 Bit Dezimalzahl vor, insgesamt 15 mal. Man bezeichnet diese Verteilung der Treffer in z–Richtung auch als z–Histogramm.

Im nächsten Modul wird dieses Histogramm vierfach gemultiplext und gelangt dann in den globalen Summierer, der die Information von allen 16 ϕ-Sektoren auswertet.

3.2.3 Triggerentscheidung

Für die Triggerentscheidung werden die einzelnen Sektor-Histogramme zu einem Gesamthistogramm zusammengefasst und ausgewertet. Man erhält dieses Gesamthistogramm wiederum durch kaskadische Addition (siehe Abb. 3.10).

Dieses Histogramm setzt sich ebenfalls aus 15 Binbereichen zusammen. In jeden Binbereich können theoretisch maximal $16_{Sektoren} \cdot 120 = 1920$ Spuren fallen. Das ist aber sehr unwahrscheinlich. Simulationen haben ergeben, dass bei einem "Kollisions-Ereignis" maximal 10% der Pads ein Signal liefern.

In Abbildung 3.11 ist das z-Histogramm dargestellt. Hier wurde von einer wesentlich höheren Teilchendichte ausgegangen, um den Mechanismus der Triggerentscheidung zu verdeutlichen. Durch Auswerten des z-Histogramms kann man eine einfache

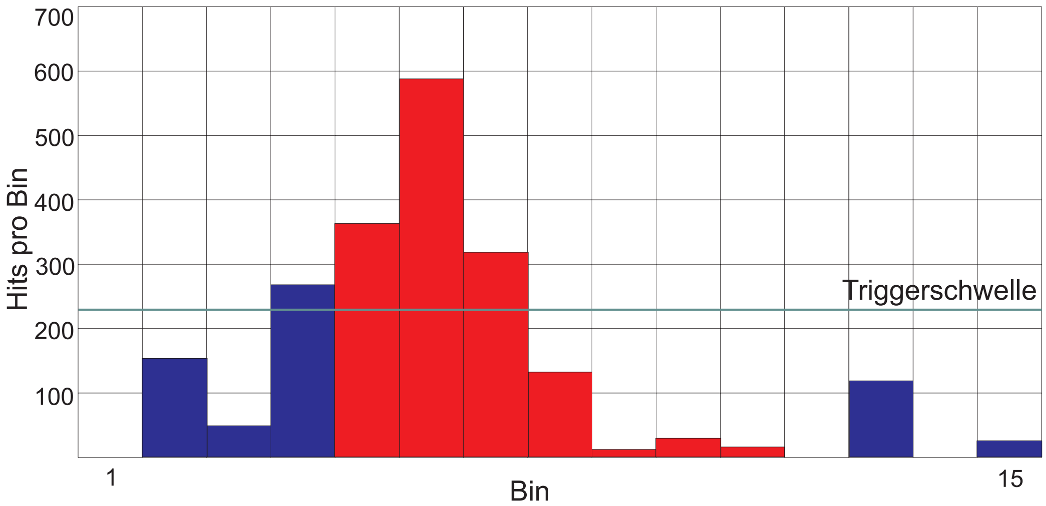


Abbildung 3.11: Histogramm der Trefferverteilung: Überschreitet in einem Binbereich die Anzahl der Hits die Triggerschwelle, wird dieser Bereich selektiert. In der Abbildung sind drei Bins im zentralen und einer im Randbereich über der Schwelle, dieses Event wird ausgewählt.

Triggerentscheidung treffen: Überschreitet in einem Binbereich die Anzahl der Hits eine Schwelle, wird dieser Bereich selektiert. Die Schwelle ist in Abb. 3.11 als helle Linie eingezeichnet. Es gibt einen Bereich in der Mitte, der eine frei bestimmbare Anzahl an Binbereichen zusammenfasst, und zwei Randbereiche. Sind im zentralen Bereich (hell) mehr Bins markiert als in den Randbereichen (dunkel), wird das Ereignis ausgewählt, ansonsten wird es verworfen. Für das Auswerten des Histogramms ist erneut eine

optimiert haben. Leider war es aus zeitlichen Gründen nicht mehr möglich, in diese Richtung Untersuchungen durchzuführen.

Summation notwendig. Sowohl für den mittleren Bereich als auch für den Randbereich werden die Binbereiche, bei denen Bin-Informationen über der Schwelle liegen, addiert. Man erhält zwei Zahlen, die nun miteinander verglichen werden können. Ist die Zahl der Bins im mittleren Bereich größer, wird dem zentralen Triggersystem mitgeteilt, dass ein Ereignis weiter untersucht werden soll, ansonsten wird das Ereignis verworfen.

3.3 Schematische Darstellung des Triggersystems

Um einen schnellen Überblick über das komplette Triggersystem zu bekommen, wird es schematisch dargestellt. Es gibt zwei wichtige Darstellungen:

- In der **funktionalen Darstellung** ist jede Komponente als Block dargestellt. Es wird besonderer Wert auf Verbindungen und Zusammenspiel gelegt.
- In der **zeitlichen Darstellung** ist die Laufzeit jeder Komponente dargestellt. Es wird besonderer Wert auf das Laufzeitverhalten gelegt.

3.3.1 Funktionale Darstellung

In Abbildung 3.12 ist das Triggercrate dargestellt. Im Triggercrate befinden sich der

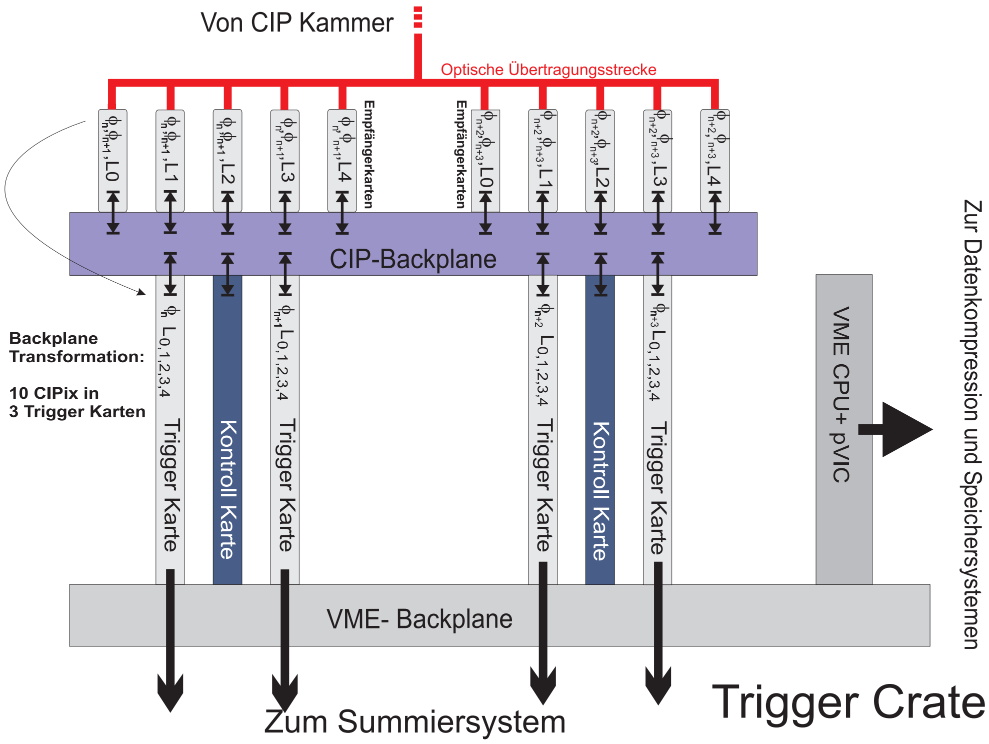


Abbildung 3.12: Funktionale Darstellung des Gesamtsystems; alle dargestellten Komponenten befinden sich in einem Triggercrate, für das gesamte System werden vier solcher Crates benötigt.

Hauptteil des Triggersystems.

Jedes Triggercrate enthält vier Triggerkarten. Eine Triggerkarte verarbeitet jeweils die Information eines ϕ-Sektors der Kammer. Auf einer Triggerkarte befinden sich Logikbausteine, in die der vollständige Triggeralgorithmus für einen ϕ-Sektor programmiert wird. Auf die verwendeten Logikbausteine wird in Kapitel 4 eingegangen. Für das komplette System werden 16 Triggerkarten benötigt, die sich auf vier Triggercrates verteilen. Die Kammerdaten gelangen über ein schnelles Lichtwellenleiter-Übertragungssystem in die Empfängerkarten, wo sie gewandelt und über die CIP-Backplane an die Trigger- und Kontrollkarten verteilt werden. Jede Empfängerkarte bearbeitet die Daten von *zwei* ϕ-Sektoren *einer* Ebene (siehe Beschriftung auf den Empfängerkarten: ϕ_n, ϕ_{n+1}, L_0). Die Triggerkarte erhält die Daten *eines* ϕ-Sektors *aller* Ebenen ($\phi_n, L_{0,1,2,3,4}$). Über zwei Kontrollkarten werden die externen Signale vom H1-Kontrollsystem in das CIP-System eingekoppelt. Jede Kontrollkarte regelt zudem das Phasenverhältnis der Taktsignale der fünf Empfängerkarten. Die Aufgabe der VME-Backplane ist es, die gespeicherten Daten von den Triggerkarten zur Weiterverarbeitung in den Auslesecomputer zu transportieren. Zudem können Informationen zu einer Steuerkonsole übertragen werden, so dass eine externe Steuerung des Triggersystems möglich ist. Es wurde beschlossen, für das Triggersystem als Frontbus das bewährte VME-Bus Konzept zu verwenden, alternativ hätte ein PCI-Bus System verwendet werden können. Für dieses System hätten alle Buskontroller neu entwickelt und getestet werden müssen. Der Datendurchsatz des VME-Buses ist für dieses System ausreichend.

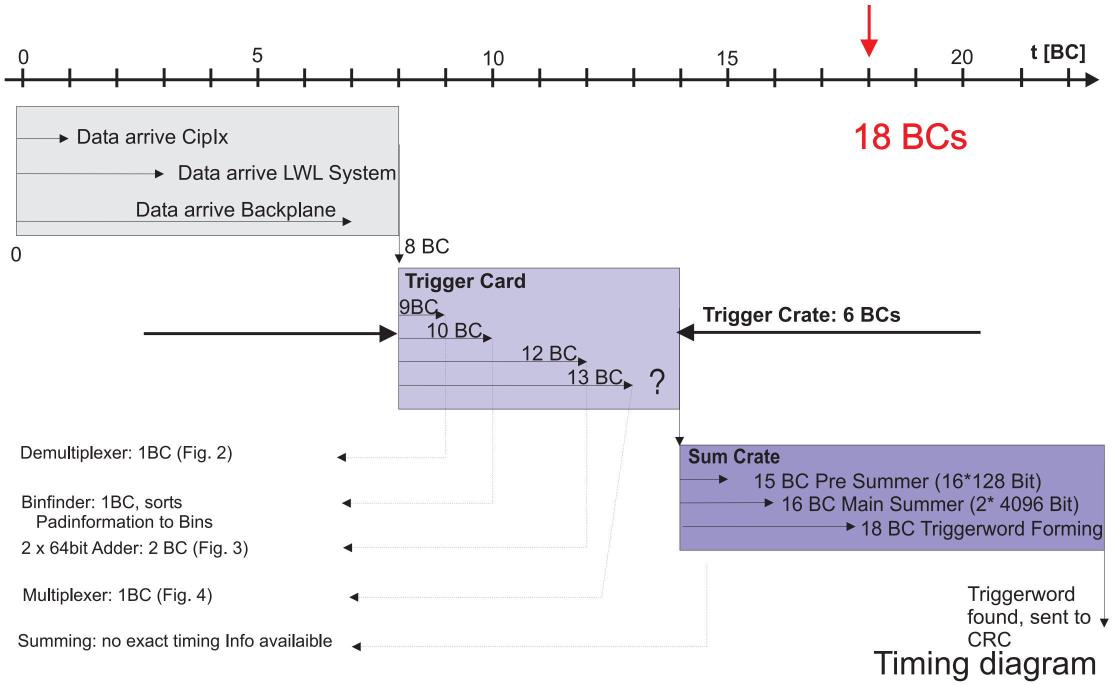


Abbildung 3.13: Laufzeiten für wichtige Systemkomponenten (siehe Text).

In der Beschreibung der endgültigen Systemkomponenten in Kapitel 5 wird die Funktion der CIP-Backplane und der Kontrollkarten sowie der endgültigen Triggerkarten beschrieben. Auf das Summiercrate und die darin befindlichen Karten wird nur sehr kurz in Kapitel 5.4 eingegangen [EW00].

3.3.2 Zeitablaufsplan

In Abbildung 3.13 sind die geforderten Laufzeiten für wichtige Systemkomponenten des Triggersystem dargestellt. Die Zeitangaben sind in Äquivalenten von BCs angegeben.

Die Kammerdaten erreichen nach einem BC den CIPix und werden dort ausgewertet und gemultiplext. Nach spätestens 3 BCs erreichen die Daten das optische Übertragungssystem und liegen nach spätestens 8 BCs an den Triggerkarten an, in denen sie vom Triggeralgorithmus weiterverarbeitet werden[5].

Die im Kasten *Trigger-Card* in Abbildung 3.13 dargestellten Zeiten wurden in Simulationen überprüft. In Abschnitt 4.4.5 wird gezeigt, dass die geforderten Laufzeiten eingehalten werden. Die genannten Zeiten für die Summierer wurden durch Überschlagsrechnungen bestimmt.

Jedes Level1 Trigger-Subsystem muss eine Triggerentscheidung innerhalb von 23 BCs an die CTC liefern. Aus der Abbildung ist ersichtlich, das das CIP-Triggersystem schon nach 18 BCs eine Triggerentscheidung liefert, aber auch diese Zahl ist eine obere Grenze, vermutlich wird die Entscheidung schon früher an der CTC anliegen. In Abbil-

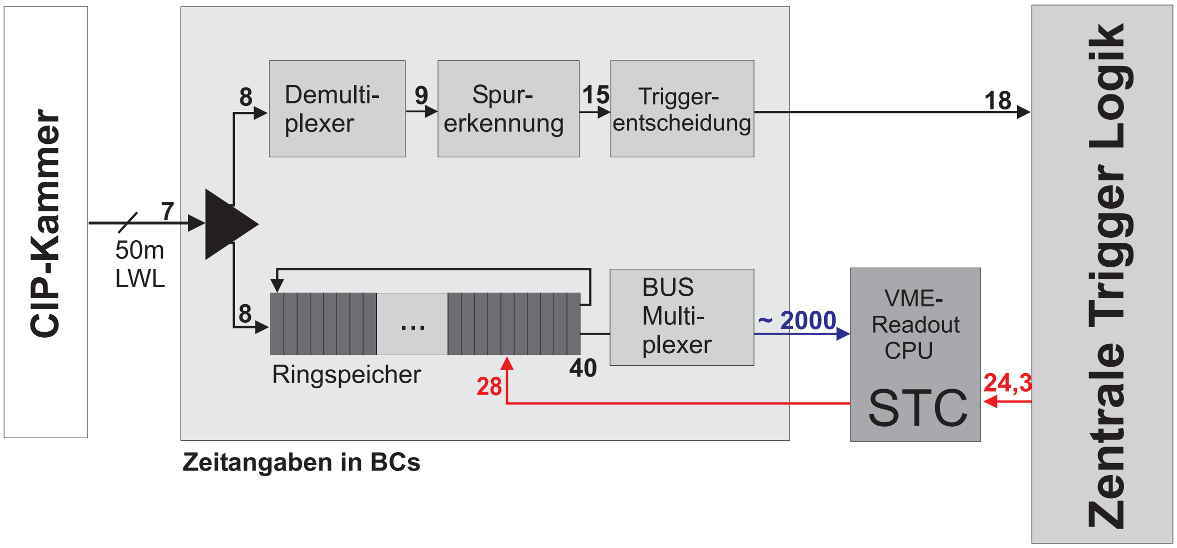


Abbildung 3.14: Blockzeitdiagramm des Triggersystems: nach 7 BCs^5 erreichen die Kammerdaten das Triggersystem, nach 8 BCs liegen sie verteilt an Demultiplexer und Ringspeicher an. Der Ringspeicher speichert die nächsten 32 BCs. Die Triggerlogik übergibt zum 18. BC die Triggerentscheidung an die CTC. Wird ein Ereignis gehalten, teilt die Triggerkontrolle dies dem Triggersystem zum 28. BC mit, die Datennahme wird gestoppt und die BCs 28-32 werden ausgelesen. Diese Auslese ist nach ca. 2000 BCs abgeschlossen.

[5]Durch Messungen wurde herausgefunden, dass die Daten bereits nach maximal 5 BCs ($(420 \pm 30)\, ns$) und nicht erst nach 8 BCs am Triggersystem ankommen.

dung 3.14 ist das *Blockzeitdiagramm* des Triggersystems dargestellt. Auf den Speicher im unteren Teil der Abbildung wird in Abschnitt 4.3 eingegangen.

Kapitel 4

Die Entwicklung des Triggersystems

In der vorangegangenen Beschreibung wurde die Funktion des Triggersystems verdeutlicht. Nun wird die Realisierung mit **F**ield **P**rogrammable **G**ate **A**rrays (FPGAs) beschrieben.

Zunächst wird ein Überblick über applikationsspezifische Hardware und deren Programmierung gegeben. Anschließend erfolgt die Beschreibung und Simulation der entwickelten Komponenten des Triggersystems.

Die in diesem Kapitel beschriebenen Schaltungen sind unter [Ur00] abgelegt.

4.1 Hardware und Software

Applikationsspezifische ICs sind Schaltungen, die für eine vom Benutzer gewünschte Anwendung entwickelt werden. Die Entwurf dieser Schaltungen erfolgt mit Entwicklungswerkzeugen, die durch die Verwendung von Rechnern große Bedeutung erlangt haben.

Durch diese Werkzeuge ist für Hardwareprojekte eine deutliche Trennung zwischen Entwurf und Entwicklung zum einen und der Verwirklichung der Schaltung im IC zum anderen entstanden. Die Entwicklung der Schaltung erfolgt in zwei Schritten. Zunächst erstellt man ein Modell, das jedes mögliche Verhalten der Schaltung beschreibt. Es ist unabhängig von der verwendeten Technologie zur Realisierung der Schaltung. Der Computer erstellt aus diesem Modell sogenannte Netzlisten[1]. Sie werden anhand von Simulationen überprüft und abgeändert, bis sich das gewünschte Verhalten zeigt. Im zweiten Schritt erfolgt das Realisieren der Schaltung in eine spezielle Hardware[2]. Jeder Anbieter muss ein spezielles Synthesewerkzeug zur Verfügung stellen, das auf die speziellen Eigenschaften seiner Hardware abgestimmt ist. Dieser Entwicklungsprozess hat die Entwicklung logischer Schaltungen auf Gatterebene abgelöst, bei der noch jeder Transistor in riesigen Schaltplänen vom Hardwareentwickler platziert und in das System eingebunden werden musste. Gleichzeitig zur Weiterentwicklung der oben genannten Entwicklungswerkzeuge steigt die Leistung solcher ICs, da immer mehr Transistoren auf gegebener Chipfläche untergebracht werden können.

[1]Netzlisten sind Dateien im ASCII-Format, die logische Verbindungen einzelner Gatter in einer integrierten Schaltung beinhalten.

[2]Den Schritt vom fertigen Entwurf zu einem lauffähigen IC nennt man synthetisieren.

In Abbildung 4.1 sind Entwicklung der Anzahl der Transistoren pro Chip sowie Anzahl logischer Gatteräquivalente in ICs gegen die Zeit aufgetragen. Die Zahl der Gatteräquivalente ist ein Maß für die Komplexität eines ICs und gibt an, wie viele Gatter man braucht, um eine solche Schaltung in herkömmlicher Technologie zu bauen. In einen üblichen TTL Baustein der 74er Serie, zum Beispiel dem Baustein 7400, sind vier logische NAND-Gatter untergebracht.

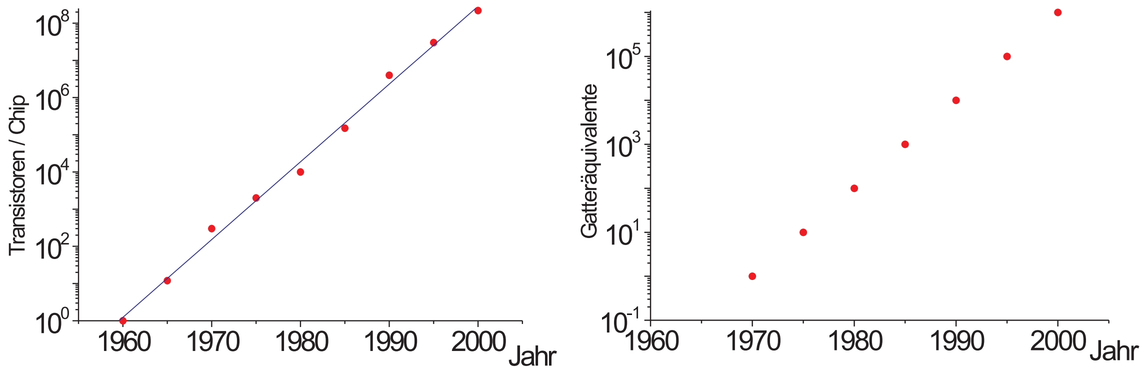


Abbildung 4.1: Entwicklung der Anzahl der Transistoren und der logischen Gatteräquivalente pro Chip nach dem empirischen *Gesetz vom Moore*.

Die Leistung und Komplexität von ICs wird annähernd durch das empirische Gesetzt von Moore beschrieben, welches besagt, dass sich die Leistung integrierter Schaltungen alle 18 Monate verdoppelt.

Man unterscheidet im wesentlichen drei Arten von anwendungsspezifischen Mikrochips:

- DSP: **D**igitale **S**ignal **P**rozessoren dienen der Signalverarbeitung. Das Design des DSPs ist vorgegeben, der Anwender kann sie programmieren und Signale entsprechend des Programms verarbeiten. DSP sind Prozessoren und können logische Schaltungen nur begrenzt abbilden.

- ASIC: **A**pplication **S**pecific **I**ntegrated **C**ircuits werden vom Anwender selber entwickelt, müssen aber in einer Chipfabrik produziert werden. Sie eignen sich vor allem, wenn eine Schaltung in hohen Stückzahlen produziert werden soll. Mit ASICs kann man fast jede beliebige digitale oder analoge Schaltung realisieren. Durch umfangreiche Entwicklungswerkzeuge ist es möglich, relativ schnell sehr komplexe Schaltungen zu entwerfen. ASICs sind allerdings sehr unflexibel bezüglich nachträglicher Designänderungen. In diesem Fall muss der Chip vollständig neu produziert werden (*neue Submission*). Auf ASICs wird nicht näher eingegangen [ASIC99].

- PLD: **P**rogrammable **L**ogic **D**evices sind fest vorgegebene integrierte Schaltungen, in die der Anwender logische Funktionen programmieren kann. Der so programmierte PLD kann eine herkömmliche Schaltung ersetzen. FPGAs sind die am weitesten entwickelten PLDs. Man kann fast jede digitale Schaltung in FPGAs abbilden. Aufgrund der sehr aufwendigen schaltungstechnischen Realisierung dieser war der Platz für digitale Schaltungen in FPGAs bisher allerdings sehr begrenzt. Das Problem ist durch weitere Miniaturisierung nahezu verschwunden. Angepasste Entwicklertools machen die Programmierung von FPGAs in hochsprachenähnlichen Programmiersprachen, sogenannten **H**ardware **D**escription **L**anguages (HDL = Hardware Beschreibungssprache), möglich. Auf HDLs wird weiter unten eingegangen (Abschnitt 4.1.2).

Die drei genannte Technologien standen für den Bau des Triggersystems zu Auswahl.

- Die Entwicklung eines ASICs wurde recht schnell ausgeschlossen, da nicht sehr viele Bausteine benötigt werden und durch die Realisierung in ASICs die Flexibilität bezüglich nachträglicher Änderung verloren geht.

- DSPs hingegen waren sehr lange die Favoriten bei der Auswahl der Hardware. Sie eignen sich gut bei Problemen, die in einem Algorithmus ausgedrückt werden können. Hingegen sind sie nicht geeignet, eine große Anzahl an Berechnungen parallel durchzuführen, so dass sehr viele Einheiten nötig wären, um den Triggeralgorithmus darin unterzubringen. DSPs sind stark bei Additions- und Multiplikations-Operationen. Im Triggeralgorithmus werden vor allem viele Bitoperationen durchgeführt, für die sich FPGAs besser eignen.

- Aber auch bei den FPGAs war es schwierig, einen so großen Algorithmus in möglichst wenig FPGAs unterzubringen. Dennoch wurde der *FPGA-Weg* weiterverfolgt.

Zunächst sollten FPGAs von Lattice verwendet werden. Abschätzungen haben ergeben, das pro ϕ-Sektor 10 FPGAs von Typ *Lattice ispL 6192* benötigt werden [La96]. Da aber zum Zeitpunkt dieser Überlegungen von *Altera* sehr hoch integrierte FPGAs vorgestellt wurden [Al99], wurde beschlossen, mit diesen FPGAs einen Versuch der Realisierung zu starten. Diese FPGAs schienen sehr gut zu dem Projekt zu passen, da sie neben einer großen Anzahl an Gattern und Anschlusspins (zwischen 204 und 858 I/O Pins) über frei adressierbaren Speicher verfügen, der zum Speichern der CIP-Kammerdaten verwendet werden kann.

4.1.1 FPGAs

FPGA können ohne große technische Ausrüstung programmiert werden. Ein Vorteil ist, dass man im Gegensatz zu ASICs weder eigene Entwicklungslabors noch Chipfabriken benötigt. *Field programmable* weist nicht auf die in FPGAs verwendeten Feldeffekttransistoren hin, sondern bedeutet, dass eine großes Areal an Gattern beliebig programmiert werden kann. Somit ist jeder Anwender in der Lage, eine integrierte Schaltung nach den eigenen Vorstellungen zu entwickeln.

Es ist kein Problem, FPGAs immer wieder umzuprogrammieren, z.B. wenn ein Fehler in der Programmierung gefunden oder eine zusätzliche Funktion gefordert wird.

Obwohl es eine Vielzahl von Herstellern gibt, ist der Aufbau der meisten FPGAs identisch. Jede FPGA setzt sich aus einer Vielzahl von konfigurierbaren logischen Einheiten (CLBs[3]) zusammen.

Eine logische Einheit enthält jeweils eine Eingangsstufe, die aus D-Flip-Flops besteht, eine Logikstufe, in der logische Funktionen in Gatter programmiert werden können, und Ausgangsregister, die das Ergebnis speichern [St98].

Das vom Anwender entwickelte Programm wird vom *Fitter* auf die vorhandenen CLBs verteilt. Die optimale Verschaltung der CLBs untereinander wird vom *Router* durchgeführt. *Router* und *Fitter* sind vom FPGA-Hersteller zur Verfügung gestellte Programme, die das erstellte Modell der Schaltung in der Hardware realisieren, sie sollen ein optimales Verhältnis zwischen Verwendung einzelnen CLBs und der Minimierung der Weglängen im FPGA finden. Für diese Optimierung sind sehr aufwendige und langwierige Rechnungen notwendig.

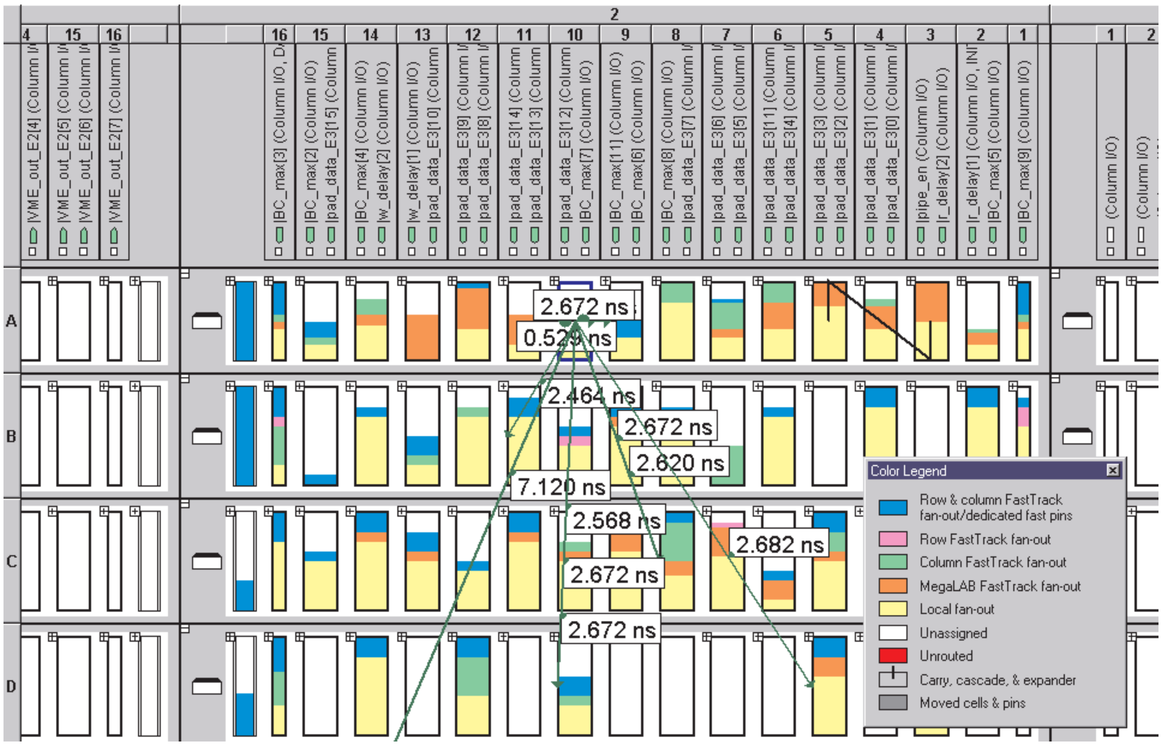


Abbildung 4.2: Interne MegaLAB Struktur in APEX 20K FPGAs: Die FPGA ist aus 104 MegaLABs zusammengesetzt, 26 in einer Spalte, 4 in einer Reihe. Hier ist ein Ausschnitt von 4 MegaLABs gezeigt, die sich jeweils aus 16 LABs zusammensetzten. LABs sind die kleineren Kästchen in der Abbildung. Die Verbindungen eines LABs ist jeweils mit Laufzeiten eingezeichnet [Al99].

In Abbildung 4.2 ist die Verknüpfung einiger CLBs in diesen FPGAs zu sehen. CLBs werden hier allerdings als LAB[4]s bezeichnet. Sie haben aber die gleiche Funktion wie

[3] **C**onfigurable **L**ogic **B**lock

[4] **L**ogig **A**rray **B**lock

CLBs. Jeder APEX 20K besteht aus MegaLABs, die jeweils aus 16 LABs und einem Speicherblock, dem sogenanntem ESB[5] zusammengesetzt sind. Die MegaLABs sind in Spalten und Reihen angeordnet. Der verwendete APEX 20K400 hat 104 MegaLABs, die in einem Gitter zu 4 Reihen und 26 Spalten angeordnet sind.

Wichtige Signale werden zwischen den MegaLABs und den Ein- und Ausgangspins über schnelle Mega LABs Verbindungen, sogenannte *Fast Track Interconnects,* verbunden. Sie nutzen spezielle Leitungen, die nicht in den *Routingprozess* einbezogen werden und ausschließlich für zeitkritische Verbindungen verwendet werden. Einzelne LABs werden über langsamere lokale Leitungen verbunden. In der Abbildung 4.2 sind die Verbindungen eines LABs dargestellt. Zu jeder Verbindung kann eine Laufzeit angegeben werden. Aufgrund der Menge der Leitungen ist eine Kontrolle nahezu ausgeschlossen und ist daher unüblich.

Im Anhang B.1 sind die Leistungseckdaten der APEX 20K FPGAs aufgeführt. Der verwendete Typ APEX 20K400 hat 10^6 Gatteräquivalente (vgl. auch Abb. 4.1).

4.1.2 Hardware Beschreibungssprachen

Wie bereits erwähnt werden FPGAs mit HDLs programmiert, höherer Programmiersprachen, erweitert um Möglichkeiten, parallel ablaufenden Prozesse elektronischer Schaltungen zu beschreiben. Hardwarenahe Datenstrukturen sind zum Beispiel für die Beschreibung von Registern oder Leitungen notwendig.

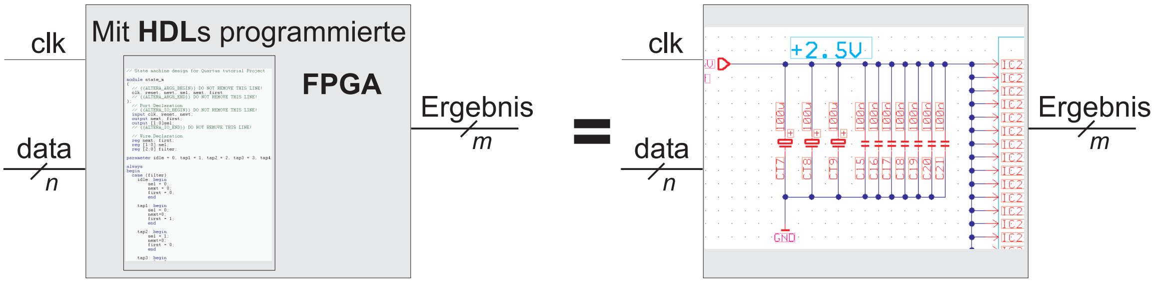


Abbildung 4.3: Eine mit einer HDL programmierte FPGA leistet das Gleiche wie eine festverdrahtete Schaltung.

Mit HDLs erstellt man ein Modell der Schaltung. Jeder mögliche Zustand, in den die Schaltung gehen kann, wird darin beschrieben.
Es gibt zwei kommerziell relevante HDLs:

- **VHDL** ist eine standarisierte HDL. Dadurch ist sichergestellt, das in VHDL programmierte Schaltungen plattformunabhängig funktionieren und das Verwenden von Simulations- und Synthese-Werkzeugen verschiedener Anbieter möglich ist. Die Abkürzung VHDL steht für VHSIC **H**ardware **D**escription **L**anguage, VHSIC bedeutet **V**ery **H**igh **S**peed **I**ntegrated **C**ircuits. Dem Namen nach ist VHDL also eine Hardwaresprache, die sich vor allem zur Beschreibung schneller und komplexer Schaltkreise eignet. Eine Darstellung der Programmiersprache ist

[5] **E**mbedded **S**ystem **B**lock

in [SE99] zu finden. Die Entwicklung von VHDL-Modellen ist in [Ha95] beschrieben.

- **Verilog** wurde noch nicht standarisiert. Jeder Anbieter von FPGA Entwicklungsumgebungen hat einen leicht veränderten Verilogdialekt. Verilog lehnt sich stark an die allerdings prozedurale Programmstruktur von Pascal an, hat aber zusätzliche Datentypen, die zur Hardwarebeschreibung notwendig sind. Einer dieser Datentypen ist das Register. Die Codezeile

$$reg\ [7:0]\ a; \tag{4.1}$$

  beschreibt ein 8 bit-Register, das taktsynchron ausgelesen werden kann.

  Der Aufbau eines Verilog Programms ist modular. In einem übergeordneten Modul werden alle Untermodule miteinander verknüpft. Jeder Programmcode befindet sich in diesen Einheiten. Der Aufbau eines Moduls ist in Abbildung 4.4 dargestellt.

```
module __module_name ( __input_name, __input_name,
      __output_name, __output_name,
      __inout_name, inout_name);
  // Port Declaration

  // Wire Declaration

  // Integer Declaration

  // Concurrent Assignment

  // Always Construct

endmodule

```

  Abbildung 4.4: Prinzipieller Aufbau eines Verilog Moduls: In der *port declaration* werden die Ein- und Ausgänge des Moduls festgelegt. In der *wire declaration* werden die Ausgänge und internen Register definiert. Hilfsvariablen werden in der *integer declaration* beschrieben, Wertzuweisungen am Programmstart werden im *concurrent Assignment*-Teil beschrieben. Der Programmhauptteil kommt in das *always Konstrukt* , wo er immer mit einem Stimulus, wie der Clock, ausgeführt wird [Qu99].

  Jedes Verilog-Programm wird aus oben erwähnten Modulen zusammengesetzt und ist damit vollständig beschrieben [Gol95].

Verilog ist früher entstanden und ist daher in den USA weit verbreitet. In Europa dominiert VHDL. Für die Implementierung des beschriebenen Triggeralgorithmus wurde Verilog verwendet, da der Hersteller der FPGA-Entwicklungsumgebung die Verwendung von VHDL noch nicht unterstützt hat.

Ist ein Modell der gewünschten Schaltung entstanden, kann man das Verhalten der Schaltung simulieren.

Da nach obigen Beschreibungen die Entwicklung der Hardware deutlich vom Bau der entsprechenden Systeme getrennt wird, soll auch hier eine Trennung zwischen Entwurf mit Simulation im Rechner und dem Bau sowie der Messungen erfolgen.

4.1.3 Simulationen im Rechner

Simulationen im Rechner liefern ein gutes Bild des Verhaltens der erstellten Schaltung. Es gibt zwei wichtige Simulationsmethoden.

Funktionale Simulation: Funktionale Simulationen analysieren die Schaltung logisch. Dabei wird das theoretische Verhalten der programmierten Schaltung wiedergegeben. In logischen Simulationen kann man das grundsätzliche Verhalten einer Schaltung sowie Abhängigkeiten einzelner Variablen überprüfen. Funktionale Simulationen berücksichtigen keine Störeffekte realer Schaltungen. Störeffekte sind vor allem Gatterlaufzeiten, Glitches und Spikes (siehe unten).

Timingsimulation: Timingsimulationen[6] dienen dazu, das Laufzeitverhalten einer Schaltung analysieren. So erkennt man Gatterlaufzeiten und unerwartetes Verhalten einer Schaltung. Deshalb erfolgt eine *Timingsimulation* erst nach der *funktionalen Simulation* und dem *Fitten* der Schaltung in eine bestimmte FPGA.

Anhand einer Addiererschaltung soll die Funktionsweise einer HDL erklärt werden und zudem die zwei Arten der Simulation verdeutlicht werden. Die *Addition* von zwei 128 Bit breiten Zahlen wird mit einer Addiererschaltung durchgeführt. Die schaltungstechnische Realisierung ist dabei nicht trivial[7].

Die Beschreibung eines Addierers hingegen ist eindeutig. In der HDL *Verilog* kann der Addierer einfach durch folgendes Programm beschrieben werden:

```
module Adder (Zahl1, Zahl2, Ergebnis)
     input [127:0] Zahl1, [127:0] Zahl2;
     output Ergebnis;
     reg [128:0] Ergebnis;

     always  (posedge clock)
          begin
               Ergebnis = Zahl1 + Zahl2;
          end
endmodule
```

Auf die Deklaration der Variablen und den Aufbau eines *Verilog*-Programms soll hier nicht eingegangen werden, wichtig ist die Zeile:

$$Ergebnis = Zahl1 + Zahl2; \tag{4.2}$$

[6]man könnte das Wort Timingsimulation etwa mit Laufzeitsimulation übersetzen

[7]Man verwendet dazu hintereinandergeschaltete Volladdierer, die sich aus jeweils vier Gattern zusammensetzten.

Bei einer funktionalen Simulation dieses Programms wird man sicher immer das erwartete Ergebnis erhalten. Problematisch ist das Übertragen des Programms in die FPGA. Das beschriebene Verhalten wird durch die Synthesewerkzeuge in die FPGA *gefittet* (Kapitel 4.1.1). Der Fitter entscheidet dabei eigenständig, wie das beschriebene Verhalten in einer Schaltung realisiert und auf die MegaLABs verteilt wird. Bei einer zu abstrakten Beschreibung der Funktion der Schaltung führt dies oft zu einer nicht funktionierenden Schaltung. In diesem Fall muss das Modell der Schaltung genauer beschrieben werden.

Gerade bei dem oben genannten Addierer entsteht eine nicht optimale Schaltung. Da 128 Bits miteinander addiert werden, ist es wahrscheinlich, dass die Berechnung nicht in einem Takt durchgeführt werden kann. Das Synthesewerkzeug kann nicht alleine entscheiden, auf wie viele Taktzyklen die Berechnung der Zahl gestreckt werden muss. Kann die Berechnung nicht in einem Takt durchgeführt werden, müssen zudem Zwischenergebnisse gespeichert werden. All diese Überlegungen müssen in der Beschreibung der Schaltung enthalten sein.

Hat der Fitter das Modell der Schaltung in der FPGA realisiert, können Timingsimulationen zum Laufzeitverhalten durchgeführt werden. Mit diesen Simulationen kann das Verhalten des Programms getestet werden, ohne dass die entsprechende Elektronik gebaut werden muss.

Sind die Simulation erfolgreich, kann die festverdrahtete Schaltung durch die FPGA ersetzt werden (Abbildung 4.3).

Auf die folgenden Störeffekte muss geachtet werden:

Gatterlaufzeiten: Gatterlaufzeiten entstehen durch Schaltzeiten der Transistoren in der FPGA. Typische Laufzeiten sind etwa 2-5 ns in Abhängigkeit der Größe der FPGA und des in die FPGA programmierten Programms.

Glitches: Glitches entstehen, wenn sich ein Signal innerhalb kurzer Zeit mehrfach ändert, z.B. wenn es gerade berechnet wird. Solange Glitches nicht kurz vor der Clockflanke auftreten, sind sie nicht kritisch.

