Entwurf und Aufbau eines Zellularlogik-Triggers für das Crystal-Barrel-Experiment an der Elektronenbeschleunigeranlage ELSA

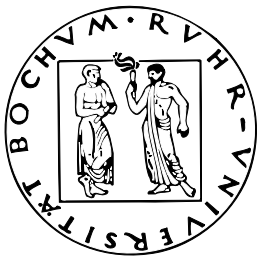
Dissertation

zur

Erlangung des Grades eines
Doktors der Naturwissenschaften

in der

Fakultät für Physik und Astronomie
der Ruhr-Universität Bochum



von

Holger Flemming
geb. in Wattenscheid

Bochum, im Juni 2001

Erster Gutachter : Prof. Dr. H. Koch (Institut für Experimentalphysik I)
Zweiter Gutachter : Prof. Dr. W. Meyer (Institut für Experimentalphysik I)

Zusammenfassung

Die vorliegende Arbeit behandelt den Entwurf, den Aufbau und verschiedene Tests eines Clustermultiplizitätstriggers für das Crystal-Barrel-Experiment am Bonner Elektronenbeschleuniger ELSA. Ziel dieses Experimentes ist die Untersuchung von Baryonenresonanzen in γ-induzierten Reaktionen unter Einsatz des aus 1380 CsI-Kristallen bestehenden Crystal-Barrel-Detektors, der zum Nachweis hochenergetischer Photonen dient. Ein durch ein Photon ausgelöster Schauer erzeugt in mehreren benachbarten Kristallen koinzidente Signale, die als Cluster bezeichnet werden. Aufgabe des Triggers ist es, diese Cluster zu identifizieren.
Der Trigger beruht auf dem für Anwendungen in der Teilchenphysik völlig neuartigen Konzept einer Zellularlogik. Er ist so aufgebaut, dass jedem einzelnen Kristall des Barrel-Detektors eine Logikzelle zugeordnet ist, die mit den seinen Nachbarkristallen zugeordneten Logikzellen Trefferinformationen austauschen kann. Alle Logikzellen bilden zusammen eine Matrix, in der die jeweiligen Verbindungen zwischen den Zellen den durch die Detektorgometrie bestimmten Nachbarschaftsbeziehungen zwischen den Kristallen entsprechen.
Für die Realisierung dieser Zellularlogik wurde als Full-Custom-Design ein Halbleiterchip in 0,8-μm-CMOS-Technik entwickelt, der 16 Logikzellen enthält. Die gesamte Logikmatrix besteht aus 105 dieser Chips, die auf zwei Platinen zusammengefasst sind, die ihrerseits zusammen mit der als VME-Einschub realisierten komplexen Steuerelektronik und den Zuführungsplatinen für die 1380 Signale in einem VNX-9-Crate untergebracht sind.
Nach seinem Einbau in das Bonner Experiment wurde der Trigger bereits bei zahlreichen Messungen erfolgreich eingesetzt. Dabei hat sich die außerordentliche Leistungsfähigkeit des ihm zugrunde liegenden Konzeptes erwiesen. Die wesentlichen Vorteile, die der Zellularlogiktrigger im Vergleich zu herkömmlichen Kalorimetertriggern bietet, deren Funktion auf der Projektion von Treffermustern beruht, sind die folgenden:

- Die Clusteridentifizierung ist vollständig unabhängig von der Topologie der registrierten Treffermuster.
- Aufgrund seiner Arbeitsweise hat der Trigger eine kurze Reaktionszeit. Die gesamte Verarbeitungszeit t_{Cl} für ein Treffermuster, das aus n Clustern besteht, liegt bei nur

$$t_{CL} = 0,8\mu\mathrm{s} + n \cdot 0,8\mu\mathrm{s}$$

Neben dem Entwurf und Aufbau des Triggers werden in der Arbeit auch erste Erfahrungen mit seinem Betrieb sowie unter seiner Verwendung erhaltene erste Meßergebnisse des CB-ELSA-Experimentes beschrieben.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Kernstück des CB-ELSA-Experimentes ist der Crystal-Barrel-Detektor, der zuvor über einen Zeitraum von etwa 10 Jahren am Antiprotonenspeicherring LEAR[1] des CERN[2] betrieben wurde und Daten mit hervorragender Qualität aus der Antiproton-Proton-Annihilation geliefert hat.
Nachdem der LEAR-Ring am Ende des Jahres 1996 abgeschaltet wurde, wurde der Crystal-Barrel-Detektor an die Bonner Elektronenbeschleunigeranlage ELSA[3] gebracht, wo er seit Januar 2000 in einer neuen Konfiguration wieder Daten liefert. Während in Genf die Mesonenspektroskopie und der Nachweis von sogenannten exotischen Zuständen Hauptziel der Untersuchungen waren, soll in Bonn vorrangig Baryonenspektroskopie mit photoninduzierten Reaktionen betrieben werden. Für solche Untersuchungen steht mit dem Crystal Barrel erstmalig ein Detektor zur Verfügung, der bei fast vollständiger Raumwinkelabdeckung mit guter Orts- und Energieauflösung Photonen aus dem Zerfall neutraler Reaktionsprodukte nachweisen kann.
Wegen der im Vergleich zu der Ereignisrate, mit der Daten von der Experimentauslese auf ein Speichermedium geschrieben werden können, sehr hohen Rate von Signalen aus der ersten Triggerstufe ist für diese Messungen eine zweite Triggerstufe erforderlich, mit der eine weitere Vorauswahl getroffen werden kann, um die Triggerrate zu reduzieren. Auch für das CERN-Experiment stand bereits ein Cluster-Multiplizitätstrigger zur Verfügung, mit dem die mit der Photonenanzahl korrelierte Clusteranzahl im Barrel ermittelt werden konnte [Br86]. Diese *Face* genannte Elektronik hatte aufgrund ihrer Arbeitsweise zwei Nachteile:

1. Abhängig von der Topologie eines Treffermusters konnten in einer Reihe von Situationen nicht alle Cluster korrekt identifiziert werden.

2. Die Verarbeitungszeit für ein typiches Treffermuster lag nur in der Größenordnung von 100 μs.

Da die Anforderungen an die zweite Triggerstufe an ELSA gegenüber dem Betrieb am LEAR wegen der an einem Photonenstrahl im Vergleich zu einem Antiprotonenstrahl deutlich unschärferen Triggerbedingungen wesentlich höher sind, musste ein neuer Multiplizitätstrigger entwickelt werden, der diese Nachteile nicht mehr aufweist.
Bei der Neuentwicklung wurde auf das für Anwendungen in der Teilchenphysik neuartige Konzept der Zellularlogik zurückgegriffen. Eine Zellularlogik besteht aus einer großen Zahl gleicher,

[1] **L**ow **E**nergy **A**ntiproton **R**ing
[2] **C**onseil **E**uropéen pour la **R**echerche **N**ucléaire, europäisches Kernforschungszentrum in Genf
[3] **El**ectron **S**tretcher **A**ccelerator

paarweise miteinander kommunizierender Schaltkreise und ermöglicht aufgrund ihrer hochgradig parallelen Arbeitsweise eine hohe Verarbeitungsgeschwindigkeit von komplexen Datenmustern. Ein wesentlicher Vorteil ist darin zu sehen, dass sich die topologischen Eigenschaften realer Detektorsysteme leicht auf die in der Zellularlogik realisierten Verbindungsmuster abbilden lassen. Dies eröffnet diesem Konzept weitreichende Anwendungsmöglichkeiten in der Teilchenphysik.

Kapitel 2

Physikalische Motivation

Für die Beschreibung der subnuklearen Physik ist heute das Standardmodell der Teilchenphysik weithin akzeptiert. Dieses Modell umfasst sowohl die Theorie der starken Wechselwirkung als auch die vereinheitlichte Theorie von elektromagnetischer und schwacher Wechselwirkung. Es unterscheidet zwischen elementaren Fermionen, die als Konstituenten der Materie dienen, und Eichbosonen, die als Träger der Kräfte zwischen diesen Materiekonstituenten wirken.
Die Konstituenten der Materie, die als elementar angenommen werden, d.h. denen keine innere Struktur zugeschrieben wird, werden wiederum in zwei Gruppen unterteilt, die Leptonen und die Quarks. Mann kennt sechs Leptonen, die in drei Familien eingeteilt werden: Das Elektron, das Myon und das τ-Lepton, sowie die dazugehörigen Neutrinos.
Ebenfalls in drei Familien werden die sechs bekannten Quarks eingeteilt. Diese Teilchen besitzen eine Reihe bemerkenswerter Eigenschaften. So tragen sie eine drittelzahlige Ladung und können im Gegensatz zu den Leptonen nicht isoliert beobachtet werden, sondern immer nur als Konstituenten von Hadronen.
Tabelle 2.1 listet die elementaren Teilchen des Standardmodells mit ihren wichtigsten Eigenschaften auf. Zu jedem aufgelisteten Teilchen gibt es zudem ein Antiteilchen.

Zwischen den Bausteinen der Materie wirken vier verschiedene Kräfte, von denen wir aus unserer täglichen Erfahrung die Gravitation und die elektromagnetische Wechselwirkung kennen. Erst im subatomaren Bereich manifestieren sich zwei weitere Wechselwirkungen, die *starke* und die *schwache Wechselwirkung*. Letztere zeigt sich z.B. im β-Zerfall des Neutrons, während die starke Wechselwirkung dafür verantwortlich ist, dass sich die Quarks zu Hadronen binden.

Fermionen	Familie 1	2	3	elektr. Ladung	Farbe	schwacher Isospin linkhdg.	rechtshdg.	Spin
Leptonen	ν_e	ν_μ	ν_τ	0	—	1/2	—	1/2
	e	μ	τ	-1		1/2	0	
Quarks	u	c	t	+2/3	r,b,g	1/2	0	1/2
	d'	s'	b'	-1/3		1/2	0	

Tabelle 2.1: Die Quarks und Leptonen. d', s' und b' entstehen durch Cabibbo-Rotation aus den Masseeigenzuständen d,s und b [Po93]

Wechselwirkung	koppelt an	Austausch-teilchen	Masse $(\mathrm{GeV/c^2})$	J^P
stark	Farbe	8 Gluonen (g)	0	1^-
elektromagn.	elektrische Ladung	Photon (γ)	0	1^-
schwach	schwache Ladung	$W^{\pm}, Z^0$	$\approx 10^2$	1
Gravitation	Masse	Graviton	0	2

Tabelle 2.2: Die vier Wechselwirkungen und ihre wichtigsten Eigenschaften[Po93]. Die Gravitation spielt in der Teilchenphysik so gut wie keine Rolle, das Graviton ist noch nicht gefunden.

Im Gegensatz zu klassischen Theorien, in denen die Kräfte durch kontinuierliche Felder beschrieben werden, müssen im mikroskopischen Bereich auch die Felder quantisiert werden. Dies führt zu Feldquanten, deren Austausch die Kraftwirkungen erzeugt (Tabelle 2.2).
Für die elektromagnetische Wechselwirkung ist die entsprechende Theorie die *Quantenelektrodynamik*, deren Vorhersagen in hervorragender Übereinstimmung mit Präzisionsmessungen stehen.
Auch bei der schwachen Wechselwirkung wurde diese Quantisierung erfolgreich durchgeführt, was zur Vorhersage der W- und Z-Bosonen als Feldquanten führte, die später auch am CERN gefunden wurden. Beide Wechselwirkungen konnten zudem zur *elektroschwachen Wechselwirkung* vereinheitlicht werden.
Für die starke Wechselwirkung gibt es ebenfalls eine Feldtheorie, die *Quantenchromodynamik*, kurz *QCD*. Sie schreibt den Quarks, zwischen denen Gluonen ausgetauscht werden, drei Farbladungen zu. Das Besondere hierbei ist, dass die Gluonen selber Farbladungen tragen und somit aneinander koppeln können.
Die QCD fordert, dass ein System nach außen farbneutral ist, was dazu führt, dass keine isolierten Quarks beobachtet werden können. Diese Eigenschaft nennt man *Confinement*.
Im Bereich hoher Impulsüberträge, wo die Quarks als asymptotisch frei angenommen werden können, lassen sich die Gleichungen der QCD störungstheoretisch lösen. Rechnungen in diesem Bereich liegen in guter Übereinstimmung mit Experimenten. Zu kleineren Impulsüberträgen hin steigt die Kopplungskonstante jedoch an und überschreitet schließlich den Wert 1, sodass eine quantenmechanische Störungsrechnung nicht mehr möglich ist.

2.1 Hadronenmodelle

Es gibt zwei Gruppen von Hadronen, die Mesonen, die aus einem Quark und einem Antiquark ($\mathrm{q\overline{q}}$) sowie die Baryonen, die aus drei Quarks (qqq) bestehen. Um trotz der oben beschriebenen Schwierigkeit Aussagen über den Aufbau der Hadronen machen zu können, wurden zahlreiche auf der QCD basierende Modelle entwickelt. Das einfachste ist das Konstituentenquark-Modell, das bereits eine Reihe der Hadroneneigenschaften, insbesondere die auftretenden Hadronenmultipletts erklärt. Die Hadronen werden dabei als aus Konstituentenquarks und -antiquarks zusammengesetzte Teilchen betrachtet. Welche Kombinationsmöglichkeiten es dabei gibt, folgt aus Symmetrieüberlegungen.

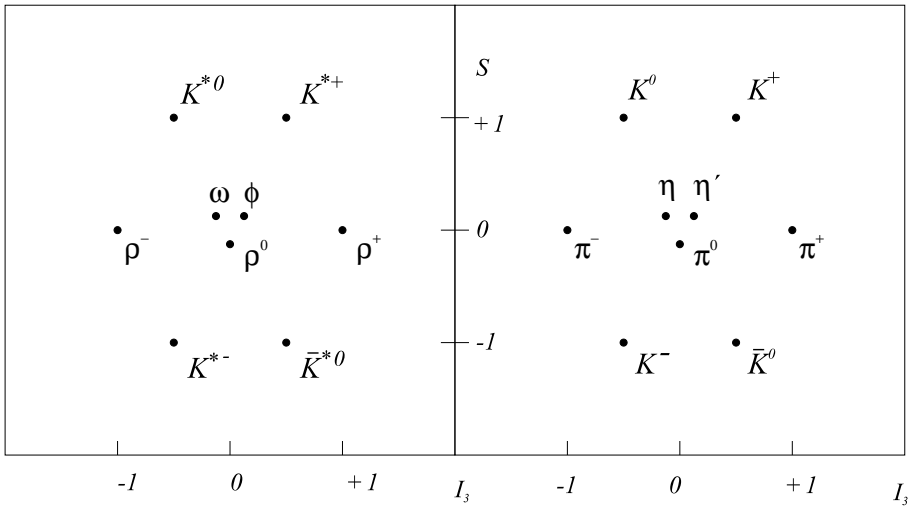


Abbildung 2.1: Die leichtesten Vektormesonen ($J^P = 1^-$) (links) und pseudoskalaren Mesonen ($J^P = 0^-$) (rechts), klassifiziert nach der dritten Komponente des Isospins I_3 und Strangeness S

2.1.1 Mesonen

Die Spins zweier Spin-$1/2$-Teilchen, wozu auch Quarks und Antiquarks gehören, können zum Gesamtspin 0 oder 1 koppeln. Je nachdem, ob das Meson einen Spin 0 oder 1 hat, spricht man von pseudoskalaren Mesonen oder von Vektormesonen. Dabei kennzeichnet das ‘pseudo’ die ungerade Parität dieser Teilchen, da skalare Größen normalerweise unter Paritätstransformation ihr Vorzeichen nicht ändern. Da die Kräfte, die zwischen den Quarks wirken, spinabhängig sind, haben Vektormesonen und pseudoskalare Mesonen gleichen Quarkinhalts stark unterschiedliche Massen.
Innerhalb dieser beiden Gruppen von Mesonen gibt es eine ganze Reihe von Zuständen. Beschränkt man sich auf leichte Mesonen, das sind solche, die nur aus u-,d- und s-Quarks bestehen, so erhält man zwei Mesonennonetts, die in Abbildung 2.1 dargestellt sind.