Powerglitches: Bei einem Powerglitch bricht kurzzeitig die Versorgungsspannung zusammen. Danach muss die FPGA normalerweise neu programmiert werden. Durch geerdete Kondensatoren, die man in die Stromversorgungsleitungen schaltet, kann man das Risiko eines Powerglitches verringern. Powerglitches entstehen zum Beispiel, wenn viele Signale gleichzeitig den Zustand von *0* zu *1* wechseln und das Stromversorgungsgerät nicht schnell genug ausreichende große Ströme liefert.

Spikes: Spikes sind Spitzen, die entstehen, wenn sich ein noch nicht fertig gebildeter Zustand ändert und so einen anderen, vom ersten abgeleiteten Zustand für einen Moment verändert.

Setup & Hold Times: Damit digitale Schaltungen zuverlässig funktionieren, müssen einige Voraussetzungen an das zeitliche Verhalten der Signale am Eingang und in der

Schaltung gemacht werden (siehe Abb. 4.5). Es gibt ein Zeitfenster um die schaltende Clockflanke, in dem jeder Signalzustand stabil sein muss, damit sichergestellt ist, dass der richtige Zustand weiterverarbeitet wird. Während der Setup & Hold-Zeiten dürfen sich die Signale nicht verändern, da das System ansonsten in einen undefinierten Zustand übergeht. Die Setupzeit wird mit t_{SU}, die Holdzeit mit t_H bezeichnet.

Die oben angeführten unerwünschten Phänomene digitaler Schaltungen sollten nicht innerhalb des S & H-Zeitfensters auftreten. Zudem wird die Zeit, in dem sich Signalzustände ändern dürfen, mit zunehmender Taktfrequenz kleiner, da die Größe des S & H-Zeitfenster fast unabhängig von der Taktfrequenz ist.

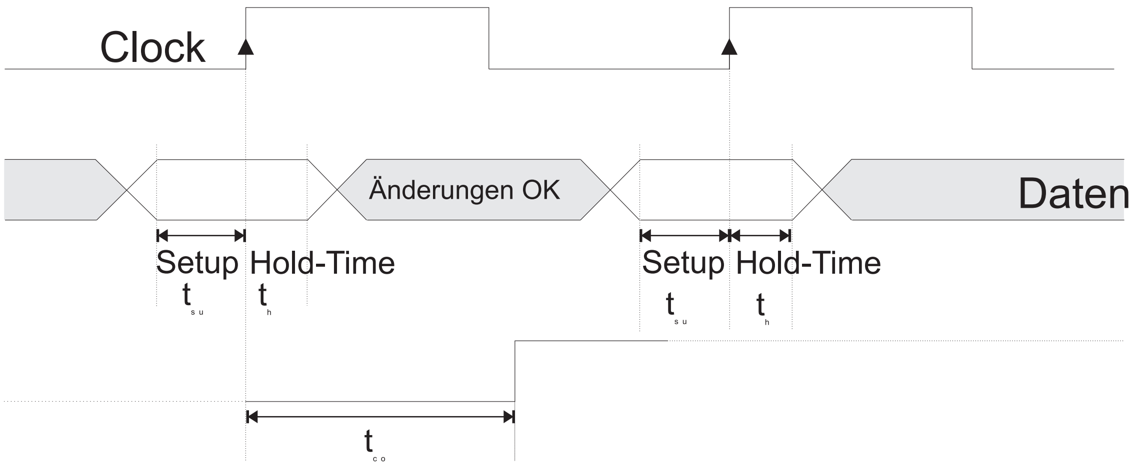


Abbildung 4.5: Zeitfenster um die Clockflanke: Setup & Hold Zeiten: Während der S&H-Zeiten dürfen Datenzustande nicht verändert werden.

- t_{SU} (clock setup Zeit): Die Setupzeit ist die Länge der Zeit, die ein Signal in einem Register sein muss, bevor die Clockflanke, die die Auslese dieses Register einleitet, anliegen muss.
- t_H (clock hold Zeit): Zeit, die das Signal im Register gehalten werden muss, bis die Clockflanke garantiert an jedem Register anliegt.
- t_{CO} (Clock to output-Verzögerung): Die Schaltverzögerungszeit gibt an, nach welcher Zeit eine durch die Clockflanke ausgelöste Aktion an ein Ausgangspin geschaltet werden kann.

In Messungen am später behandelten Testsystem werden diese Zeiten für den APEX 20K-400 bestimmt (siehe Abschnitt 6.1.3 in Kapitel 6).

Die Entwicklungsumgebung Eine Entwicklungsumgebung ist eine dem Anwender zur Verfügung gestellte Software, mit der die komplette Entwicklung eines Hardwareprojekts durchgeführt wird. Die von Altera speziell für FPGAs der APEX-Serie entwickelte Entwicklungsumgebung heißt Quartus. Alle Programme wurden mit dieser Software erstellt, simuliert und in die FPGA gefittet. Die Software bietet eine Projektumgebung, in der alle Stufen der Entwicklung integriert sind, die nacheinander verwendet werden:

- Der *Design Entry* ist der Rahmen eines Projekts. Jede Schaltung besteht aus Modulen, die eine vorgegebene Aufgabe erfüllen. Im *Design Entry* werden die einzelnen Module zu einem Modell der Schaltung verbunden. Jedes Modul kann im sogenannten **B***lock* **F***ile* **D***iagramm* (BDF) als Block dargestellt werden. Diese Blöcke enthalten die Verilogprogramme. Das Blockdiagramm verbindet die Ein- und Ausgänge der einzelnen Module. In Abbildung 4.6 ist das Blockdiagramm für den Triggeralgorithmus mit 60 Pads dargestellt.

- Die *Verilog Programmierumgebung* ermöglicht das Programmieren der Module, die im *Design Entry* als Blöcke dargestellt sind. Mit einem Kompiler können die Programme übersetzt werden, um sie auf Fehler zu überprüfen.

- Der *Funktionale Simulator* bietet die Möglichkeit, Simulationen am Modell der Schaltung durchzuführen.

- *Router* und *Fitter* berechnen die Schaltung für eine beliebige APEX FPGA.

- Mit dem *Laufzeit Simulator* kann man das Verhalten der Schaltung in der FPGA simulieren.

- Mit dem *Programmierer* kann man die Daten in die FPGA oder EEPROMs auf der fertigen Platine übertragen.

Die Quartus Software bietet neben der Entwicklungsumgebung eine Vielzahl an vorkonfigurierten Modulen, die an die spezielle Struktur der FPGA angepasst werden. So gibt es z.B. sehr viele Module, die den im APEX FPGA vorhandenen Speicher ansprechen, dies erspart die Erstellung eigener Module.

4.2 Das Testsystem

Im Rahmen der Diskussion über den Bau des Triggersystems wurde beschlossen, ein Testsystem zu entwickeln. Das Testsystem enthält eine FPGA mit einer beschränkten Anzahl an Leitungen. Für das Testsystem wird ein Triggeralgorithmus entwickelt, der nicht die volle Anzahl Pads eines ϕ-Sektors verwendet. Durch die reduzierte und dadurch handlichere Anzahl von Pads können viele Entwicklungen übersichtlich durchgeführt werden. Die Entwicklung des Testalgorithmus macht einen Großteil der Arbeit aus, die Erweiterung auf die volle Pad-Anzahl wurde anschließend einfach möglich. Mit dem Testsystem werden Messungen zur Leistungsfähigkeit der FPGAs und des Algorithmus ermöglicht. Auf den Bau, die Eigenschaften, Inbetriebnahme und Tests dieses Systems wird in Kapitel 6 eingegangen, zunächst wird die Entwicklung des Testalgorithmus und des endgültigen Triggeralgorithmus beschrieben.

4.2.1 Testschaltungen

Der Testalgorithmus bezieht nur 60 Pads in den Triggerprozess ein. Dennoch müssen hier alle notwendigen Module entwickelt und miteinander verbunden werden. Folgende Module wurden nacheinander entwickelt und simuliert:

- Trigger_control
- Demultiplexer
- Trackfinder
- Hitlistsummierer

Die Speicherfunktion wurde erst sehr spät ins Testsystem integriert und soll gesondert betrachtet werden (Kapitel 4.3).

Optional können Filter zwischen Demultiplexer und Trackfinder geschaltet werden, die eine Vorselektion der Daten ermöglichen. Auf diese Funktion soll nicht näher eingegangen werden. Aufgrund der flexiblen Programmierung der FPGAs können diese Funktionen nachträglich programmiert werden.

In Abbildung 4.6 ist das BDF des Testalgorithmus abgebildet. Die Funktion der Module wird nun im einzelnen erklärt.

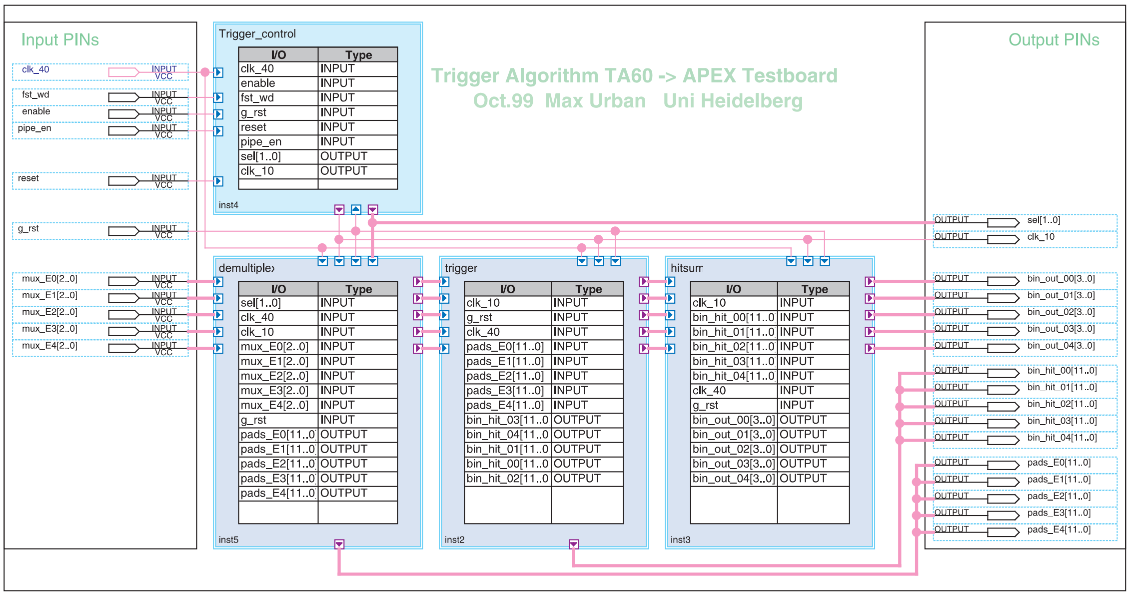


Abbildung 4.6: Blockdiagramm vom Testalgorithmus (TA60). Dieser Testalgorithmus enthält schon alle wichtigen Elemente des Triggeralgorithmus (siehe Text), verarbeitet aber nur die Padinformation von 60 Pads. Die Verbindungen zwischen den Blocks entsprechen Verbindungsleitungen. Eingangspins sind im Blockdiagramm links, Ausgangspins rechts angeordnet. In jedem Block stehen die Bezeichnungen der Ein- und Ausgangssignale, die die Schnittstelle zwischen Block und Gesamtsystem herstellen.

4.2.2 Trigger_control

Das Trigger_control-Modul ist die zentrale Steuer- und Kontrolleinheit (Abb. 4.6, oben links). Es überwacht alle Signale, die das Triggersystem von außen beeinflussen und gibt Änderungen an die anderen Module weiter. Das Trigger_control-Modul reagiert auf folgende externe Signale:

- clk_40: Die *40 MHz* Clock ist der zentrale Taktgeber des Systems. Alle Schaltvorgänge werden durch die positive Flanke dieses Signals ausgelöst[8]. In der Hardwarebeschreibung gibt es eine Konstruktion, die dieses Verhalten ausdrückt:

$$always \quad @(posedge \quad clk\_40) \quad begin...end \tag{4.3}$$

  Alle Anweisungen, die innerhalb einer solchen *always*–Konstruktion stehen, werden genau dann ausgeführt, wenn eine positive Taktflanke dieses Signals kommt.

- g_rst: Wenn ein globales Reset gesendet wird, muss das Triggersystem sofort in einen definierten Anfangszustand gehen. Das Resetsignal ist asynchron und muss nicht mit dem *40 MHz* Signal synchronisiert werden. Die oben genannte *always*–Konstruktion bietet hierfür eine Erweiterung:

$$always \quad @(posedge \quad clk\_40 \quad or \quad posedge \quad g\_rst) \quad begin...end \tag{4.4}$$

  In diesem Fall wird die *always*–Konstruktion auch bei einem Resetsignal durchlaufen.

- fst_wd: Das *First-word*-Signal (fst_wd) dient zum Abgleich der Datenpakete aus dem CIPix. Die Signale werden im CIPix vierfach gemultiplext, deswegen wird beim Entschlüsseln im Demultiplexer ein Signal benötigt, damit jeder verschlüsselte Datensatz richtig entschlüsselt wird. Fst_ed ist dieses Normierungssignal (Der Demultiplexer ist in Abschnitt 4.2.3 beschrieben).

- pipe_en: Das *Pipe enable (pipe_en)* Signal kommt vom CTC. Es gibt an, ob die Datenpipeline arbeiten kann, also der Detektor Daten nimmt (*pipe_en*= 1), oder ob die Datennahme zur Untersuchung eines eventuellen Ereignisses unterbrochen wird (*pipe_en*= 0) und Daten aus der FPGA ausgelesen werden (siehe auch Kapitel 4.3).

Neben der Überwachung der externen Steuersignale muss das Trigger_control-Modul für Demultiplexer- und Multiplexermodule einen vierstufigen Zähler erzeugen, der mit jeder clk_40 Flanke inkrementiert und mit dem fst_wd Signal auf 0 zurückgesetzt wird.

Zudem wird das 10,4 MHz Signal (clk_10) im Trigger_control-Modul erzeugt (Um ein definiertes Phasenverhältnis zu bekommen, wird nicht das externe 10,4 MHz-Signal verwendet). Alle Module werden auf das clk_10 Signal synchronisiert.

Statemachine: Die genannten Forderungen werden in einem Zustandsdiagramm beschrieben, das mit einer Statemachine realisiert wird. Jeder Zustand, den das System annehmen kann, sowie jeder denkbare "Bedingungs"-Pfad zwischen Zuständen wird in dieser Statemachine dargestellt. Dabei muss sichergestellt sein, dass das System sich immer in einem der möglichen Zustände befindet.

[8] Die im Triggersystem verwendete Clock wird 41,6 MHz betragen. In allen Simulationen wurde zur Vereinfachung und besseren Ablesbarkeit von einer 40 MHz Clock ausgegangen. Das gleiche gilt für die 10,4 MHz Clock

Wird eine bestimmte Bedingung wahr, geht das System in den nächsten möglichen Zustand über. Durch ein Reset geht das System in einen fest definierten Anfangszustand, der *Idle* genannt wird. Statemachines erfordern taktsynchrone Zustandsänderungen. In [St98] oder [Ti91] wird die Funktion von Statemachines genauer dargestellt. In Abbildung 4.7 ist die Statemachine für das Kontrollmodul gezeigt.

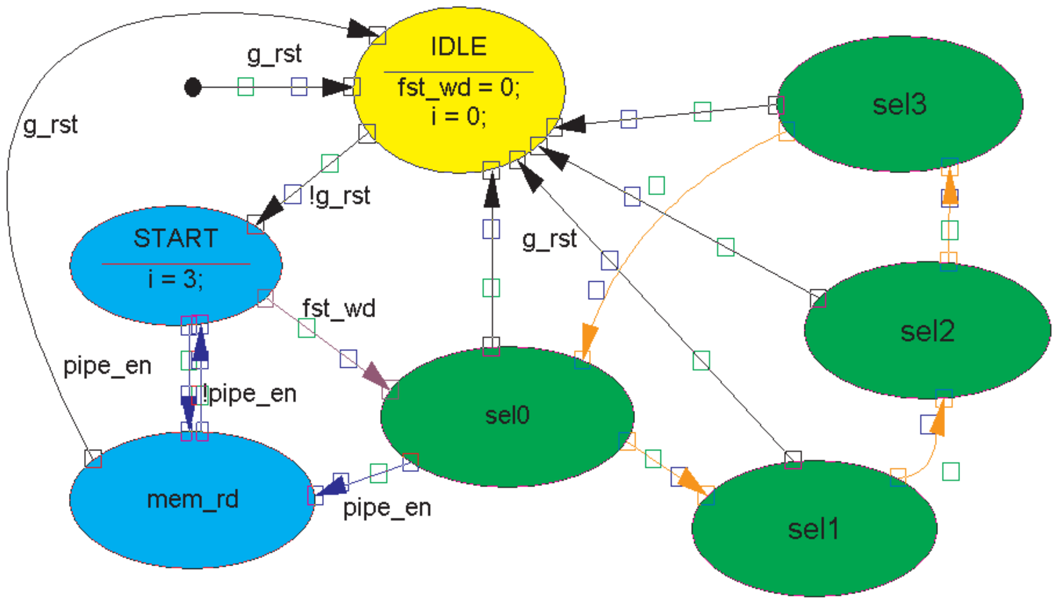


Abbildung 4.7: Zustandsdiagramm des Trigger_Control-Moduls, in Abhängigkeit externer Steuersignale wird ein 4 Bit Zähler erzeugt.

Solange das g_rst Signal nicht null ist, befindet sich das System im Zustand *Idle*. Erst wenn *g_rst* auf "0" gesetzt wird, wird der Zustand *start* erreicht, der solange anliegt, bis das System das fst_wd Signal erhält oder durch pipe_en in den *mem_rd* springt. Geht das System in den *mem_rd*-Zustand, übernimmt das Speicher_kontroll-Modul (Abschnitt 4.3) die Kontrolle über das System. Geht das System in den Zustand *sel0*, läuft es immer im Kreis zwischen den Zuständen sel0, sel1, sel2 und sel3, die den vierstufigen Zähler bilden. Dieser Zyklus kann nur im Zustand sel0 durch ein pipe_en oder in allen Zuständen durch ein g_rst verlassen werden.

In Abbildung 4.8 ist die funktionale Simulation der Trigger_control Statemachine zu sehen. Es ist zu erkennen, dass das clk_10 Signal immer einen Takt vor dem fst_wd Signal anliegt. Dadurch ist sichergestellt, dass mit der positiven clk_40-Flanke das clk_10 Signal anliegt, zu der das fst_wd Signal gehört. Dadurch läuft allerdings das clk_10 Signal dem fst_wd Signal ein BC nach (zu erkennen in der Abbildung bei 200 und 500 ns).

Erfolgt ein globales Reset (g_rst), geht das System sofort in den *Idle*-Zustand, bei einem Ausschalten von pipe_en läuft der Zähler noch zu Ende, dann geht das System in den *Idle*-Zustand.

Die Timingsimulation wurde zusammen mit der des Demultiplexers durchgeführt, der zunächst geschildert wird.

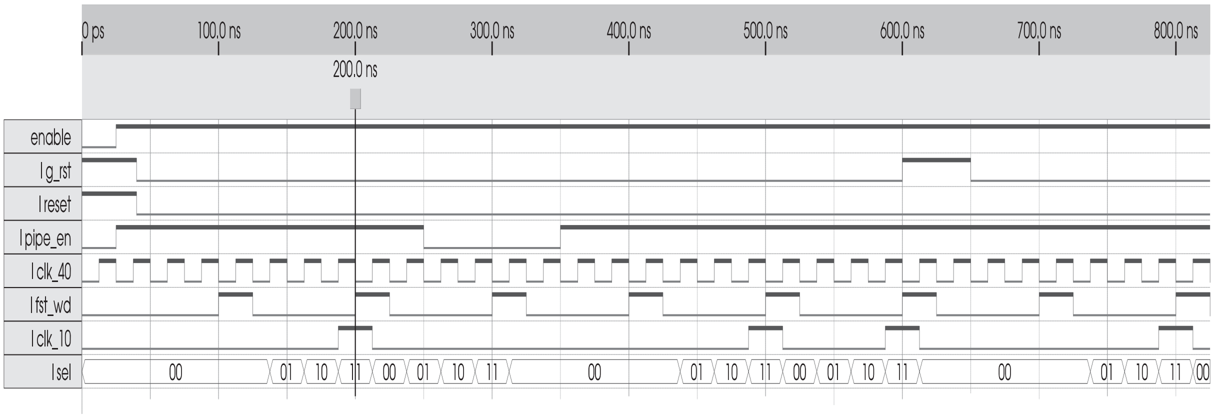


Abbildung 4.8: Funktionale Simulation der Statemachine (siehe Text)

4.2.3 Demultiplexer

Um möglichst wenig Ausgangskanäle auf dem CIPix unterzubringen, werden die Signale dort vierfach gemultiplext. Die von den Multiplexern erzeugten Signale müssen im Triggersystem zurückverwandelt werden. Dies geschieht mit einem Demultiplexer (Abbildung 4.9).

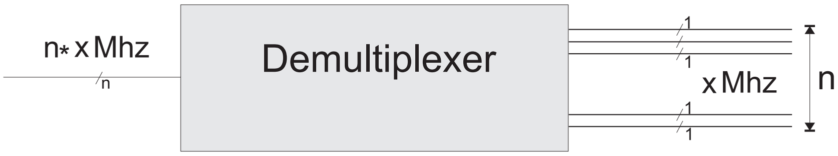


Abbildung 4.9: Schematische Darstellung eines Demultiplexers: Ein Signal mit n-facher Frequenz wird in n Signale einfacher Frequenz demultiplext.

Da die CIP-Kammerdaten vierfach gemultiplext sind, muss der Demultiplexer Signale mit *40 MHz* in vier Signale mit *10 MHz* verwandeln.

Der Demultiplexer besteht intern aus zwei Teilen: Einem Zähler, der vier Zustände liefert(das *Trigger_control* Modul), und einem Modul, das bei jedem der vier Zustände ein Datenpaket einem der vier Ausgangsleitungen zuordnet.

Das Datenpaket, welches taktgleich mit dem *fst_wd* Signal kommt, ist per Definition das Erste. Es kommt mit der positiven *40 MHz* Taktflanke und dem ersten Zählerzustand (sel = 0) und wird an die Ausgangsleitung 0 gelegt (mux = demux_0). Entsprechend werden die Daten beim zweiten Zustand (sel = 1) an Ausgang zwei (mux = demux_1), beim dritten (sel = 2) an Ausgang drei (mux = demux_2) und beim vierten (sel = 3) an Ausgang vier (mux = demux_3) gelegt.

Die zugeordneten Signale werden nicht sofort an die entsprechenden Ausgangsleitungen geschaltet, sondern zunächst zwischengespeichert. Wenn drei der vier Signale in den entsprechenden Zwischenspeicher geschrieben wurden, werden sie taktgleich mit dem vierten Signal an die Ausgänge gelegt. Durch das Zwischenspeichern stellt man sicher, dass die Signale für 100 ns, also einen 10 MHz Zyklus, gleichzeitig an den entsprechenden Ausgangsregistern anliegen, andernfalls bestünde die Gefahr, dass Signale verschiedener BCs zusammenfallen.

In Abbildung 4.10 ist die funktionale Simulation des Demultiplexers dargestellt. Die Timingsimulation erfolgt am Gesamtsystem (Abb. 4.13)

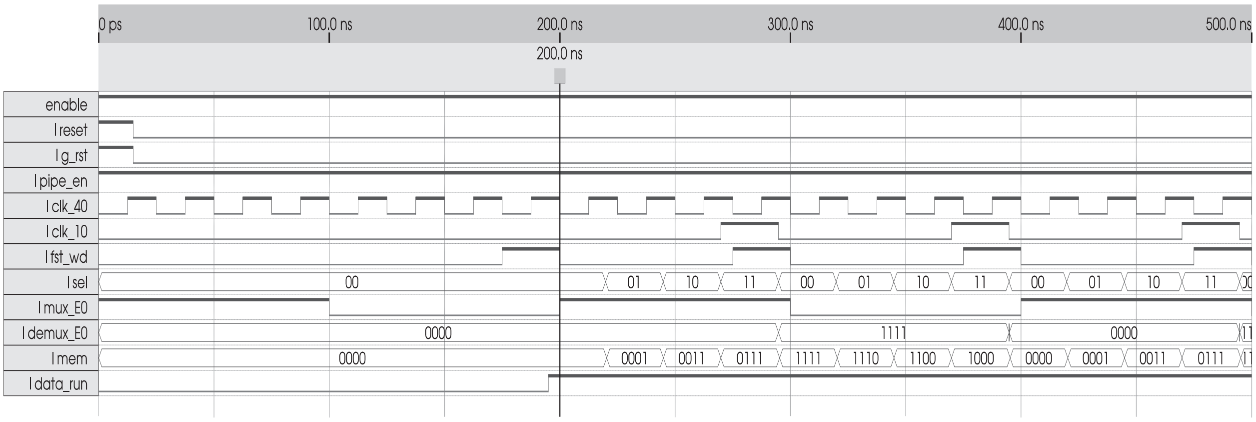


Abbildung 4.10: Abbildung der Funktion des Demultiplexers: Mit der 40 MHz Taktflanke wird der Zähler inkrementiert ("sel" = 1,2,3,4) und die Daten in den Zwischenspeicher "mem" geschrieben. Von dort werden sie mit dem 4. Zählerzustand an den Ausgang "demux_E0" geschaltet.

Aus dem obigen Beschreibungen und aus den Ergebnissen der Simulation geht hervor, dass das Demultiplexen der Kammerdaten 4 *40 MHz* Zyklen dauert. Die demultiplexten Daten liegen nach einem Bunch Crossing vor (vgl. auch Abbildung 3.13 in Abschnitt 3.3.2).

4.2.4 Spurfinder

Die Schaltung wird in ein Logikarray programmiert, in dem gewünschte Pads "ver*und*et" oder "ver*oder*t" werden. Für den ersten Testalgorithmus werden zunächst nur 60 Pads einbezogen, eine lokale Umgebung enthält fünf Spuren. Es wird an dieser Stelle zudem angenommen, dass die Pads nicht in *projektiver Geometrie* angeordnet sind. Es gibt 12 zentrale Pads, also können maximal 60 Spuren gefunden werden. In Abbildung 3.3 in Kapitel 3.2.1 ist der verwendete vereinfachte Spuralgorithmus schematisch dargestellt. In Abbildung 4.11 ist die zur Beschreibung notwendige Funktion dargestellt.

Für diesen Algorithmus ist die Hitlist eine zweidimensionale Matrix mit 5 Spalten und 12 Zeilen entsprechend der 5 Binbereiche mit maximal 12 Treffern pro Bereich.

4.2.5 Addierer

Der in Kapitel 3.2.2 beschriebene Kaskadenaddierer kommt bei diesem "*pad-reduzierten*" Algorithmus noch nicht zum Einsatz. Die zwölf 1 Bit breiten Zahlen werden hier von in der Entwicklungssoftware enthaltenen Standardaddierern zusammengezählt. Diese Addierer garantieren eine gute Laufzeit, können aber maximal 32 Bit breite Zahlen addieren. Deshalb kommen sie für den endgültigen Addierer nicht in Frage.

```


always @ (posedge clk_40  or posedge g_rst)
  begin
    if (g_rst)  // reset Abfrage
      begin
        assign bin_hit_00 = 0;
        assign bin_hit_01 = 0;
        assign bin_hit_02 = 0;
        assign bin_hit_03 = 0;
        assign bin_hit_04 = 0;
      end
    if (clk_10) // warten auf die clk_40 Flanke, bei der clk_10 = 1 ist
     begin     // Spurfinder

      bin_hit_00[0]  = ( pads_E4[0] & pads_E3[0]              & pads_E2[0] &  pads_E1[0]             & pads_E0[0]);
      bin_hit_00[1]  = ( pads_E4[3] & (pads_E3[3]|pads_E3[2]) & pads_E2[2] & (pads_E1[2]|pads_E1[1]) & pads_E0[1]);
      bin_hit_00[2]  = ( pads_E4[5] & (pads_E3[5]|pads_E3[4]) & pads_E2[4] & (pads_E1[4]|pads_E1[3]) & pads_E0[3]);
      bin_hit_00[3]  = ( pads_E4[4] & (pads_E3[4]|pads_E3[3]) & pads_E2[3] & (pads_E1[3]|pads_E1[2]) & pads_E0[2]);

                // 42 weitere Aufrufe, für Demonstration ausgeblendet
                //      ...                 ...                 ...

      bin_hit_04[10]  = ( pads_E4[10] & pads_E3[10] & pads_E2[10] & pads_E1[10] & pads_E0[10]);
      bin_hit_04[11]  = ( pads_E4[11] & pads_E3[11] & pads_E2[11] & pads_E1[11] & pads_E0[11]);
     end

  end
endmodule
```

Abbildung 4.11: Programmkern des Spurfinders

Es wurden keine gesonderten Simulationen für den Addierer durchgeführt. Da der Addierer an den Ein- und Ausgängen Register hat, die nur mit der positiven clk_10 Flanke ausgelesen werden können, beträgt die Laufzeit des Addierers 100 ns (ein BC). Die reine Bearbeitungszeit im Addierer ist weitaus geringer. In der Gesamtsimulation kann man dieses Verhalten ablesen (Abbildung 4.13). Die vom Addierer gelieferten fünf 4 Bit breiten Zahlen entsprechen der Anzahl der Treffer pro Binbereich und stellen das Histogramm der Trefferverteilung dar.

4.2.6 Zusammenspiel der Module

Nach Test der einzelnen Module wurden diese zu einem Gesamtsystem zusammengesetzt und getestet. Für diese Simulationen wurde das in Abbildung 4.6 dargestellte System verwendet. Die im realen System erwarteten Steuersignale wurden im Simulator nachempfunden.

In Abbildung 4.12 ist das Ergebnis der funktionalen Simulation zu sehen. Im direkten Vergleich dazu ist die Timingsimulation (Abb.4.13) derselben Schaltung zu sehen. In beiden Simulationen wurden alle multiplexten Eingangssignale auf "1" geschaltet, um das Laufzeitverhalten der einzelnen Module übersichtlich darzustellen. Im Trigger wurden einige Spurmuster deaktiviert, um das Verhalten des Addierers testen zu können.

In der funktionalen Simulation ist das Verhalten der Schaltung gut zu erkennen:

- Das g_rst Signal wird nach 62,5 ns auf "0" geschaltet, dadurch ist sichergestellt, dass alle Module die Werte in den "Reset-Zustand" gesetzt haben.

- Mit der positiven clk_40 Taktflanke bei 87,5 ns verlässt die Statemachine den *Idle*-Zustand und startet in folgenden Takt den Zähler, da das fst_wd Signal gesetzt ist. Die clk_10 wird erst im dritten Zählerzustand gestartet, sie läuft dem fst_wd Signal vier 40 MHz-Takte hinterher und dient allen Modulen zum Auslesen der Ausgangsregister alle 100 ns.

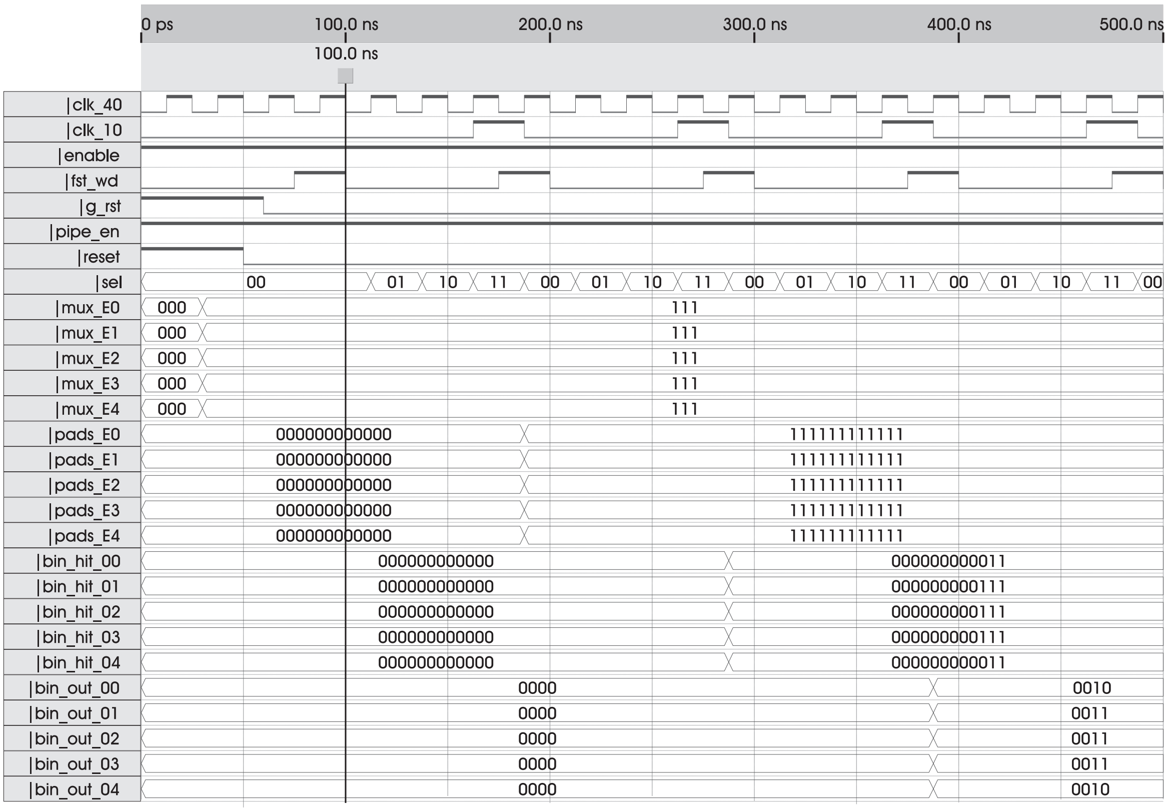


Abbildung 4.12: funktionales Verhalten des kompletten Testalgorithmus (siehe Text).

- Der erste Zählerwert (sel=0) liest die Daten vom Dateneingang in den Demultiplexer, rechtzeitig zum nächsten sel=0 liegen die demultiplexten Daten am Eingang des Spurfindermoduls.

- Im Triggermodul werden die Daten sehr schnell berechnet, nach 10 ns bei 225 ns liegen die Daten schon am Eingang des Addierers. im ersten Binbereich *bin_hit_00* (Abb.4.13) wurden zwei Spuren gefunden (der Rest ist abgeschaltet, mehr konnten nicht gefunden werden).

- Der Addierer darf die Daten aber erst mit der folgenden clk_10 Flanke auswerten. Am Ausgang liegen nach 300 ns (bei 400 ns) die addierten Werte, im Binbereich *bin_hit_00*wurden z.B. zwei Suren gefunden.

In einer folgenden Simulation ist der Dateneingang mit wechselnden Eingangsdaten beschaltet (Abb.4.14). Der Demultiplexer liefert alle 100 ns an die Eingänge des Spurfinders gemultiplexte Daten. Die Funktion ist in der Abbildung für den Kanal *mux_E2* gut zu erkennen (siehe Text zur Abbildung).

In Kapitel 6.2.1 wird über das Verhalten der Testschaltung unter Laborbedingungen am Testsystem berichtet.

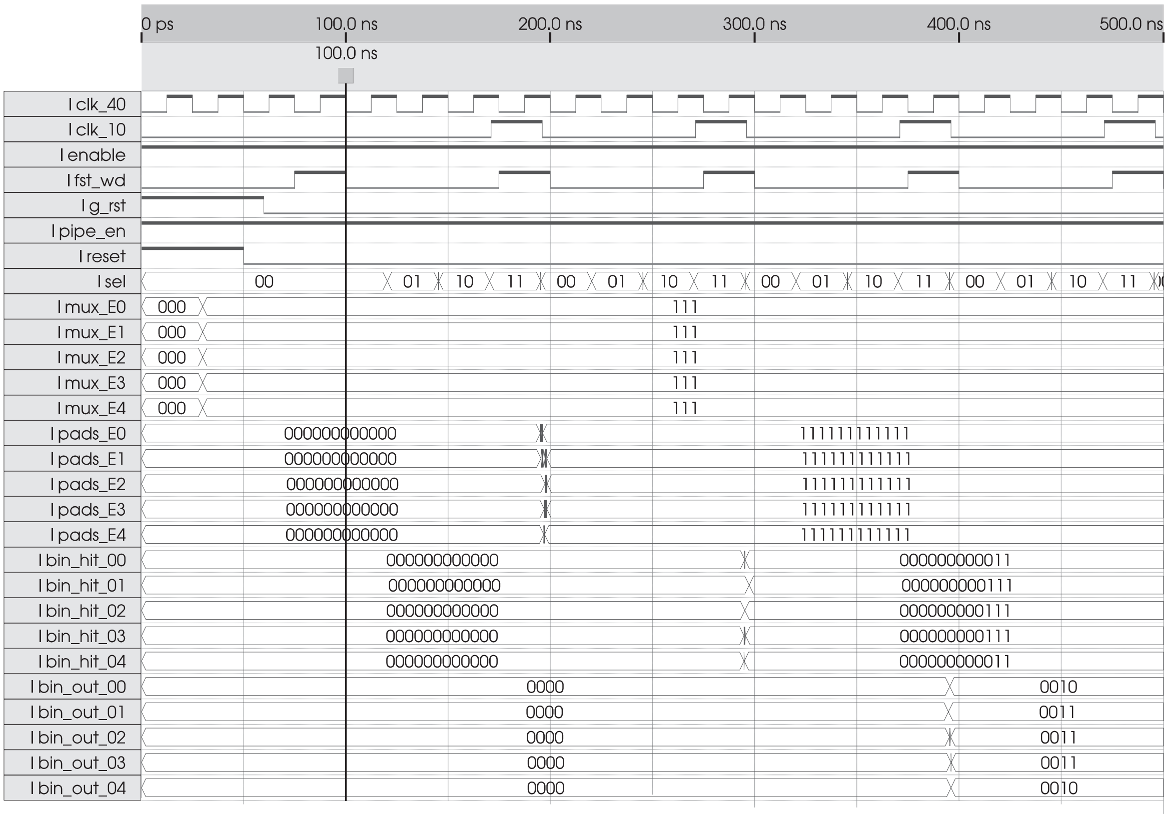


Abbildung 4.13: Timingdiagramm für den kompletten Testalgorithmus. Im Gegensatz zur Abbildung 4.12 erkennt man hier die endlichen Schaltzeiten (z.B. bei 200 ns).

4.3 Speicher

Die Kammerdaten jedes Events werden in der FPGA in einem Ringspeicher für 3,2 μs gespeichert. In dieser Zeit ist das Triggersystem in der Lage, zu entscheiden, ob die gespeicherten Daten aus den FPGAs zu weiteren Untersuchung ausgelesen werden sollen, oder ob sie gelöscht werden können. In Abbildung 4.15 ist der zeitliche Verlauf eines "Speicher-Lese-Zyklus" im Ringspeicher dargestellt. Die zeitlichen Anforderungen an das Triggersystem wurden in Kapitel 3.3.2 behandelt. Entscheidend für das Speichersystem ist das in Kapitel 4.2.2 erwähnte *pipe_en*-Signal. Haben die L1-Trigger entschieden, dass ein Ereignis weiter untersucht werden soll, wird es spätestens 26 BCs nach der Kollision vom CTC auf "0" gesetzt und die Daten aus der FPGA werden ausgelesen.