2.1.2 Baryonen

Die andere Gruppe der Hadronen sind die Baryonen, die aus drei Quarks zusammengesetzt sind. Da hier drei Spin $1/2$-Teilchen beteiligt sind, können Baryonen einen Gesamtspin von $1/2$ oder von $3/2$ haben. Baryonen sind somit Fermionen, d.h. die Gesamtwellenfunktion dieser Teilchen muss total antisymmetrisch gegenüber Vertauschung zweier Quarks sein. Dies hat Folgen für die möglichen Baryonenzustände.
Die Gesamtwellenfunktion eines Baryons setzt sich aus den Orts-, Flavour-, Spin- und Farbwellenfunktionen zusammen:

$$\psi_{total} = \xi_{Ort} \cdot \zeta_{Flavour} \cdot \chi_{Spin} \cdot \phi_{Farbe} \tag{2.1}$$

Bei Baryonen mit dem Gesamtdrehimpuls $3/2$ müssen alle Quarkspins parallel stehen, sodass die Spin-Wellenfunktion χ_{Spin} total symmetrisch ist. Beschränkt man sich auf Zustände mit

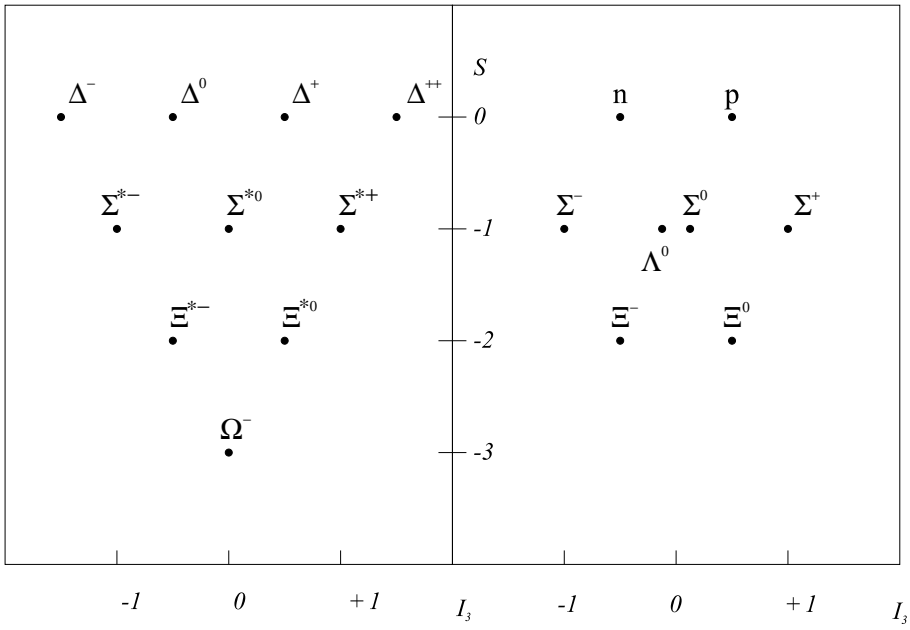


Abbildung 2.2: Die Zustände des Baryonendekupletts mit $J^P = 3/2^+$ und des Baryonenoktetts mit $J^P = 1/2^+$, klassifiziert nach der dritten Komponente des Isospins I_3 und Strangeness S

Drehimpuls $l = 0$, ist auch die Ortswellenfunktion total symmetrisch.
Da das Baryon ein farbneutrales Objekt ist, muss die Farbwellenfunktion dagegen total antisymmetrisch sein. Es gilt:

$$\phi_{Farbe} = \frac{1}{\sqrt{6}} \sum_{\alpha=r,g,b} \sum_{\beta=r,g,b} \sum_{\gamma=r,g,b} \epsilon_{\alpha\beta\gamma} |q_\alpha q_\beta q_\gamma > \tag{2.2}$$

wobei $\epsilon_{\alpha\beta\gamma}$ der antisymmetrische Tensor ist.
Um sicherzustellen, dass die Gesamtwellenfunktion total antisymmetrisch bleibt, muss die Flavour-Wellenfunktion $\zeta_{Flavour}$ symmetrisch sein. Betrachtet man wieder nur die drei leichtesten Quarks erhält man bei einem System aus drei Quarks 10 mögliche symmetrische Wellenfunktionen. Die dazugehörigen Teilchen sind in Abbildung 2.2 auf der linken Seite eingetragen.
Koppeln die Quarkspins dagegen zu einem Gesamtspin von $1/2$, führt das dazu, dass die Spinwellenfunktion nicht mehr rein symmetrisch, aber auch nicht rein antisymmetrisch gegenüber Vertauschung zweier Spinvektoren ist. Will man wieder eine total antisymmetrische Gesamtwellenfunktion erreichen, muss auch die Flavour-Wellenfunktion gemischt symmetrisch sein. Dies führt dazu, dass Spin-$1/2$-Baryonen mit den Quarkkombinationen uuu, ddd und sss nicht möglich sind. Somit ist kein Zustand mit Strangeness -3 möglich, mit Strangeness 0 sind nur noch zwei Zustände vorhanden: die bekannten Nukleonen. Die verbleibenden Zustände sind in der Abbildung 2.2 rechts eingetragen.
Die hier nur vereinfacht wiedergegebene Herleitung der möglichen Baryonenzustände lässt sich auch quantitativ mit gruppentheoretischen Argumenten zur SU(6)-Symmetrie der Quarks untermauern [Cl79].
Im Gegensatz zu den Mesonen, bei denen Teilchen und Antiteilchen jeweils aus einem Quark-Antiquarkpaar bestehen und somit im gleichen Multiplett auftauchen, bestehen bei den Baryo-

nen die Antiteilchen aus drei Antiquarks. Somit gibt es zu den Multipletts aus Abbildung 2.2 äquivalente Antibaryonenmultipletts.

2.1.3 Genauere Modelle

Baryonen lassen sich nicht nur im Grundzustand, sondern auch in angeregten Zuständen beobachten. Dies lässt sich nicht mehr mit einem statischen Konstituentenquarkmodell erklären, sondern erfordert andere Modelle. Ein erstes Modell, das in der Lage war, das Anregungsspektrum der Baryonen im Ansatz auch quantitativ zu beschreiben, wurde bereits Ende der sechziger Jahre entwickelt [Fa68].
Dieses Modell betrachtet das Baryon als harmonischen Oszillator dreier Konstitutentenquarks. Erweitert man dieses Modell noch um ein spin-abhängiges Hyperfeinpotential, erhält man, gemessen an den groben Vereinfachungen dieses Modells, bereits hervorragende Aussagen über das Baryonenspektrum und die Baryonenstruktur. Ein wesentlicher Schwachpunkt dieses Modells ist jedoch, dass es nicht relativistisch ist.
Derzeit existieren im Wesentlichen fünf Ansätze, um die Physik der Baryonen zu beschreiben. Die Modelle des Baryonenspektrums von ISGUR ET. AL. basieren auf einem langreichweitigen Confinement-Potential und einer kurzreichweitigen Komponente, die sich aus dem *Ein-Gluon-Austausch* ergibt. Mit solchen relativ einfachen, semirelativistischen Ansätzen gelang es, eine Vielzahl von baryonischen Anregungszuständen zu beschreiben [Ca86][Go85].
Einem anderen Ansatz folgend, kann die QCD auf einem *Gitter* in vielen ihrer Eigenschaften modelliert werden. Dabei können z.B. Mesonen und Baryonenmassen relativ genau berechnet werden, wobei jedoch unrealistisch hohe Quarkmassen verwendet werden müssen. Allerdings ist dieses Modell bisher nur im Bereich der Grundzustände erfolgreich anwendbar, womit jedoch ein wesentlicher Teil der Baryonenstruktur verloren geht, der sich im Anregungsspektrum und den Übergangsmatrixelementen zeigt.
Im Bereich niedrigerer Energien zeigt die Quantenchromodynamik einige bemerkenswerte Eigenschaften. Vernachlässigt man die Masse der leichtesten Quarks, weist die QCD eine *chirale Symmetrie* auf, wodurch wieder eine Störungsrechnung möglich ist. In der chiralen Störungstheorie treten acht masselose Bosonen auf, die sogenannten *Goldstone-Bosonen*. Durch die Masse der Quarks wird diese Symmetrie jedoch spontan gebrochen, was in der Theorie zur Entstehung eines $\mathrm{q\overline{q}}$-Kondensats und zu den Konstituentenquarks führt. Gleichzeitig erhalten die Goldstone-Bosonen Masse und können mit dem pseudoskalaren Mesonenoktett identifiziert werden[Bh88] [Me98].
Ein weiterer Ansatz, die QCD im Bereich niedriger Energien zu beschreiben, beruht auf *Instantonen*. Versuche, die Spektroskopie der leichten Mesonen auf einen Instantonen-Ansatz zurückzuführen, erwiesen sich als ausgesprochen erfolgreich.[Me98a]
Gittereichtheorien scheinen die Hypothese zu belegen, dass die qualitativen Eigenschaften der QCD nicht von der Anzahl der Farben N_c abhängt. Eine Reihe von Effekten konnte so in einer Störungstheorie in $1/N_c$ verstanden werden. Allerdings treten in diesen Entwicklungen keine Instantonen auf, sodass, sollten Instantonen eine wichtige Rolle in der starken Wechselwirkung spielen, dieser Ansatz Mängel aufweist.
Bislang fehlt noch eine Reihe von wichtigen Daten, um zwischen diesen Ansätzen unterscheiden zu können. Um diese Situation zu verbessern, sind am Bonner Elektronenbeschleuniger ELSA eine Reihe von Untersuchungen geplant.

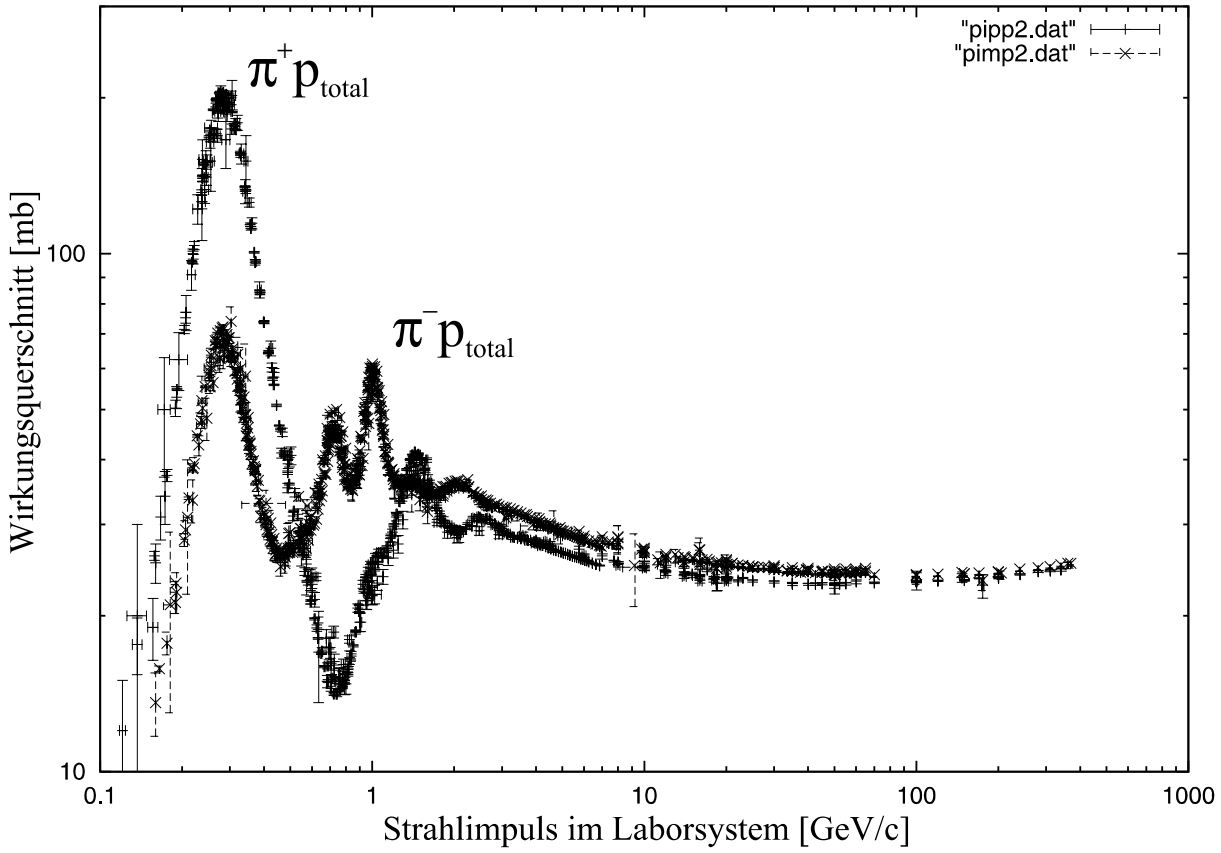


Abbildung 2.3: Totaler Wirkungsquerschnitt der π^+-Proton- und der π^--Proton-Streuung. Die Daten wurden aus verschiedenen Experimenten von der Particle Data Group zusammengestellt

2.2 Baryonenanregungen

Aus allen Modellen folgt, dass es eine ganze Reihe von Anregungszuständen der Baryonen gibt. Diese Zustände werden nach Masse, Bahndrehimpuls in der Nukleon-Pion-Streuung, Gesamtdrehimpuls und Isospin klassifiziert. Die Delta-Resonanz wird somit in der üblichen Notation als

$$P_{33}(1232)$$

bezeichnet. Dies bedeutet, dass diese Resonanz in der Pion-Nukleonstreuung als P-Welle ($l = 1$) beobachtet wird und eine Masse von 1232 MeV/c^2 hat. Der erste Index ist der doppelte Isospin, der zweite Index der doppelte Gesamtdrehimpuls der Resonanz. P_{33} bedeutet also, dass diese Resonanz einen Isospin von 3/2 und einen Gesamtdrehimpuls von 3/2 hat.
Solche Resonanzen wurden auch im Experiment gefunden. Abbildung 2.3 zeigt den totalen Wirkungsquerschnitt der π^+-Proton- und der π^--Proton-Streuung. In beiden Wirkungsquerschnitten ist bei einem Strahlimpuls von etwa 300 MeV/c deutlich die Delta-Resonanz zu sehen. Auch im weiteren Verlauf bis etwa zu einem Impuls von 3 GeV/c sind noch, jedoch deutlich schwächer, Resonanzen zu erkennen.
Die meisten Resonanzen, insbesondere deren Quantenzahlen, können aus diesen Wirkungsquer-

schnitten jedoch erst durch eine Betrachtung der Winkelverteilungen und durch detaillierte Partialwellenanalysen identifiziert werden.
Obwohl inzwischen in einer Vielzahl von Experimenten das Spektrum der Baryonen untersucht wurde, gibt es noch erhebliche Diskrepanzen zwischen der Theorie und den experimentellen Befunden. So wird von der Theorie eine Reihe von Anregungszuständen vorhergesagt, die bisher nicht experimentell beobachtet werden konnten.
Tabelle 2.3 zeigt den gegenwärtigen Stand der Suche nach Resonanzen. Lediglich die mit drei

$I = \frac{1}{2}$ Zustände	Status in $\Delta\pi$	$I = \frac{3}{2}$ Zustände	Status in $\Delta\pi$
$P_{11}(1440)$	***		
$D_{13}(1520)$	****		
$S_{11}(1535)$	*		
$S_{11}(1650)$	***	$P_{33}(1600)$	***
$D_{15}(1675)$	****	$S_{31}(1620)$	**
$F_{15}(1580)$	****		
$D_{13}(1700)$	**	$D_{33}(1700)$	***
$P_{11}(1710)$	**		
$P_{13}(1720)$	*		
		$S_{31}(1900)$	*
		$F_{35}(1905)$	**
		$P_{31}(1910)$	*
		$P_{33}(1920)$	**
$F_{15}(2000)$	*	$F_{37}(1950)$	****

Tabelle 2.3: Status der N*- und Δ-Resonanzen im $\Delta\pi$-Zerfall. Nur *** und **** Resonanzen sind als etabliert anzusehen [Ca98]

oder vier Sternen gekennzeichneten Resonanzen können als gesichert betrachtet werden. Insgesamt gibt es noch große Lücken im Baryonenspektrum.
Für diesen Befund gibt es zwei Erklärungsansätze. Der eine wurde bereits 1969 von LICHTENBERG gebracht, der vorschlug, dass Baryonen eine Quark-Diquark-Struktur besitzen [Li69]. Damit wäre ein innerer Freiheitsgrad des Baryons eingefroren, was automatisch zur Folge hätte, dass sich die Zahl der möglichen Anregungszustände veringern würde.
Eine andere Erklärung, die heute als wahrscheinlicher gilt und inzwischen auch von neueren Rechnungen untermauert wird, zielt darauf ab, dass bisher fast alle Untersuchungen des Baryonenspektrums mit πN-Streuexperimenten durchgeführt wurden. Somit besteht die Möglichkeit, dass diese fehlenden Resonanzen bisher lediglich nicht beobachtet wurden, weil sie nicht an den πN-Kanal koppeln. Statt dessen könnten solche Resonanzen jedoch an Ausgangskanäle wie $\Delta\pi$, Nη, Nρ, Nω oder Nη' koppeln.
Zur Klärung dieser Frage ist es somit notwendig, die Resonanzanregung von Nukleonen auch in diesen Kanälen zu untersuchen, wobei insbesondere die neutralen Kanäle interessante Untersuchungsmöglichkeiten bieten.