Speichern: Pro BC und ϕ-Sektor liefert die CIP-Kammer 600 Signale an das Speichersystem. In jeder positiven 40 MHz Taktflanke werden somit 150 Signale in den Speicher geschrieben, so dass nach vier Zyklen die Daten eines BC in den Speicher übertragen wurden (siehe auch Abbildung 3.1, insgesamt werden pro clk_40 Flanke 2400 Signale übertragen, pro ϕ-Sektor sind es 150). Um die Kammerdaten eines BCs zu speichern, müssen diese im Gegensatz zur Analyse nicht demultiplext werden. Sie

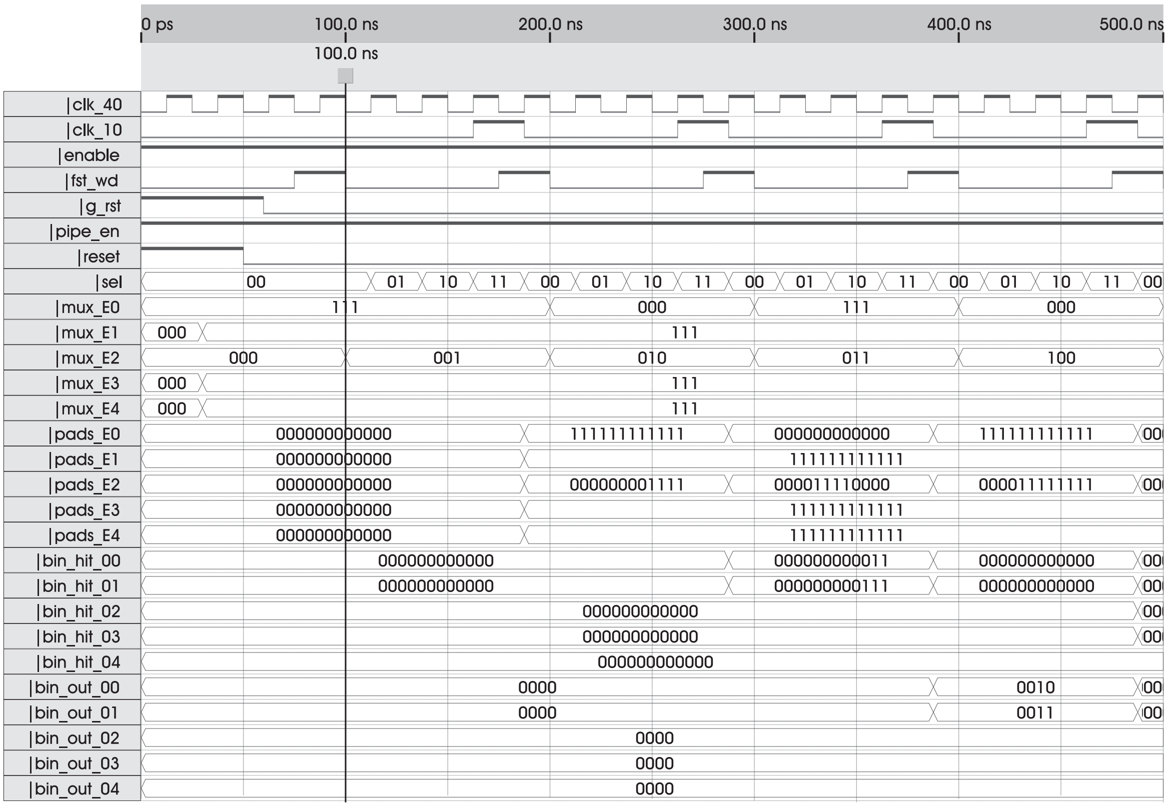


Abbildung 4.14: Funktionale Simulation für wechselnde Eingangsdaten: Es liegen die Daten "0 0 1" am Eingang an, nach 100 ns liegt an *pads_E2* der demultiplexte Wert "0000.0000.1111" vor. Entsprechend verhalten sich die folgenden Werte (0 1 0 $\Rightarrow$ 0000.1111.0000, 0 1 1 $\Rightarrow$ 0000.1111.1111). Der Wert von *bin_hit_01* wird erst bei 300 ns "000.000.000.011", die Quersumme ist "2". Dieser Wert kann an *bin_out_01* nach weiteren 100 ns abgelesen werden (bei 400 ns)

werden direkt in die Speicherzellen geschrieben. Obwohl jeder ϕ-Sektor durch die projektive Geometrie der Kammer nur 534 Pads hat, überträgt das CIPix Auslesesystem 600 Signale, die es bei einer symmetrischen Kammer hätte. Der Speicher speichert somit insgesamt

$$mem_{max} = 150_{Signale} \cdot 32_{BCs} \cdot 4_{SchreibzyklenproBC} = 19200 \quad Pad\ Bits. \tag{4.5}$$

Erfolgt keine Datenauslese, werden die Daten nach 32 BCs zyklisch nach dem FIFO-Prinzip überschrieben. Der Speicher ist in den FPGAs der Triggerkarten untergebracht. Zum Zeitpunkt der Entwicklung des Speichers wurde angenommen, dass sich auf einer Triggerkarte zwei FPGAs befinden (in Abschnitt 4.4 wird auf diese Entscheidung eingegangen). Die 150 Eingangssignale pro Flanke werden symmetrisch auf beide FPGAs aufgeteilt, jede FPGA speichert 75 Signale.

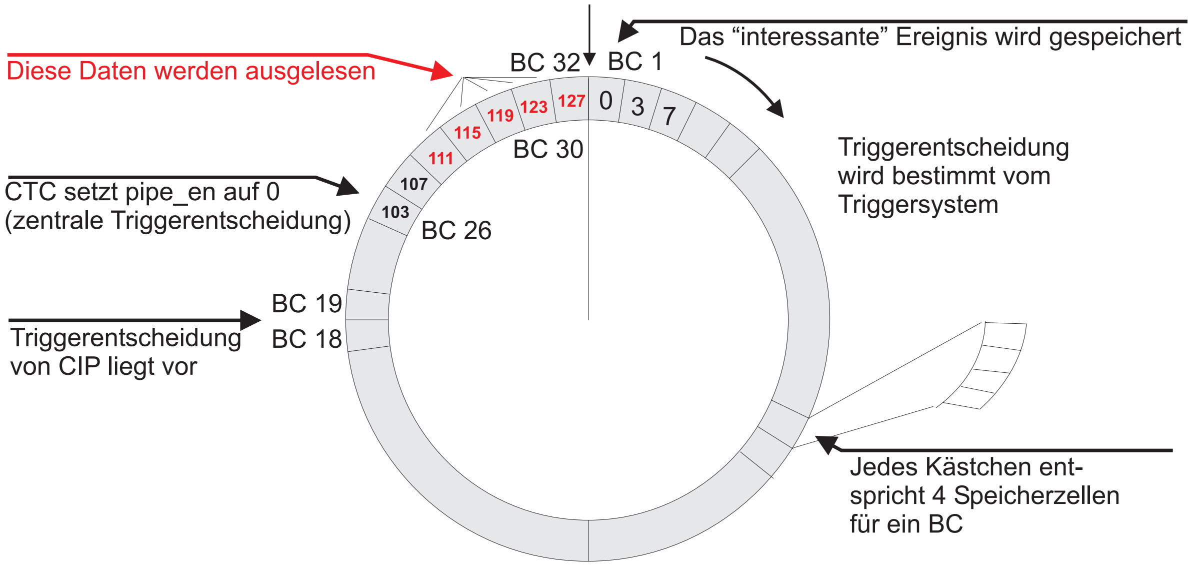


Abbildung 4.15: Darstellung eines Zyklus des Ringspeichers: Der Adressinkrementor wird inkrementiert, hat er den Wert 127 erreicht, springt er auf 0 (siehe Text).

Auslese: Die Auslese des Speichers erfolgt über den VME-Bus[9]. Der VME Bus kann in einem Auslesezyklus 16 Bit von jeder FPGA auslesen, insgesamt werden je Zyklus 32 Bit gelesen. Aus Symmetriegründen werden aber nur 15 Signale pro FPGA auf den VME-Bus gegeben, das 16. Bit dient jeweils als Kontrollbit.

Die Information eines BCs besteht aus 300 Bits pro FPGA, wenn pro Lesezyklus 15 Bits gelesen werden, sind die Daten eines BCs in **20** Lesezyklen ausgelesen. Die Auslesereihenfolge ist wie folgt festgelegt (Abbildung 4.16):

- Zuerst werden die Daten des ersten Multiplexerkanals (sel0) ausgelesen. Hintereinander werden die Information der einzelnen Ebenen ausgelesen, das sind pro FPGA 15 Informationen. Dies dauert fünf Lesezyklen ($75/15 = 5$).

- Anschließend werden die Daten des zweiten, dritten und vierten Multiplexerkanals ausgelesen, was auch jeweils 5 Lesezyklen dauert. Nach 20 Lesezyklen sind also die Daten des ersten BCs ausgelesen.

- Es werden nacheinander die Daten der BCs BC_{n-1}, BC_n , BC_{n+1} und BC_{n+2} übertragen. BC_n ist dabei das BC, in dem sich vermutlich die gesuchte Teilchenkollision ereignet hat. Um das Ereignis besser rekonstruieren zu können, werden die Daten von zwei BCs davor (BC 31 und BC 32) und danach (BC 28 und BC 29) mit abgespeichert.

- Nach 100 Lesezyklen sind alle Daten an die VME-CPU übertragen.

[9] Der VME-Bus wird in fast allen H1-Systemen als Service und Datenbus verwendet. Auf eine genaue Spezifikation des VME-Buses wird nicht eingegangen [Sp87].

Somit ist die

$$Anzahl\ der\ Lesezyklen = 4_{Multiplexerkanäle} \cdot 5_{Ebenen} \cdot 5_{BCs} = 100. \qquad (4.6)$$

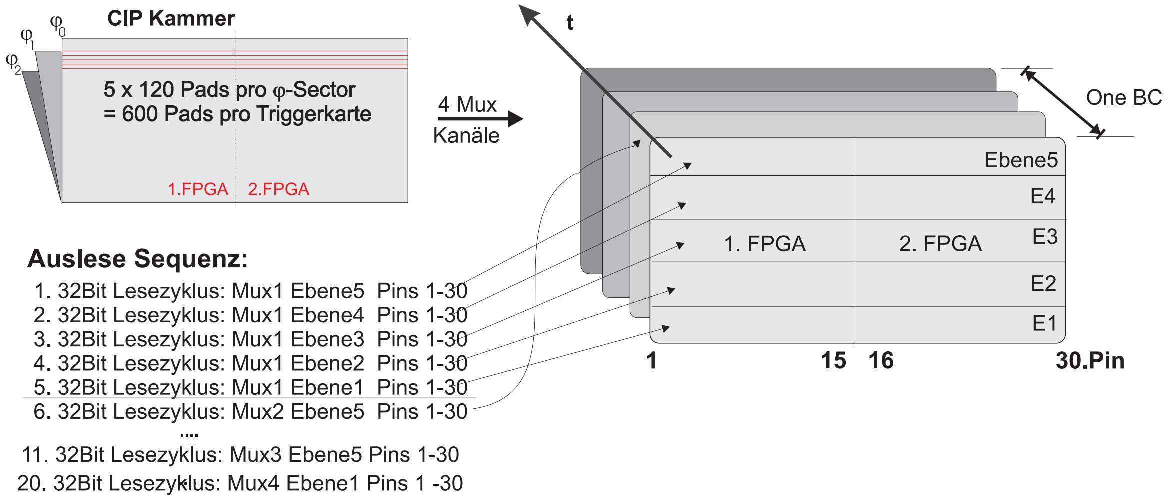


Abbildung 4.16: Auslese der Daten über den VME-Bus. Im ersten Lesezyklus werden die Daten des ersten Multiplexerkanals (sel 0) der 5. Ebene gelesen, das ist ein 30 Bit breites Signal - von jede FPGA 15 Bits. In 20 Lesezyklen sind die 5 Ebenen der 4 Multiplexerkanäle ausgelesen.

Simulation: Der Speicher wird in die im APEX integrierten ESBs programmiert. Altera stellt für die Verwendung dieses Speichers verschiedene vorkonfigurierte Module zur Verfügung, mit denen man verschiedene Arten von Speicher programmieren kann.

Es gibt allerdings keine Möglichkeit, Schieberegister[10] in ESBs zu implementieren, so dass letztendlich ein RAM verwendet wird, dessen Adressenzugriff durch einen Adressinkrementor definiert wird. Damit erreicht man die Funktionalität eines Schieberegister und hat zudem die Schaltzeiten minimiert, da nur Zeiger verschoben werden und nicht die Daten. Allerdings ist eine aufwendige Speicherkontrolle notwendig.

Insgesamt gibt es drei Blöcke:

- Das Speicher-Kontrollmodul enthält zwei Statemachines, eine Lese- und eine Schreibmachine. Durch einen Zustandswechsel des *pipe_en*-Signals wechselt das System vom Lese- in den Schreibzustand und umgekehrt. Für das Schreiben in den Speicher wird eine andere Clock verwendet als für die Auslese. Die Schreibmachine läuft taktsynchron mit der von der HERA Clock abgeleiteten 40 MHz Clock und die Lesemachine mit der vom VME-Bus vorgegebenen VME Clock (wr_vme) mit ca. 5 MHz. Da beide Statemachines taktsynchron arbeiten, müssen alle Steuerleitungen jeweils auf beide Clocks synchronisiert werden.

[10] Der Ringspeicher ist ein zyklisch beschreibbares Schieberegister.

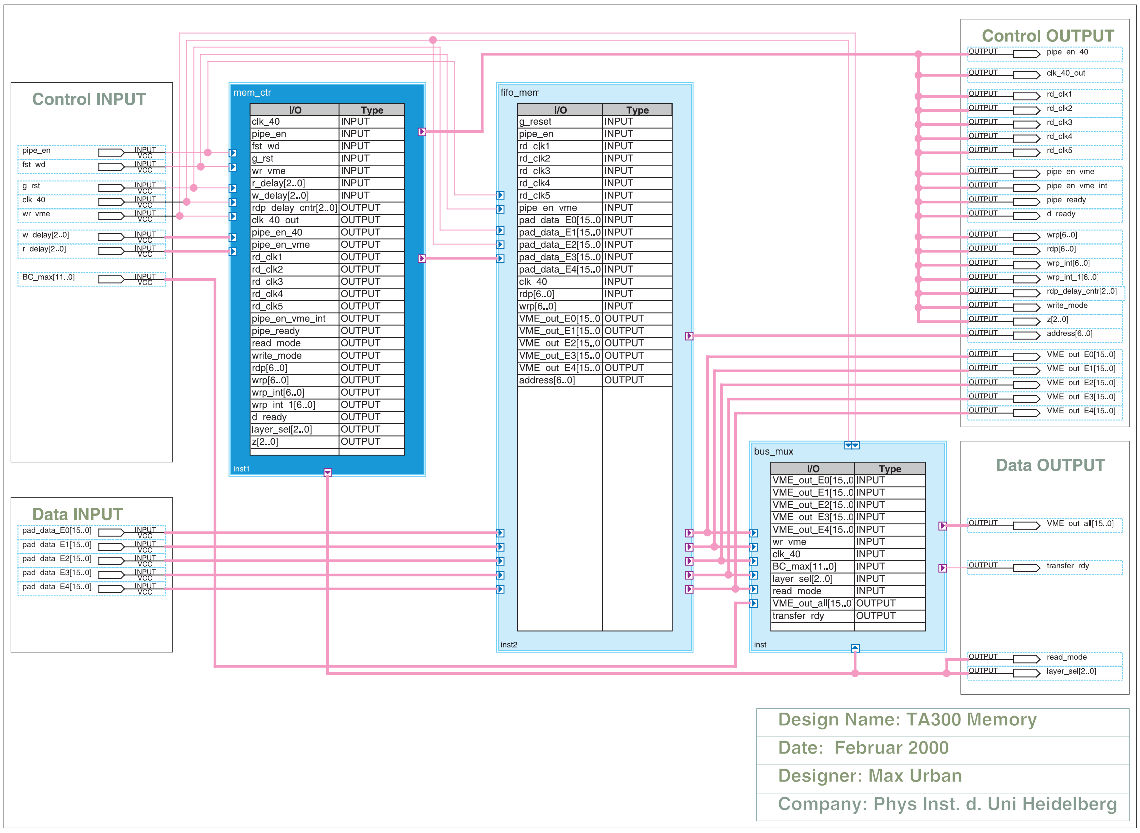


Abbildung 4.17: Blockdiagramm des Speichers (die Blöcke sind im Text erklärt).

- Das Speichermodul enthält die vorgegebenen Funktionen zum Lesen und Schreiben des Speichers. Das Speicher-Kontrollmodul übergibt alle notwendigen Variablen an das Speichermodul und ruft die Funktionen zum Lesen und Schreiben auf.

- Das Busmultiplexermodul fasst die bisher parallel im Speicher befindlichen Daten der einzelnen Ebenen in einen 15 Bit breiten Datenkanal zusammen und übergibt die Daten an den VME-Bus. Mit jedem VME-Bus Takt gibt der Busmultiplexer eines der fünf Datenpakete (der jeweiligen Ebene) an den VME-Bus.

In den Abbildungen 4.18 und 4.19 ist die Timingsimulationen eines Schreib-Lesezyklus dargestellt. Für die Steuerung des Speichers sind sehr viele Variablen und Steuersignale notwendig, die nicht im einzelnen erklärt werden. Im Anhang B ist die Funktion der einzelnen Signale erläutert.

Schreibzyklus: Die simulierten Kammerdaten *pad_Data_E0 - E4* werden mit der clk_40 in den Speicher geschrieben und der Adressinkrementor *wrp* wird erhöht. Wenn in den Speicher geschrieben wird, erhöht *wrp* den gesamten Adresszähler, der aus der Summe der Schreib- (*wrp*) und Lese- (*rdp*) Adresszeiger gebildet wird. In dem Moment, in dem *pipe_en* $= 1$ wird, wechselt das System in den Lesemodus, nachdem alle

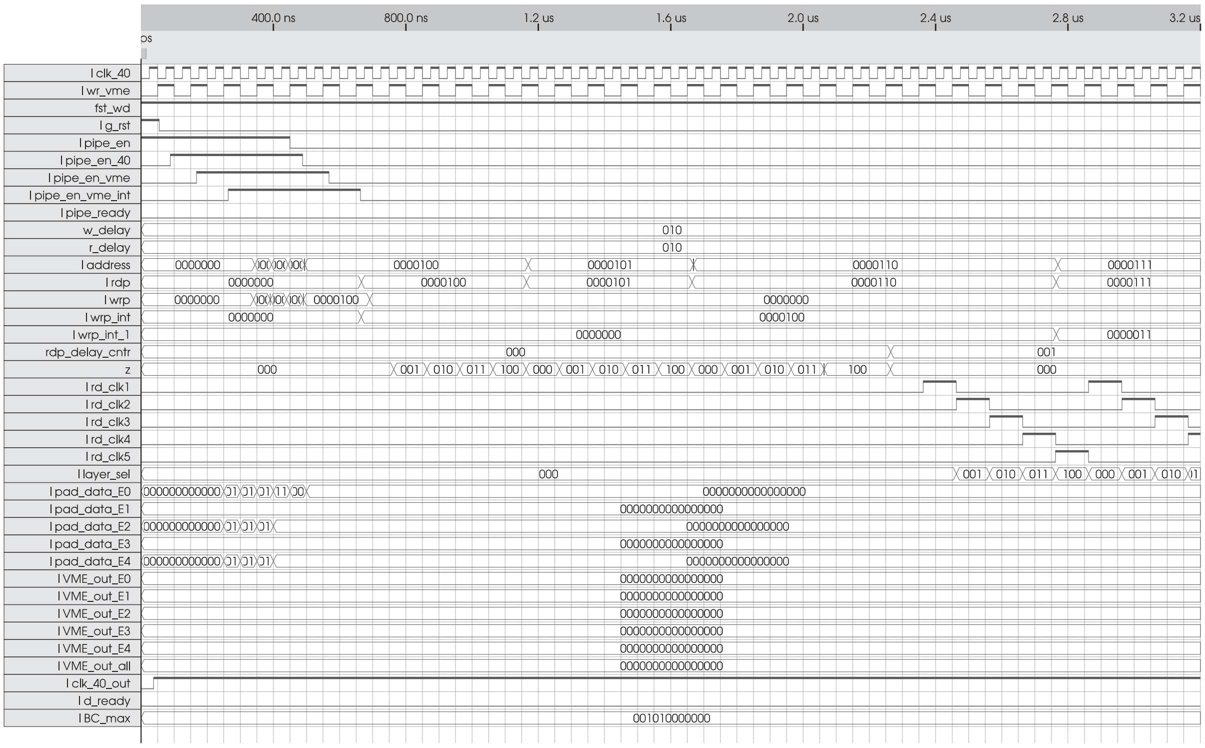


Abbildung 4.18: Timingsimulation des Speichers von 0 ns bis 3,2 μs (siehe Text). Im Anhang B.2 befindet sich diese Abbildung vergrößert.

Steuersignale auf die Lesemachine synchronisiert wurden. Durch das Synchronisieren wird sichergestellt, dass das System nicht in undefinierte Zustände wechseln kann.

Lesezyklus: Nun (bei 700 ns in der Abbildung) beginnt der Lesezylus. Da aber nur in die ersten vier Speicherplätze Daten geschrieben wurden, wird zunächst nur "0" gelesen. Erst bei 63,3 μs zeigt der Adressinkrementor (nach dem 128.) auf den ersten Speicherplatz. Die an diesen Adressen gespeicherten Daten (*VME_out_E0 - E4*) werden ausgelesen (in Abbildung 4.19 bei 63,4 ns). Die Daten liegen an den Ausgängen *VME_out_E1 - E4*. Der Busmultiplexer legt diese Signale der Reihe nach an den *VME_out_all*-Ausgang. Diese Daten werden direkt auf den VME-Bus gegeben.

Das Verhalten des Speichers wurde in Messungen bestätigt (siehe Abschnitt 6.2.2).

4.4 Die endgültigen Triggerkarten

Nach erfolgreichen Tests des kompletten Testsystems und des Speichers wurde mit der Entwicklung des endgültigen Systems begonnen. Es hat sich auch hier als sinnvoll erwiesen, die Komponenten des Systems getrennt voneinander zu entwickeln und zu simulieren.

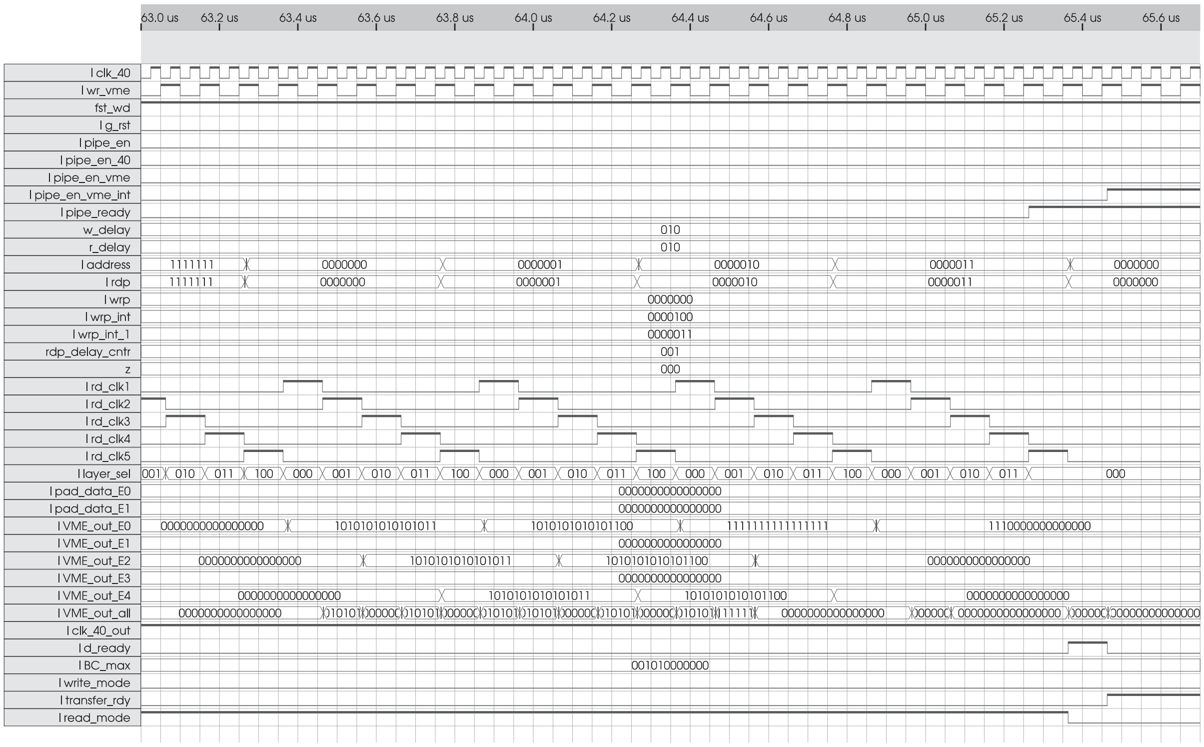


Abbildung 4.19: Timingsimulation des Speichers von 63 μs bis 65,6 μs (siehe Text).Im Anhang B.3 befindet sich diese Abbildung vergrößert.

4.4.1 Dimension des Systems

Platzbedarf des Triggeralgorithmus: Aus Hochrechnungen des Leistungs- und Gatterbedarfs der einzelnen Komponenten musste schon relativ früh entschieden werden, ob man das Programm des Triggeralgorithmus in ein oder in zwei FPGAs programmieren kann. Diese Information ist wichtig, damit parallel zu der Entwicklung der Algorithmen die Triggerkarte entwickelt werden kann.

Addiert man die Auslastung der einzelnen Komponenten, bekommt man eine grobe obere Grenze des Ressourcenbedarfs. Es ist nach allgemeiner Aussage nicht sinnvoll mit mehr als 60% der Gatter in einer FPGA zu planen. In diesem Fall steigt der Routingaufwand stark an, zudem wird die Erweiterbarkeit eingeschränkt. (Das Fitten und Routen der endgültigen Programme in die FPGA hat 14 Stunden gedauert)

Um die Frage zu klären, ob eine oder zwei FPGAs benötigt werden, wurden die Komponenten des Systems zunächst so entwickelt, dass man mit **einem** APEX20K[11] auskommt. In Tabelle 4.1 ist eine Auflistung der von einzelnen Komponenten benötigten Gatteräquivalenten und Speicherzellen für Module in einer FPGA angeführt. Simulationen, in der alle Module zum Gesamtsystem zusammengesetzt werden, zeigen, dass das zusammengesetzte System einen geringeren Platzbedarf hat als die Summe der Einzelkomponenten (s. Tabelle 4.1 unten).

Da die Kapazität einer FPGA mit dem verwendeten Algorithmus fast erschöpft ist,

[11] Altera bietet mittlerweile größere FPGAs an. Zum Zeitpunkt der Entscheidung war nicht bekannt, bis wann diese FPGAs lieferbar sein werden.

	Gatterbedarf [%]	Speicherbedarf [%]
Trigger_control	3	0
Demultiplexer	8	0
Spurfinder	7	0
Addierer	33	0
Multiplexer	5	0
Speicher_control	4	0
Speicher	4	27
Busmultiplexer	5	0
theo. Summe für eine FPGA	69	27
effektive Summe für eine FPGA	58	27
effektive Summe FPGA 1	29	15
effektive Summe FPGA 2	38	15

Tabelle 4.1: Auslastung der FPGA durch genannte Module für das Gesamtsystem. Das Simulationsprogramm gibt nach erfolgreichem Fitten in die FPGA die Anzahl der benötigten Gatter und Speicherzellen an. Diese Werte sind hier zusammengetragen.

wurde beschlossen, **zwei** FPGAs zu verwenden.

Verteilung des Algorithmus auf zwei FPGAs: In die Triggerkarten kommen pro ϕ-Sektor 150 Signalleitungen. Der Demultiplexer erhält diese Signale nach Ebenen getrennt als

$$DATA\_in\_E0[29..0] \qquad bis \qquad DATA\_in\_E4[29..0]. \tag{4.7}$$

Da zwei FPGAs verwendet werden, gibt es in jeder FPGA einen eigenen Demultiplexer. An jede FPGA werden nur die Signale der zu analysierenden Seite und die für Überlappbereiche der lokalen Umgebungen an den Rändern benötigen Pins geliefert. In Abbildung 4.20 ist der definierte Überlappbereich dargestellt. Jedes Kästchen bedeutet einen multiplexten Kanal. Die vom Triggeralgorithmus benötigten Pins sind dunkelgrau markiert. Hellgrau sind die Pins schraffiert, die tatsächlich an die FPGAs geführt werden. Um möglichst flexibel zu bleiben, ist diese Zone größer als die momentan benötigte Zone. Der Demultiplexer in der linken FPGA erhält die Signale

$$DATA\_in\_E0[17..0] \qquad bis \qquad DATA\_in\_E4[17..0]. \tag{4.8}$$

Der Demultiplexer in der rechten FPGA erhält die Signale

$$DATA\_in\_E0[29..12] \qquad bis \qquad DATA\_in\_E4[29..12]. \tag{4.9}$$

Die linke FPGA wertet damit 60 lokale Umgebungen von zPAD_0 bis zPAD_59 aus und liefert so 60 15 Bit breite Einträge in der Hitlist. Die rechte FPGA wertet nur 46 lokale Umgebungen aus. Um die Programme für linke und rechte FPGA möglichst

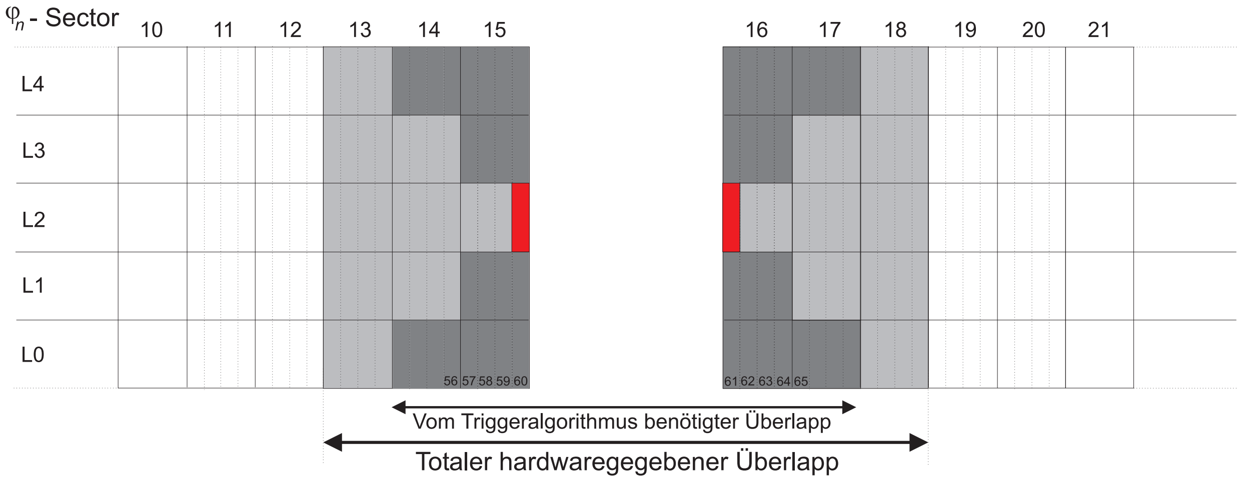


Abbildung 4.20: Darstellung des Überlappbereichs zwischen den beiden FPGAs: dunkelgrau ist der vom Algorithmus benötigte Überlapp markiert, hellgrau sind die tatsächlich an beide FPGAs übertragenen Pads abgebildet.

gleich zu gestalten, sind dies die zentralen Pads 7 bis 52 [12]. In beiden FPGAs wird die Hitlist durch die in 3.2.2 beschriebenen Kaskadenaddierer ausgewertet, auf dessen Implementierung eingegangen wird. Nach der Addition wird das Teilhistogramm über 90 Leitungen zur 2.FPGA übertragen. Die 90 Leitungen setzen sich aus $15_{Bins} \cdot 6_{Bit}$ zusammen (6 Bit, weil $2^6 = 64 > 60$ zentrale Pads der 1.FPGA).

In der zweiten FPGA werden die Histogrammdaten beider FPGAs zum Gesamthistogramm zusammengefasst. Dieses Histogramm hat nun $15_{Bins} \cdot 7_{Bit} = 105$ Leitungen. Aus Platzgründen wird es vierfach gemultiplext. Danach bleiben 32 Leitungen übrig[13], die als vier 8 Bit Zahlen ausgegeben werden:

$$BIN\_mux\_00[7..0] \qquad bis \qquad BIN\_mux\_03[7..0]. \tag{4.10}$$

In den Abbildung 4.21 und 4.22 sind die BDF-Files des auf die zwei FPGAs verteilte Triggeralgorithmus abgebildet. Im Anhang sind diese Abbildungen größer dargestellt. Diese BDF-Files enthalten noch alle Debugleitungen, da mit ihnen die Messungen am vollständigen System durchgeführt werden.

Überlegungen zur Weiterverwendung des TA60 und Neugestaltung: Folgende Komponenten des Testalgorithmus (TA60) konnten ohne größere Änderungen für das endgültige System übernommen werden:

- Das Trigger_control-Modul konnte vollständig wiederverwendet werden.
- Der Demultiplexer musste an die größere Anzahl von Eingangspads angepasst werden. Dafür wurde die Demultiplexerfunktion statt 12 mal nun jeweils 150 mal parallel aufgerufen. Die Funktion blieb unverändert.

[12] ab dem 8.Pad sind die lokalen Umgebungen nicht mehr durch den Rand links und rechts eingeschränkt

[13] 105 : 4 = 26.25, die restlichen Leitungen wurden wiederum für Erweiterungen reserviert.

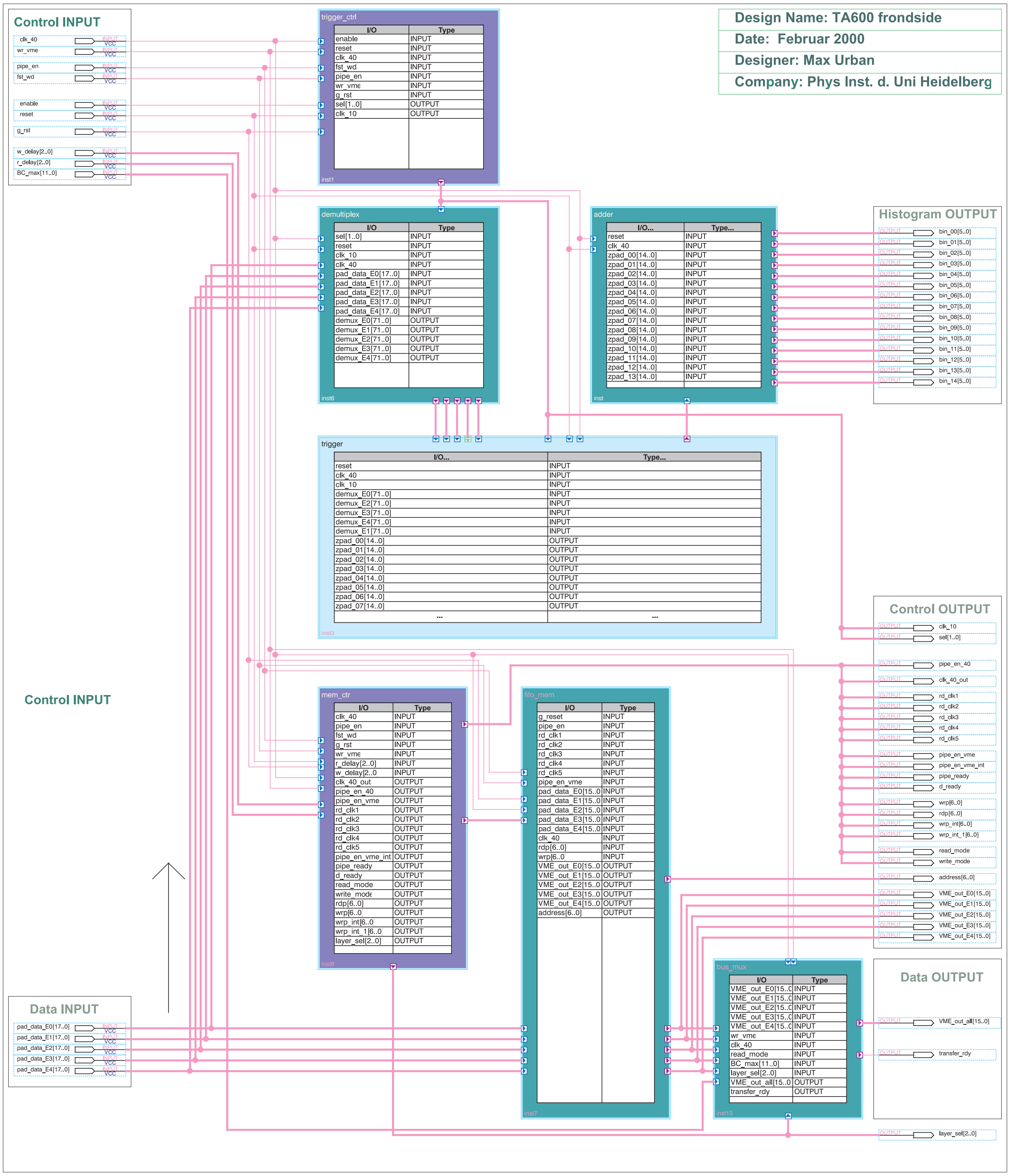


Abbildung 4.21: Das Blockschaltbild des in die linke FPGA programmierten Algorithmus (Diese Abbildung ist im Anhang vergrößert zu finden).

Der Speicher für das endgültige Programm wurde separat vom Testalgorithmus entwickelt (Abschnitt 4.3). Er konnte direkt übernommen werden.

Neugestaltet und überdacht werden mussten

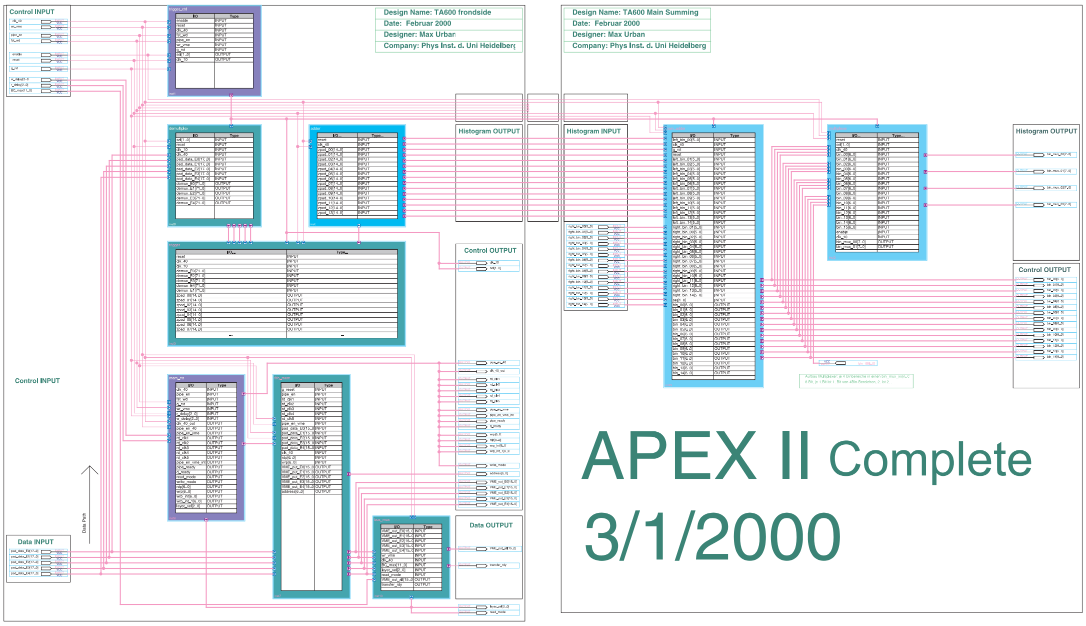


Abbildung 4.22: Das Blockschaltbild des in die rechte FPGA programmierten Algorithmus, hier sind zusätzlich noch Addierer und Demultiplexer für beide FPGAs enthalten (Diese Abbildung ist im Anhang vergrößert zu finden).

- Spurfinder
- Rand- und Grenzbereiche der FPGAs
- Addierer
- Multiplexer

4.4.2 Spurfinder

Der Spurfinder besteht wie auch beim TA60 ausschließlich aus kombinatorischer Logik. Da im Mittel jedes Spurmuster aus 5,5 logischen Verknüpfungen besteht und jede logische Verknüpfung 4 Gatter benötigt, werden bei 15 Binbereiche und 106 zentralen Pads für $15 \cdot 106 = 1590$ mögliche Spurmuster

$$5,5 \cdot 4 \cdot 1590 = 41580 \quad Gatteräquivalente \tag{4.11}$$

für einen vollständigen ϕ-Sektor gebraucht. Durch die projektive Geometrie der Kammer ist nur eine Funktion nötig, die insgesamt 60 (46) mal pro FPGA aufgerufen wird. Diese Funktion benötigt als Eingabevektor alle Pads der lokalen Umgebung und gibt als Ausgabevektor die 15 Bit breite eindimensionale Hitlist zurück.

Die Spurfindung in der Funktion erfolgt nach aus Simulationen gewonnen Spurmusterinformationen. In Abbildung 4.23 sind die zugrundegelegten Spurmuster für eine lokale Umgebung dargestellt. Die Pads, die jeweils von einem Spurbereich berührt

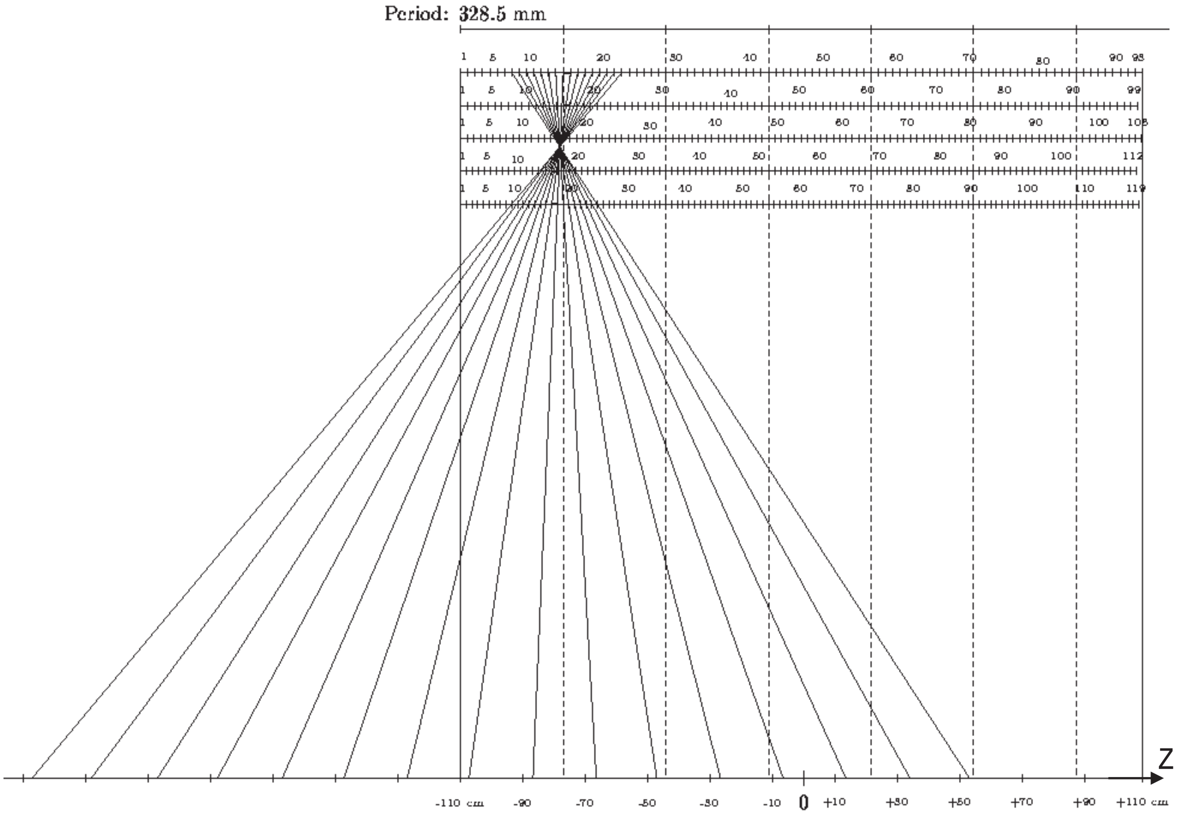


Abbildung 4.23: Aus Simulationen gefundene ideale Padkombination einer lokalen Umgebung [Be98]. Die von zwei Strahlen aufgespannte Fläche berührt die Pads, die ausgelöst werden müssen für eine Zuordnung in den entsprechenden Binbereich.

werden, werden miteinander ver*und*et. Werden zwei Pads der gleichen Ebene berührt, werden diese ver*oder*t.(siehe auch Abschnitt 3.2.1).