2.2.1 Suche nach $\Delta(1232)\eta$-Resonanzen in der Photoproduktion

Eine Fragestellung, die mit CB-ELSA bearbeitet werden soll, ist die Suche nach Δ-Resonanzen, die unter Emission eines η in den $\Delta(1232)$-Grundzustand zerfallen.
Im Bereich der Baryonen gibt es ein auffälliges, sich wiederholendes Muster von Resonanzen, die unter Emission eines η in ihren Grundzustand zerfallen. So zerfällt das N(1535)S_{11} mit einer hohen Verzweigungsrate von (15-35)% in Nη. Bei anderen Resonanzzuständen, insbesondere beim N(1650)S_{11}, das die gleichen Quantenzahlen besitzt, kann man dagegen keinen signifikanten Nη-Zerfall beobachten. Trotz der bedeutenden Rolle des N(1535)S_{11} als leichteste Dipolresonanz gibt es stark abweichende Erklärungen ihrer Struktur und der Anomalie ihres Zerfalls.
Neben Modellen von ISGUR und KARL [Is77] und von GLOZMAN und RISKA [Gl96], in denen die Zerfalls-Anomalie eine natürliche Erklärung durch eine Mischung von zwei S_{11}-Zuständen mit $s = 1/2$ und $s = 3/2$ (ISGUR und KARL) bzw. durch die Einführung eines Ein-Pion-Austausches als wesentliche Quark-Quark-Wechselwirkung (GLOZMAN und RISKA) findet, gibt es andere Modelle, in denen keine echte Nukleonresonanz benötigt wird. So erklären WEISE ET. AL.[Ka95] dieses Muster durch die Dynamik eines gekoppelten ΣK-pη-Systems. Dies wird durch eine Analyse von HÖHLER[Hö98] unterstützt, der in einer Amplitudenanalyse von η-Produktionsdaten ohne einen Resonanzpol für N(1535)S_{11} auskommt.
Dieses Muster wiederholt sich im Bereich der Λ-Resonanzen. Lediglich im Bereich der Δ-Resonanzen wurde diese Zerfallsanomalie noch nicht beobachtet. Um zwischen den unterschiedlichen Modellen zu unterscheiden und damit die Struktur der wichtigen N(1535)S_{11}-Resonanz zu klären, wäre es daher von großer Bedeutung zu sehen, ob sich dieses Muster auch im Bereich der Δ-Resonanz wiederholt. Dazu soll an CB-ELSA-Messdaten intensiv der Kanal γp $\rightarrow$p$\pi^0\eta$ studiert werden, in dem solche Resonanzen zu beobachten sein müssten [Sm99].

2.3 Exotische Materie

Neben der Existenz von Mesonen und Baryonen sagt die QCD aufgrund der Tatsache, dass Gluonen selbst Farbladungen tragen, auch exotische Teilchen voraus, die nur aus Gluonen oder aus Quark- Antiquarkpaaren mit einer Gluonenbeimischung bestehen: sogenannte *Gluebälle* und *Hybride*. Zudem wären Zustände aus vier ($\overline{\text{q}}\text{q}\overline{\text{q}}\text{q}$) und 6 (qqqqqq,qqq$\overline{\text{qqq}}$) Quarks denkbar.
Inzwischen wurden im Spektrum der Mesonen auch eine Reihe von Zuständen gefunden, von denen vermutet wird, dass sie gute Kandidaten für solche exotischen Zustände sind. Ein Beispiel dafür ist das $\pi_1(1400)$. Dieser Zustand hat einen Drehimpuls von $J^{PC} = 1^{-+}$, eine Quantenzahlkombination, die für normale q$\overline{\text{q}}$-Zustände nicht möglich ist. Offen hingegen ist die Frage, ob es sich dabei um ein Hybrid, also um einen q$\overline{\text{q}}$-Zustand mit Gluonbeimischung oder um einen Vier-Quark-Zustand handelt.
Diese Frage könnte mit Photoproduktionsexperimenten geklärt werden. Abbildung 2.4 zeigt den vorhergesagten Verlauf des Wirkungsquerschnittes in der Photoproduktion dieser Resonanz für den Fall, dass es sich um ein Hybrid handelt. Bei einer angenommenen Masse von 1,4GeV/c^2 sollte er bis auf nahezu 1 μb ansteigen. Der Wirkungsquerschnitt für die Photoproduktion unter der Annahme, dass es sich um einen Vier-Quark-Zustand handelt, ist nicht bekannt, sollte aber deutlich kleiner sein, sodass sich anhand von Photoproduktionsdaten entscheiden ließe, um welche Art eines exotischen Teilchen es sich beim $\pi_1(1400)$ handelt.
Zwei andere Resonanzen im Mesonenspektrum, deren Untersuchung Aufschlüsse über das Vor-

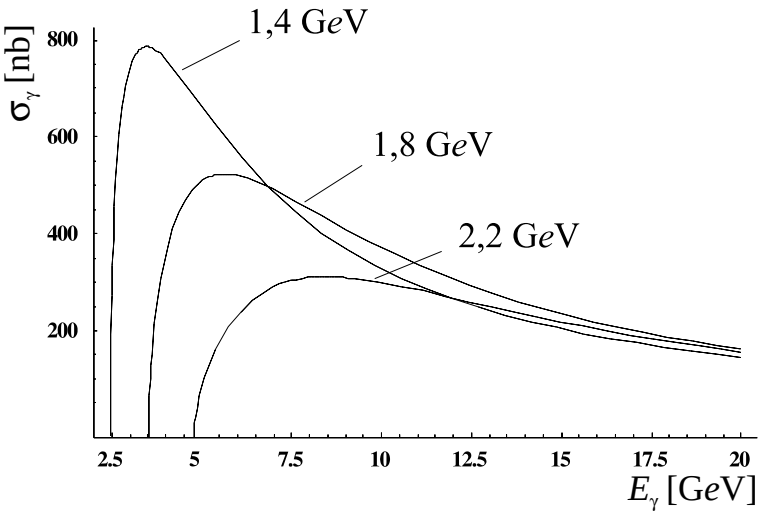


Abbildung 2.4: Wirkungsquerschnit für die Photoproduktion (γp $\rightarrow H$p) eines Hybrides H mit den Quantenzahlen $J^{PC} = 1^{-+}$ der Masse 1,4, 1,8 und 2,2 GeV/c^2als Funktion der Photonenergie nach Rechnungen von Afanasev und Page [Af98]

handensein exotischer Materie geben könnte, sind das $a_0(980)$ und das $a_2(1320)$. Die Interpretation des $f_0(1500)$ als Grundzustand eines skalaren Glueballs stützt sich darauf, dass es sich beim $a_0(980)$ nicht um einen $q\overline{q}$-Zustand handelt. In der Tat vermutet man, dass es sich beim $a_0(980)$ um ein $K\overline{K}$-Molekül handelt; bisher fehlt jedoch eine klare experimentelle Evidenz dafür [Sm99].
Eine Untersuchung der Produktionsverhältnisse von $a_0(980)$ und $a_2(1320)$ mit verschiedenen Targetmaterialien sollte eine Evidenz dafür liefern, da ein lose gebundenes $K\overline{K}$-Molekül in Kernmaterie schnell dissoziieren sollte und das Produktionsverhältnis mit z.B. einem Kohlenstofftarget somit kleiner sein sollte als mit einem Wasserstofftarget.