In Abbildung 4.24 ist die Funktion abgebildet, die alle markierten Spuren einer lokalen Umgebung erkennt.

Alle 96 ns schreibt diese Funktion die Trefferverteilung des entsprechen zentralen Pads in das Hitlistregister. Dieses Register hat in der ersten FPGA 900 (15 x 60) und in der zweiten 690 (15 x 46) Einträge, die mit der darauf folgenden 10 MHz Clockflanke in den Addierer geführt und ausgewertet werden.

In den Simulationen hat sich der Spurfinder als funktionsfähig erwiesen. Bei 60 parallelen Aufrufen im endgültigen Triggeralgorithmus ist die Anzahl der Zwischenzustände und Spikes in Timingsimulationen höher als beim Testalgorithmus mit nur 10 Spuren. Bis zur folgenden S&H-Zeit sind aber alle Berechnungen abgeschlossen und das Ergebnis liegt vor. Das Programm mit 60 Aufrufen braucht 3 ns (vgl Abb. 4.27) länger als das Testprogramm mit 12 Aufrufen.

Randbereiche der Kammer: Sieben zentrale Pads am linken und rechten Rand der Kammer haben keine vollständige lokale Umgebung. Für jedes dieser 14 Pads muss die lokale Umgebung gesondert definiert werden. Entweder programmiert man diese

```
  always @(posedge clk_40)
    begin
      if (clk_10)
      begin
        // This is the Trigger-Kernel
          out[0]  = E0[0]  + (E1[3]|E1[4])   + E2 + (E3[6]|E3[7])    + E4[14];
          out[1]  = E0[1]  +  E1[4]          + E2 +   E3[7]          + E4[13];
          out[2]  = E0[2]  + (E1[4]|E1[5])   + E2 +   E3[8]          + E4[12];
          out[3]  = E0[3]  +  E1[5]          + E2 +   E3[8]          + E4[11];
          out[4]  = E0[4]  +  E1[6]          + E2 +   E3[8]          + E4[10];
          out[5]  = E0[5]  + (E1[6]|E1[7])   + E2 +   E3[9]          + E4[9];
          out[6]  = E0[6]  +  E1[7]          + E2 +   E3[9]          + E4[8];
          out[7]  = E0[7]  + (E1[7]|E1[8])   + E2 +   E3[10]         + E4[7];
          out[8]  = E0[8]  +  E1[8]          + E2 +   E3[10]         + E4[6];
          out[9]  = E0[9]  +  E1[9]          + E2 +   E3[10]         + E4[5];
          out[10] = E0[10] + (E1[9]|E1[10])  + E2 +   E3[11]         + E4[4];
          out[11] = E0[11] +  E1[10]         + E2 +   E3[11]         + E4[3];
          out[12] = E0[12] + (E1[10]|E1[11]) + E2 +   E3[12]         + E4[2];
          out[13] = E0[13] +  E1[11]         + E2 +   E3[12]         + E4[1];
          out[14] = E0[14] +  E1[12]         + E2 +  (E3[12]|E3[13]) + E4[0];
          // End Trigger-Kernel
      end
//      else
//        begin
//          out = 0;
//        end
      end
// end Trigger Submodule
```

Abbildung 4.24: Diese Spurfunktion (in Verilog) wird für jedes zentrale Pad aufgerufen. Sie ist aus dem Spurmuster in Abbildung 4.23 abgeleitet. Da der Kreuzungspunkt der Strahlen in dieser Abbildung nicht genau auf der zweiten Ebene liegt (bedingt durch die unterschiedlich großen Pads), folgt die Bildung der Spurmuster keinem regelmäßigem Muster. Die Funktion kann aber beliebig umprogrammiert werden.

Randbereiche so, dass das entsprechende zentrale Pad wie jedes andere 15 Spuren finden kann, wobei die Zuordnung jeweils nur aus vorhandenen Pads gebildet wird, oder die lokale Umgebung liefert nur die Spuren, die vollständig mit vorhandenen Pads gebildet werden können. Für das linke Rand-Pad wären das entsprechend acht Spuren.

Es wurden beide Möglichkeiten realisiert. Je nach Anwendung kann einer der beiden Algorithmen in die FPGA programmiert werden. Im Algorithmus wurde für jedes (zentrale) Rand-Pad eine eigene Funktion geschrieben, die für die Pads der entsprechend veränderten lokalen Umgebung aufgerufen wird.

Überlappbereich der FPGAs: Im Grenzbereich zwischen den FPGAs erhalten sieben zentrale Pads jeder FPGA Signale von der anderen Kammerhälfte (siehe dazu Abbildung 4.20). Simulationen zum Grenzbereich liegen nicht vor, da immer nur die Schaltung in einer FPGA simuliert werden kann. Durch die flexible Änderung der Programme in der FPGA können die Algorithmen bei nicht einwandfreier Funktion angepasst werden.

4aus5 Option: Falls durch Defekte an der Kammer oder am Auslesesystem die Information einer Ebene nicht mehr geliefert werden kann, muss der Triggeralgorithmus unter Verwendung von vier Ebenen weiterarbeiten können. Die Anpassung an dieses

System wird *4aus5-Option* genannt.

- Fällt die mittlere Ebene aus, wird die Information aus den Pads der zweiten und vierten Ebene mit gleicher Nummerierung betrachtet. Sind beide Pads an, wird der Algorithmus normal durchlaufen, ansonsten gibt es keine Spur. Zusätzlich werden Spuren erkannt, bei dem nur das zentrale Pad fehlt. In diesem Fall wird ein ausgelöstes zentrales Pad angenommen.

- Fällt die zweite oder die vierte Ebene aus, werden diese Pads im Algorithmus nicht berücksichtigt, ansonsten wird nichts verändert.

- Fällt die erste oder fünfte Ebene aus, werden die Pads der Ebene nicht berücksichtigt und in der Ebene darüber (darunter) werden keine Pads mehr ver*oder*t wie beim ursprünglichen Algorithmus. Es muss das Pad dieser Ebene ausgelöst werden, das beim Einzeichnen des Spurmusters dem ausgefallenen Pad am nächsten ist.

- fallen zwei Ebenen aus (3aus5-Option), dürfen das nur die Ebenen *zwei und vier, eins und zwei* oder *vier und fünf* sein. Ansonsten muss der Algorithmus vollständig überarbeitet werden.

4.4.3 Addierer

Der Kaskadenaddierer ist so programmiert, dass er mit 40 MHz und nicht mit 10 MHz wie die anderen Module arbeitet. Wie schon in Kapitel 3.2.2 gezeigt, kann der Addierer die 60 (bzw. 46) Werte der Hitlist in 6 Schritten auswerten, das Ergebnis liegt nach 150 ns vor und kann mit der folgenden 10 MHz Taktflanke, also nach 2 Bunch Crossings, ausgelesen werden.

Jede Stufe im Addierer hat ein Ausgangsregister, in das die Zwischenergebnisse geschrieben werden. Diese Register sind in der Abbildung 4.25 als *data_buf1[31..0]* bis *data_buf4[9..0]* sowie *data_pre1[5..0]* dargestellt. Auf Grund der ineinanderlaufenden Kaskadenstruktur werden sie von Stufe zu Stufe kleiner. Die Inhalte der Register werden für die Testuntersuchung des Addierers an Ausgangspins gelegt. Dadurch ändert sich das zeitliche Verhalten des Addierers nicht.

In der Abbildung ist ein 8 Bit Addierer dargestellt, bei dem die Berechnung zwei Taktzyklen länger dauert als bei den in den FPGAs endgültig verwendeten 6 Bit Addierern. Die Simulationsergebnisse sind ansonsten identisch, so dass für die Simulationen der 8 Bit Addierer verwendet werden kann. Die Simulationen haben ergeben, dass dieser zuverlässig funktioniert, das Verhalten kann in der Gesamtsimulation abgelesen werden (siehe Abb. 4.27).

4.4.4 Multiplexer

Um möglichst wenig Leitungen zum Summiersystem zu übertragen, sollen die Daten vierfach gemultiplext werden. Ein Multiplexer funktioniert genau umgekehrt wie der oben beschriebene Demultiplexer. Auch hier ist ein 4 Bit Zähler notwendig, um die Zuordnung von vier Kanälen auf vier hintereinanderfolgende Datenpakete zu realisieren.

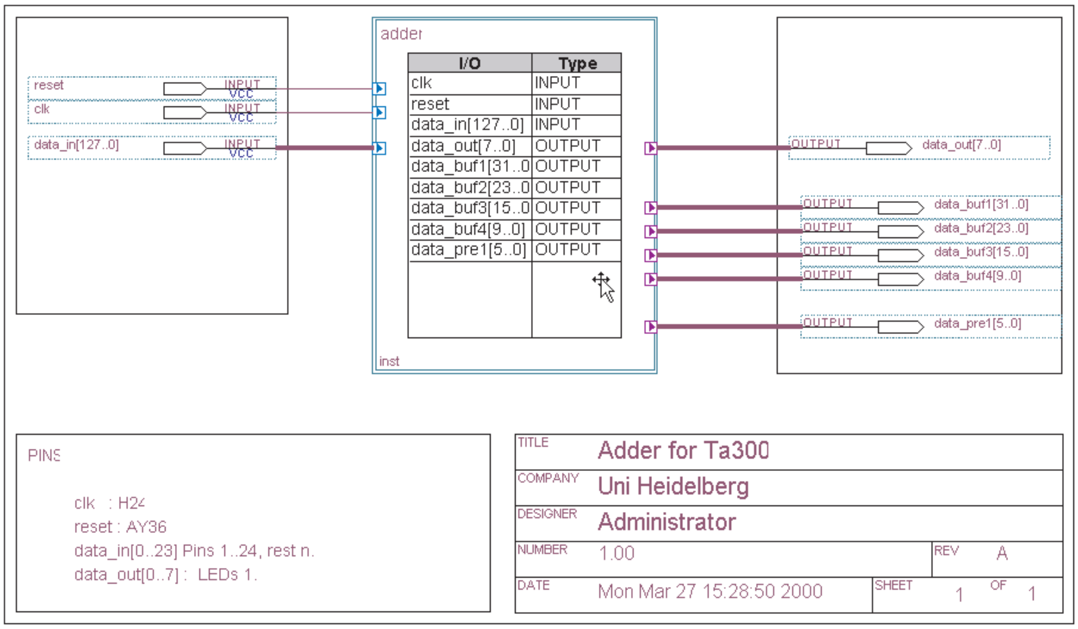


Abbildung 4.25: Blockschaltbild des Addierers, die Leitungen databuf1 - 4 sind zum Debuggen notwendig.

Es kann derselbe Zähler verwendet werden wie beim Demultiplexer, da es sich um ein zusammenhängendes System handelt, das in allen Modulen denselben Takt verwendet. Beim Multiplexer müssen vier Eingangskanäle auf einen Ausgangskanal reduziert werden, der die Daten mit der vierfachen Frequenz ausgibt. Der Zählerzustand (sel0 bis sel3) gibt jetzt die Position an, in der die Daten abfolgen (Abb. 4.26).

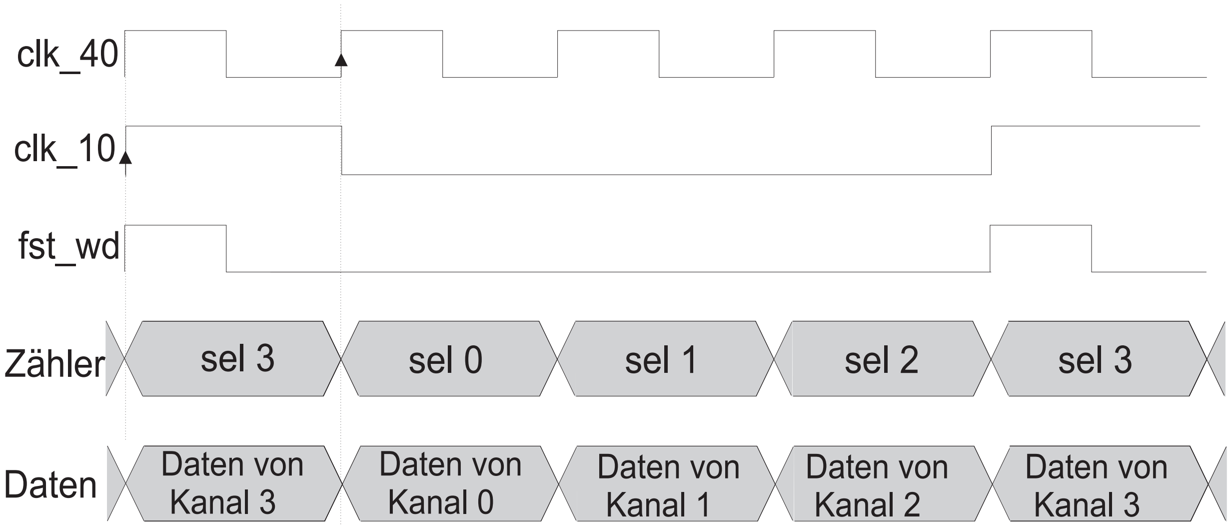


Abbildung 4.26: Zeitliches Verhalten des Multiplexes: Zeitgleich mit den Zählerzuständen werden die Daten der Kanäle 1-4 auf den Ausgangskanal gelegt.

An die erste Position werden die Daten immer zeitgleich mit dem *sel 0* Signal geschrieben. Es wurden Simulationen zum Verhalten des Multiplexers durchgeführt. Auch hier wurde das Trigger_control-Modul zum Steuern verwendet. Auf diese Simulationen wird auf Grund der starken Ähnlichkeit zu denen des Demultiplexers nicht eingegangen.

4.4.5 Simulation am Gesamtsystem

In Abbildung 4.27 ist das Verhalten des Triggeralgorithmus in Timingsimulationen dargestellt.

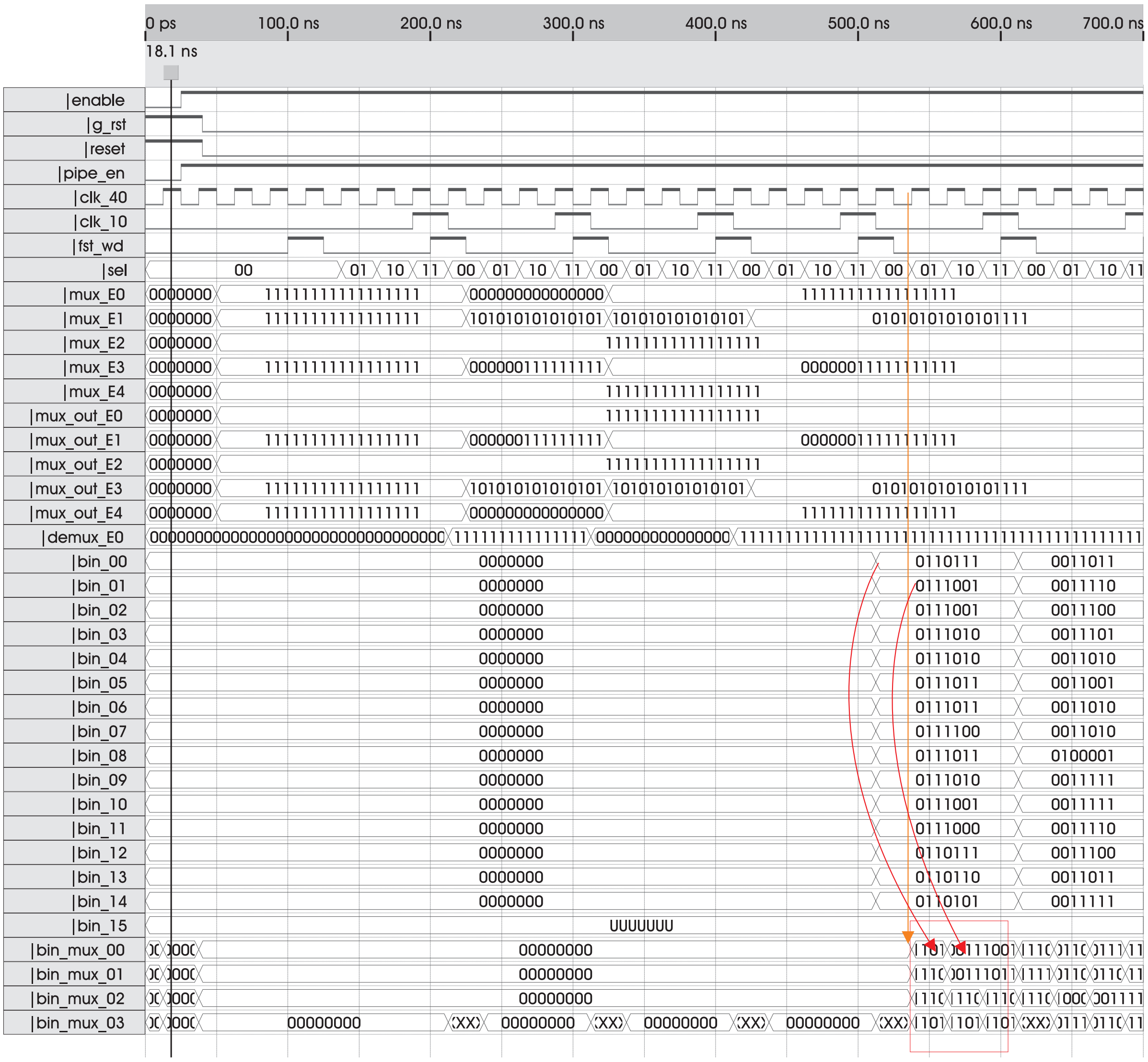


Abbildung 4.27: Logische Simulation der Gesamtschaltung (Das Laufzeitverhalten wird im Text behandelt.)

Steuersignale: Das Trigger_Control-Modul wurde vom Testalgorithmus übernommen, somit ist das Verhalten der Signale *enable, g_rst, reset, pipe_en, clk_40* und *clk_10* zu denen im TA60 identisch. Durch die nun stärker ausgelastete FPGA sind im TA300 mehr Spikes in den Schaltphasen zu erkennen (vgl. Signal "sel" in Abb. 4.13 und Abb. 4.27). Die Signale liegen später an den Ausgangspins an (t_{co} wird größer, vgl. Abschnitt 6.1.3). Im TA60 folgt die steigende clk_10 Flanke 10,5 ns nach der zugehörigen clk_40 Flanke, im TA300 nach 17 ns. Das Phasenverhältnis des *fst_wd*-Signals lässt sich beliebig umprogrammieren, in diesen Simulationen wurde es so programmiert, dass es einen Takt vor dem ersten multiplexten Datenpaket eines BCs anliegt. Zum Zeitpunk der Simulationen stand noch nicht fest, wie das im CIPix erzeugte Signal zum Zähler des Demultiplexers stehen wird.

In Abbildung 4.28 ist die selbe Simulation für mit der HERA-Clock wechselnde Daten dargestellt.

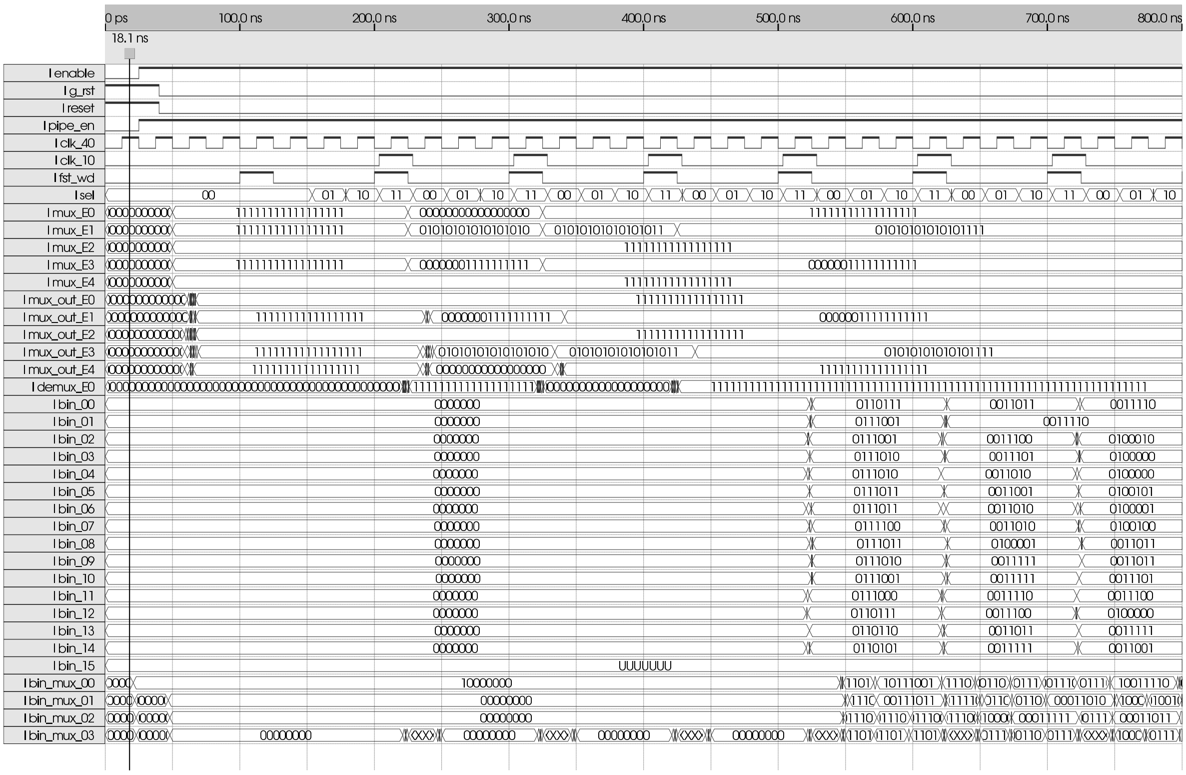


Abbildung 4.28: Gesamtsystemsimulation für sich ändernde Eingangsignale (Das Laufzeitverhalten wird im Text behandelt)

Demultiplexer: Der Demultiplexer liefert rechtzeitig zum ersten Zählerzustand jedes 10 MHz Zyklus die demultiplexten Daten des vorangehenden BCs. In Abbildung 4.27 ist zu erkennen, dass dies nach z.B. 225 ns passiert.

Spurfinder: Der Spurfinder liefert innerhalb von 100 ns die Hitlist an den Addierer. Der Wert des Ausgangsregisters des Spurfinders ist in der Simulation nicht dargestellt.

Aus vorangehenden Simulationen ist bekannt, das dieser Wert mit der dritten 10 MHz Flanke in den Addierer gelangt, hier also nach 325 ns.

Addierer: Der Addierer benötigt 6 40 MHz Zyklen, um die Hitlist auszuwerten (vgl. Abschnitt 4.4.3). Nach 200 ns bei 512,5 ns in der Simulation liegen die Anzahl der gefundenen Spuren vor.

Multiplexer: Der Multiplexer gibt die Daten des halben Histogramms in vier 8 Bit Zahlen aus. Hier ist ein Fehler im Programm des Triggeralgorithmus aufgetaucht: Der Multiplexer schreibt das Ergebnis nicht zunächst in ein Zwischenregister, welches erst mit der folgenden clk_10-Flake an den Multiplexerausgang gegeben wird (siehe Kasten bei 550 ns in Abb. 4.27), sondern übergibt die Daten direkt an den Ausgang. Dadurch sind die Daten am Ausgang jeweils um eine 40 MHz Clock verschoben. Das Multiplexerprogramm muss abgeändert werden. Da daraufhin alle Programmen neu kompiliert, simuliert und gefittet werden müssen, wird an dieser Stelle von einer Korrektur abgesehen.

Kapitel 5

Die endgültigen Systemkomponenten

Dieses Kapitel enthält eine Übersicht über die endgültigen Komponenten. Es dient als Dokumentation der entstandenen Schaltungen.

5.1 Die CIP-Backplane

Die Backplane wurde in der Elektronikwerkstatt der Universität Heidelberg entworfen [EW00]. Aufgabe der Backplane ist es, die Kammerdaten der Empfängerkarten so zu verteilen, dass je eine Triggerkarte die Daten eines ϕ-Sektors erhält. Jede Empfängerkarte liefert jeweils die Daten einer der fünf Ebenen von je zwei ϕ-Sektoren. Die Daten müssen also von einer *ebenenweisen* Führung in eine *sektorweise* Führung transformiert werden; hierzu ist eine aufwendige Signalführung notwendig. Über die Backplane werden zudem Signale von übergeordneten Systemen verteilt. In Abbildung 5.1 ist das Schaltbild der Backplane dargestellt.

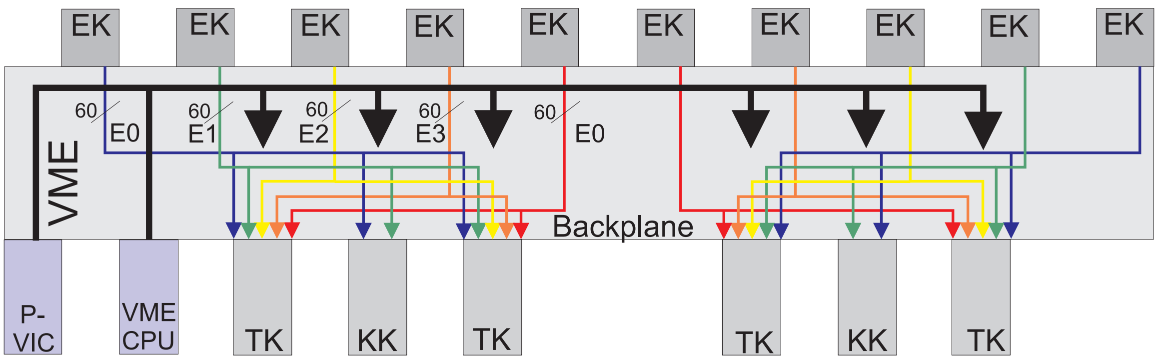


Abbildung 5.1: Signalverteilung in der Backplane. Die eingezeichneten Pfeile zeigen die Verteilung der Daten von den Empfängerkarten an die Trigger- und Kontrollkarten. Es bedeutet TK : Trigger-Karte, KK : Kontroll-Karte, EK = Empfänger-Karte.

In eine CIP-Backplanekarte werden auf der Rückseite zehn Empfängerkarten, auf der

Frontseite vier Triggerkarten und zwei Kontrollkarten eingesteckt. Die Kontrollkarten erhalten jeweils nur die Informationen der ersten beiden Ebenen E_0 und E_1. Jede Empfängerkarte gibt die Informationen einer Ebene an zwei Triggerkarten weiter - an die Triggerkarte gelangt also die Informationen von fünf Ebenen (fünf Pfeile) der beiden ϕ-Sektoren.

Über die Backplane werden die Versorgungsspannungen 2,5 V, -2,5 V, 3,3 V, 5 V und -5 V an die einzelnen Karten verteilt. Zudem enthält die Backplane die VME-Bus-Leitungen des erweiterten 32 Bit-VME-Busses.

Die Backplane enthält keine aktiven elektronischen Schaltungen. Im Anhang C sind die Schaltpläne, in Anhang B die Steckerbelegungen abgebildet.

5.2 Die Triggerkarte

Die Triggerkarte wurde ebenfalls in der Elektronikwerkstatt der Universität Heidelberg entwickelt. Es wird nun auf die Komponenten und Anschlüsse der Triggerkarten eingegangen. Diese sind schematisch in Abbildung 5.2 dargestellt. Im Anhang C.1 befindet sich der Bestückungsplan, in dem die Komponenten und Anschlüsse wiedergefunden werden können.

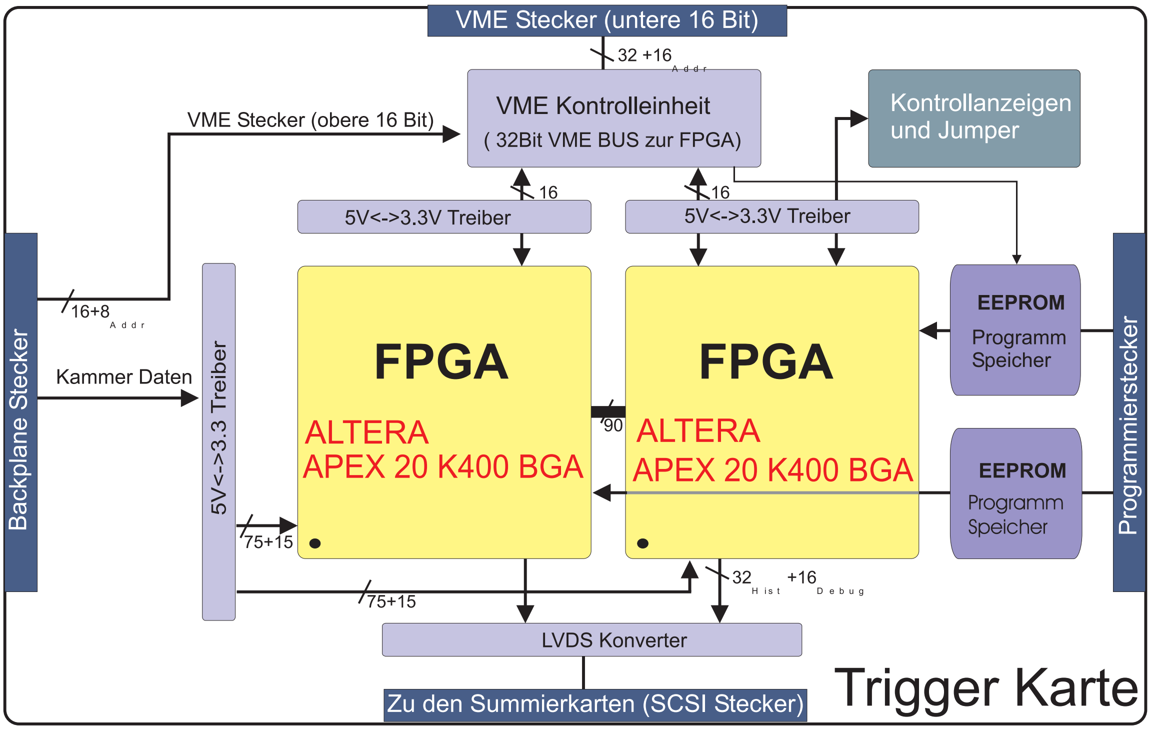


Abbildung 5.2: Schematische Darstellung der Triggerkarten (siehe Text)

5.2.0.1 Funktion der Komponenten der Triggerkarte

FPGAs Die verwendeten Altera APEX 20K400GCC/3 FPGAs werden auf der Triggerkarte in einem *Ball-Bonding*-Verfahren angebracht. Die FPGAs haben 652 Pins, von denen 502 programmierbar sind. Die anderen 150 Anschlüsse sind Masseverbindungen, Stromversorgungsanschlüsse und Leitungen, mit denen die FPGA eingestellt wird. Die Belegung dieser Anschlüsse ist in der Betriebsanleitung der FPGA beschrieben [Al99]. Die gewählten Einstellungen sind dem Schaltplan im Anhang zu entnehmen. Aus Abbildung 5.2 geht hervor, wie und mit wievielen Leitungen die FPGAs mit den anderen Systemkomponenten verbunden sind.

VME-Bus-Controller Auf den Triggerkarten ist ein VME-Bus-Controller, der die Datenverbindung mit dem VME-Bus sicherstellt. Jede FPGA hat auf alle 32 Datenleitungen des VME-Busses Zugriff und kann über einen 16 Bit breiten "Adressenraum" verfügen. Nur für die Übertragung der in den FPGAs gespeicherten Daten an die VME-CPU werden für die erste FPGA die unteren 16 und für die zweite FPGA die oberen 16 Bit des Busses fest zugewiesen. Der VME-Bus-Controller ist in einen Lattice *ispL*-2048 programmiert [La96], [EW00].

Signal-Treiber Bevor die Kammerdaten in die FPGAs gelangen, werden sie von 3,3 V-Signal-Verstärkern (Treibern) des Typs $Cy - 74LVCH16240$ [CY00] auf einen eindeutigen Pegel gesetzt, da die Eingangssignaltoleranz der APEX-FPGAs nach Angaben von Altera [So99] sehr gering ist. Alle Signale der Backplane, des VME-Bus-Steckers und des Programmiersteckers werden mit diesen *Treibern* verstärkt.

LVDS-Konverter Um die Histogramminformation mit 40 MHz an die Summierkarte zu übertragen, wird sie in **L**ow **V**oltage **D**ifferential **S**ignals (LVDS[1]) konvertiert. Da es sich bei diesen um Differenzsignale handelt, werden doppelt soviele Leitungen wie zuvor und deswegen sehr große Stecker benötigt.

Kontrollanzeigen und Steckpins Auf der Triggerkarte befinden sich LEDs, die Fehlfunktionen am laufenden System anzeigen. Zusätzlich sind Steckpins auf der Platine enthalten, über die Signale von den FPGAs ausgelesen und eingespielt werden können (siehe Schaltplan 7/7 der Triggerkarte in Anhang **??**). Die LEDs und Pins sind frei programmierbar.

EEPROMs Mit den EEPROMs soll eine schnelle Programmierung der FPGAs ermög-licht werden. Das Programm wird einmalig in den EEPROMs gespeichert; sie übergeben es dann selbständig an die FPGAs, z.B. wenn durch einen Powerglitch deren Programmierung verloren geht. Vor allem bei einem Neustart des Systems, der durch das globale Reset-Signal ausgelöst wird, übertragen die EEPROMs die Programmdaten automatisch in die FPGAs.

5.2.0.2 Anschlüsse

Folgende Anschlüsse stehen auf der Triggerkarte zur Verfügung:

[1]Diese Signale haben die Differenzpegel +2V und -2V.

Backplane-Stecker Über den CIP-Backplane-Stecker werden:

- die Kammerdaten eines ϕ-Sektors in die Triggerkarte geführt.
- alle Steuer- und Clockleitungen zugeführt.
- die oberen 16 Bit des VME-Busses in die Triggerkarte gebracht.

Die Belegung des Backplane-Steckers ist im Anhang B.7 aufgeführt.

VME-Bus-Stecker Über den VME-Bus-Stecker werden:

- die gespeicherten Kammerdaten in die VME-CPU transportiert, wo sie komprimiert und gespeichert werden.
- zudem Testmuster in die FPGAs eingespielt, mit denen das Triggersystem getestet werden kann.
- Programmierung und Überwachung der FPGAs erfolgen. Dazu notwendige Systeme sind noch nicht entwickelt worden.

SCSI-Stecker Es gibt zwei SCSI-Stecker auf der Triggerkarte:

- Der 68-polige Stecker dient dem Transport des Histogramms zur Summierkarte.
- Der 50-polige Stecker wird für Debugzwecke verwendet, hier können externe Patterngeneratoren oder Messgeräte angeschlossen werden.

Programmierstecker Über den Programmierstecker werden die EEPROMs oder die FPGAs direkt programmiert.

5.3 Die Kontrollkarte

5.3.1 Aufgabe der Kontrollkarte

Die Kontrollkarte hat im wesentlichen vier Aufgaben:

1. Sie soll das Phasenverhältnis der Clock synchronisieren, damit sichergestellt ist, dass alle fünf Empfängerkarten die Daten gleicher BCs gleichzeitig an die Triggerkarten legen.
2. Sie soll alle "wichtigen" Steuerleitungen überwachen. Die Funktion der überwachten Steuerleitungen wurde in Abschnitt 4.2.2 beschrieben.
3. Sie soll die Kammerdaten der neuen CIP-Kammer so aufbereiten und an das alte z-Vtx-Triggersystem übergeben, dass dieses mit den Daten eine Triggerentscheidung finden kann.
4. Sie soll aus den *Empty_word*-Signalen, die der CIPix liefert, einen *Cosmics*-Trigger zusammensetzen.

Anhand der schematischen Darstellung in Abb.5.3 erfolgt eine knappe Beschreibung der Komponenten des Kontrollmoduls.

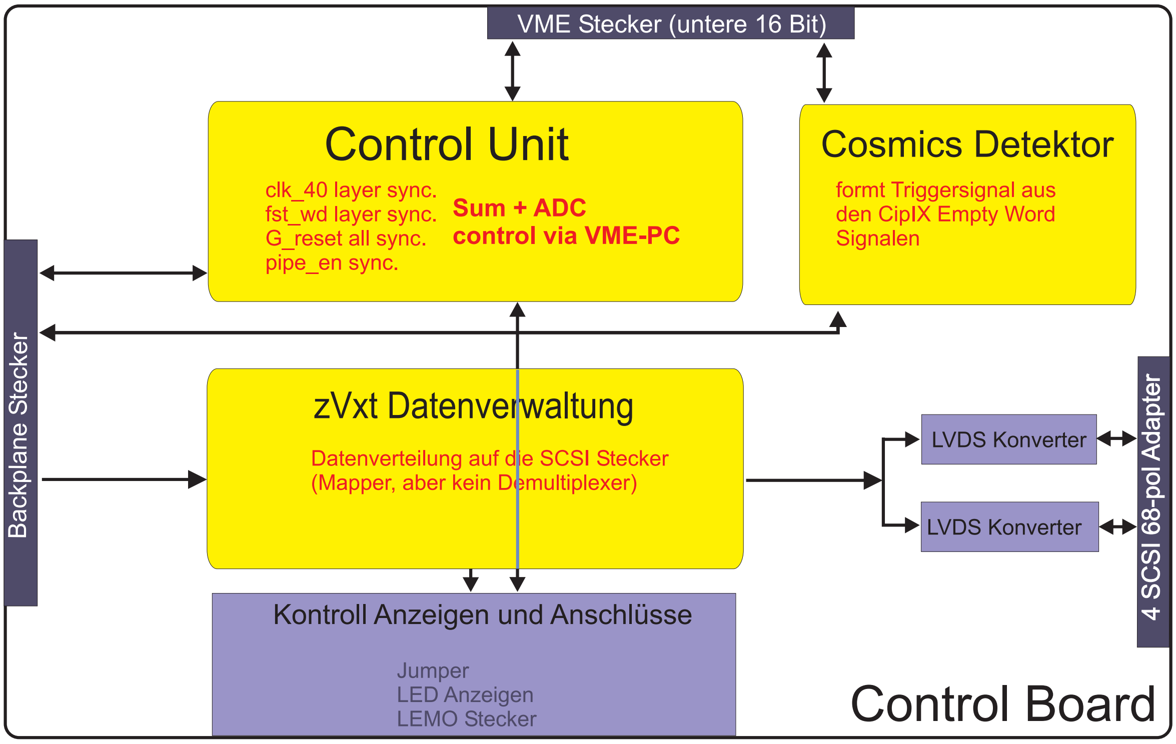


Abbildung 5.3: Funktionale Beschreibung des Kontrollmoduls

5.3.1.1 Funktion der Komponenten der Kontrollkarte

zVtx-Datenverwaltung Die Kammerdaten, die die Kontrollkarte über den Backplanestecker erhält, sollen an den alten zVtx-Trigger weitergeleitet werden. Dazu werden sie in LVDS konvertiert und auf vier SCSI-Stecker verteilt. Die Verteilung erfolgt so, dass vier in z-Richtung unterteilte Bereiche entstehen.

Das alte Triggersystem benötigt nur die Daten der ersten zwei Ebenen. Jede Kontrollkarte leitet die Daten von zwei ϕ-Sektoren weiter. Da das alte Triggersystem nur einen Teil der Daten benötigt (an den Rändern fallen je etwa 10% weg), werden am linken und rechten Rand jeweils 4 Padkanäle pro Ebene pro Sektor weggelassen.

Es ergibt sich folgende Rechnung:

- 56 Datenleitungen:

$$Daten_{Stecker} = \frac{\frac{120_{PadsproEbene} \cdot 2_{Ebenen} \cdot 2_{\phi-Sektoren}}{4_{mux}} - 8_{weggefallen}}{4_{Stecker}} \cdot 2_{LVDS} = 56 \quad (5.1)$$

- 8 Steuerleitungen: clk_40, fst_wd (first word), g_rst (globales reset) und pipe_en (pipe enable). Diese Leitungen werden auch jeweils in LVDS-Signale konvertiert, also ergeben sich 8 Leitungen.

- 4 Ground-Leitungen

Insgesamt werden 68 Signale pro Stecker und damit 272 Signale an das alte Triggersystem übertragen.

Kontrolleinheit Die Kontrolleinheit überwacht die folgenden Signale:

10,4 MHz-Clock: Die 10,4 MHz-Clock wird für jede CIPix-Karte getrennt regelbar von der HERA-Clock abgeleitet. Die 10,4 MHz-Signale gelangen in die Empfängerkarten und von dort in die PLLs[2] der CIPix-Karten.