2.4 Experimentelle Voraussetzungen

Um die beschriebenen Probleme experimentell zu klären, ist der Einsatz eines Detektors erforderlich, der mit guter Raumwinkelabdeckung und hoher Effizienz die aus dem Zerfall der neutralen Mesonen stammenden Photonen nachweisen kann. Dazu ist der Crystal-Barrel-Detektor (CB), der an der Elektronenbeschleunigeranlage ELSA der Universität Bonn für diese Untersuchungen eingesetzt wird, in idealer Weise geeignet.
Zunächst ist dabei die Untersuchung von Reaktionen wie γ p $\rightarrow \Delta^+\pi^0 \rightarrow$ p $\pi^0\pi^0$[Th99],γ p $\rightarrow$ p $\pi^0\eta$ oder γ p $\rightarrow$ p η[Fö99] geplant, um das Baryonenspektrum im $\Delta\pi^0$und $\Delta\eta$-Kanal zu untersuchen. Um neben den neutralen Teilchen, die im Crystal-Barrel-Kalorimeter nachgewiesen werden, auch das Rückstoßproton unter kleinen Winkeln nachweisen zu können, stehen verschiedene Vorwärtsdetektoren zur Verfügung. Im folgenden Kapitel wird der experimentelle Aufbau des CB-ELSA-Experimentes genauer beschrieben.

Kapitel 3

Der experimentelle Aufbau

Kernstück des experimentellen Aufbaus des CB-ELSA-Experimentes ist der Crystal-Barrel-Detektor, der bis 1996 beim Experiment PS197 am Antiprotonenspeicherring LEAR des CERN im Einsatz war [CB92]. Nachdem der LEAR Ende des Jahres 1996 abgeschaltet wurde, wurde der Crystal-Barrel-Detektor nach Bonn gebracht und am Elektronenbeschleuniger ELSA installiert.

3.1 Die Elektronenbeschleunigeranlage ELSA

Die Bonner Beschleunigeranlage ELSA, deren Aufbau in Abbildung 3.1 dargestellt ist, erzeugt Elektronenstrahlen mit einer Energie von 0,5 bis 3,5 GeV. Zur Verfügung stehen sowohl eine herkömmliche Elektronenquelle mit elektronenstrahlgeheizter Glühkathode als auch eine weitere Quelle, die in der Lage ist, durch Bestrahlung eines GaAs-Einkristalls mit intensivem polarisiertem Laserlicht polarisierte Elektronen zu erzeugen. Beiden Quellen ist ein eigener Linearbescheuniger (Linac I und Linac II) nachgeschaltet, der die Elektronen für den Einschuss in das Synchrotron auf 20 MeV vorbeschleunigt.
Nach dem Einschuss in das Synchrotron werden die Elektronen dort auf eine Energie von 1,6 GeV beschleunigt, um dann in den Stretcherring ELSA injiziert zu werden. In Kreisbeschleunigern gibt es eine feste Beziehung zwischen Teilchenenergie, Feldstärke der Dipolmagneten und Bahnradius. Da in einem Synchrotron der Sollbahnradius konstant gehalten wird, muss die Führungsfeldstärke proportional zum Teilchenimpuls anwachsen. Deshalb ist es nicht möglich, mit einem Synchrotron einen kontinuierlichen Strahl zu erzeugen. Statt dessen entstehen beim Füllen des Rings mit dem Linac ein oder mehrere Elektronenpakete, sogenannte *Bunches*, die dann gemeinsam bis zur Endenergie beschleunigt werden. Die Zeit zwischen zwei aufeinanderfolgenden Füllungen beträgt beim Bonner Synchrotron 20 ms.
Die Hauptaufgabe des ELSA-Rings besteht darin, die Elektronenbunches, die vom Synchrotron in ELSA injiziert werden, longitudinal zu strecken, um das Tastverhältnis zu verbessern und einen fast kontinuierlichen Strahl zu erzeugen. Es besteht aber auch die Möglichkeit, die Elektronen in ELSA noch weiter auf Energien bis zu 3,5 GeV zu beschleunigen. Je nachdem von welcher Betriebsart Gebrauch gemacht wird, spricht man vom *Stretcher mode* oder vom *Post accelerator mode*. Tabelle 3.1 listet die wichtigsten Strahleigenschaften von ELSA auf. Der extrahierte Strahl kann zu drei Experimentierplätzen geführt werden, von denen derzeit nur zwei

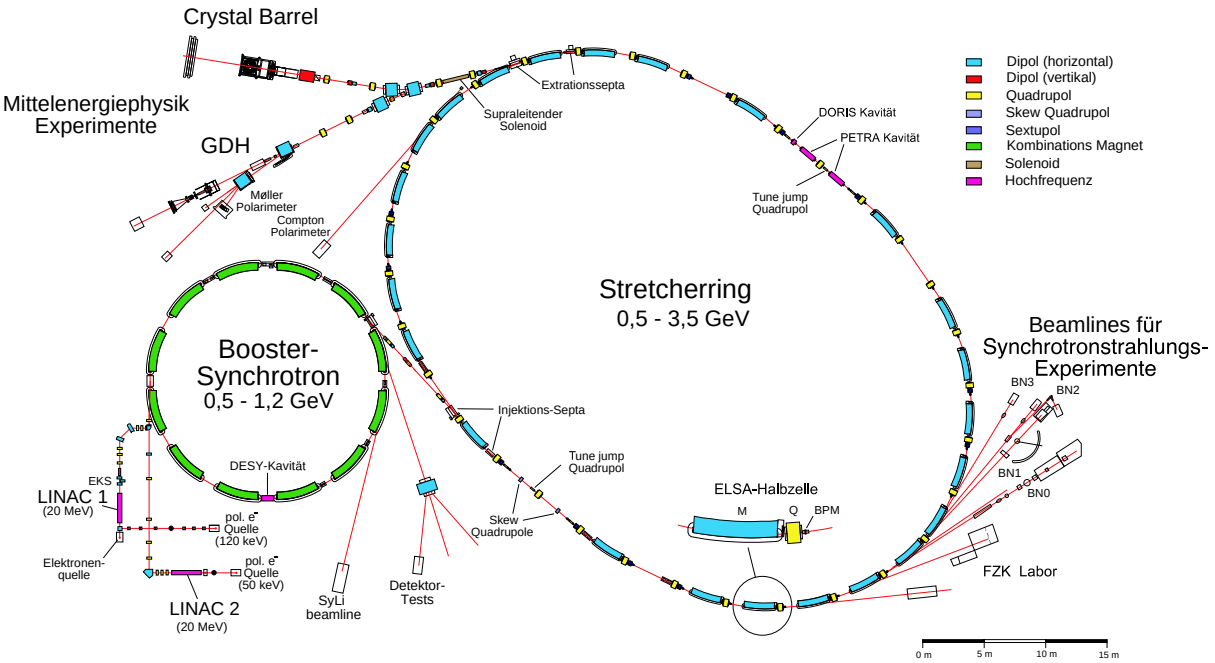


Abbildung 3.1: Die Beschleunigeranlage ELSA am Physikalischen Institut der Universität Bonn

besetzt sind. In Abbildung 3.1 erkennt man oben links das CB-ELSA-Experiment und direkt darunter das GDH-Experiment.

3.2 Das Crystal-Barrel-Experiment an ELSA

Abbildung 3.2 zeigt den Aufbau des Crystal-Barrel-Experimentes, wie er an ELSA realisiert wurde. Der in der Abbildung von links einfallende Elektronenstrahl trifft zunächst auf das Bremsstrahltarget, das mit einem Goniometer genau auf den Strahl ausgerichtet werden kann. Während der Primärelektronenstrahl durch den Taggermagneten abgelenkt und im Beam-Dump gestoppt wird, gelangen die erzeugten Bremsstrahlphotonen in das Flüssig-Wasserstoff-Target. Um dieses Target herum sind ein Faserhodoskop zur Detektion geladener Teilchen und das Barrel-Kalorimeter angeordnet. Im Abstand von einigen Metern folgt zudem zum Nachweis der in Vorwärtsrichtung emitterten Protonen die Flugzeitwand. Die einzelnen Detektorkomponenten werden im Folgenden genauer beschrieben.