41,6 MHz-Clock: Im PLL wird aus diesen Signalen die 41,6 MHz-Clock erzeugt, die für die Multiplexer im CIPix verwendet wird. Die 41,6 MHz-Clock gelangt über die Empfängerbords in die Kontrollkarten und wird dort im Regelsystem mit den 41,6 MHz-Signalen der anderen CIPix-Karten verglichen.

first word: Gleichzeitig zu den 41,6 MHz-Signalen wird das *first Word*-Signal für die Phaseninformation der multiplexten Signale erzeugt (s. auch Abschnitt 4.2.2). Das fst_wd-Signal ist per Definition phasenstarr zum 41,6 MHz-Signal, der Phasenwinkel ist nicht bekannt und muss vor Inbetriebnahme gemessen werden (s. Abbildung 5.4).

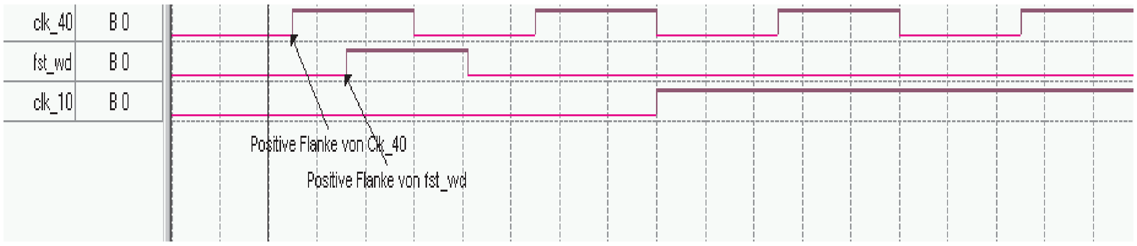


Abbildung 5.4: Das Phasenverhältnis zwischen clk_40 und fst_wd ist fest, muss aber erst bestimmt werden.

Durch die Regelkreise in der Kontrollkarte erhält man ein 41,6 MHz- und ein fst_wd-Referenzsignal, welches an die Triggerkarten und das alte zVtx-Triggersystem weitergegeben wird. In Abbildung 5.5 ist der vollständige Regelkreis aus Kontrollkarten und jeweils fünf CIPix-Karten angegeben.

Das Regelsystem besteht aus einem analogem Addierer, der die fünf 41,6 MHz-Signale addiert, integriert und anschließend digitalisiert. Durch die Integration wird vermieden, dass Spikes digitalisiert werden, die vom digitalen Regelsystem nicht richtig interpretiert werden. Nachdem das Signal digitalisiert ist, werden die 10,4 MHz-Signale über digitale Delays so geregelt, dass das digitalisierte Signal minimiert wird. In Abbildung 5.6 ist der Regelkreis schematisch dargestellt.

g_rst: Das übergeordnete globale Reset-Signal setzt das Triggersystem in den Neustart-Zustand.
Das Signal wird auf Grund der globalen Bedeutung über einen LEMO-Stecker in das System gekoppelt und ohne Veränderung auf die Backplane geführt. Bei einem Systemreset wird es gesetzt; es muss mindestens 200 ns (2 HERA-Clocks) auf /em actice high /em ("1") sein. Dadurch ist sichergestellt, dass das asynchrone Signal nicht in

[2]Phase Lock Loops, Regelkreise zur Regelung und Erzeugung zyklischer Signale

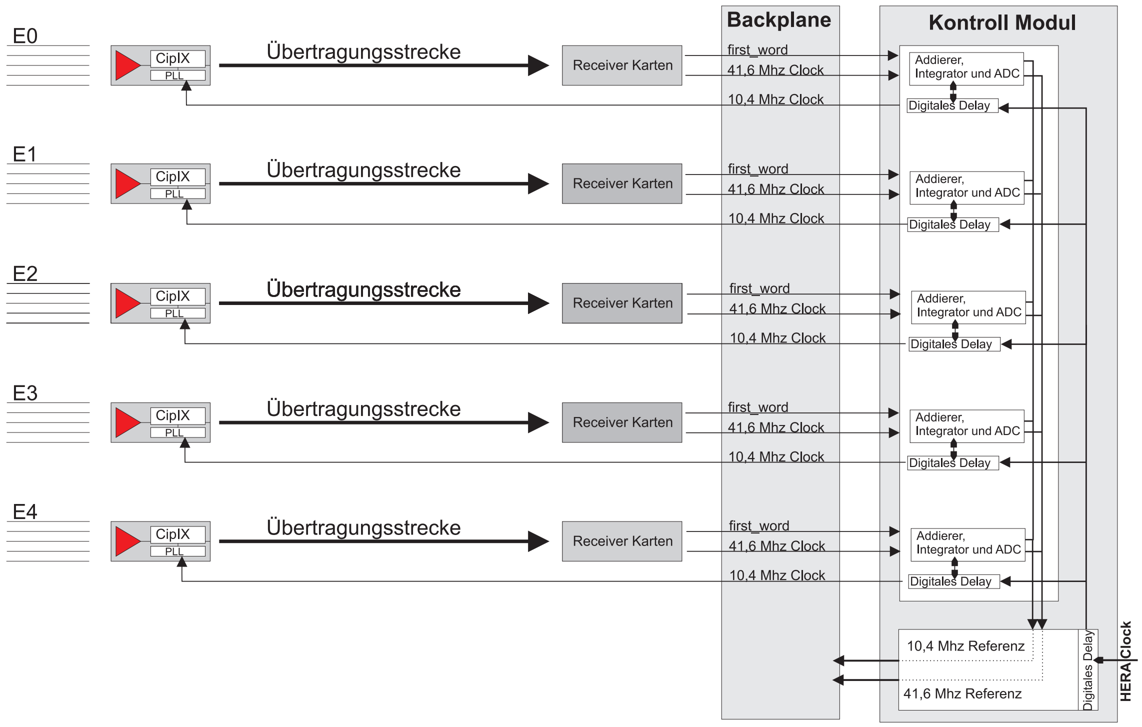


Abbildung 5.5: Der Verlauf der Clock-Signale: Die eingekoppelte HERA-Clock wird verzögert und dann in die Regeldelays geleitet. Danach gelangen die Signale an die PLLs in den CipIX-Karten. Dort entstehen die fst_wd- und clk_40-Signale, die nach der Übertragungsstrecke wieder in das Kontrollmodul geleitet werden. Damit ist der Regelkreis geschlossen.

einer Schaltphase nur einzelne Komponenten erreicht.
Um einen systeminternen Reset hervorzurufen, wird ein Taster in den Signalweg geschaltet.

5.3.1.2 Anschlüsse

Backplane-Stecker Über den Backplane-Stecker werden alle Signale des CIP-Kammer-Systems in die und aus der Kontrollkarte geführt. Im Anhang **??** ist die Steckerbelegung des Backplane-Steckers angegeben.

Insbesondere die geregelte HERA-Clock wird über die Kontrollkarte in das CIP-Kammer-System eingekoppelt. Neben den Clock-Signalen wird auch das globale Reset-Signal verteilt. Dieses Signal wird sechsmal auf die Backplane gelegt: fünf Leitungen gehen an die Empfängerstecker, eine geht an je zwei Triggerkarten.
Das Kontrollmodul gibt zudem die Signale der ersten zwei Ebenen der Kammer an den alten zVtx-Trigger weiter. Diese Signale werden auch über den Backplane-Stecker auf das Kontrollmodul geführt.

VME-Bus-Stecker Über den VME-Bus-Stecker werden Fehlfunktionen an externe Überwachungssysteme weitergeleitet. Zudem wird die Kontrollkarte über den VME-Bus gesteuert. Der VME-Stecker ist nach den VME-Konventionen beschaltet. Eine

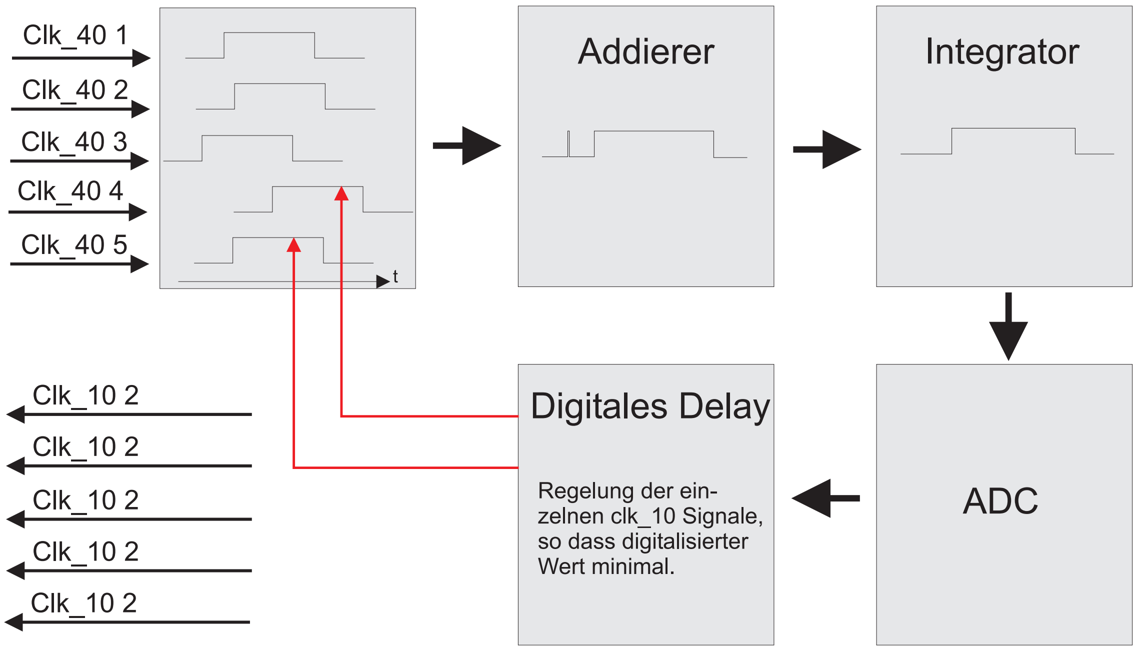


Abbildung 5.6: Schematische Darstellung des Regelkreises. Die Phase der im oberen rechten Kästchen abgebildeten Signale wird so zueinander verändert, dass die addierte "Fläche" minimal wird.

Beschreibung der Steckerbelegung ist in diesem Fall nicht notwendig [Sp87].

SCSI-Stecker Über den SCSI-Stecker werden die Daten an den alten zVtx-Trigger geleitet. Zudem werden die clock_40- und fst_wd-Signale an das alte zVtx-Kammer-Triggersystem weitergeleitet. Diese Stecker erfüllen nicht das SCSI-Protokoll.

Anschlüsse an der Frontblende An der Frontblende befinden sich LEMO-Steckanschlüsse, über die externe Signale in das CIP-Kammersystem gelangen und zudem Signale zur Kontrolle abgegriffen werden können. Vorhandene Anschlüsse an der Frontblende[3] sind folgend aufgezählt[4]:

- Hera-Clock (Eingang)
- Pipe_enable (Eingang)
- globales Reset-Signal (Eingang)
- pipe_enable (Ausgang)
- vom Cosmics Trigger erzeugte Triggersignal (Ausgang)
- fünf first word-Signale (Kontrollausgänge)

[3] TTL an 50Ω

[4] Die aufgezählten Signale wurden in vorangegangenen Kapiteln beschrieben, u.a. in Abschnitt 4.2.2 und Kapitel 4.3.

- fünf 41,6 MHz-Signale (Kontrollausgänge)
- fünf 10,4 MHz-Signale (Kontrollausgänge)

Leuchtdioden: Alle erzeugten Signale werden über LEDs angezeigt. Dadurch lässt sich die Funktion des Systems schnell überprüfen. Es folgt eine Aufzählung der LEDs:

- 20 Empty_word-Signale (20 LEDs)
- Triggersignal aus dem Cosmics Detektor (eine LED)
- die HERA-Clock (eine LED)
- die 41,6 MHz-Clock (eine LED)
- die 10,4 MHz-Clock (eine LED)
- fst_wd (eine LED)
- pipe_enable (eine LED)

Taster und Schalter: Mit einem Reset-Taster kann man einen Hardwarereset des Systems durchführen; dieser Taster hat die gleiche Priorität wie der globale Reset.

Kontrollanzeigen

LEDs Die oben aufgeführten LEDs werden durch Mono-Flops etwa 5 ms auf *active high* gehalten, das Signal wird vor dem Anzeigen integriert. Dadurch ist sichergestellt, dass beispielsweise eine auf high stehen gebliebene Clock nicht durch eine leuchtende LED angezeigt werden kann. Zudem können beliebige Signale direkt in die WatchDog-Schaltung geführt werden.

WatchDog Die fst_wd-Signale werden über eine WatchDog-Schaltung ausgelesen. Falls ein externes Signal durch eine Fehlfunktion ausfällt, wird das über den VME-Bus mitgeteilt. Der WatchDog wird in eine Lattice-FPGA programmiert.

LVDS-Konverter Die LVDS-Konverter dienen der Umwandlung des 3,3V-Signals aus den Empfängerkarten in ein Differenzsignal. Dadurch ist sichergestellt, dass die Signale ohne Veränderungen am alten Triggersystem ankommen.

5.3.2 Bau der Kontrollkarte

Der Bau der Kontrollmoduls wurde in der Elektronikwerkstatt der Universität Heidelberg entwickelt. Die Kontrollkarte besteht aus einer Hauptplatine und zwei Steckkarten (Piggy Bags), die oben und unten in die Hauptkarte gesteckt werden. Die Karte erstreckt sich über drei VME-Bus-Slots. In den Steckkarten befinden sich die SCSI-Stecker und die Leuchtdioden, in der zentralen Karte befinden sich die LEMO-Steckbuchsen. Die Schaltpläne zur Kontrollkarte sind sich im Anhang abgebildet.

Insgesamt werden 10 Kontrollkarten gebaut; sie werden vermutlich Anfang Juni fertiggestellt sein.

5.4 Die Summierkarten

Die Funktion der Summierkarten wurde in Abschnitt 3.2.3 geschildert. Es erfolgt nun eine knappe Beschreibung der Anforderungen und Realisierungsansätze. Die Karten werden vermutlich im *September 2000* gebaut.

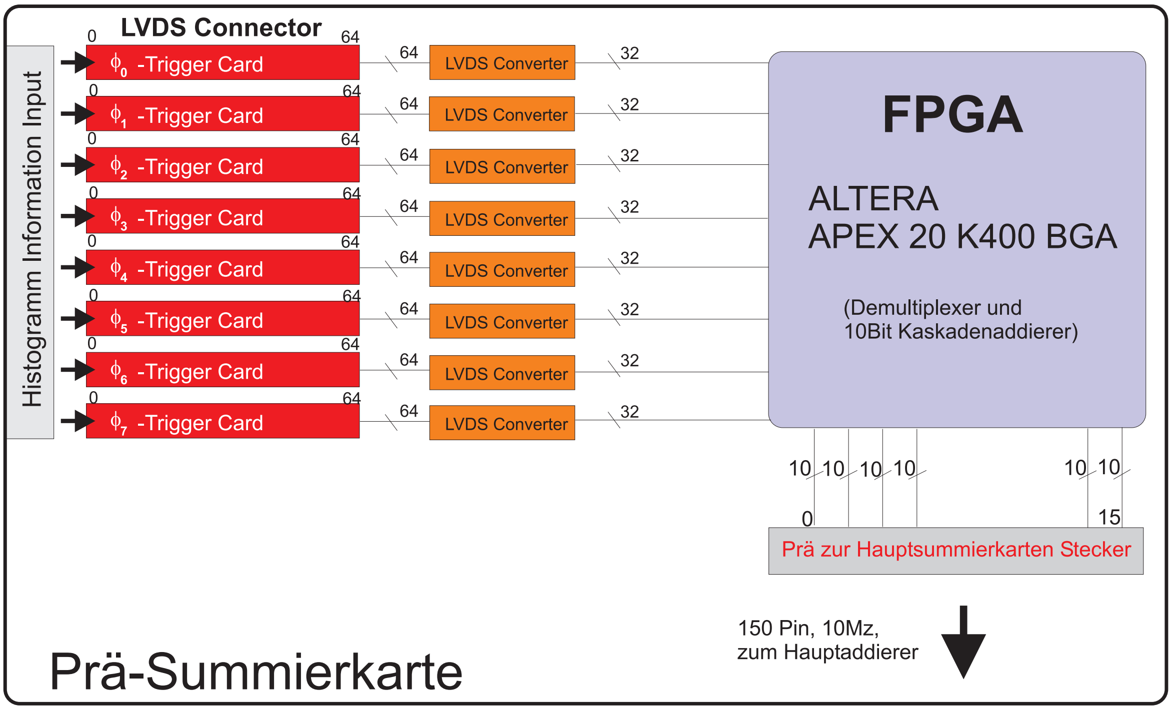


Abbildung 5.7: Die Prä-Summierkarte enthält eine FPGA, die die Daten von acht ϕ-Sektoren auswertet und an die Hauptsummierkarte weitergibt

Prä-Summierkarten: Es gibt zwei Prä-Summierkarten, die jeweils die Information von acht ϕ-Sektortriggerkarten zusammenaddieren. Jede Karte enthält acht 68-polige SCSI-Stecker, über die das ϕ-Sektorhistogramm in die Summierkarten gelangt. Die Daten werden von den Triggerkarten als vierfach gemultiplexte LVDS-Signale übergeben. Der Addierer wird so programmiert, dass er die multiplexten Signale direkt addiert.

Diese Additionen erfolgen in einem weiteren Altera APEX-Baustein. Das Ergebnis der Addition ist ein Histogramm über acht ϕ-Sektoren. Es besteht aus 15 jeweils 10 Bit breiten Zahlen und wird mit 10 MHz an die zentrale Summierkarte übertragen. Der schematische Entwurf der Prä-Summierkarte ist in Abbildung 5.7 dargestellt.

Hauptsummierkarten: In der Hauptsummierkarte wird aus den zwei "halben" Histogrammen das Gesamthistogramm gebildet. Die Triggerentscheidung wird aus den nun 11 Bit breiten Werten der 15 Binbereiche gebildet (siehe Kapitel 3.2.3). Beim Aufaddieren der ϕ-Histogramme geht die Information über ϑ verloren. Über noch freie Leitungen kann aber für jedes ϕ-Histogramm die Entscheidung von den Triggerkarten an die Hauptsummierkarte übertragen und ausgewertet werden.

Die Triggerentscheidung wird als 16 Bit breites Triggerwort an die zentrale Triggerkontrolle (CTC) weitergegeben. Zuden kann das Gesamthistogramm ausgelesen werden, welches sich aus den 15 jeweils 11 Bit breiten Binbereichsinformationen zusammensetzt.

Es ist geplant, auch für die Summierkarten APEX FPGAs zu verwenden, da sowohl Entwicklungsumgebung als auch die Hardware vertraut sind.

In Abbildung 5.8 ist die schematische Schaltung dargestellt.

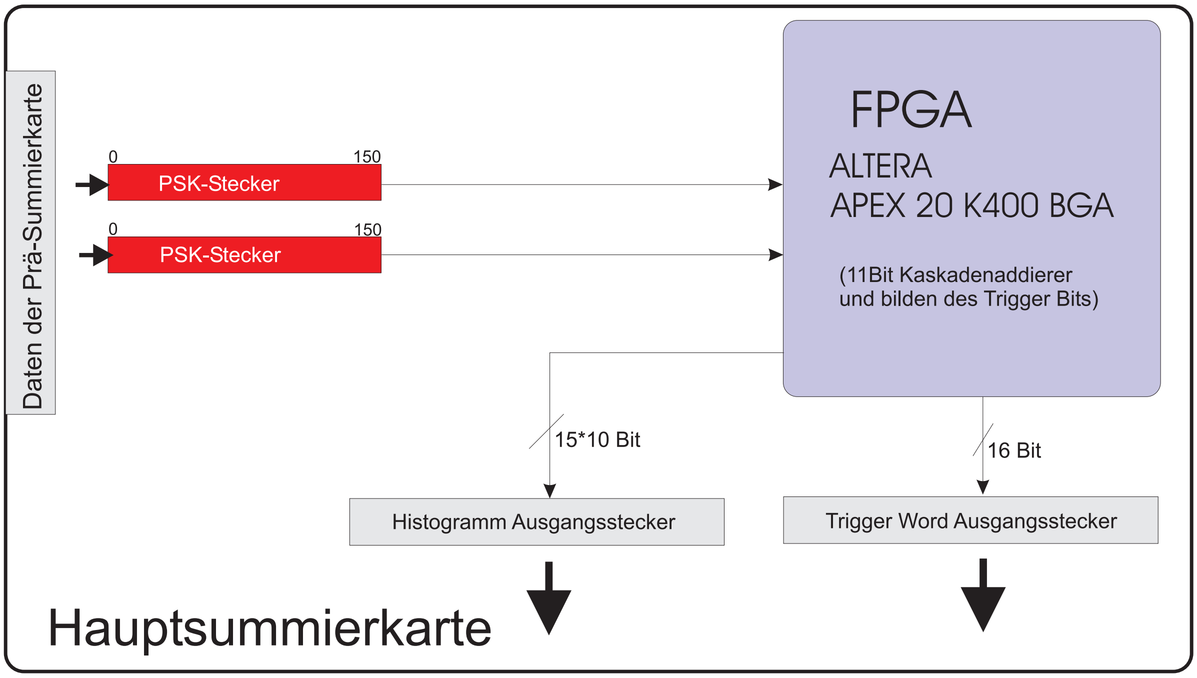


Abbildung 5.8: Die zentrale Summierkarten werten die Daten aller Sektoren aus und treffen die Triggerentscheidung.

Kapitel 6

Messungen und Hardwaretests

Während der Entwicklung des Triggersystems wurden drei große Messreihen durchgeführt:

- Messungen zur grundsätzlichen Funktion von FPGAs am Testsystem
- Messungen mit dem Patterngenerator am Testsystem
- der globale Systemtest mit Kammer, Auslesesystem und endgültigen Triggerkarten

6.1 Bau und Messungen am Testsystem

6.1.1 Bau des Testsystems

In der Elektronikwerkstatt der Universität Heidelberg wurde im Juli 1999 ein Testsystem entworfen [Ra00], mit dem der Einsatz von FPGAs für die Triggerelektronik geprüft werden sollte. Bevor die Simulationen mit den Algorithmen abgeschlossen waren, konnten so Simulationen direkt durch Messungen überprüft werden.

Das Testsystem besteht aus einem VME-Crate, in das ein Testboard[1] gesteckt wird. Es enthält eine Altera FPGA des Typs APEX 20K-400 (siehe auch Abb. B.1), die erst seit Juli 1999 erhältlich ist.

Features des Testsystems:

- Die FPGA wird als Pin Grid Array (PGA) mit 655 Pins geliefert und in einen Nullforce-Sockel[2] der Firma Yamaichi Electronics gesetzt, der speziell für den APEX entworfen wurde. Die Anzahl der Pins der FPGA hat alles vorher da gewesene übertroffen.
- Das Testboard verwendet 80 der 502 frei programmierbare Leitungen, über die die Signale von einem Patterngenerator oder die Kammersignale in den Chip

[1] Der Begriff Testboard hat sich sehr schnell gefestigt und soll aus diesem Grund beibehalten werden.
[2] Ein Nullforce-Sockel ermöglicht beliebigen Ein- und Ausbau der FPGA ohne mechanischen Kraftaufwand.

geführt oder ausgelesen werden. Für funktionale Tests können Ausgänge an LED-Anzeigen gelegt werden.

- Auf dem Board ist eine Downloadeinrichtung, die das Laden der Programmdaten über das *Altera Masterblasterkabel* mit einem USB-Stecker vom PC ermöglicht. Die Programmierung der FPGAs wird durch Abschalten der Spannung gelöscht und muss dann neu heruntergeladen werden. Auf der endgültigen Triggerkarte werden EEPROMs verwendet, über die das Programm in die FPGAs gelangt.
- Als Schnittstelle mit anderen Systemen ist ein 16 Bit-VME-Bus-Controller gedacht, der das Lesen und Schreiben externer Daten ermöglicht.
- Ein auf dem Board befindlicher Clockgenerator kann beliebige Clocksignale von 1-100 MHz erzeugen. Für Clocks im Bereich einiger Hz existieren Eingänge.

In Abbildung 6.1 ist der Altera APEX 20K400-PGA abgebildet.

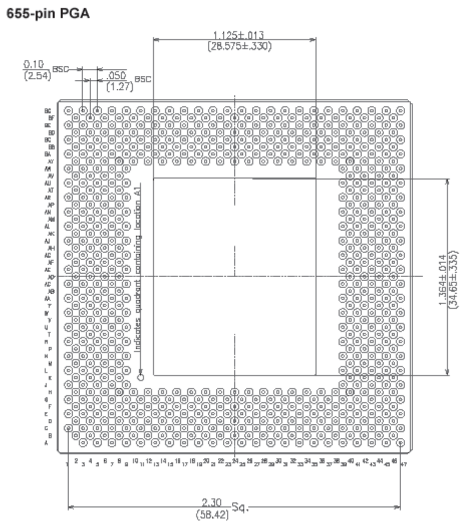


Abbildung 6.1: Die 655 Pins des Altera APEX 20K400-PGA

6.1.2 Erste Schritte

Nachdem das Testboard vom Hersteller geliefert wurde, wurde es schrittweise bestückt und in Betrieb genommen. Zuerst wurden der VME-Bus-Controller und die internen Clockgeneratoren in Betrieb genommen und getestet. Beide Einheiten sind in eine FPGA vom Typ *Lattice Ispl 1024* [La96] programmiert. Danach wurde die APEX FPGA programmiert.

Es gibt zwei Programmiermodi, einen *passiv-serial-Modus* (PS-Mode), in dem man nur den Chip programmiert, und einen *JTAG*[3]-Modus, in dem man den Chip programmieren und testen kann. Die FPGAs werden im PS-Mode programmiert. Dazu werden die Daten vom PC byteseriell (bitparallel) in den SRAM (statischer Speicher)

[3] Join Testing Action Group

des Chips geladen und weiter über ein chipinternes Protokoll in die Makrozellen (LABs) programmiert. In Abbildung 6.2 ist die Beschaltung der FPGAs mit dem Downloadstecker beschrieben.

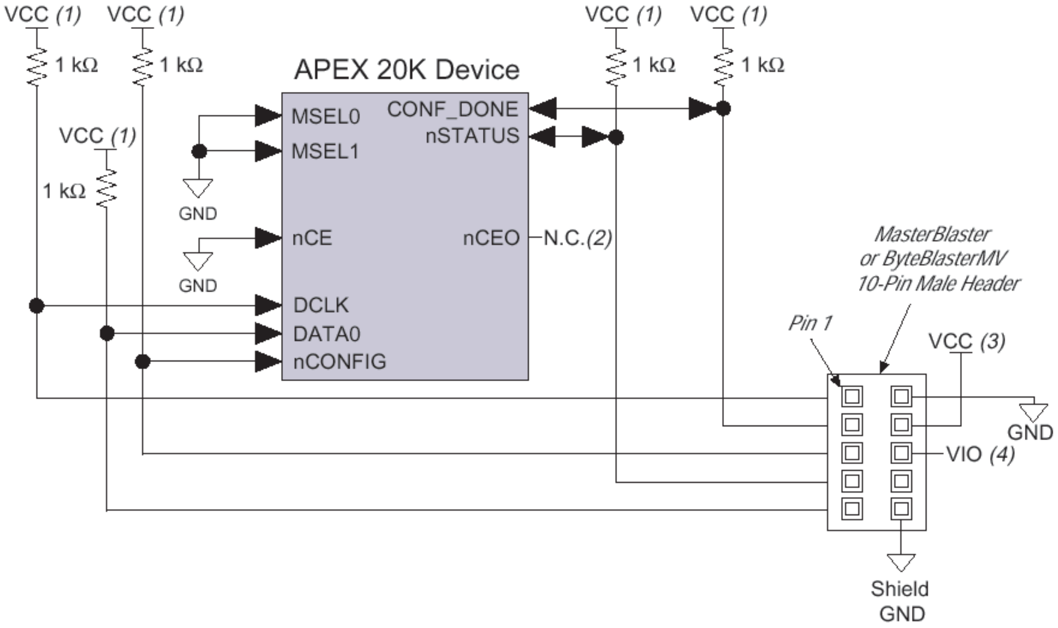


Abbildung 6.2: Schaltplan zu *Download* -Prozedur [AN120]

Im Abbildung 6.3 ist das Timingdiagramm abgebildet, mit dem ein Datentransfer initialisiert wird:

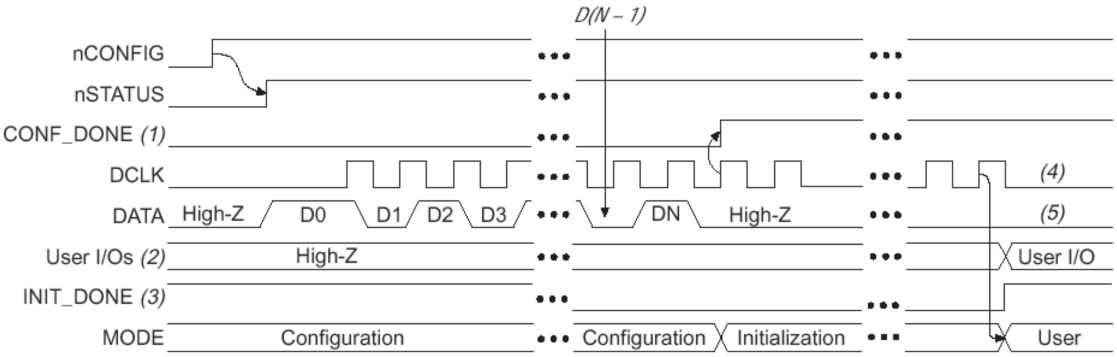


Abbildung 6.3: Timing-Diagramm der *Download* -Prozedur [AN120]

Programmieren der FPGAs: Wird das *nConfig*-Bit des Chips gesetzt, schaltet er in den Programmiermodus. Ist der Programmiermodus erreicht, wird das *nStatus*-Bit gesetzt, und der Datentransfer in die SRAMS beginnt. Das erste Datenbit startet einen

Taktgeber (DCLK). Sind alle Daten in die SRAMs transferiert, wird das *Conf_done*-Bit gesetzt. Die FPGA schaltet vom Configuration-Mode in den Initialisation-Mode. Die Daten werden aus dem SRAM in die Makrozellen geladen. Dieses geschieht taktsynchron zur DCLK. Sind alle Daten aus den SRAMs in die LABs geladen, wird das *Init_Done*-Bit gesetzt, und die Schaltung in der FPGA ist betriebsbereit.

6.1.3 Versuche am Testboard

Die simulierten Schaltungen des Testalgorithmus wurden in den APEX programmiert. Der Testaufbau ist in Abbildung 6.4 dargestellt.

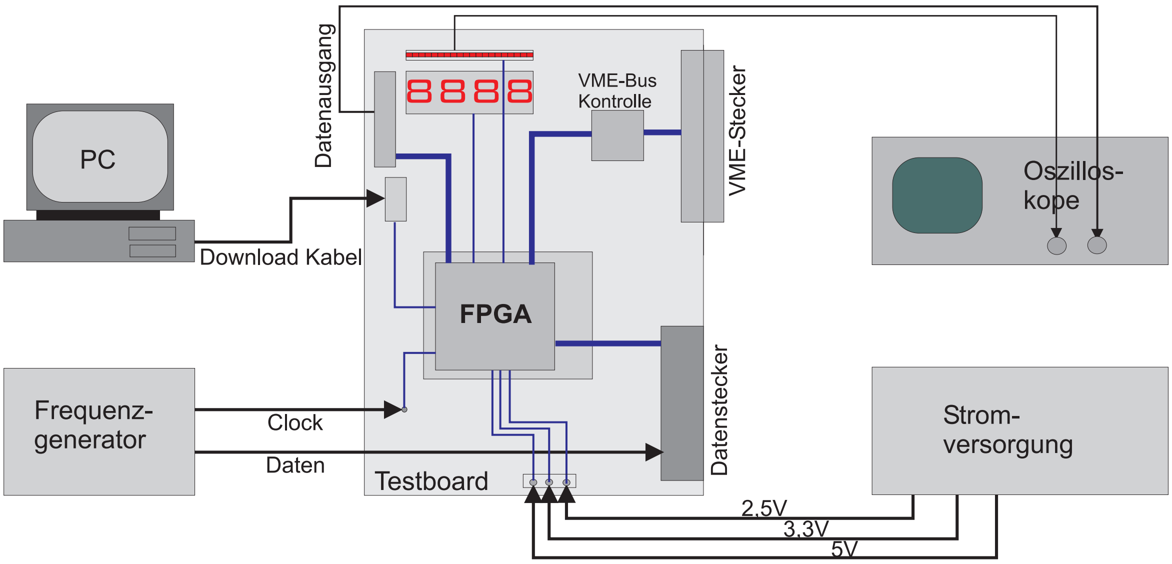


Abbildung 6.4: Schema des Testaufbaus: Im Testboard mit der FPGA laufen alle Daten zusammen. Die Programmierdaten kommen vom PC. Über eine externe Stromversorgung kann das Testboard unabhängig vom Crate betrieben werden. Mit einem Oszilloskop können alle Pins der FPGA gemessen werden.

Zunächst wurde untersucht, wie sich das Laufzeitverhalten einer Schaltung ändert, wenn man die Signalausgänge unterschiedlichen Pins zuordnet. Zum Zeitpunkt der Messung waren keine Informationen zum "Pin Assignment" veröffentlicht. Die Messung fand mit einer Taktfrequenz von 40 MHz statt. Es hat sich gezeigt, dass die Laufzeit nahezu unabhängig von der Zuordnung der Pins ist (Tabelle 6.1).

Als erste Schaltung wurde das Trigger_control-Modul in die FPGA programmiert. Die Schaltung wurde zunächst mit einem Takt von 1Hz überprüft; dafür wurden die Ausgänge an LEDs gelegt. Zur Erzeugung der Taktsignale wurde eine selbstentwickelte Schaltung verwendet, die Clocksignale in einem Bereich von 0,1 Hz bis 10 kHz generieren kann. Das Verhalten der Simulation in Abbildung 4.8 wurde bestätigt.

Nachdem die Schaltung mit 1 Hz funktionierte, wurde in einer Messreihe die Frequenz stufenweise bis auf 200 MHz erhöht und jeweils die zwei Bits des Vierzustandszählers (00, 01, 10,11 aus dem Trigger_control Modul) im Verhältnis zur Clock gemessen. In

	Clock to output-Verzögerung (t_{CO}) in ns
Pin A5	8,4 ± 0,5
Pin BB22	8,8 ± 0,5
Pin AY36	8,1 ± 0,5
Pin A17	7,9 ± 0,5

Tabelle 6.1: Messung von der Laufzeit für verschiedene Zuordnung der Ausgangspins

Tabelle 6.2 wird die Signalverzögerung zwischen Taktflanke und gemessener Signalflanke am Ausgang angegeben. Der Wert wird auf die Taktfrequenz normiert (10 MHz ⇒ 100 ns, hier kam die Signalflanke 30 ns später ⇒ 30%). Bei höheren Taktfrequenzen wird die Signalverzögerung am Ausgang größer.

Frequenz in Hz	Signalverzögerung in % des Clockzyklusses
1	–
100	≤ 10 ± 5
1k	10 ± 5
10k	10 ± 5
100k	10 ± 5
1M	20 ± 5
10M	30 ± 5
100M	35 ± 5
166M	50 ± 5
200M	–

Tabelle 6.2: Messung der Signalverzögerung zwischen Signalausgang und Taktgeber für verschiedene Taktfrequenzen

In einer weiteren Messung sollte das Laufzeitverhalten der FPGA bei verschiedener Auslastung der LABs gemessen werden. Es wurden unterschiedliche Programme verwendet, deren Funktion nicht beachtet wurde. In den Programmen Wurde das Eingangssignal clk_40 auf einen Ausgangspin gelegt und die Zeit zwischen Eingangssignal und Signal am Ausgang gemessen.
Nach diesen Messungen wurde der Demultiplexer getestet. Für diese Tests wurden zwei Frequenzgeneratoren benötigt. Einer lieferte die Clock, der andere ein Signal, das demultiplext werden sollte. Es gab nicht die Möglichkeit, Signale mit einem frei einstellbaren Muster zu erzeugen, zudem konnte nur jeweils ein Kanal demultiplext werden. Jeder weitere Kanal hätte einen weiteren Frequenzgenerator erfordert.

Über eine externe Verzögerungsschaltung kann das Phasenverhältnis zwischen Taktgeber und Datensignal abgestimmt werden. Der Frequenzgenerator, der die Daten generiert, liefert ein Signal mit der halben Taktfrequenz, so dass an den ersten beiden Ausgängen "1", an den anderen beiden Ausgängen "0" erwartet wird. Der Reset erfolgt mit einem Taster.

Am Ausgang der FPGA liegen die demultiplexten Signale, die mit dem Oszilloskop

Auslastung in %	Signalverzögerung in ns
≤ 1	7 ± 0.5
2	7.5 ± 0.5
6	8 ± 0.5
18	9 ± 0.5
29	10 ± 0.5
58	13 ± 0.5

Tabelle 6.3: Messung der Signalverzögerung zwischen Signalausgang und Taktgeber für verschiedene Auslastung der FPGA

gemessen werden können. In Abbildung 6.5 ist das Ausgangssignal der FPGA (Kanal 4) sowie das multiplexte Eingangssignal (Kanal 3), das clk_40- (Kanal 1) und das fst_wd-Signal (Kanal 2) dargestellt.

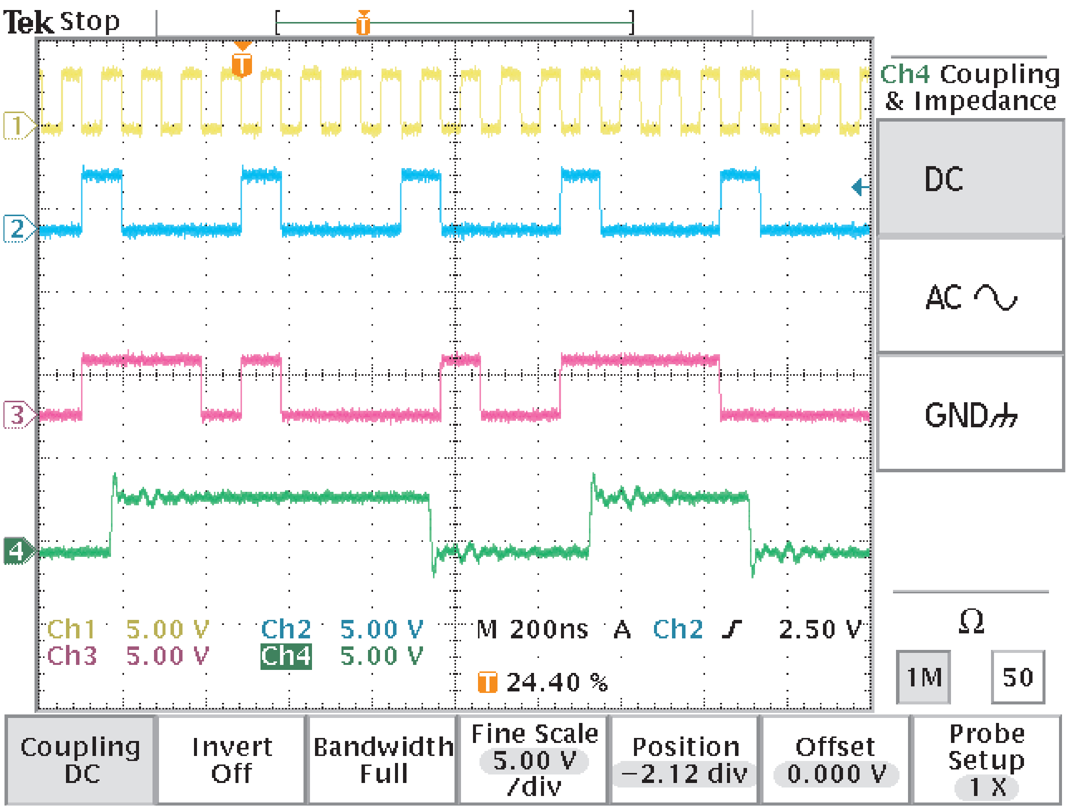


Abbildung 6.5: Gemessenes Ausgangssignal am Demultiplexer: An Kanal 1 wird das clk_40-Signal gemessen, an Kanal2 das fst_wd-Signal, an Kanal3 der Daten-Eingangskanal und an Kanal4 der Ausgang der FPGA. Hier werden die demultiplexten Daten angezeigt.

Mit diesen Messungen konnte die einwandfreie Funktion des Demultiplexermoduls nachgewiesen werden.

Nun wurden der Spurfinder und der Addierer in die Schaltung eingebracht. Um das

Verhalten der Schaltung zu überprüfen, müssen die Eingangskanäle mit Mustern aus einem Patterngenerator beschaltet werden, der zu diesem Zeitpunkt nicht zur Verfügung stand. Es konnte nur eine funktionale Überprüfung der Schaltung vorgenommen werden. Durch Taster können Kammersignale simuliert werden. Auf vier 7-Segment-Anzeigen erschien nach drei Takten das Ergebnis der Spurfindung. Im folgenden Kapitel (Abschnitt 6.2) werden die Messungen mit einem Patterngenerator beschrieben. Auf dem Foto 6.6 ist die funktionierende Schaltung im Labor abgebildet.