	Stretcher Mode	Post-accelerator Mode
Strahlenergie	0,5 - 1,6GeV	0,5 - 3,5 GeV
Intensität	10 pA-100 nA	1 pA-20 nA
Tastverhältnis	98% (Makroskopisch)	bis 95% (Makroskopisch)

Tabelle 3.1: Die wichtigsten Strahleigenschaften von ELSA

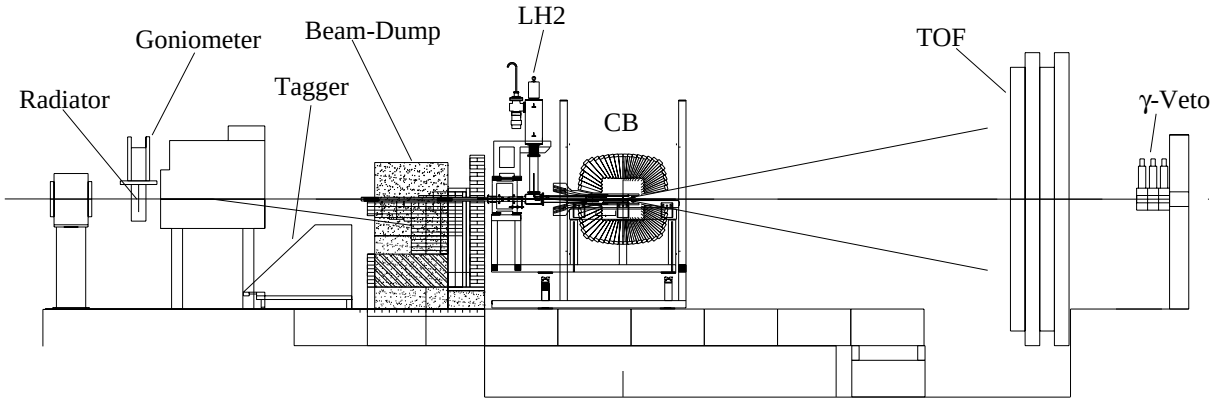


Abbildung 3.2: Der Experimentaufbau von CB-ELSA. Eine Beschreibung des Aufbaus findet sich im Text

3.2.1 Der Photonentagger

Die Experimente, die mit dem Crystal-Barrel-Detektor an ELSA durchgeführt werden sollen, erfordern einen Photonenstrahl, der in einem Bremsstrahltarget erzeugt wird, das sich hinter dem letzten Quadrupolmagneten im Strahlführungssystem für die Elektronen befindet. Als Target stehen wahlweise Folien aus Kupfer mit Dicken von $1/1000$, $3/1000$ oder $1/100$ Strahlungslängen zur Verfügung. Zudem besteht die Möglichkeit, mit einem Diamanteinkristall aus einem polarisierten Elektronenstrahl polarisierte Photonen zu erzeugen.
Die bei der Erzeugung der Photonen abgebremsten Elektronen werden von einem Magneten nach unten abgelenkt und in einem aus zwei Komponenten bestehenden Tagging-Spektrometer nachgewiesen, das vom Saphir-Experiment [Sc94] übernommen wurde.
Um ein schnelles Signal zur Bestimmung des Eintrittszeitpunktes des Photons in das Experiment zu erhalten, werden Szintillationszähler verwendet. Dazu wurde eine sogenannte Taggingleiter aus 14 Szintillatoren aufgebaut, die beidseitig von Photomultipliern ausgelesen werden. Um Akzeptanzlücken zu vermeiden, überlappen sich diese Szintillatoren leicht.
Eine genauere Ortsbestimmung und die damit verbundene bessere Impulsbestimmung der im Magnetfeld abgelenkten Elektronen wird mit zwei Drahtkammern mit 144 bzw. 208 Drähten erreicht, die vor den Szintillatoren angebracht sind. Damit gestattet das Tagging-Spektrometer eine Impulsbestimmung des Elektrons mit einer Auflösung von etwa 2 %, abhängig vom Impuls des Elektrons. Derzeit wird der Tagger mit einer über alle Kanäle summierten Gesamtrate von ca 5 MHz betrieben. Die Anordnung der Szintillatoren und Drahtkammern ist in der linken Bildhälfte der Abbildung 3.3 dargestellt.
Aus dem Impuls des abgebremsten Elektrons und der bekannten Energie des Primärstrahls lässt sich die Energie des erzeugten Bremsstrahlphotons ermitteln und das Photon somit *markieren*. Der Tagger arbeitet in einem Energiebereich von ca 30 % bis 95 % der Energie der Primärelektronen. Somit lassen sich bei einer Maximalenergie von 3,5 GeV von ELSA mit diesem Tagger Photonen bis zu einer Energie von 3,3 GeV markieren. Ein typisches Bremsstrahlspektrum mit dem Bereich, in dem die Photonen markiert werden können, ist rechts in Abbildung 3.3 dargestellt.
Der größte Teil der Elektronen des Primärstrahls passiert das Bremsstrahltarget ohne ein Photon zu erzeugen. Diese Elektronen werden vom Taggingmagneten weniger stark abgelenkt, flie-

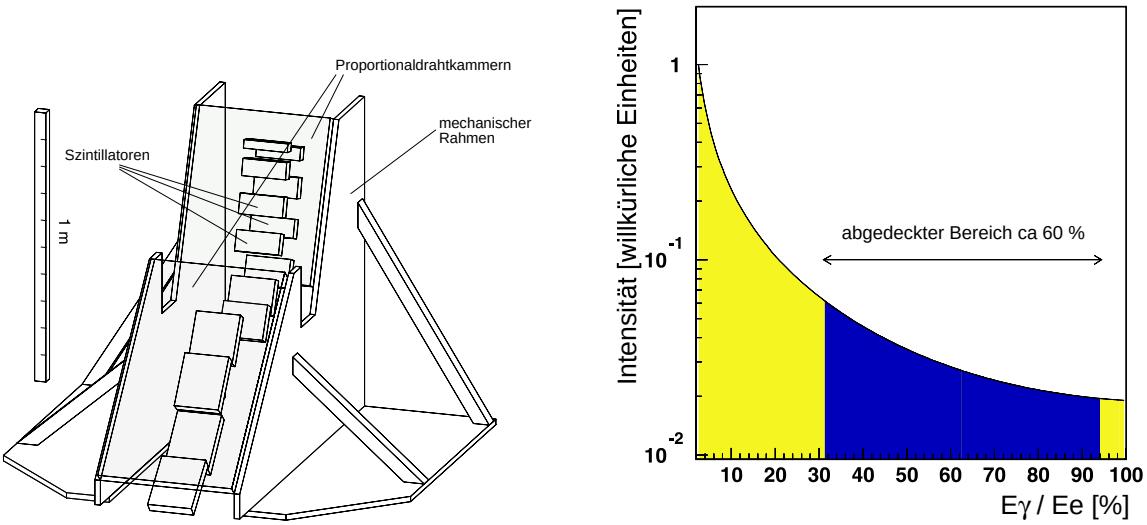


Abbildung 3.3: Taggingleiter des CB-ELSA Photon-Taggers (links) und ein typisches Bremsstrahlspektrum (rechts)

gen deshalb an der Taggingleiter vorbei und werden schließlich in einem Beam-Dump gestoppt, der aus mehreren Lagen Blei, Eisen, Polyethylen und Borkarbid besteht. Diese ausgeklügelte Zusammensetzung aus verschiedenen Materialien ist erforderlich, um den Detektor möglichst vollständig gegen den beim Auftreffen der Elektronen auf den Beamdump entstehenden Untergrund von Photonen und Neutronen abzuschirmen.