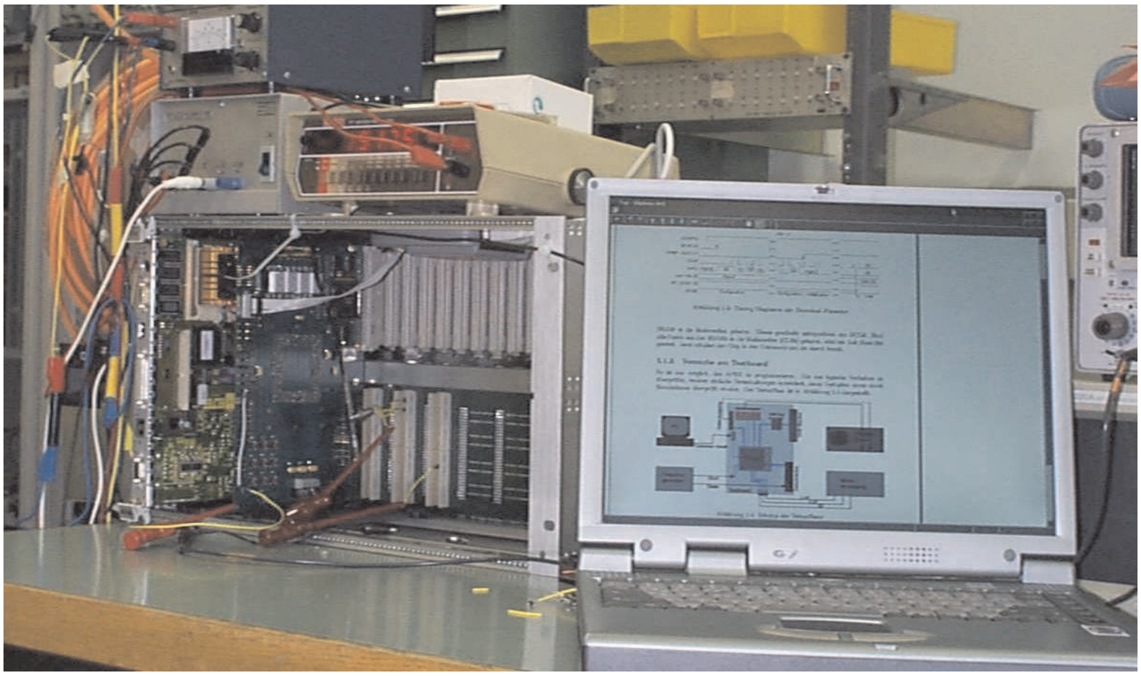


Abbildung 6.6: Der Testaufbau; das Testcrate mit dem Testboard ist im linken Teil des Fotos zu erkennen.

6.2 Systemtest mit dem Patterngenerator

Für die Testreihe wurde ein Patterngenerator vom Typ Tektronix DG2020A [Tex99] verwendet, der 24 programmierbare Muster mit 40 MHz an die Eingänge der FPGA lieferte, so dass Signale am Testalgorithmus gemessen werden konnten.

6.2.1 Messung mit dem Testalgorithmus

Über einen Adapterstecker wurden die Ausgänge des Patterngenerators mit den Eingängen der FPGAs verbunden. Je ein Kanal wurde mit den Eingängen clk_40, fst_wd, pipe_en und g_rst verbunden. 15 Kanäle waren mit den programmierten Dateneingängen der FPGA verbunden. Drei multiplexte Signale pro Ebene ergeben jeweils 12 demultiplexte Padinfomationen. Für fünf Ebenen sind das die 60 für den Testalgorithmus (TA60) benötigten PADs.

An den Ausgängen lagen die Daten des z-Histogramms; sie wurden mit einem Oszilloskop gemessen. In den Patterngenerator wurde ein Muster programmiert, das

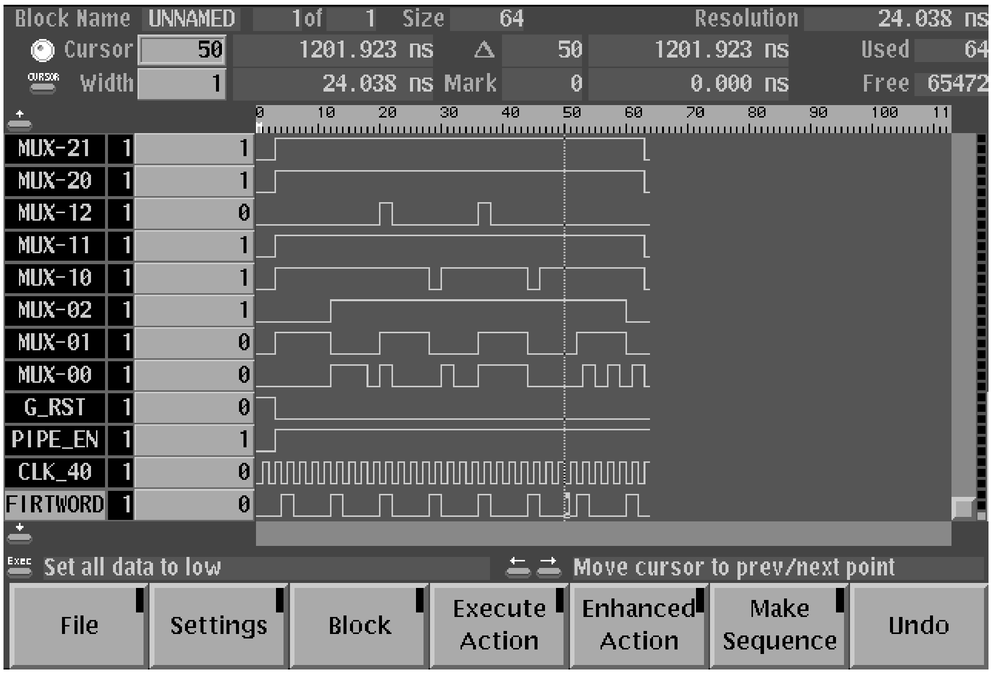


Abbildung 6.7: Im Patterngenerator programmiertes Muster für den Testalgorithmus: Nach 4 clk_40-Flanken sind die Daten eines BCs in das System übertragen, insgesamt wird die Information von sieben BCs eingespielt. Die Padmuster am Eingang wurden so gewählt, dass in bestimmten Binbereichen Spuren erwartet werden. Zudem wurden zufällige Padmuster programmiert, die nicht zu einer Spurerkennung führen dürfen.

einmalig zu Beginn einen Reset auslöst und dann das pipe_en-Signal auf "1" setzt. Anschließend folgt eine sich wiederholende Sequenz, in der 28 multiplexte Signale an die jeweils 15 Eingangskanäle gelegt werden. Das fst_wd-Signal wird alle vier Takte für einen Takt auf eins gesetzt, ansonsten ist es null. Das Muster ist in Abbildung 6.7 dargestellt.

Ergebnis:

- Die Messung hat das in der Simulation (Abb 4.13 in Abschnitt 4.2.1) gezeigte Verhalten des Testalgorithmusses bestätigt. Nach drei HERA-Clockzyklen liegt das z-Histogramm an den Ausgangspins an.
- Die multiplexten Testmuster in den ersten vier simulierten BCs (siehe Abb. 6.7) liefern in den Binbereichen die erwartete Anzahl gefundener Spuren.

- Die multiplexten Testmuster in den letzten drei simulierten BCs liefern keine Spuren, obwohl sehr viele Pads "1" sind. Sie wurden aber so programmiert, dass kein zusammenhängendes Spurmuster entsteht.

6.2.2 Die VME-Schnittstelle

Neben dem Einspielen der Signale über den Patterngenerator wurde das Einspielen und Auslesen von Signalen über den VME-Bus getestet. Da das Zusammenspiel zwischen VME-Bus und FPGAs nicht simuliert werden kann, ist hier die Messung unersetzlich. Ziel der Tests war es, den Datentransport zwischen Modulen, die mit dem VME-Bus verbunden sind, und der FPGA zu untersuchen. Über den VME-Bus wurden die gespeicherten Daten ausgewählter Ereignisse ausgelesen. Zudem sollte versucht werden, über den VME-Bus zuvor ausgelesene Daten zu einem späteren Zeitpunkt in die FPGA zurückzuschreiben, um das Triggersystem ohne Kammer oder Ausleseelektronik mit echten Kammerdaten testen zu können.

Auslesen der Daten: Das Auslesen der Daten sollte nach dem im Abschnitt 4.3 beschriebenen Prinzip erfolgen: Der VME-Bus-Master[4] steuerte den Transfer der Daten von der FPGA über den VME-Bus. Ein Programm in der FPGA, das Daten in der FPGA gespeichert hatte, sollte nun Daten über den VME-Bus ausgeben. Der VME-Bus-Master war ein PC, der über eine PC-Einsteckkarte mit dem VME-Bus verbunden wurde [Ra96]. Ein Schreibzyklus wurde mit der **wr**-Clock, ein Lesezyklus mit der **rd**-Clock synchronisiert [Ra00]. Die Daten aus der FPGA konnten im PC angezeigt werden.

Der Versuchsaufbau ist in Abbildung 6.8 dargestellt.

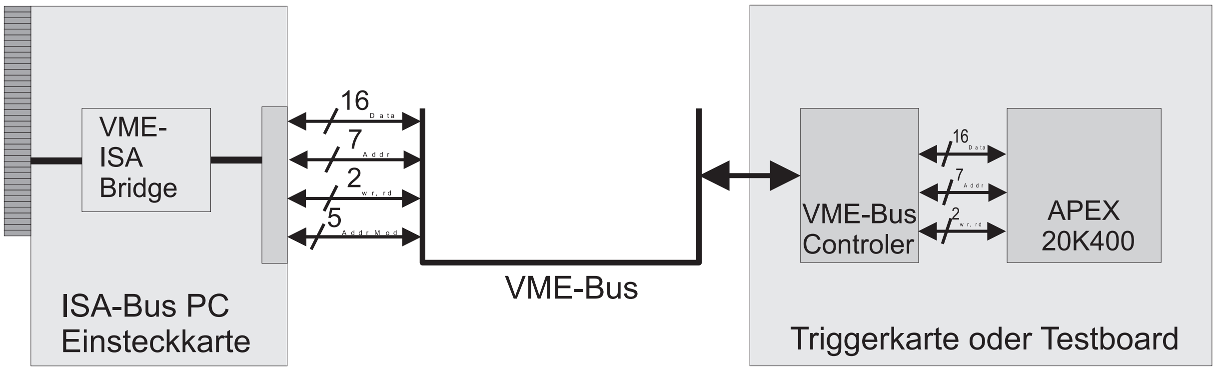


Abbildung 6.8: Darstellung des VME-Bus Versuchsaufbaus: Die Triggerkarte wird über eine PC-Einsteckkarte angesprochen, die die Aufgabe der VME-CPU übernimmt. Der VME-Controler auf der Triggerkarte organisiert die Buszugriffe und ermöglicht dem APEX Daten Lesen und Schreiben von Daten mit der Bus-Clock **wr** (Schreiben) und **rd** (Lesen)

[4]Der VME-Bus benötigt ein Gerät, das die Steuerung des Datentransports und die Adressvergabe organisiert und den Bustakt vorgibt. Dieses Gerät wird als VME-Bus-Master bezeichnet. [Sp87]

Zur Simulation des funktionierenden Systems wurde eine Bitmap auf die FPGA geschrieben, gespeichert und anschließend wieder ausgelesen.

Schreiben der Daten: Im Prinzip wurde in der obigen Messung schon gezeigt, dass das Schreiben von Daten auf die FPGA funktioniert. Es sollte zudem versucht werden, Kammerdaten über den VME-Bus zu schreiben, die auf dem PC zwischengespeichert waren. Diese Daten sollten in die FPGA übertragen werden und dann von dort mit 40 MHz als simulierte Kammerpads ausgewertet werden. Da dazu ein sehr aufwendiges Programm in der FPGA notwendig wäre, konnte diese Messung im Rahmen der Diplomarbeit aus Zeitgründen nicht mehr durchgeführt werden.

6.3 Der globale Systemtest

Zuletzt wurde die gesamte Auslesekette (siehe auch Abb. 2.9 in Abschnitt **??**) getestet. Dafür wurde eine CIP-Testkammer gebaut, die im Gegensatz zur endgültigen Version nur 1/3 der Länge und nur eine Ebene hat. Diese Kammer wurde an der Universität Zürich gebaut und ist seit Juli 1999 in Betrieb. An dieser Kammer wurde die Frontend-Elektronik (CIPix und CIPix-Karten) getestet. Das auf den CIPix-Karten befindliche optische Übertragungssystem bildet mit den in der Backplane befindlichen Empfängerkarten eine Einheit und kann mit Kammerdaten getestet werden.

Ein genauer Bericht über diese Tests ist in [HI00] zu finden, hier soll speziell auf die Resultate der Messungen mit der Backplane und den Triggerkarten eingegangen werden. Die Messung unterteilte sich in zwei Schritte:

- Messung der an der Backplane anliegenden Versorgungsspannungen, Steuersignale und der von der Kammer ankommenden Signale an der Frontseite

- Messung der von den Triggerkarten bearbeiteten und ausgegebenen Signale. Da nur zwei Kanäle vom CIPix an die Backplane übertragen werden konnten, konnte dabei nur der Demultiplexer getestet werden. Eine Spurfindung wäre nicht sinnvoll mit acht gemultiplexten Signalen. Dennoch konnte mit diesem Test gezeigt werden, dass die Triggerkarte mit externen Signalen arbeiten kann und als Glied der Kette funktioniert.

6.3.1 Inbetriebnahme der Triggerkarten

Nachdem die Triggerkarten fertig bestückt waren, wurden sie schrittweise in Betrieb genommen. Hier wurde ähnlich wie beim Testboard vorgegangen. Der Stromverbrauch entspricht für alle drei Spannungen (2,5 V, 3,3 V und 5 V, siehe Tabelle 6.4) den erwarteten Werten. Bei der 5V-Stromversorgung wurde ein Strom vom 1530 mA gemessen, allerdings braucht die VME-Backplane im Triggercrate bereits ohne Einsteckkarten schon 1110 ± 20 mA.

Abbildung 6.9 ist ein Foto der Triggerkarte.

	ohne Clock	mit 40 MHz-Clock
2.5 V	0 mA ± 20 mA	250 mA ± 20 mA
3.3 V	35 mA ± 20 mA	100 mA ± 20 mA
5 V	400 mA ± 20 mA	600 mA ± 20 mA

Tabelle 6.4: Messung des Stromverbrauchs der Karte bei den Spannungen 2,5 V, 3,3 V und 5 V ohne Clock und mit angeschalteter 40 MHz-Clock. Es wurde zuvor die Demultiplexerschaltung in eine FPGA geladen.

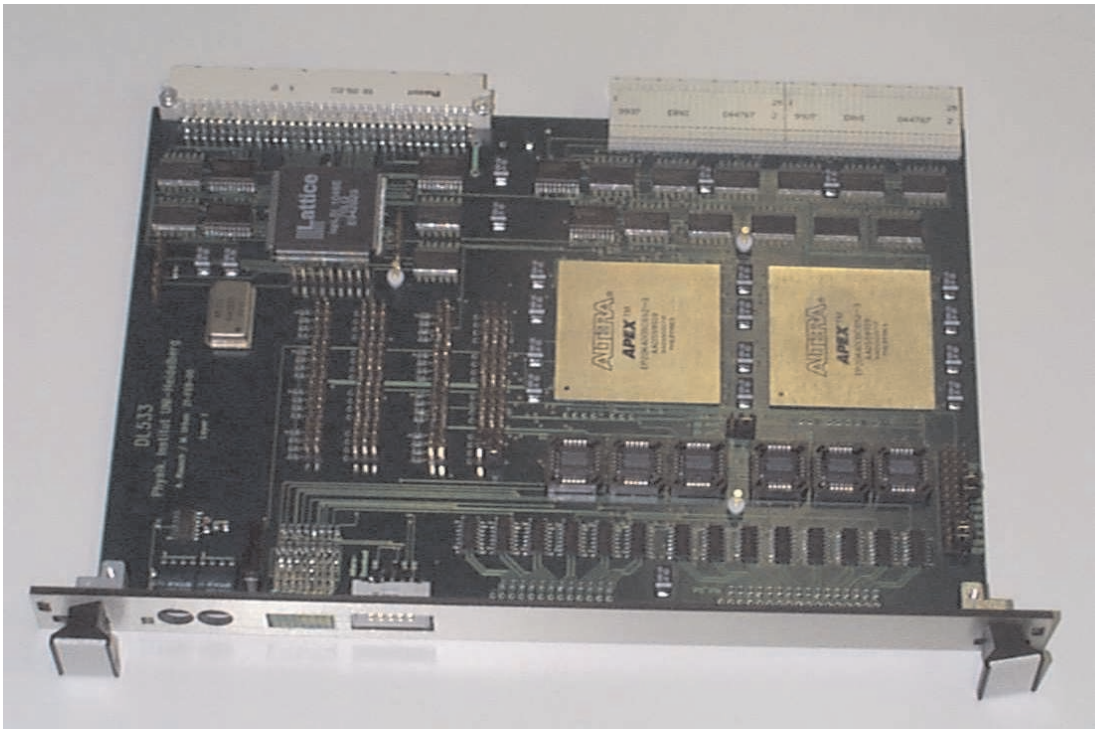


Abbildung 6.9: Die Triggerkarte; die SCSI-Stecker (siehe Dokumentation zur Triggerkarte in Abschnitt 5.2), über die das z-Histogramm ausgelesen wird, sind noch nicht bestückt.

Programmierung der EEPROMs: Die Programmierung der EEPROMs erfolgte im oben erwähnten "JTAG-Mode", den Altera für die Programmierung von EEPROMs oder FPGAs (-Devices) vorschlägt. Sie wurden in einer Kette hintereinandergeschaltet, so dass ein geschlossener Kreis mit dem Programmierstecker entstand, der über ein Programm der Entwicklungsumgebung angesprochen wurde. Jedes Device wurde gesondert programmiert. Die Schaltung zur Programmierung ist in [AN116] abbgebildet.

Bei der Inbetriebnahme musste zunächst mit jedem Device in der JTAG-Kette eine Verbindung hergestellt werden. Es wurde dann über eine JTAG-ID und die Nummer in der Kette identifiziert. Bei der Inbetriebnahme wurden die sechs EEPROMs und eine FPGA erkannt, die zweite FPGA wurde nicht erkannt. Dennoch konnten die EEPROMs programmiert werden, die dann automatisch nach einem Neustart das

Programm in die FPGAs übertragen sollen. Jedoch konnten auch die EEPROMs nur eine FPGA ansprechen; es ist bis jetzt ungeklärt, warum die zweite FPGA nicht auf die Programmierung reagiert.

6.3.2 Integration in das Gesamtsystem

In Abbildung 6.10 ist der Gesamtaufbau, an dem die weiteren Messungen durchgeführt wurden, dargestellt.

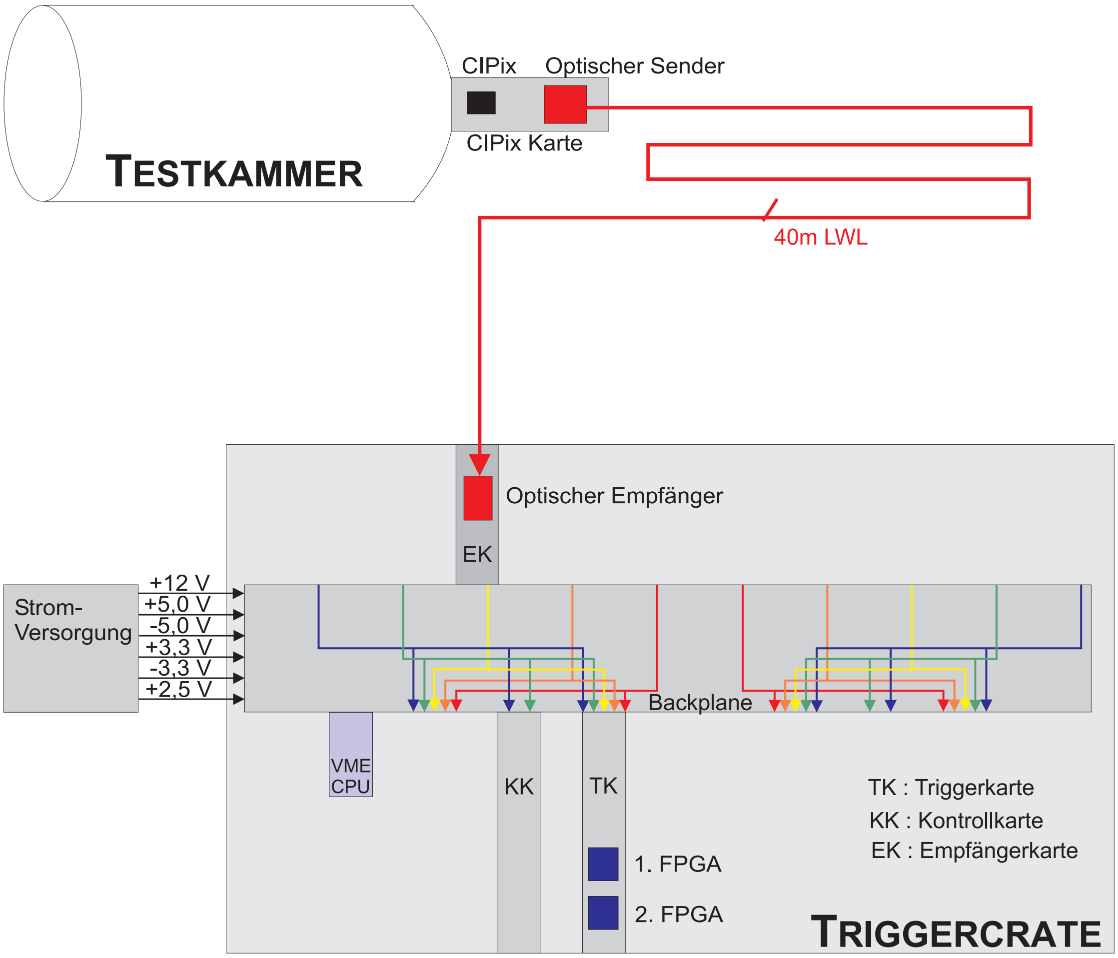


Abbildung 6.10: Versuchsaufbau für den Systemtest: Die Testkammer mit einer aufgesteckten CIPix-Karte, das Lichtwellenleiter-Übertragungssystem und das Triggercrate bilden das Test-CIP-Kammersystem.

Für erste Tests des CIP-Kammer-Systems wurde die Testkammer mit einer CIPix-Karte verbunden. Der auf der Karte befindliche CIPix-Chip bearbeitete die Signale der Kammer und gab sie an die optischen Sender weiter, die die Daten in die Backplane transportierten. Über die optischen Empfängerkarten im Triggercrate wurde zudem die clk_10 an die CIPix-Karten übertragen.

Weitere Steuersignale werden bis jetzt noch nicht an die CIPix-Karten übertragen. Momentan steckt nur eine Empfängerkarte im Triggercrate. Der Dokumentation der

Backplane (siehe Kapitel 5.1) ist zu entnehmen, das die Position der Empfängerkarte in der Backplane entscheidet, welche Ebene mit den Daten der Empfängerkarte versorgt wird. Steckt die Empfängerkarte im mittleren Steckplatz der linken Steckerreihe, können Daten der dritten Ebene der Sektoren ϕ_n und ϕ_{n+1} gemessenen werden. Insgesamt gibt es 10 Steckplätze, die in zwei Fünfergruppen angeordnet sind.

Die Triggerkarte kann auf der anderen Seite der Backplane entweder links oder Rechts, aber nicht in der Mitte der linken Dreiergruppe eingesteckt werden. Steckt sie links, werden Daten des ϕ_n-Sektors gemessen, ansonsten Daten des ϕ_{n+1}-Sektors. Da die Kontrollkarten noch nicht bereitstehen, werden an der Stelle, an der sie später eingesteckt werden sollen, zur Zeit noch über Klemmstecker die notwendigen Signale eingekoppelt.

6.3.3 Robustheit der Triggerkarten

In den Simulationen und Tests mit dem Patterngenerator wurde von "idealen" Daten ausgegangen, die in einem fest vorgegebenen Verhältnis zur Taktflanke das Triggersystem erreichen. Da aber jede Triggerkarte die Kammerdaten verschiedener Ebenen über fünf verschiedene Empfängerkarten erhält, die wiederum die Daten von verschiedenen CIPix-Karten erhalten, werden die Daten mit einer zeitlichen Versatz an den Triggerkarten ankommen. Die Robustheit der Triggerkarten gibt an, wie stark die Phasenverschiebung zwischen Kammerdaten und Taktflanke vom idealen Verhältnis abweichen darf, damit das Triggersystem die Daten noch eindeutig erkennt. Zum einen können die Daten zueinander so stark versetzt sein, dass eine eindeutige Zuordnung zu einem BC nicht mehr möglich ist - in diesem Fall wird der Trigger nicht funktionieren. Zum anderen besteht die Gefahr, dass die Daten während der S&H-Zeiten die FPGAs erreichen und deshalb nicht richtig erkannt werden. In Abbildung 6.11 sind die vom Triggeralgorithmus und der FPGA vorgegebenen Zeitfenster, in der die Daten das System erreichen dürfen, dargestellt.

Die in den Simulationen bestimmten S&H-Zeiten betragen $t_S = 8\,\text{ns}$ und $t_h = 2\,\text{ns}$. Diese Zeiten wurden am APEX 20K400-BGA/3 simuliert (die "3" in der Bezeichnung ist eine Geschwindigkeitsangabe). Die maximalen garantierten Schaltzeiten sind beim APEX /3 größer als beim APEX /1, wo $t_S = 4\,\text{ns}$ und $t_h = 1\,\text{ns}$ beträgt. Diese FPGAs waren nicht lieferbar, werden aber in der Hauptserie verwendet werden.

Bei einer Taktfrequenz von 40 MHz gibt es alle 25 ns eine positive Taktflanke. Werden die Kammerdaten mit der positiven Taktflanke erwartet, so dürfen sie in einem Zeitfenster von 0,5 ns vor und 14,5 ns nach der Flanke anliegen; werden sie mit der negativen Flanke erwartet, so dürfen sie 4 ns davor und 10 ns danach ankommen (siehe Text zur Abbildung 6.11).

6.3.4 Messungen an der CIP-Backplane

In diesen Messungen sollte geprüft werden, ob in welcher Phase die 40 MHz-Clock zu den Daten steht. Die Daten müssen in den dargestellten Zeitfenstern (Abb. 6.11) das Triggersystem erreichen. In Abbildung 6.12 ist diese Messung dargestellt, die direkt an der Backplane durchgeführt wurde.

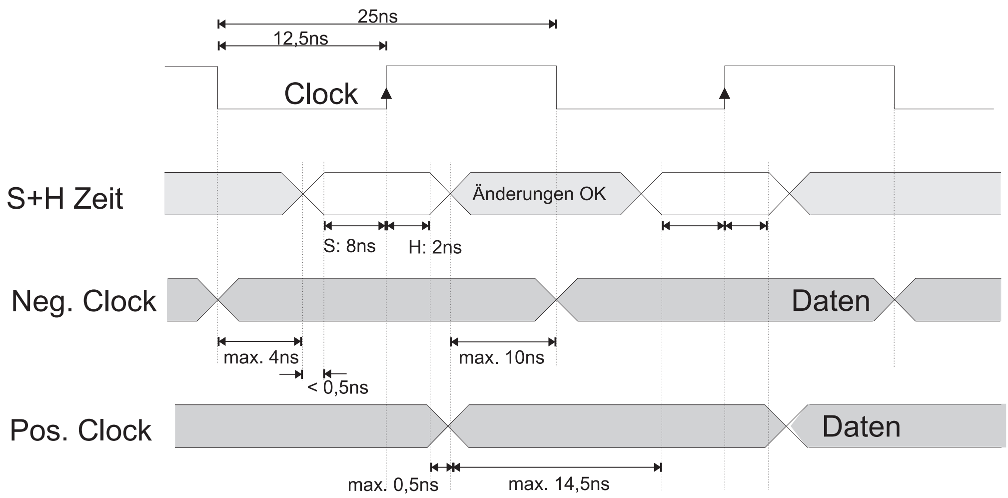


Abbildung 6.11: Darstellung des Bereichs, in dem Eingangssignale zur Clock verschoben sein dürfen: Wenn die Daten mit der negativen clk_40-Flanke synchronisiert übertragen werden, müssen sie 4 ns nach und 10 ns vor der Clockflanke anliegen, um nicht in den S&H-Bereich zu kommen. Für eine positive Clockflanke müssen sie bis maximal 14,5 ns nach der Flanke anliegen. In beiden Fällen werden die Daten in den FPGAs mit der positiven clk_40-Flanke eingelesen.

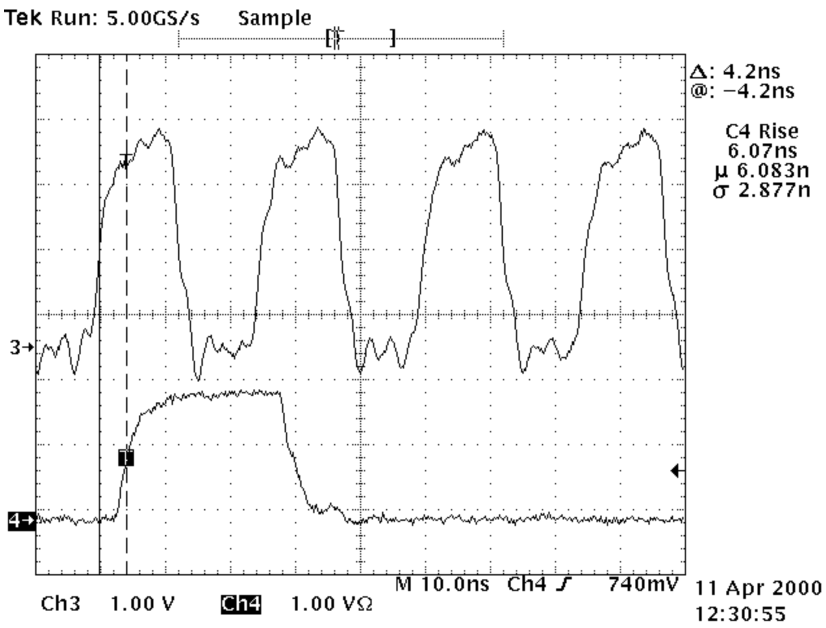


Abbildung 6.12: Signalqualität der Eingangssignale: oben ist die 40 MHz-Clock dargestellt, 6,07 ns später liegt das multiplexte Datenpaket an der Backplane an (unten).

6.3.5 Messungen an der Triggerkarte

Die Messungen an der Triggerkarte konnten nur eingeschränkt durchgeführt werden, da die zweite FPGA noch nicht angesprochen werden kann. In den nächsten Wochen wird ein zweiter Triggerkarten-Prototyp erwartet. Es wird überprüft werden, ob auch bei diesem dasselbe Problem auftritt oder ob es sich um einen Produktionsfehler handelte.

In einer Messung sollte nun versucht werden, Signale von der CIP-Backplane in die FPGAs zu führen und sie anschließend am Ausgang einer FPGA zu messen. Für die Messung wurde ein Programm in die funktionierende FPGA programmiert, das das clk_40-Signal in ein Ausgangsregister schreibt, an dem es abgegriffen werden kann. Über die Backplane gelangt die clk_40 in die Triggerkarten und von dort über Signaltreiber in die FPGA. Die Funktion der Systemkette, bestehend aus CIP-Kammer, CIPix, Übertragungsstrecke, Backplane und Triggerkarte konnte durch diese Messung gezeigt werden.

Wenn die zweite FPGA funktioniert, wird in weiteren Messungen die erfolgreich simulierte Funktion des vollständige Algorithmus geprüft werden. Man erwartet ähnliche Resultate wie bei den Messungen am Testsystem.

Im nächsten Systemtest (im Juli 2000) werden Messungen mit Daten der ersten drei Ebenen der endgültigen CIP-Kammer durchgeführt werden; zu diesem Zeitpunkt werden zwei Triggerkarten und die Kontrollkarte vorhanden sein.

Zusammenfassung:

In dieser Diplomarbeit wurde das Triggersystem für die neue CIP-Kammer entwickelt, die im Rahmen des *H1-Upgrade-2000*-Projekts gebaut wird.

Das Triggersystem ist in der Lage, über einen Bereich von 2,2 m Teilchenspuren zu erkennen, und aus der Verteilung der Spuren längs der Strahlachse eine schnelle Triggerentscheidung zu treffen. Die für einen Level 1-Trigger vorgegebene Zeit, in der eine Triggerentscheidung an der zentralen Triggerkontrolle vorliegen muss, unterbietet das System um 0,5 μs. Die Entscheidung liegt 1,8 μ nach der Teilchenkollision, auf die sie sich bezieht, vor.

Das Triggersystem besteht aus 16 Triggerkarten, drei Summierkarten, acht Kontrollkarten und fünf Prozessorkarten, die auf fünf Crates verteilt werden. Die vier Triggercrates wurden bereits entwickelt. Jedes Triggercrate enthält vier Triggerkarten, zwei Kontrollkarten und einen Prozessor. Auf die Prozessoren wurde in dieser Arbeit nicht näher eingegangen. Auf den Triggerkarten befinden sich FPGAs der Firma Altera, die über ca. 10^6 Gatteräquivalente verfügen. Es wurde ein Algorithmus entwickelt, der mittels der Kammerdaten Teilchenspuren erkennt und eine Zuordnung der Spuren in z-Achsenabschnitte ermöglicht. Dieser Triggeralgorithmus wird vollständig in die FPGAs programmiert. Er wurde in der Hardware-Beschreibungssprache *Verilog* implementiert. Durch die Verwendung von FPGAs auf den Triggerkarten kann das System flexibel an das Experiment und geforderte Aufgaben angepasst werden. Über die Kontrollkarten werden alle Steuerleitungen des Systems überwacht und zudem Kammerdaten der neuen CIP-Kammer aufbereitet und an das alte Triggersystem weitergegeben, so dass dieses mit den Daten ebenfalls eine Triggerentscheidung treffen kann. Die Kontrollkarten werden momentan gebaut.

Im Rahmen dieser Arbeit wurde ein Testsystem entwickelt, das die Funktion des Triggeralgorithmus und die Anwendung von FPGAs für dieses Projekt überprüfen soll. Da die Messungen an diesem Testsystem erfolgreich waren, wurden anschließend die endgültigen Triggerkarten gebaut.

Das ausführliche Testen der entstandenen Triggerkarten konnte in dieser Arbeit nicht mehr durchgeführt werden. Zudem müssen noch die Summierkarten gebaut werden, welche einzelne Informationen der Triggerkarten zusammenfassen und eine Triggerentscheidung treffen.

Das Testen der Triggerkarten wird in einem Systemtest, in dem das komplette System aus Kammer, Ausleseelektronik und Triggersystem aufgebaut wird, im *August 2000* erfolgen. Der Bau der Summierkarten beginnt im *September 2000*.

Anhang A

Liste verwendeter Abkürzungen

In diesem Anhang sind häufig verwendete Abkürzungen alphabetisch aufgeführt.

ADC	Analog Digital Converter
ASIC	Application Specific Integrated Circuit
BC	Bunch Crossing
BDF	Block Diagramm File
CLB	Configurable Logic Array
CIP	Central Inner Proportional
CIPix	Central Inner Proportional Readout Chip
CIZ	Central Inner Z
COZ	Central Outer Z
CTC	Central Trigger Control
DAQ	Daten AQuisation
DSP	Digitaler Signal Prozessor
EEPROM	Electric Erasable Programmable Read Only Memory
FADC	Flash Analog Digital Wandler
FIFO	First In First Out
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
HERA	Hadron-Elektron-Ring-Anlage
IC	Integrated Circuit
JTAG	Join Testing Action Group
L1	Level1
LAB	Logic Array Block
LUT	Look Up Table

LVDS	Low Voltage Differential Signal
NAND	Not AND
PLD	Programmable Logic Device
PLL	Phase Lock Loop
S&U	Setup and Hold
SRAM	Static Random Access Memory
STC	System Trigger Crate
TA60	Trigger Algorithmus mit 60 Pads
TA300	Trigger Algorithmus mit 300 Pads
TTL	Transistor-Transistor-Logik
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

Anhang B

sonstige Abbildungen

Tabelle der APEX-Bausteine von Altera: In Tabelle B.1 sind die Eckdaten der FPGAs aus der APEX-Serie von Altera zusammengefasst. In der Tabelle sind neben der Anzahl der Gatteraquivalente die Anzahl der ESBs und die Anzahl der Userpins von Bedeutung. Zum Zeitpunkt der Entscheidung, welche FPGA verwendet werden sollte, waren nur FPGAs bis hin zu APEX 20K400 mit der geringsten Schaltgeschwindigkeit erhältlich.

Table 1. APEX 20K Device Features *Notes (1), (2)*

Feature	EP20K60E	EP20K100E EP20K100	EP20K160E	EP20K200E EP20K200	EP20K300E	EP20K400E EP20K400	EP20K600E	EP20K1000E	EP20K1500E
Maximum system gates	162,000	263,000	404,000	526,000	728,000	1,052,000	1,537,000	1,771,520	2,524,416
Typical gates	60,000	100,000	160,000	200,000	300,000	400,000	600,000	1,000,000	1,500,000
LEs	2,560	4,160	6,400	8,320	11,520	16,640	24,320	38,400	54,720
ESBs	16	26	40	52	72	104	152	160	228
Maximum RAM bits	32,768	53,248	81,920	106,496	147,456	212,992	311,296	327,680	466,944
Maximum macrocells	256	416	640	832	1,152	1,664	2,432	2,560	3,648
Maximum user I/O pins	204	252	316	382	408	502	624	716	858

Notes:
(1) The embedded IEEE Std. 1149.1 Joint Test Action Group (JTAG) boundary-scan circuitry contributes up to 48,000 additional gates.
(2) This information is preliminary.

Abbildung B.1: Eckdaten der FPGAs aus der APEX-Serie von Altera

Simulation am Speicher: Die in den Abbildungen B.2 und B.3 enthaltenen Signale sollen hier kurz beschrieben werden:

clk_40	41,6 MHz-Clock
wr_vme	VME-Bus-Clock

g_rst	globales Resetsignal
pipe_en	Pipe Enable
pipe_en_40	auf die Schreibmachine synchronisiertes Pipe Enable-Signal
pipe_en_vme	auf die Lesemachine synchronisiertes Pipe Enable-Signal
pipe_en_vme_int	auf die pipe_en_vme synchronisiertes Pipe Enable-Signal
pipe_ready	wird gesetzt, wenn Datenauslese beendet
w_delay	Verzögerungszeit, nach der Schreiben in den Speicher beginnen soll
r_delay	Verzögerungszeit, nach der Lesen in den Speicher beginnen soll
adress	gesamter Adresszeiger, gebildet aus rdp + wrp
rdp	Lese-Adresszeiger, Adresszeiger für Lesemachine
wrp	Schreib-Adresszeiger, Adresszeiger für Schreibmachine
wrp_int	Adressregister, speichert wrp beim Wechsel der Schreib- zur Lesemachine
wrp_int	Adressregister, speichert wrp beim Wechsel der Lese- zur Schreibmachine
rdp_delay_cntr	Zählvariable für Verzögerungszeiten
z	Zählvariable für Verzögerungszeiten
rd_clk1-4	Clock für ebenenweises Auslesen der Daten aus dem Speicher
layer_sel	teilt Busmultiplexer mit, welche Ebene ausgelesen werden soll
pad_data_E0-4	Dateneingang, hier kommen Daten der CIP-Kammer an
VME_out_E0-4	zeigt gespeicherte Daten der einzelnen Ebenen an
VME_out_all	Datenausgang, diese Daten werden auf den VME-Bus gelegt
clk_40_out	Signal zur Messung der Signalverzögerung in der FPGA (s. Kap. 6.1.3)
d_ready	Auslese der Daten fertig

BDFs des endgültigen Triggeralgorithmus TA300: In den Abbildungen B.4 und B.5 sind die Blockschaltbilder der Programme abgebildet, die in die erste und zweite FPGA programmiert werden. Da in die zweite FPGA noch der Gesamtaddierer für den φ-Sektor und der Multiplexer programmiert werden müssen, enthält dieses Schaltbild mehr Blöcke.

Belegungspläne der CIP-Backplanestecker: In den Abbildungen B.6 bis B.9 sind die Pinbelegungen der CIP-Backplanestecker auf Vorder- und Hinterseite dargestellt. Die CIP-Backplane hat in der Elektronikwerkstatt der Universität Heidelberg die Projektbezeichnung **AS16**. Nähere Informationen können unter [EW00] gefunden werden.

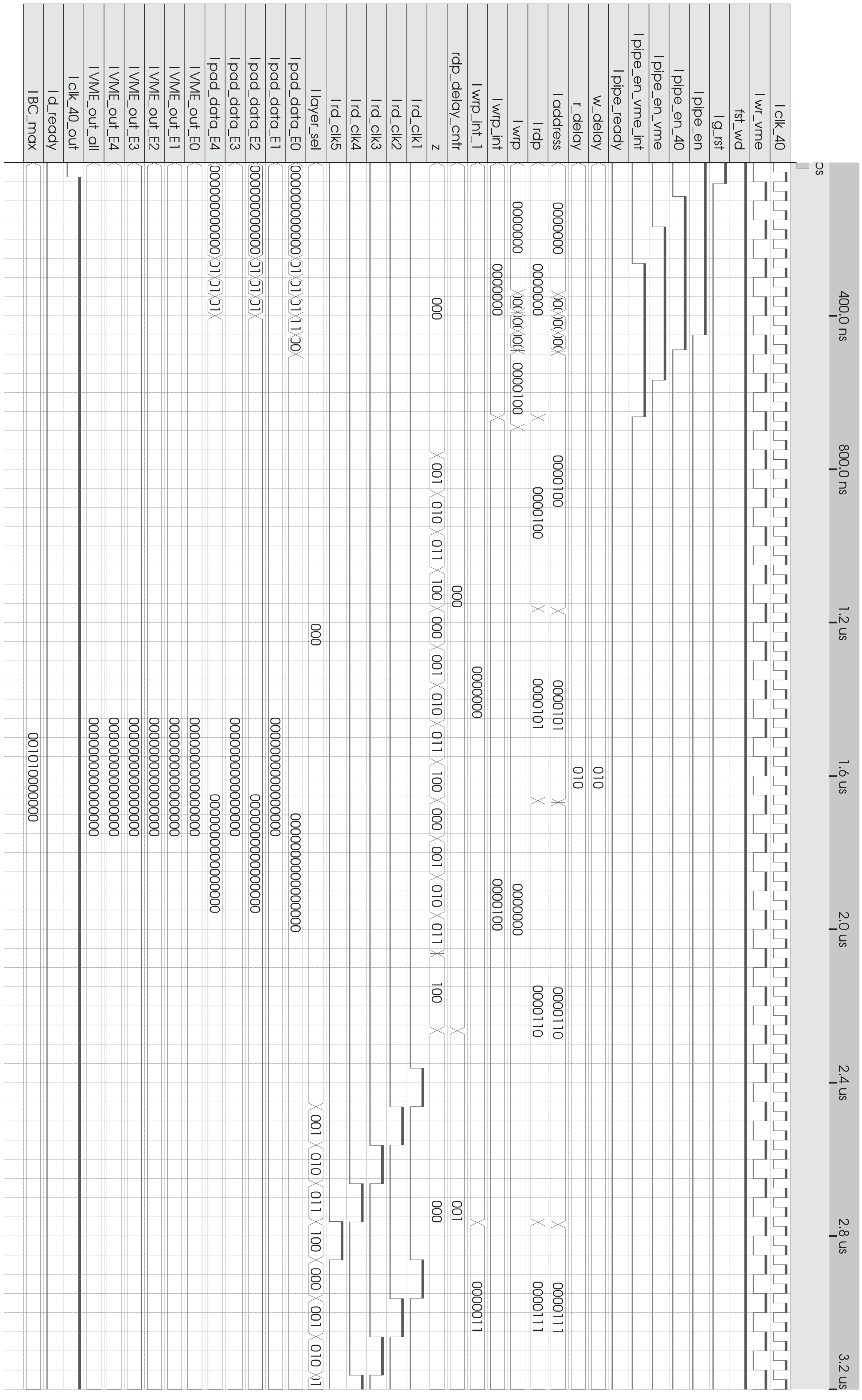


Abbildung B.2: Timingsimulation des Speichers Teil A

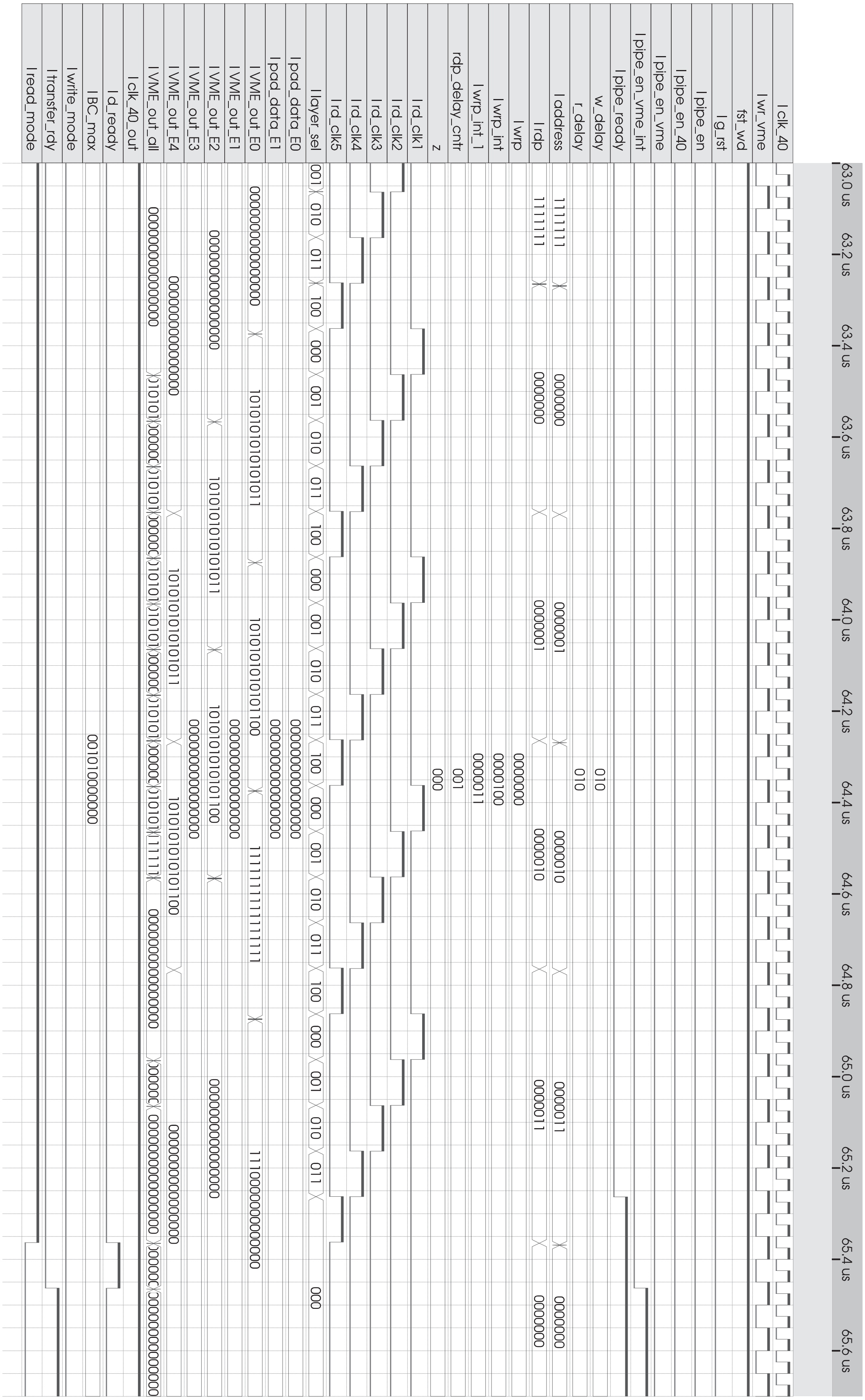


Abbildung B.3: Timingsimulation des Speichers, Teil B

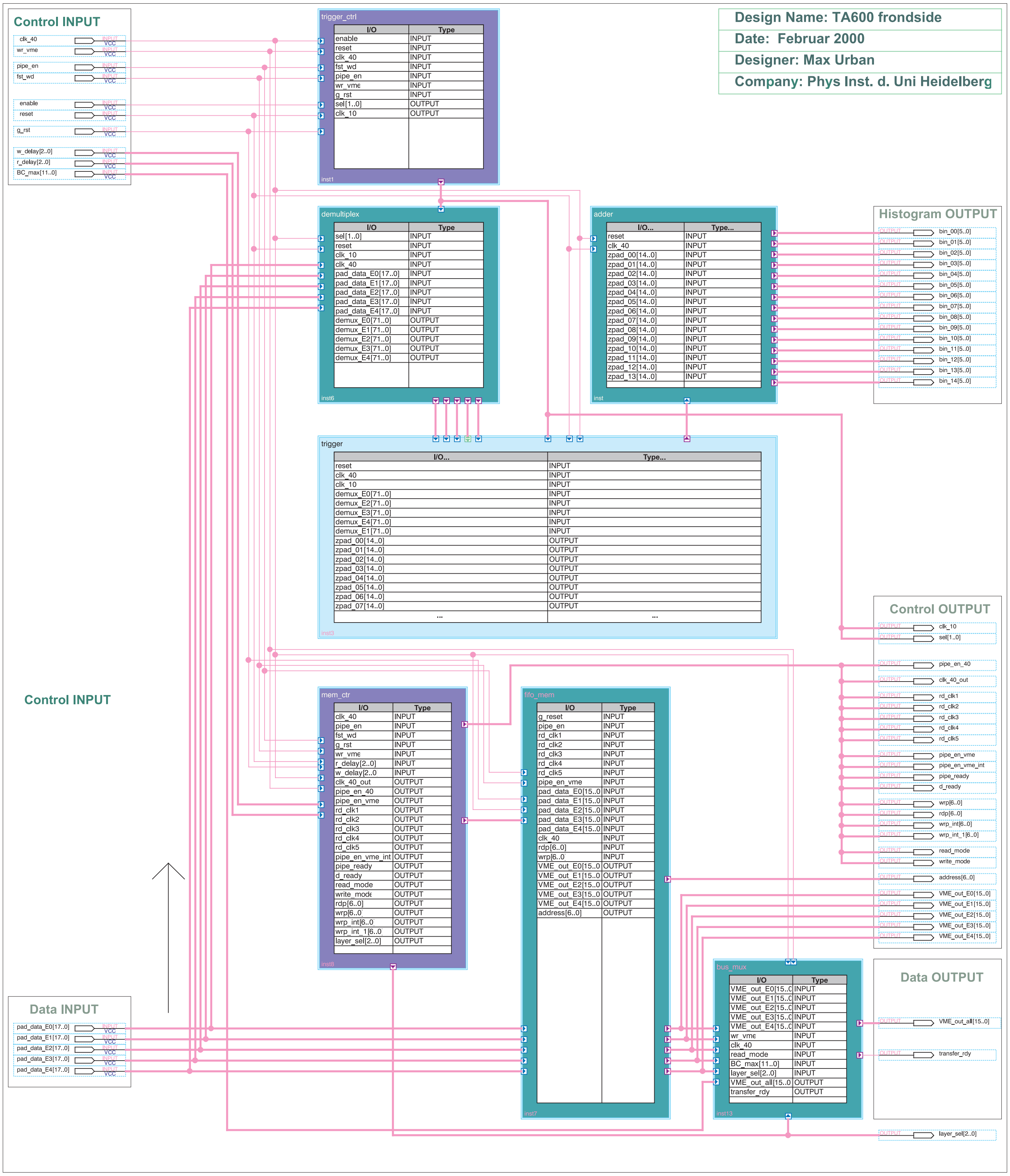


Abbildung B.4: Blockschaltbild des in die linke FPGA programmierten Algorithmusses

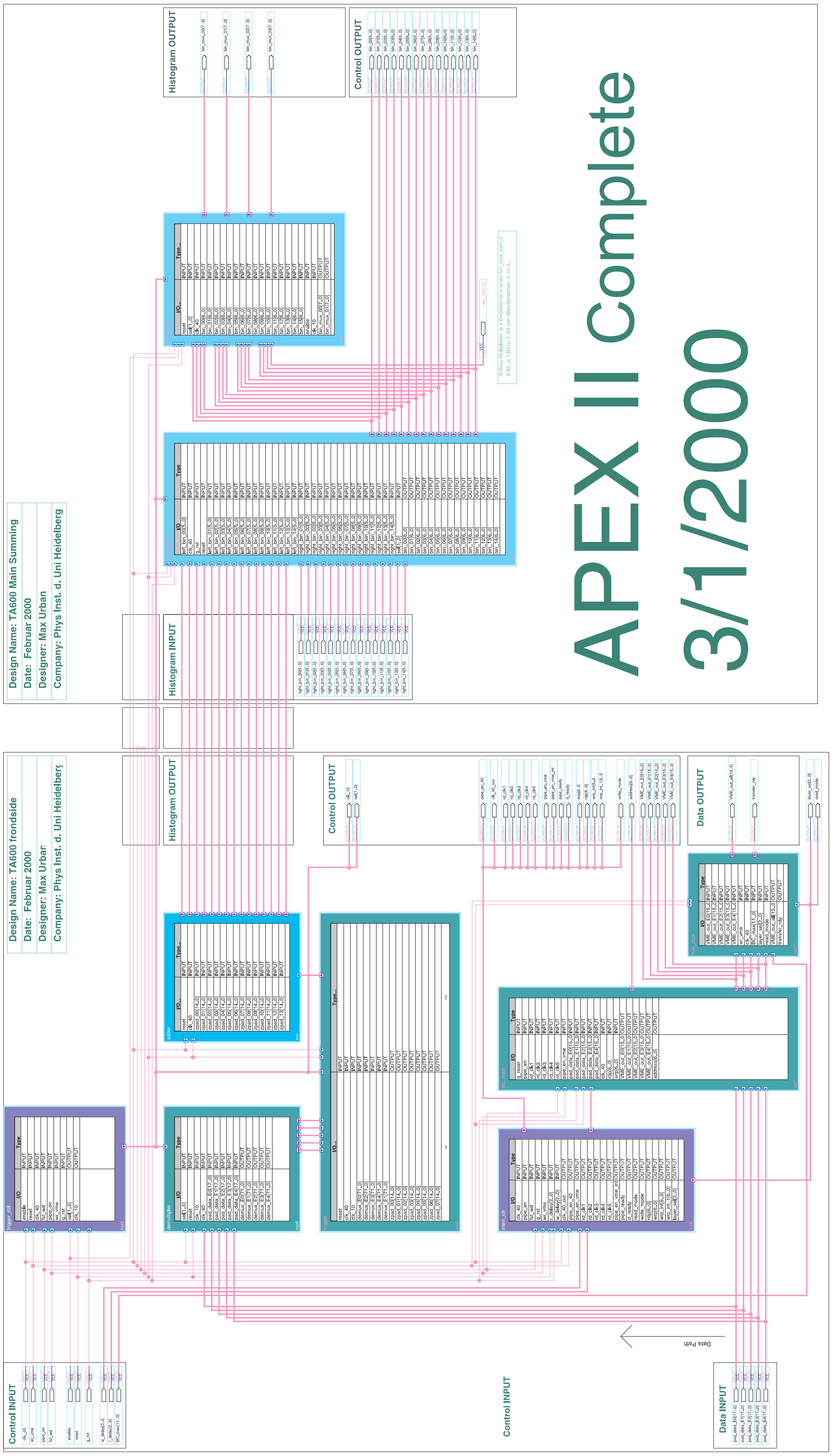


Abbildung B.5: Blockschaltbild des in die rechte FPGA programmierten Algorithmusses

Legende: Sxx = Segment / S01 > Segment 0 u. 1
Ex = Ebene / E0 bis E4
Nxx = Nippel für 4-bit / Schaltplan wird übersichtlicher!
Cxx = Clock
FWx = FirstWord
EMx = EMpty
VDxx = VME-DATA 16-31

Pinbelegung / Pinbezeichnung vom Optical Link Interface Steckplatz # S1

PIN\Reihe	A	B	C
1	S1E0N15	/GBRESET	S0E0N15
2	S1E0N16	GND	S0E0N16
3	S1E0N17	GND	S0E0N17
4	S1E0N18	S01E0C10 1,2*	S0E0N18
5	S1E0N19	S01E0C10 1,2*	S0E0N19
6	S1E0N20	GND	S0E0N20
7	S1E0N21	GND	S0E0N21
8	S1E0N22	+3.3V	S0E0N22
9	S1E0N23	+3.3V	S0E0N23
10	S1E0N24	GND	S0E0N24
11	S1E0N25	GND	S0E0N25
12	S1E0N26	+5VA	S0E0N26
13	S1E0N27	+5VA	S0E0N27
14	S1E0N28	GND	S0E0N28
15	S1E0N29	GND	S0E0N29
16	S1E0N0	-5VA	S0E0N0
17	S1E0N1	-5VA	S0E0N1
18	S1E0N2	GND	S0E0N2
19	S1E0N3	GND	S0E0N3
20	S1E0N4	+3.3V	S0E0N4
21	S1E0N5	+3.3V	S0E0N5
22	S1E0N6	GND	S0E0N6
23	S1E0N7	GND	S0E0N7
24	S1E0N8		S0E0N8
25	S1E0N0		S0E0N0
26	S1E0N10		S0E0N10
27	S1E0N11	S01E0FW	S0E0N11
28	S1E0N12	GND	S0E0N12
29	S1E0N13	GND	S0E0N13
30	S1E0N14	S01E0C40 1*	S0E0N14
31	S1E0EM60*	S01E0C40	S0E0EM60*
32	S1E0EM0*	GND	S0E0EM0*

Anmerkung: 1* Steuersignale von Ebene 2 gehen an J1, J2 u. J3
Steuersignale von Ebene 0-1 u. 3-4 gehen nur an J2 zur Gleichlaufkontrolle
2* Clock 10.4 MHz ist hier parallel geschaltet ! / kommt jedoch separat von J2 für jede Ebene

Abbildung B.6: Belegungssplan der CIP-Backplane, Seite 1

Pinbelegung / Pinbezeichnung vom SECTOR-0 Board Stecker # J1

PIN #	a	b	c	d	e
1	VD16	VD20	/GBRESET	VD24	VD28
2	VD17	VD21	S01Spare	VD25	VD29
3	VD18	VD22	GND/DASY0	VD26	VD30
4	VD19	VD23	DASY1	VD27	VD31
5	-	-	-	-	-
6	S0E4N15	S0E3N15	S0E2N15	S0E1N15	S0E0N15
7	S0E4N16	S0E3N16	S0E2N16	S0E1N16	S0E0N16
8	S0E4N17	S0E3N17	S0E2N17	S0E1N17	S0E0N17
9	GND	GND	GND	GND	GND
10	S0E4N18	S0E3N18	S0E2N18	S0E1N18	S0E0N18
11	S0E4N19	S0E3N19	S0E2N19	S0E1N19	S0E0N19
12	S0E4N20	S0E3N20	S0E0N20	S0E1N20	S0E0N20
13	+2,5V	+2,5V	+2,5V	+2,5V	+2,5V
14	S0E4N21	S0E3N21	S0E2N21	S0E1N21	S0E0N21
15	S0E4N22	S0E3N22	S0E2N22	S0E1N22	S0E0N22
16	S0E4N23	S0E3N23	S0E2N23	S0E1N23	S0E0N23
17	GND	GND	GND	GND	GND
18	S0E4N24	S0E3N24	S0E2N24	S0E1N24	S0E0N24
19	S0E4N25	S0E3N25	S0E2N25	S0E1N25	S0E0N25
20	S0E4N26	S0E3N26	S0E2N26	S0E1N26	S0E0N26
21	+2,5V	+2,5V	+2,5V	+2,5V	+2,5V
22	S0E4N27	S0E3N27	S0E2N27	S0E1N27	S0E0N27
23	S0E4N28	S0E3N28	S0E2N28	S0E1N28	S0E0N28
24	S0E4N29	S0E3N29	S0E2N29	S0E1N29	S0E0N29
25	GND	GND	GND	GND	GND
26	S0E4N0	S0E3N0	S0E2N0	S0E1N0	S0E0N0
27	S0E4N1	S0E3N1	S0E2N1	S0E1N1	S0E0N1
28	S0E4N2	S0E3N2	S0E2N2	S0E1N2	S0E0N2
29	+2,5V	+2,5V	+2,5V	+2,5V	+2,5V
30	S0E4N3	S0E3N3	S0E2N3	S0E1N3	S0E0N3
31	S0E4N4	S0E3N4	S0E2N4	S0E1N4	S0E0N4
32	S0E4N5	S0E3N5	S0E2N5	S0E1N5	S0E0N5
33	GND	GND	GND	GND	GND
34	S0E4N6	S0E3N6	S0E2N6	S0E1N6	S0E0N6
35	S0E4N7	S0E3N7	S0E2N7	S0E1N7	S0E0N7
36	S0E4N8	S0E3N8	S0E2N8	S0E1N8	S0E0N8
37	+3,3V	+3,3V	+3,3V	+3,3V	+3,3V
38	S0E4N9	S0E3N9	S0E2N9	S0E1N9	S0E0N9
39	S0E4N10	S0E3N10	S0E2N10	S0E1N10	S0E0N10
40	S0E4N11	S0E3N11	S0E2N11	S0E1N11	S0E0N11
41	GND	GND	S01PIPEEN	GND	GND
42	S0E4N12	S0E3N12	S0E2N12	S0E1N12	S0E0N12
43	S0E4N13	S0E3N13	S0E2N13	S0E1N13	S0E0N13
44	S0E4N14	S0E3N14	S0E2N14	S0E1N14	S0E0N14
45			S01E2FW		
46			S01E2C40		
47			S0E2EM60*		
48			S0E2EM0*		
59			S1E2EM60*		
50			S1E2EM0*		

Abbildung B.7: Belegungssplan der CIP-Backplane, Seite 2

Pinbelegung / Pinbezeichnung vom Contro1-Board S01 Stecker # J2

PIN	a	b	c	d	e
1	VD16	VD20	/GBRESET	VD24	VD28
2	VD17	VD21	S01Spare	VD25	VD29
3	VD18	VD22	-	VD26	VD30
4	VD19	VD23	-	VD27	VD31
5	S01E4C10 > S5	S01E3C10 > S4	S01E2C10 > S3	S01E1C10> S2	S01E0C10 > S1
6	S0E1N15	S0E0N15		S1E1N15	S1E0N15
7	S0E1N16	S0E0N16		S1E1N16	S1E0N16
8	S0E1N17	S0E0N17		S1E1N17	S1E0N17
9	GND	GND	GND	GND	GND
10	S0E1N18	S0E0N18		S1E1N18	S1E0N18
11	S0E1N19	S0E0N19		S1E1N19	S1E0N19
12	S0E1N20	S0E0N20		S1E1N20	S1E0N20
13	+2,5V	+2,5V	+2,5V	+2,5V	+2,5V
14	S0E1N21	S0E0N21		S1E1N21	S1E0N21
15	S0E1N22	S0E0N22		S1E1N22	S1E0N22
16	S0E1N23	S0E0N23		S1E1N23	S1E0N23
17	GND	GND	GND	GND	GND
18	S0E1N24	S0E0N24		S1E1N24	S1E0N24
19	S0E1N25	S0E0N25		S1E1N25	S1E0N25
20	S0E1N26	S0E0N26		S1E1N26	S1E0N26
21	+2,5V	+2,5V	+2,5V	+2,5V	+2,5V
22	S0E1N27	S0E0N27		S1E1N27	S1E0N27
23	S0E1N28	S0E0N28		S1E1N28	S1E0N28
24	S0E1N29	S0E0N29		S1E1N29	S1E0N29
25	GND	GND	GND	GND	GND
26	S0E1N0	S0E0N0		S1E1N0	S1E0N0
27	S0E1N1	S0E0N1		S1E1N1	S1E0N1
28	S0E1N2	S0E0N2		S1E1N2	S1E0N2
29	+2,5V	+2,5V	+2,5V	+2,5V	+2,5V
30	S0E1N3	S0E0N3		S1E1N3	S1E0N3
31	S0E1N4	S0E0N4		S1E1N4	S1E0N4
32	S0E1N5	S0E0N5		S1E1N5	S1E0N5
33	GND	GND	GND	GND	GND
34	S0E1N6	S0E0N6		S1E1N6	S1E0N6
35	S0E1N7	S0E0N7		S1E1N7	S1E0N7
36	S0E1N8	S0E0N8		S1E1N8	S1E0N8
37	+3,3V	+3,3V	+3,3V	+3,3V	+3,3V
38	S0E1N9	S0E0N9		S1E1N9	S1E0N9
39	S0E1N10	S0E0N10		S1E1N10	S1E0N10
40	S0E1N11	S0E0N11		S1E1N11	S1E0N11
41	GND	GND	S01PIPEEN	GND	GND
42	S0E1N12	S0E0N12		S1E1N12	S1E0N12
43	S0E1N13	S0E0N13		S1E1N13	S1E0N13
44	S0E1N14	S0E0N14		S1E1N14	S1E0N14
45	S01E4FW	S01E3FW	S01E2FW	S01E1FW	S01E0FW
46	S01E4C40	S01E3C40	S01E2C40	S01E1C40	S01E0C40
47	S0E4EM60*	S0E3EM60*	S0E2EM60*	S0E1EM60*	S0E0EM60*
48	S0E4EM0*	S0E3EM0*	S0E2EM0*	S0E1EM0*	S0E0EM0*
59	S1E4EM60*	S1E3EM60*	S1E2EM60*	S1E1EM60*	S1E0EM60*
50	S1E4EM0*	S1E3EM0*	S1E2EM0*	S1E1EM0*	S1E0EM0*

Abbildung B.8: Belegungssplan der CIP-Backplane, Seite 3

Pinbelegung / Pinbezeichnung vom SECTOR-1 Board Stecker # J3

PIN #	a	b	c	d	e
1	VD16	VD20	/GBRESET	VD24	VD28
2	VD17	VD21	S01Spare	VD25	VD29
3	VD18	VD22	DASY2	VD26	VD30
4	VD19	VD23	DASY3/to J4.C3	VD27	VD31
5	-	-	-	-	-
6	S1E4N15	S1E3N15	S1E2N15	S1E1N15	S1E0N15
7	S1E4N16	S1E3N16	S1E2N16	S1E1N16	S1E0N16
8	S1E4N17	S1E3N17	S1E2N17	S1E1N17	S1E0N17
9	GND	GND	GND	GND	GND
10	S1E4N18	S1E3N18	S1E2N18	S1E1N18	S1E0N18
11	S1E4N19	S1E3N19	S1E2N19	S1E1N19	S1E0N19
12	S1E4N20	S1E3N20	S1E2N20	S1E1N20	S1E0N20
13	+2,5V	+2,5V	+2,5V	+2,5V	+2,5V
14	S1E4N21	S1E3N21	S1E2N21	S1E1N21	S1E0N21
15	S1E4N22	S1E3N22	S1E2N22	S1E1N22	S1E0N22
16	S1E4N23	S1E3N23	S1E2N23	S1E1N23	S1E0N23
17	GND	GND	GND	GND	GND
18	S1E4N24	S1E3N24	S1E2N24	S1E1N24	S1E0N24
19	S1E4N25	S1E3N25	S1E2N25	S1E1N25	S1E0N25
20	S1E4N26	S1E3N26	S1E2N26	S1E1N26	S1E0N26
21	+2,5V	+2,5V	+2,5V	+2,5V	+2,5V
22	S1E4N27	S1E3N27	S1E2N27	S1E1N27	S1E0N27
23	S1E4N28	S1E3N28	S1E2N28	S1E1N28	S1E0N28
24	S1E4N29	S1E3N29	S1E2N29	S1E1N29	S1E0N29
25	GND	GND	GND	GND	GND
26	S1E4N0	S1E3N0	S1E2N0	S1E1N0	S1E0N0
27	S1E4N1	S1E3N1	S1E2N1	S1E1N1	S1E0N1
28	S1E4N2	S1E3N2	S1E2N2	S1E1N2	S1E0N2
29	+2,5V	+2,5V	+2,5V	+2,5V	+2,5V
30	S1E4N3	S1E3N3	S1E2N3	S1E1N3	S1E0N3
31	S1E4N4	S1E3N4	S1E2N4	S1E1N4	S1E0N4
32	S1E4N5	S1E3N5	S1E2N5	S1E1N5	S1E0N5
33	GND	GND	GND	GND	GND
34	S1E4N6	S1E3N6	S1E2N6	S1E1N6	S1E0N6
35	S1E4N7	S1E3N7	S1E2N7	S1E1N7	S1E0N7
36	S1E4N8	S1E3N8	S1E2N8	S1E1N8	S1E0N8
37	+3,3V	+3,3V	+3,3V	+3,3V	+3,3V
38	S1E4N9	S1E3N9	S1E2N9	S1E1N9	S1E0N9
39	S1E4N10	S1E3N10	S1E2N10	S1E1N10	S1E0N10
40	S1E4N11	S1E3N11	S1E2N11	S1E1N11	S1E0N11
41	GND	GND	S01PIPEEN	GND	GND
42	S1E4N12	S1E3N12	S1E2N12	S1E1N12	S1E0N12
43	S1E4N13	S1E3N13	S1E2N13	S1E1N13	S1E0N13
44	S1E4N14	S1E3N14	S1E2N14	S1E1N14	S1E0N14
45			S01E2FW		
46			S01E2C40		
47			S0E2EM60*		
48			S0E2EM0*		
59			S1E2EM60*		
50			S1E2EM0*		

Lagenaufbau der Backplan / 10 Lagen Multilayer

Abbildung B.9: Belegungssplan der CIP-Backplane, Seite 4

Anhang C

Schaltpläne zu den entwickelten Komponenten

In diesem Anhang sind die Schaltpläne der CIP-Backplane und der Triggerkarte aufgeführt.

Bestückungsplan der Triggerkarte: Eine Liste der verwendeten Bausteine kann im Archiv der Elektroniwerkstat der Universität Heidelberg unter der Projektbezeichnung **DL533** gefunden werden [EW00].

Schaltpläne der Triggerkarte: In den Abbildungen C.2 bis C.8 sind die Schaltpläne der Triggerkarten dargestellt.

Schaltpläne der CIP-Backplane: In den Abbildungen C.9 bin C.13 sind die Schaltpläne der CIP-Backplane dargestellt.

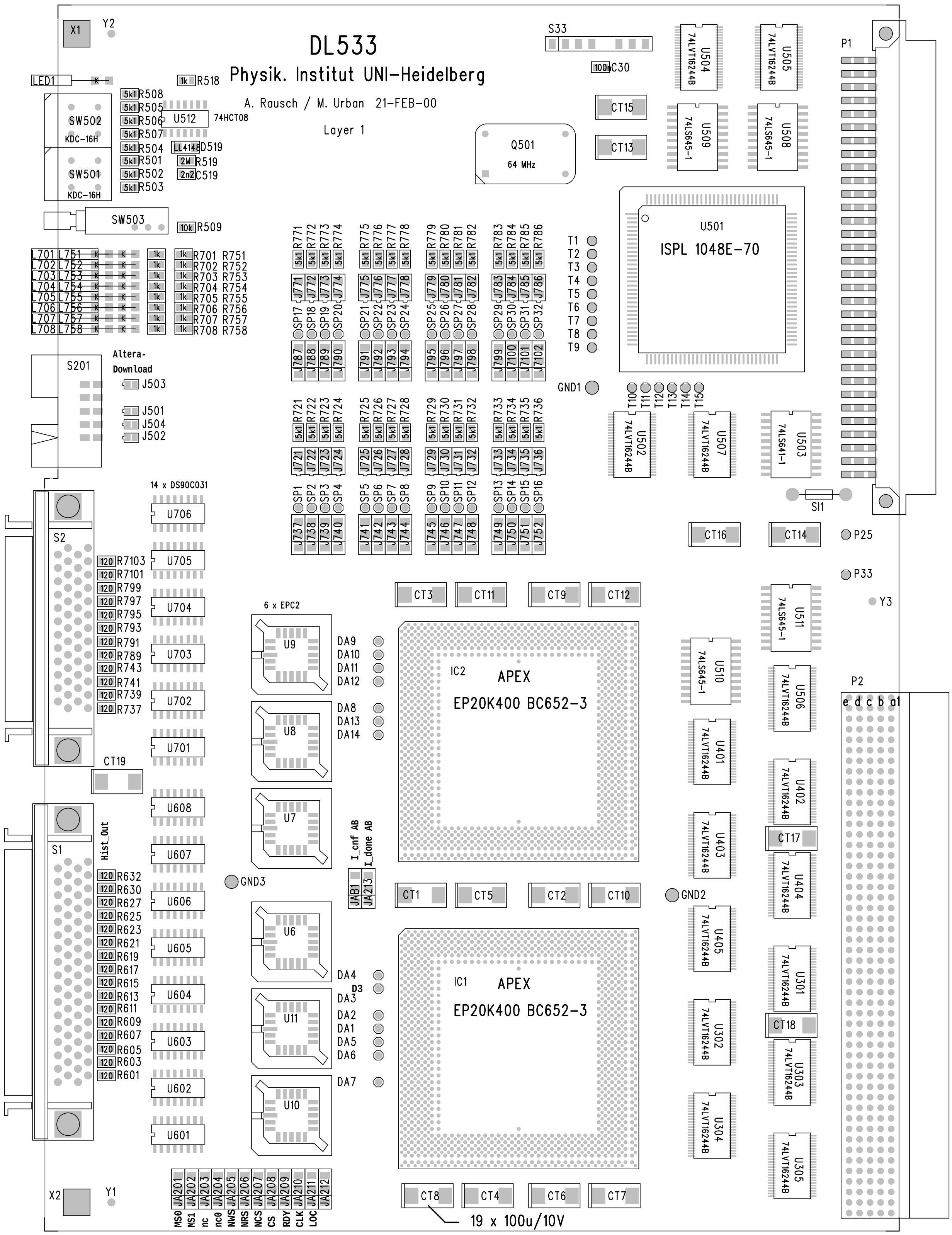


Abbildung C.1: Bestückungsplan der Triggerkarte

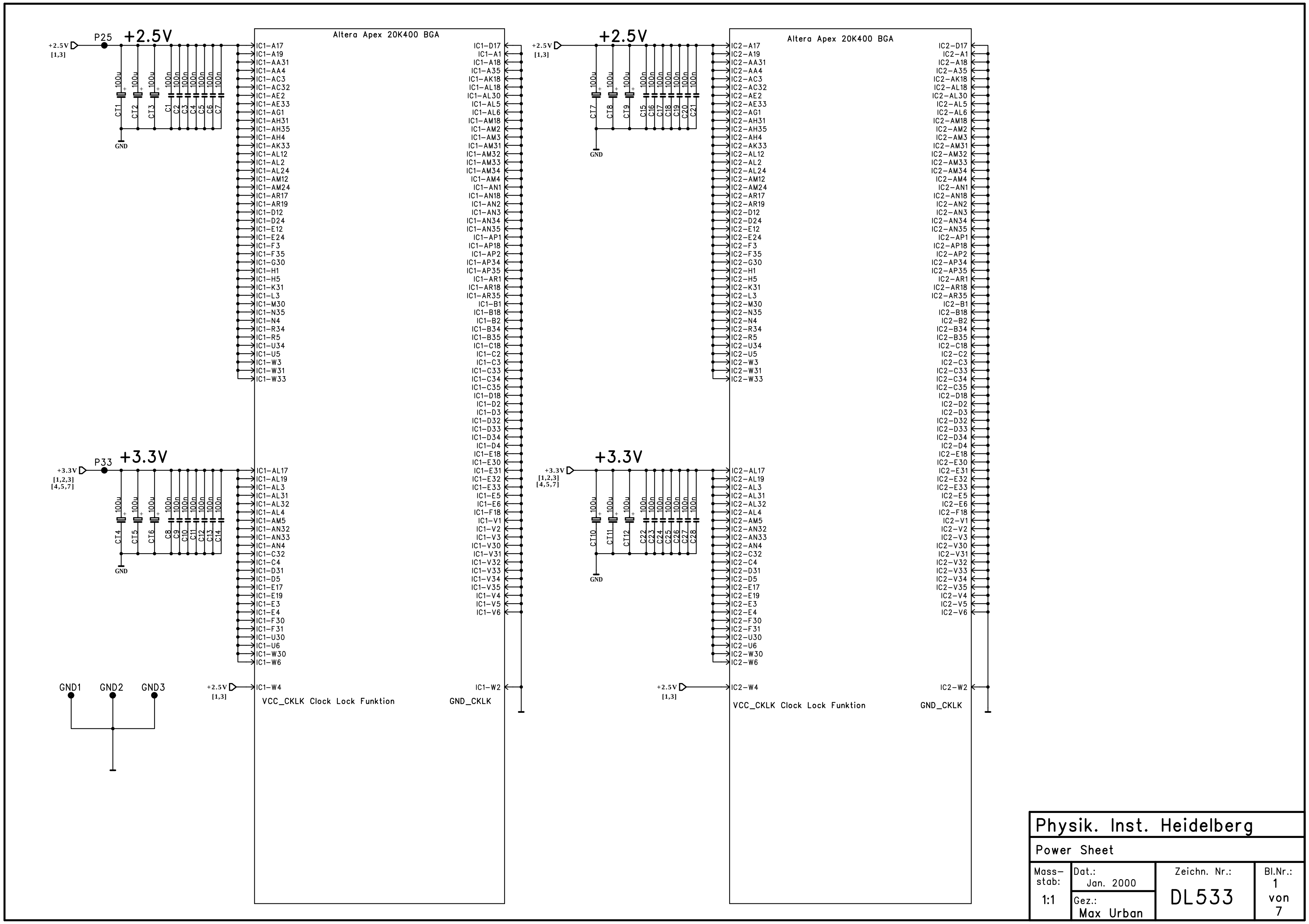


Abbildung C.2: Schaltplan der Triggerkarte, Lage 1/7

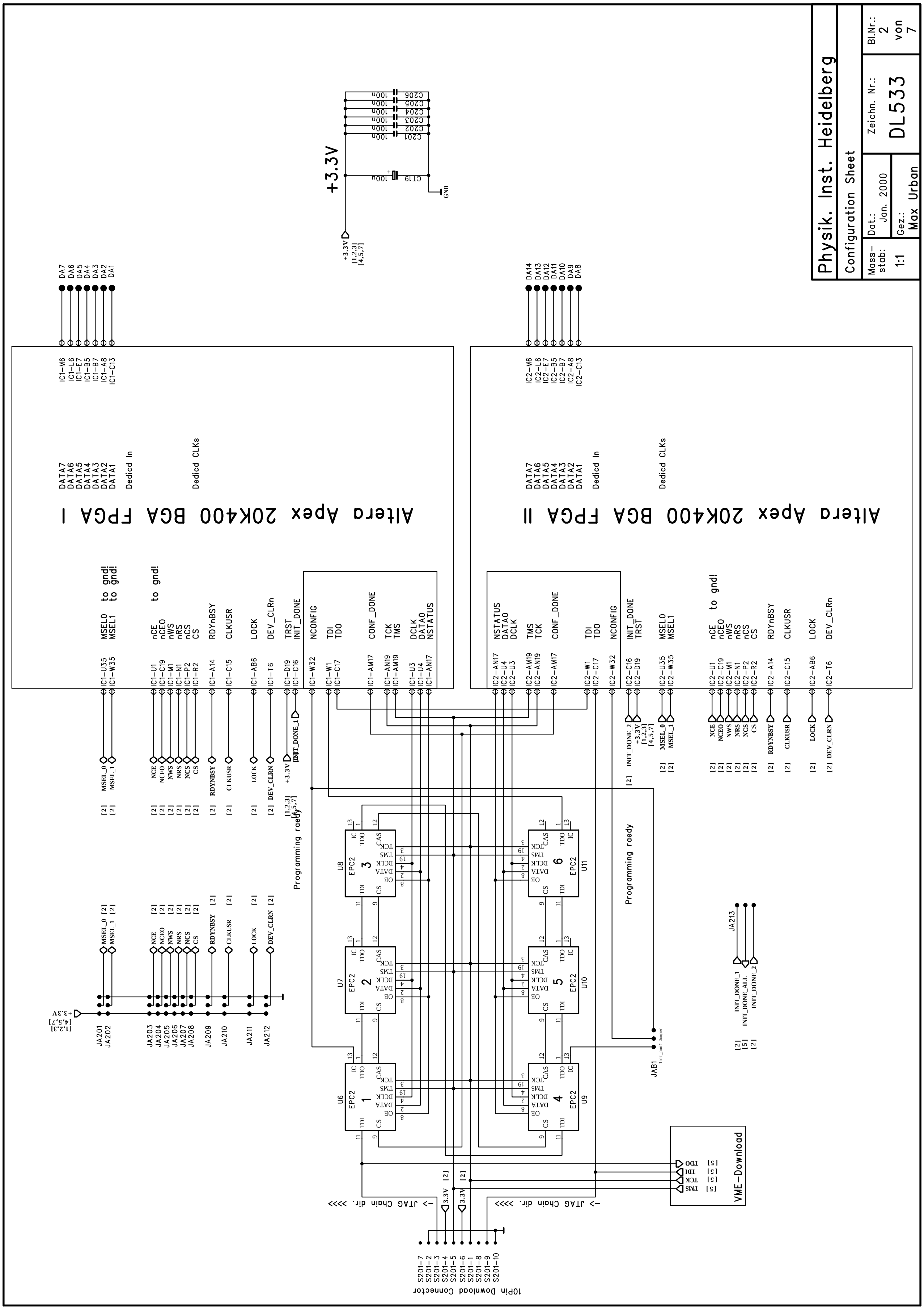
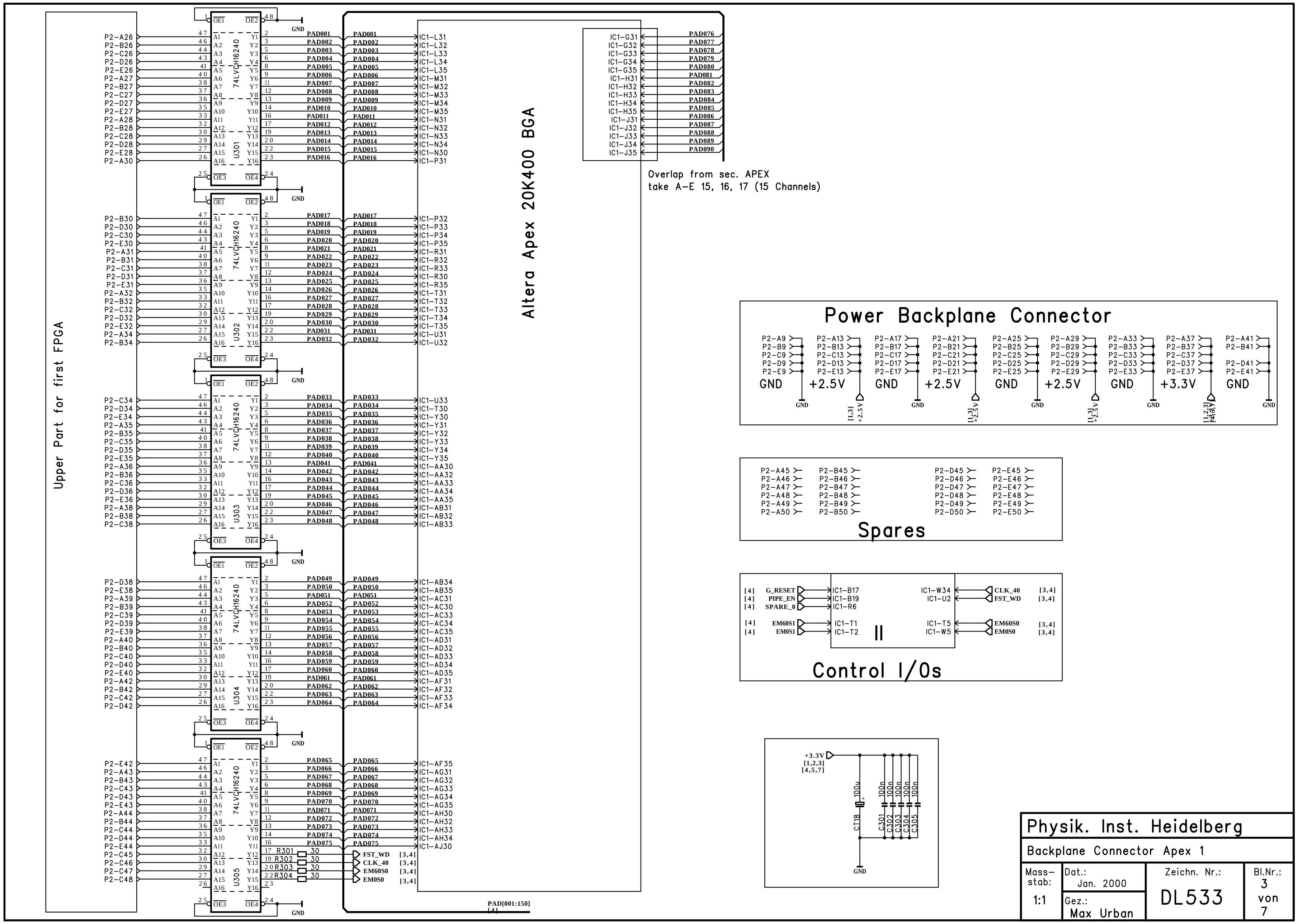


Abbildung C.3: Schaltplan der Triggerkarte, Lage 2/7

Abbildung C.4: Schaltplan der Triggerkarte, Lage 3/7



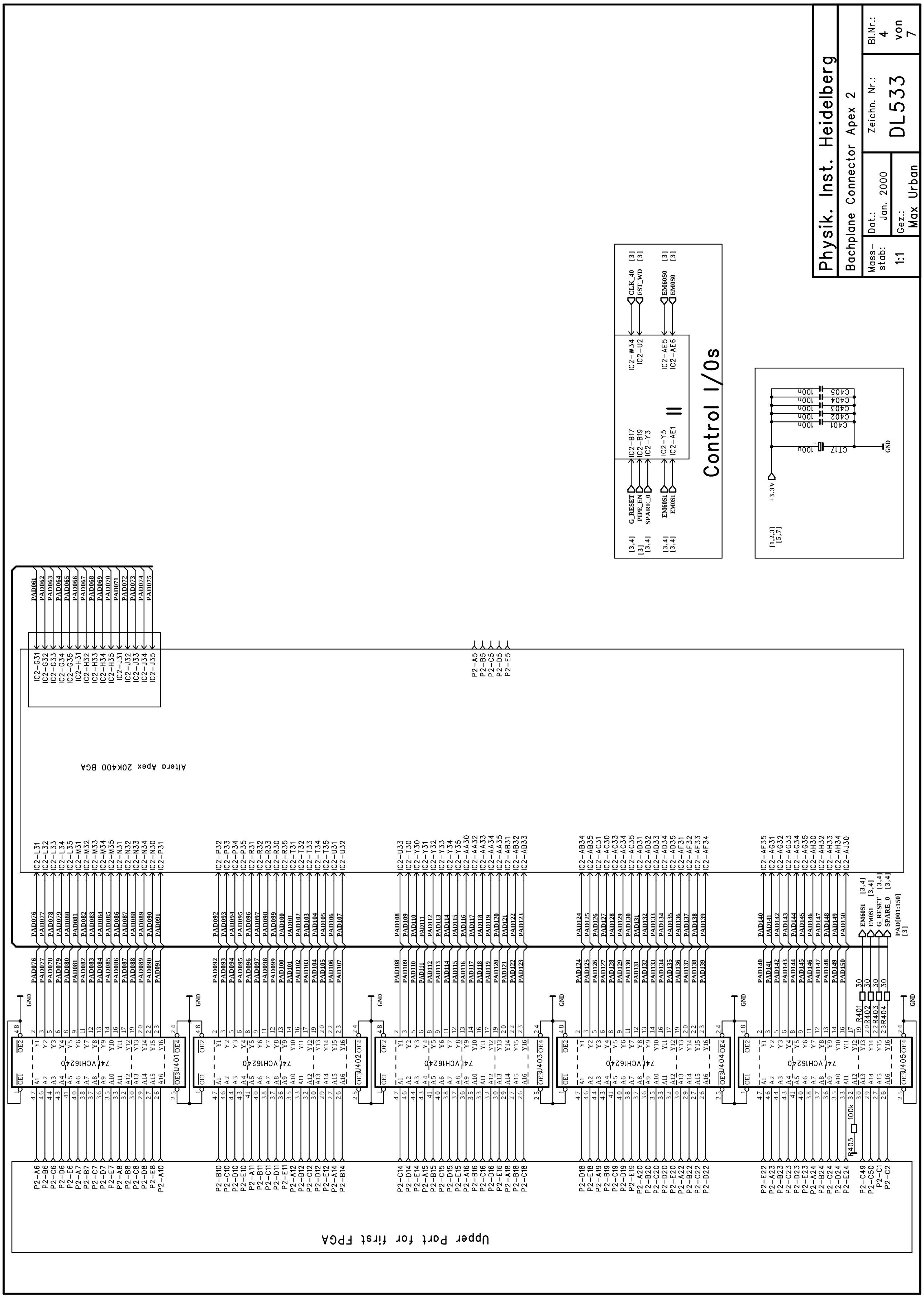


Abbildung C.5: Schaltplan der Triggerkarte, Lage 4/7

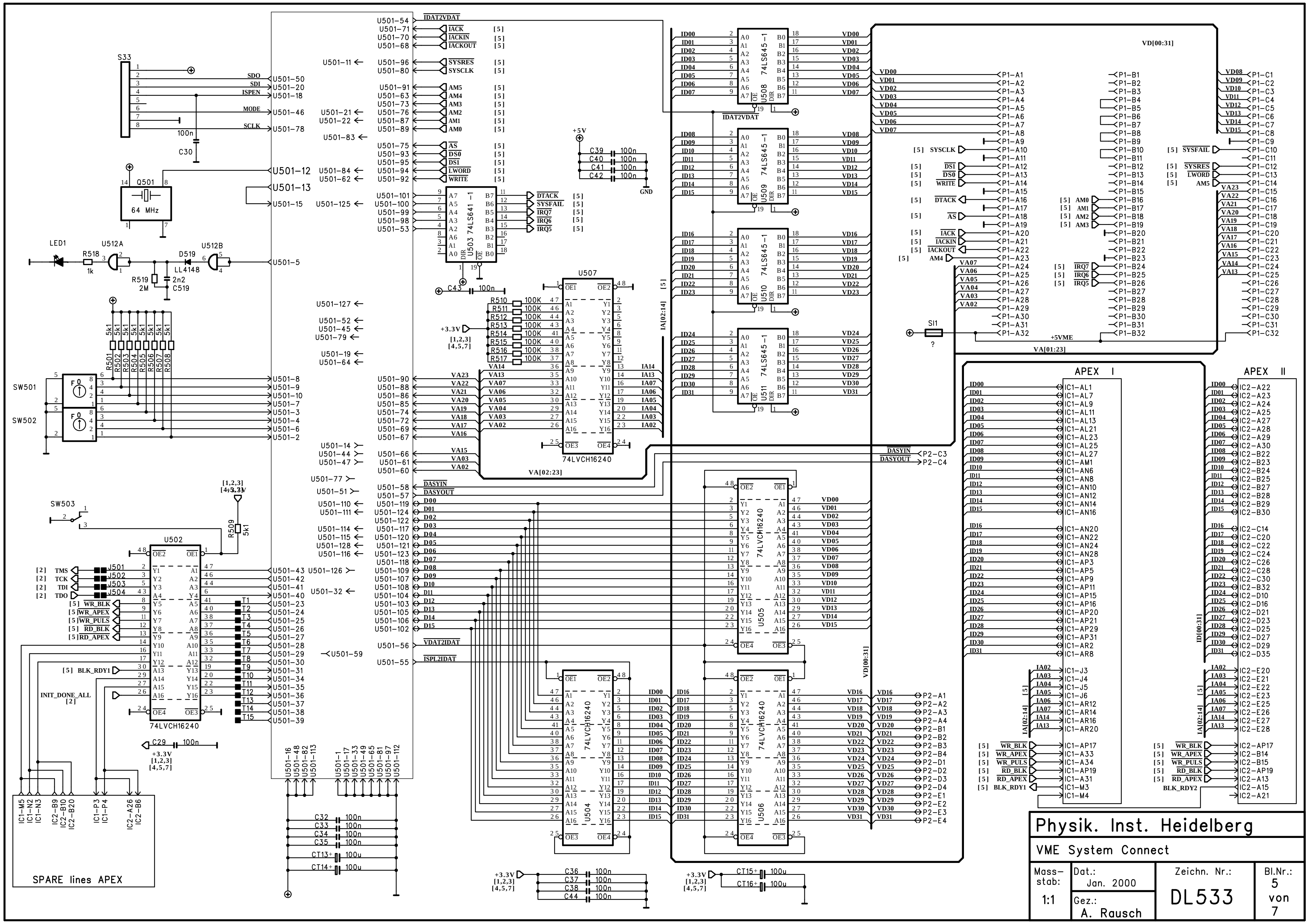


Abbildung C.6: Schaltplan der Triggerkarte, Lage 5/7

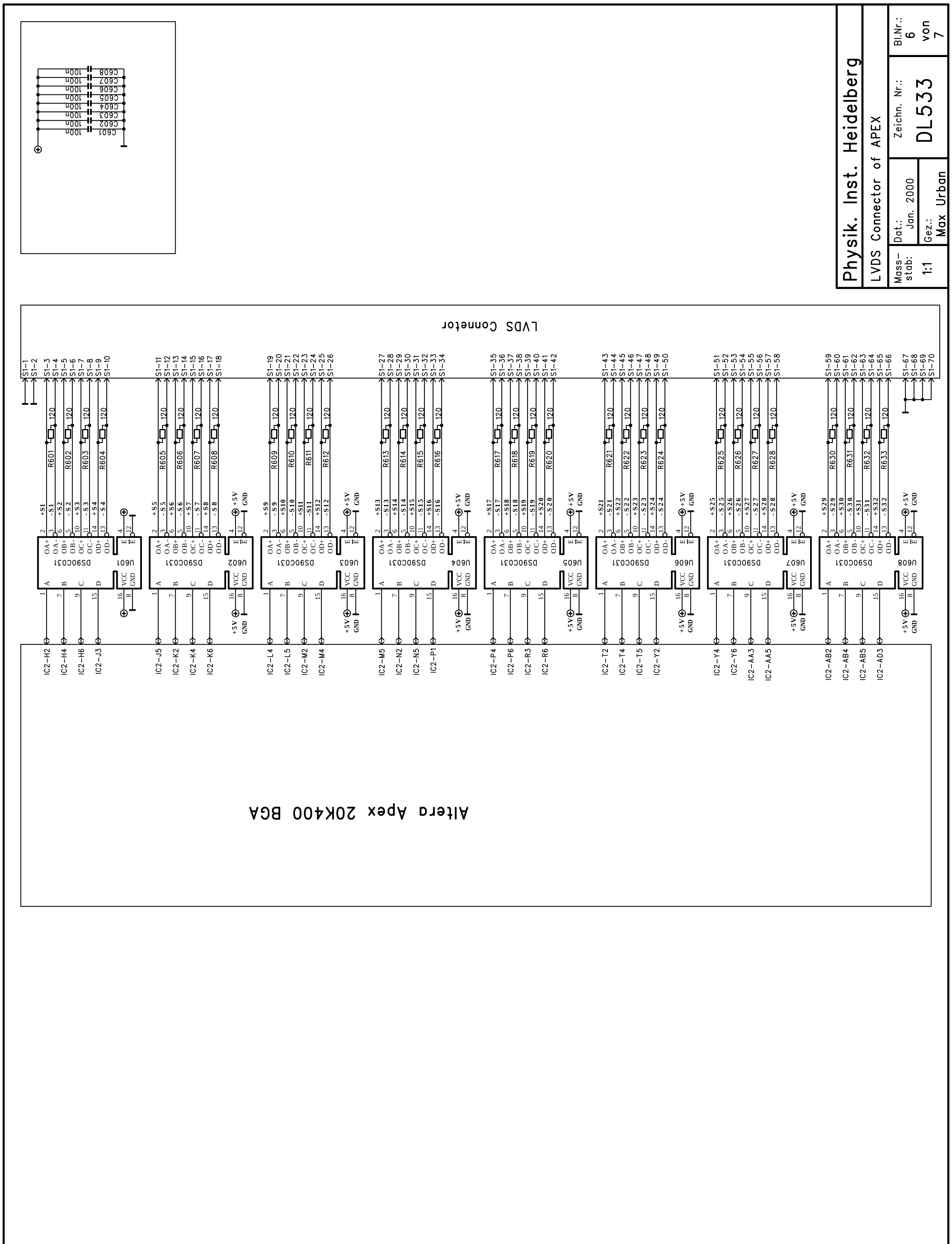


Abbildung C.7: Schaltplan der Triggerkarte, Lage 6/7

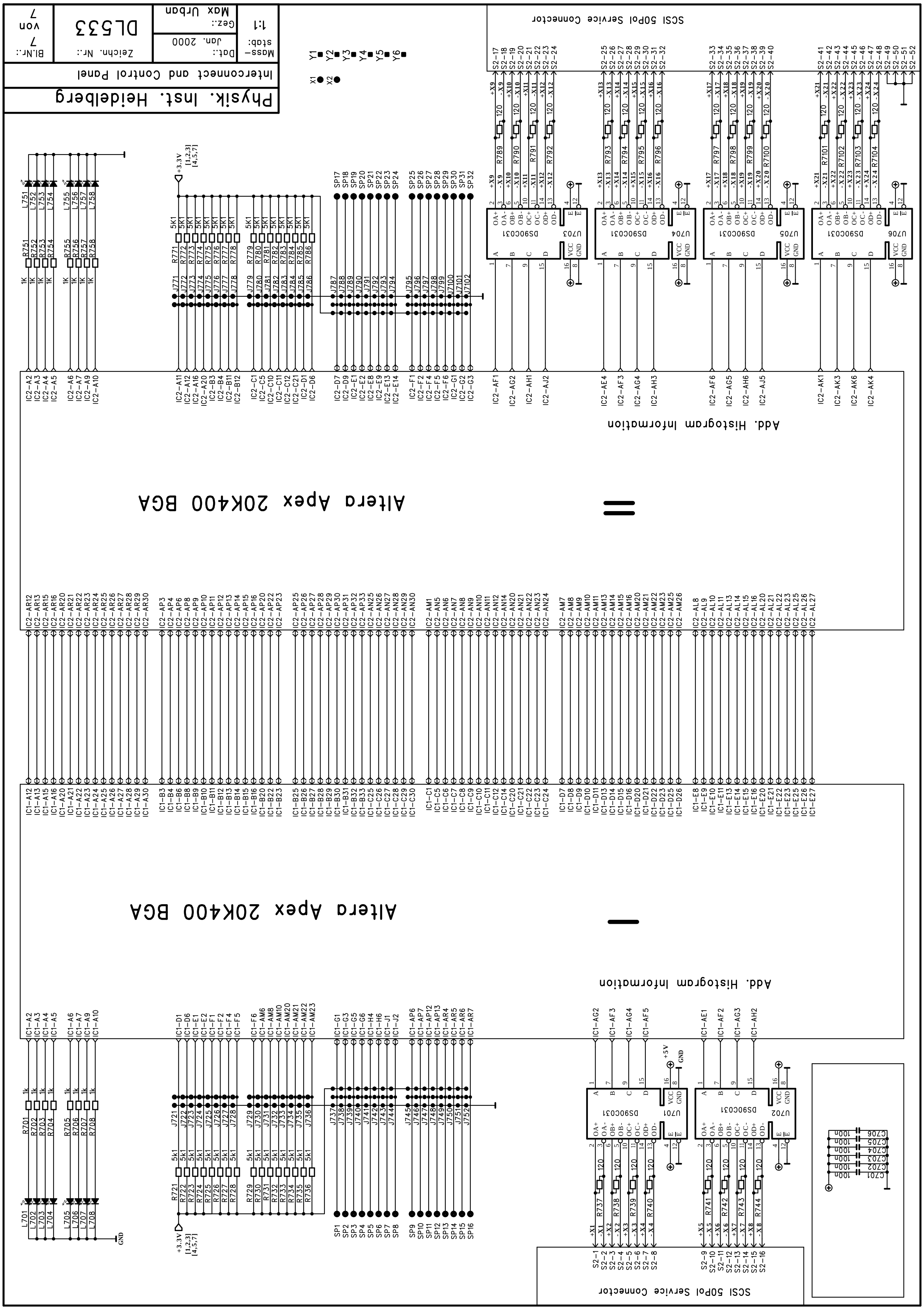


Abbildung C.8: Schaltplan der Triggerkarte, Lage 7/7

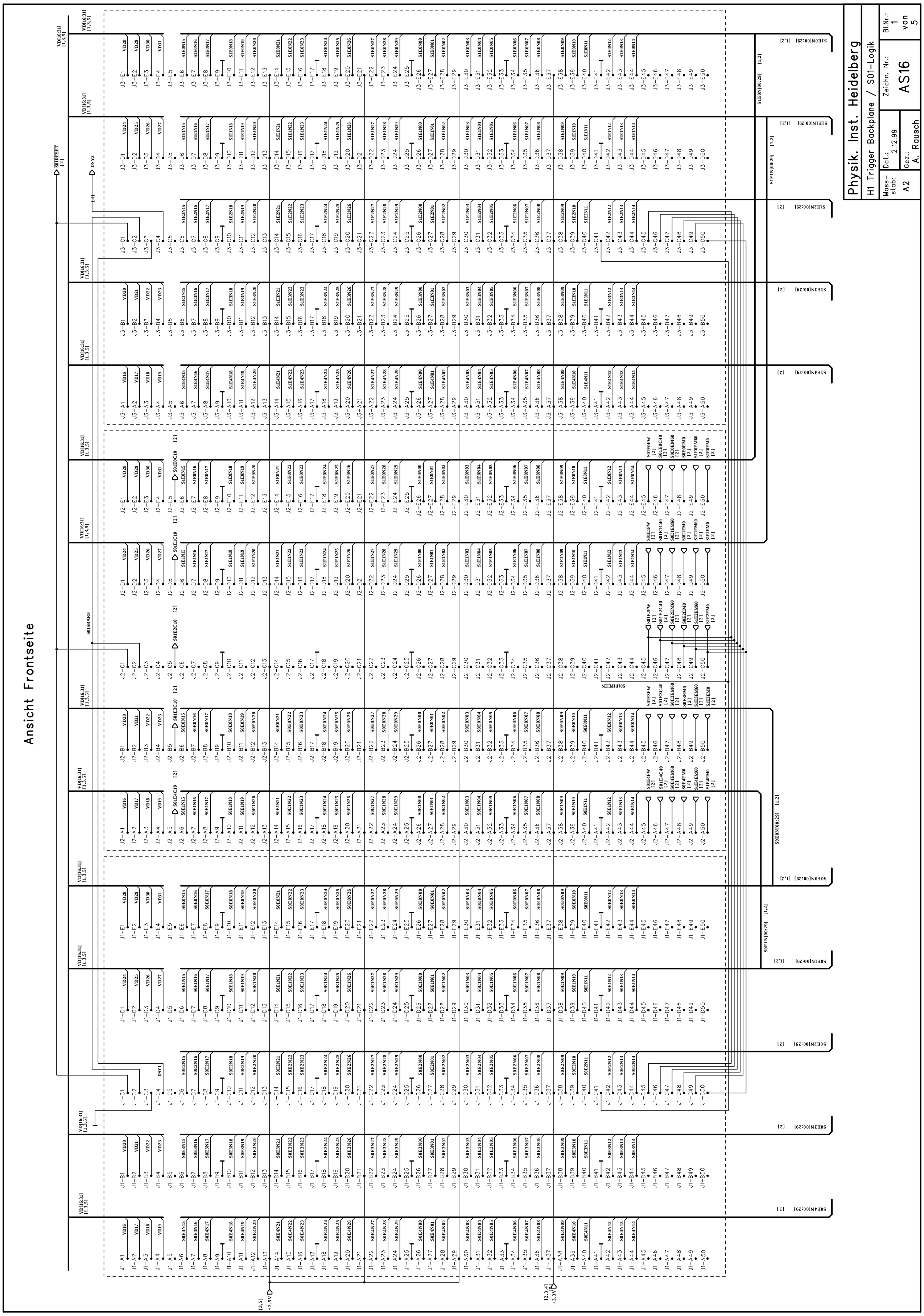


Abbildung C.9: Schaltplan der CIP-Backplane, Lage 1/5

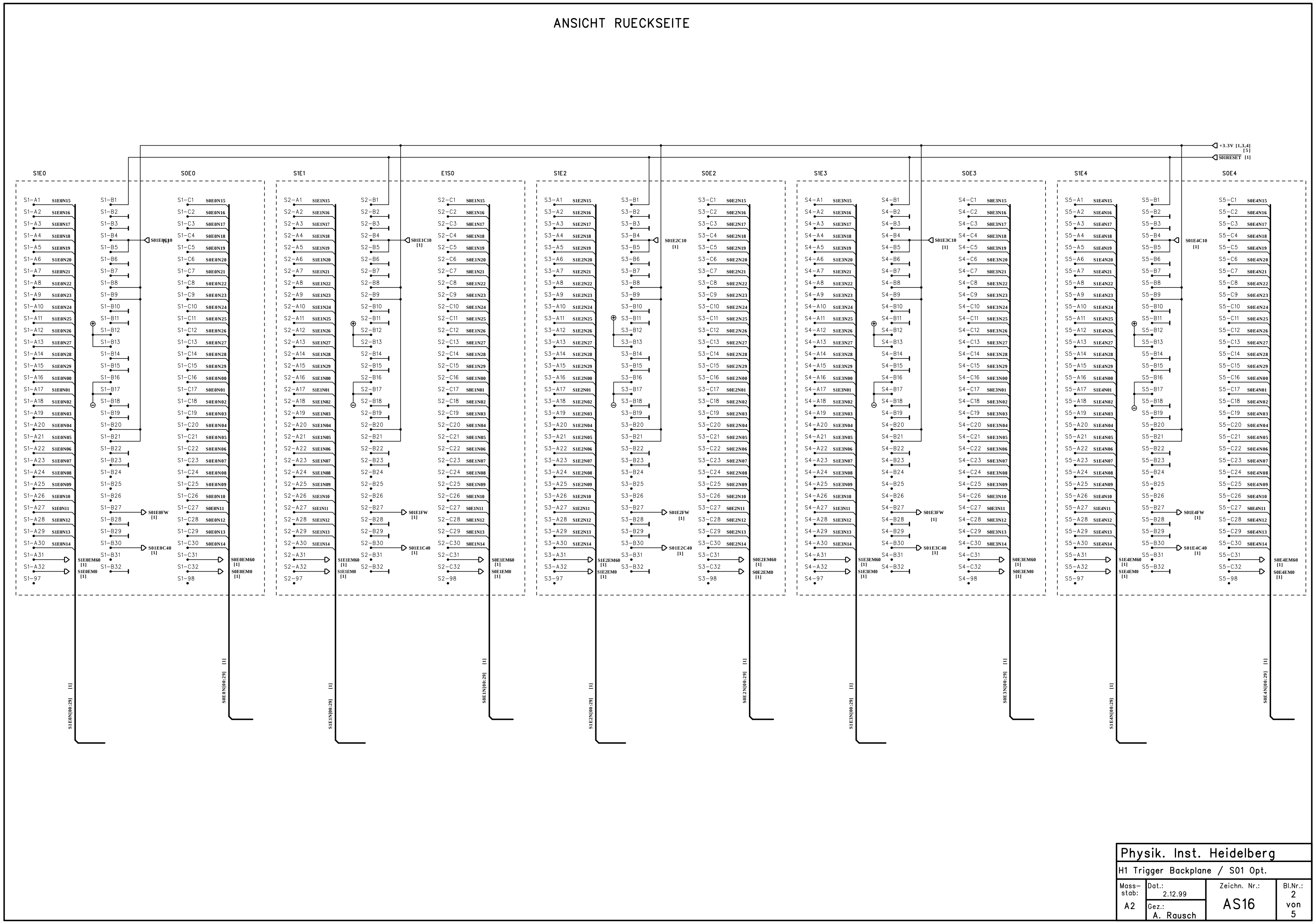


Abbildung C.10: Schaltplan der CIP-Backplane, Lage 2/5

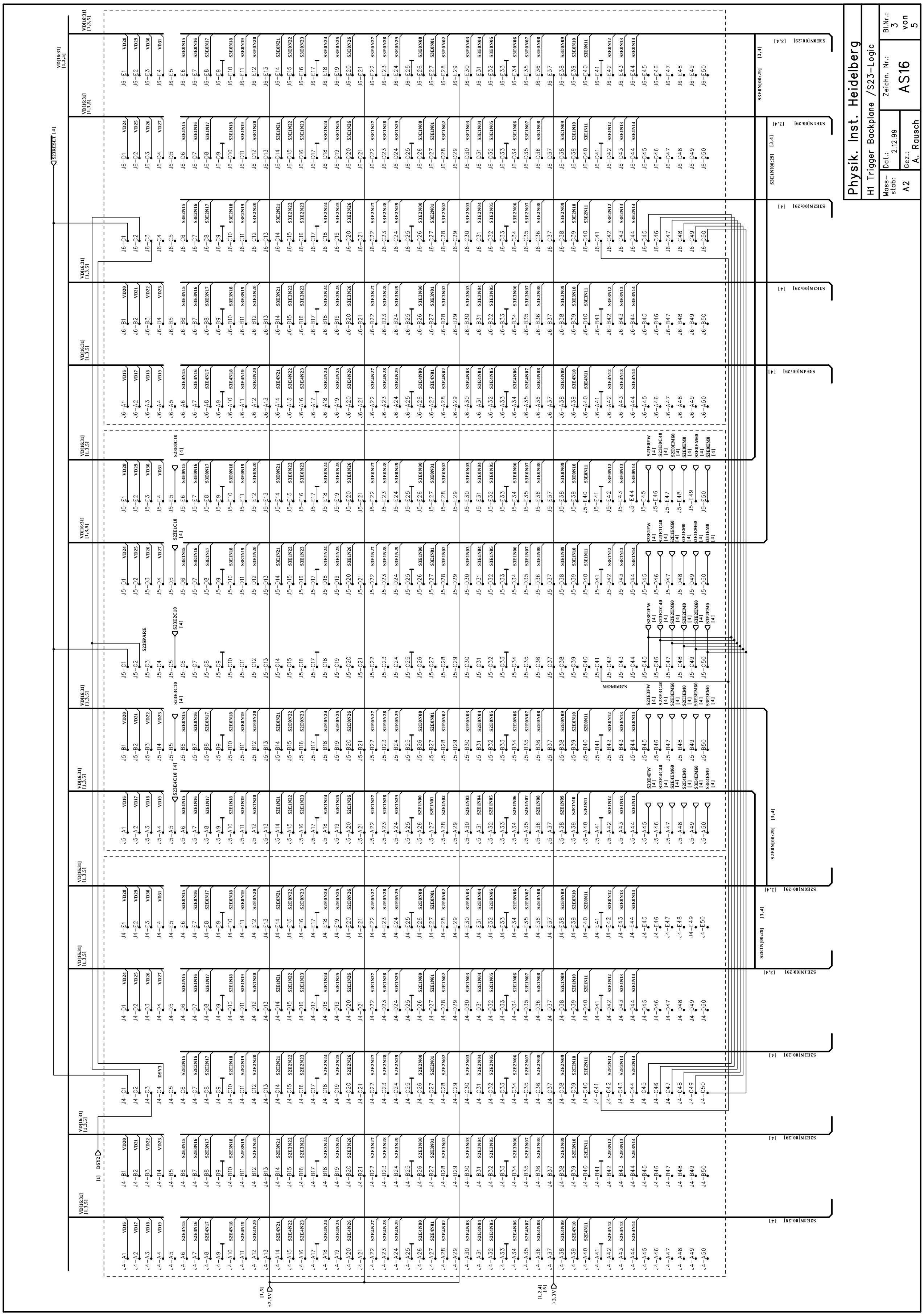


Abbildung C.11: Schaltplan der CIP-Backplane, Lage 3/5

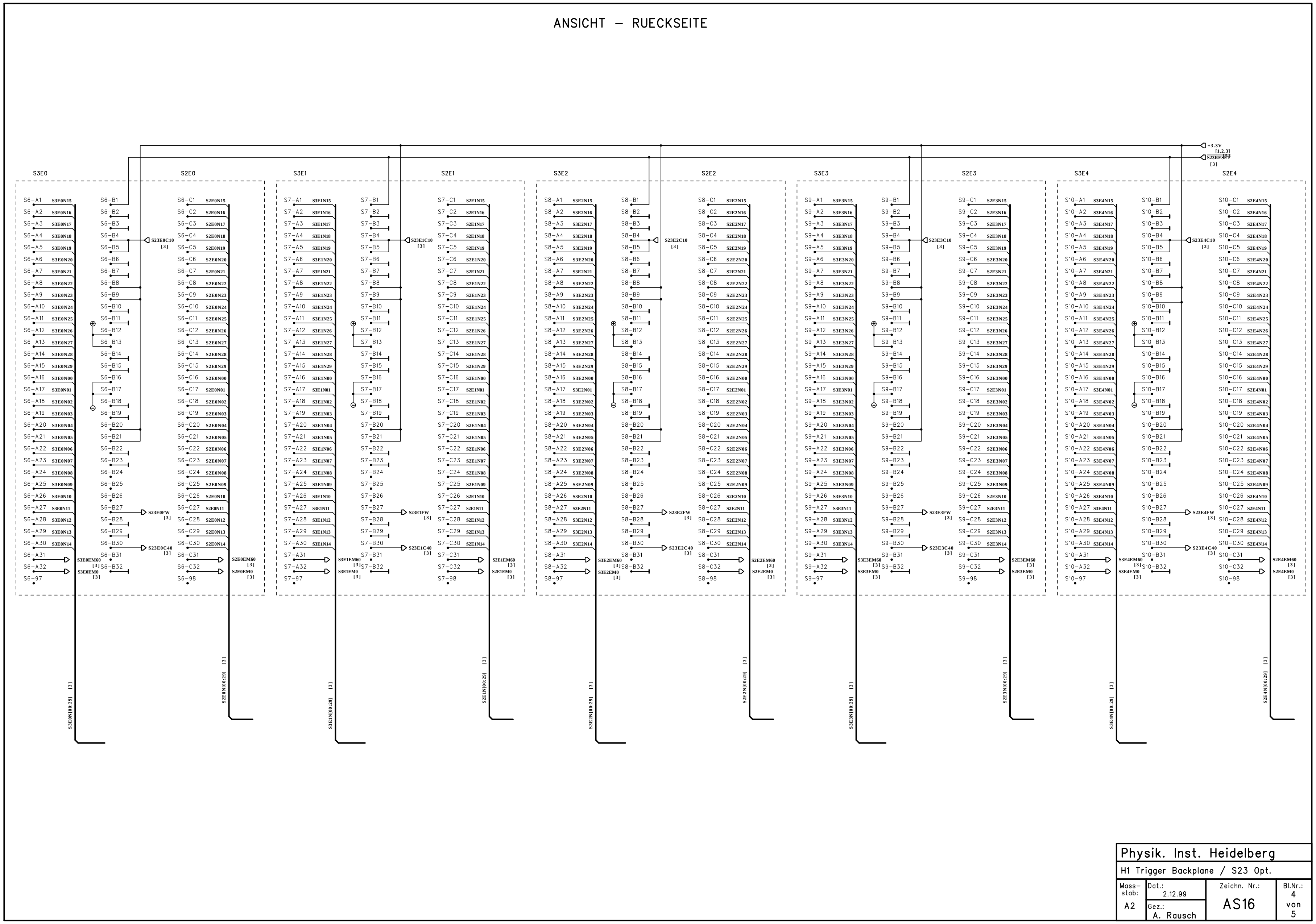


Abbildung C.12: Schaltplan der CIP-Backplane, Lage 4/5

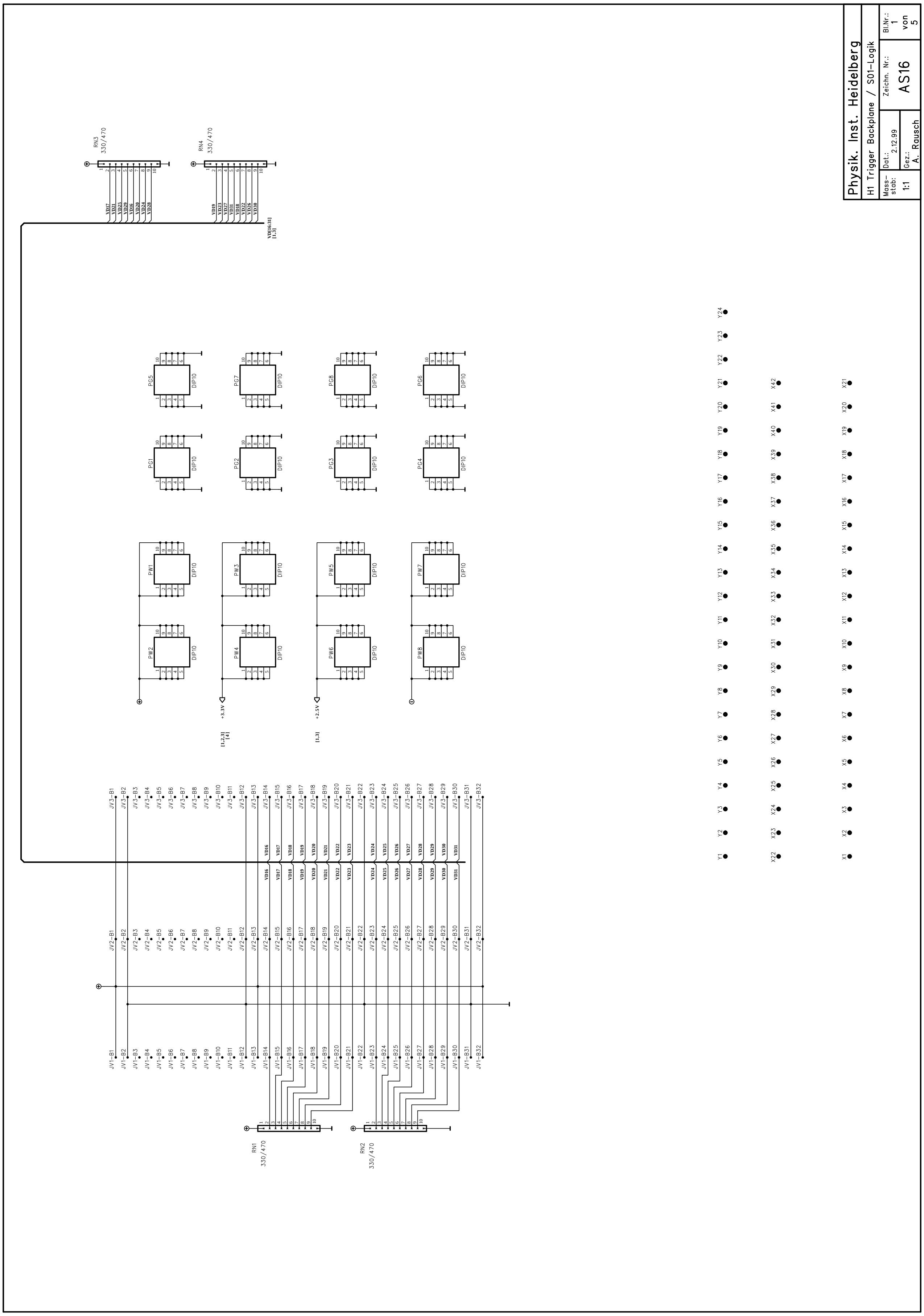


Abbildung C.13: Schaltplan der CIP-Backplane, Lage 5/5

Literaturverzeichnis

[Al99] *APEX20K Data Sheet Ver.2*, Altera, Mai 1999

[AN116] AN116 - Internal Note zur Programmierung der FPGAs, Altera, 1999

[AN80] AN80 - Internal Note zur Pinbelegung der FPGAs, Altera, 1999

[AN120] Altera, *MasterBlaster Serial/USB Communication Cable, ver.1*, 4/1999

[ASIC99] ASIC Labor, Universität Heidelberg, http://wwwasic.ihep.uni-heidelberg.de/asic/

[Ba98] Daniel Baumeister, *Entwicklung des CIPix* , Physikalisches Institut der Universität Heidelberg, 10/1998

[Be98] J.Becker,*Trigger Hardware Simulation* , Physikalisches Institut der Universität Heidelberg, 1998, http://www.desy.de/ jbecker/

[Be00] J.Becker, *CIP2000-Trigger and z-Vtx-Trigger* , Vortrag für H1-Trigger-Workshop in Ringberg, Januar 2000, http://www.desy.de/ jbecker/

[CIP00] Internet: CIP-Upgrade Seite, http://www.physi.uni-heidelberg.de/groups/he/h1/cipupgrade/cip.html

[CY00] Cypress Homepage, 5/2000, Internet: www.cypress.com

[DESY99] Internet: DESY Homepage, 3/2000, http://www.desy.de/

[Tex99] Internet: Tektronix Homepage, 5/2000,
http://www.tektronix.com/Measurement/signal_sources/

[EW00] Archiv der Elektronik-Werstatt, *Projekte DL532, DL533 und AS16*, Universität Heidelberg, 1999, 2000

[Gol95] U. Gloze,*VLSI-Entwurf eines RISC-Prozessors* , Vieweg, 1995

[H100] Internet: H1-Homepage, http://www-h1.desy.de/

[Ha95] K. ten´Hagen, *Abstrakte Modellierung digitaler Schaltungen.* , Springer-Verlag, Heidelberg, 1995

[HE98] HERA B Homepage, 3/2000, Internet: http://www-hera-b.desy.de/

[HER98] HEMES Homepage, 3/2000, Internet: http://dxhra1.desy.de/

[HI00] M. Hildebrandt, *Protokoll H1 CIP-Upgrade Meeting 05.04.00, zum globalen Systemtest*, Universität Zürich, 4/2000

[H198] H1 High Luminosity Upgrade 2000,*Progress Report on CIP and Level1 Vertex Trigger*, Juni 1998

[HU99] Beschreibung des L2-Triggers, 1999, Internet : http://www-h1.desy.de/h1/www/general/home/intra_home.html/Level2_Trigger

[HU98] H1 Collaboration, H1 Upgrade proposual, *ep-Physics bejond 1999* , H1-10/97-531

[Hx98] Helix Online Manual V2.1, 2/1999, http://wwwasic.ihep.uni-heidelberg.de/ feuersta/projects/Helix/helix/helix.htm

[Ko98] Michael Kollak, *Entwicklung einer Stripline-Auslese für Kathodensignale einer langen, zylindrischen Vieldrahtproportionalkammer*, Physikalisches Institut der Universität Heidelberg, 12/1998

[La96] Lattice Semiconductor Corporation, San Jose, *Mature Programmable High Density Devices* , (c)2000, http://www.latticesemi.com/products/devices/

[Lo98] Sven Löchner, *Charakterisierung und Entwicklung eines CIP-Auslese-ASIC für das H1-Upgrade-Projekt 2000* , Physikalisches Institut der Universität Heidelberg, 12/1998

[Ur00] M. Urban, *Programmcode zum Triggeralgorithmus* , Universität Heidelberg, 3/2000, http://www.physi.uni-heidelberg.de/ murban

[Lu99] S. Lüders,*Optical Link Elektronics - Interface to Backplane*, ETH Zürich, 3/2000

[Qu99] Screenshot aus der Quartus-Software, Version 1999.10, Copyright (c) 1995 - 2000 Altera Corporation

[Ra96] A. Rausch, *Dokumentation der VME-to-ISA Karte* , Universität Heidelberg, 1996

[Ra00] A. Rausch, *persönliche Mitteilungen* , Universität Heidelberg, 2000

[SE99] V. Lindenstruth, F. Lesser, *VHDL Einführungskurs im Rahmen des Graduiertenkollegs*, Universität Heidelberg, Graduiertenkollegs Heidelberg, 1999

[So99] R. Sonnleitner, M. Herrlein, *diverse Telefonate mit den Herren Sonnleitner und Herrlein, EBV-Elektronik*, München, 1999

[St00] Uwe Stange, *Charakterisierung und Weiterentwicklung des CIPix*, Physikalisches Institut der Universität Heidelberg, 12/1998

[St98] U. Straumann, *Skriptum zur Elektronikvorlesung* , 1998, Internet: http://www.physik.unizh.ch/ strauman/elektronik/

[St98] U. Straumann, *Definition der Padbezeichnung der neuen CIP-Kammer, Version 22 April 1999* , 1999, http://www.physik.unizh.ch/ strauman/nummerierung.ps, Internet: http://www.physik.unizh.ch/ strauman/elektronik/

[Sp87] VME International Physics Association (VIPA), *VME-Standard*, spezifiziert in ANSI/IEEE STD1014-1987, 1987

[Ti91] U. Tietze, Ch. Schenk, *Halbleiter-Schaltungs-Technik* , 10.Auflage, Springer-Verlag Heidelberg, 1991

Erklärung:

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

Heidelberg, den ....

Max Christoph Urban