3.2.2 Das Wasserstofftarget

Die im Bremsstrahltarget erzeugten Photonen treffen im Zentrum des Crystal-Barrel-Detektors auf ein Flüssig-Wasserstoff Target ((1) in Abbildung 3.4) [Ko01], das ein Zweikreiskühlsystem verwendet. Im Primärkreis wird flüssiger Wasserstoff, der aus einem Verflüssiger ((5) in Abbildung 3.4) stammt, über ein 150 cm langes, horizontales Rohr zu einem in unmittelbarer Nähe des Targets befindlichen und aus Kupfer bestehenden Wärmetauscher geführt. Im Sekundärkreis wird dem Wärmetauscher gasförmiger Wasserstoff zugeführt, der daran zu flüssigem Wasserstoff kondensiert und über kurze Kaptonröhrchen in die Targetzelle fließt. Die Targetzelle selbst hat einen Durchmesser von 30 mm und eine Länge von 50 mm.
Dass in dem verwendeten Zweikreissystem die Targetzelle mit ihrem eigenen Wasserstoffreservoir ein abgeschlossenes System bildet, hat den Vorteil, dass eine eindeutige Beziehung zwischen dem Füllstand der Zelle und dem Druck in diesem System besteht. Somit lässt sich der Füllstand ständig überwachen. Zudem ergibt sich daraus die Möglichkeit, für Messungen an Neutronen Deuterium als Targetmaterial zu benutzen, ohne dass der gesamte Verflüssiger mit teurem Deuterium gefüllt werden müsste.
Sowohl bei der Verlegung der Rohrleitungen für die beiden Kühlkreisläufe, als auch bei der Positionierung des Wärmetauschers wurde darauf geachtet, den durch diese Komponenten erzeugten Untergrund so klein wie möglich zu halten.

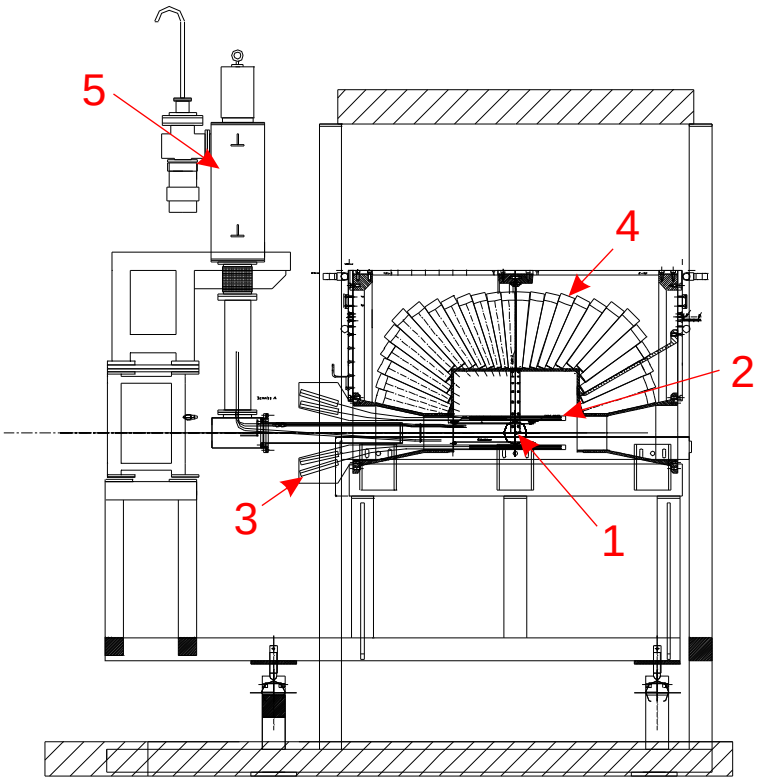


Abbildung 3.4: Schnittzeichnung durch den Aufbau des Crystal-Barrel-Detektors. Zu sehen sind 1. die Flüssig-Wasserstoff-Targetzelle, 2. das Faserhodoskop, 3. die Photomultiplier zur Auslese des Faserhodoskops, 4. die CsI-Kristalle des Kalorimeters und 5. der Wasserstoffverflüssiger

3.2.3 Das Faserhodoskop

Um das Target herum ist ein Faserhodoskop als Innendetektor zum Nachweis geladener Teilchen ((2) in Abbildung 3.4) angebracht. Dieser Detektor besteht aus drei konzentrisch aufgebauten Lagen von szintillierenden Fasern. Als Träger für die Szintillationsfasern dienen drei 40 cm lange Kohlefaserzylinder mit Radien von 5,8 cm, 6,1 cm und 6,4 cm. In diese Zylinder sind Rillen eingefräst, in die die szintillierenden Fasern eingeklebt sind.
Um die Durchstoßpunkte der Teilchentrajektorien bestimmen zu können, sind die Fasern gegeneinander geneigt. Die äußere Lage verläuft parallel zur Strahlachse, die inneren Lagen sind um $\pm 25°$ gegen die Strahlrichtung geneigt.
Insgesamt besteht der Detektor aus 513 szintillierenden Fasern, die über Lichtleitfasern an Multianoden-Photomultipier ((3) in Abbildung 3.4) zur Auslese angekoppelt sind.

3.2.4 Der Crystal-Barrel-Detektor

Das Faserhodoskop wird von dem aus 1380 CsI(Tl)-Kristallen ((4) in Abbildung 3.4) bestehenden Crystal-Barrel-Detektor [CB92] umgeben, der als elektromagnetisches Kalorimeter zum Nachweis von Photonen dient. Der gesamte Detektor ist in zwei spiegelsymmetrische Hälften aufgeteilt, in denen die Kristalle in mehreren Ringen angeordnet sind. Jede Hälfte besteht aus 10 Ringen mit je 60 Kristallen und einer Endkappe aus drei weiteren Ringen mit je 30 Kristallen. Damit ergibt sich, dass jeder Kristall einen Winkel von $6°$ in Polar- und $6°$ in Azimutrichtung, bzw. in der Endkappe von $12°$ in Azimutrichtung, abdeckt.
Der Detektor umfasst einen Polarwinkelbereich von $12°$ bis $168°$. Die Raumwinkelabdeckung

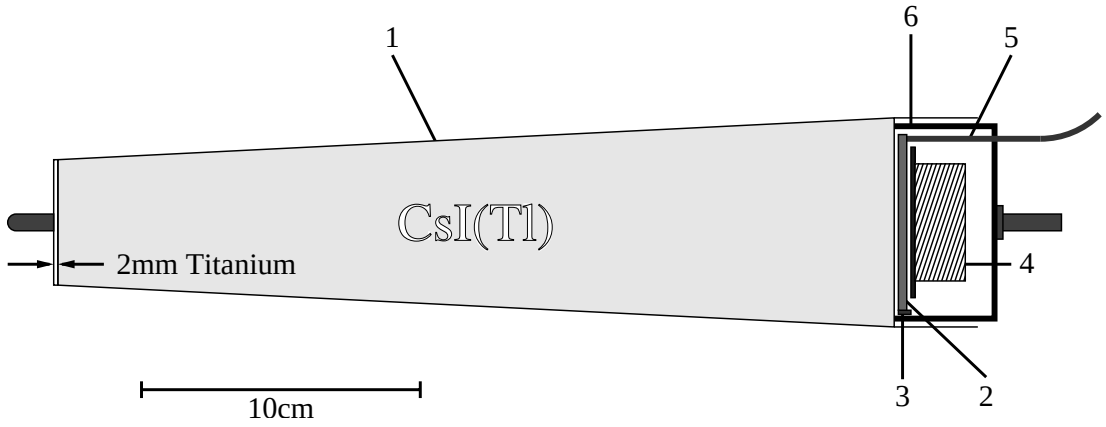


Abbildung 3.5: Schematische Darstellung eines Kristallmoduls des Crystal-Barrel-Detektors. 1 Cäsiumjodid-Kristall mit Titanhülle, 2 Wellenlängenschieber, 3 Fotodiode, 4 Leiterplatte mit Elektronik, 5 Lichtleitfaser, 6 Haltestruktur [CB92]

beträgt somit im Laborsystem nahezu 95 % von 4π. Es ist vorgesehen, für Experimente mit Vorwärtsdetektoren, die eine grössere Polarwinkelabdeckung als $\pm 12^\circ$ haben, die hintere Endkappe zu entfernen, um somit nur noch den Winkelbereich von 30° bis 168° mit dem Barrel abzudecken.
Abbildung 3.5 zeigt den Aufbau eines Kristallmoduls. Als Detektormaterial wird ein thalliumdotierter Cäsiumjodid-Kristall benutzt, der von einer dünnen Titanhülle umgeben ist. Der Kristall hat eine Länge von 30 cm, was 16 Strahlungslängen entspricht. An der Frontseite befindet sich eine 2 mm dicke Titanplatte, an der ein Führungsstift angebracht ist, mit dem der Kristall an der inneren Haltestruktur des Detektors befestigt wird. Auf der Rückseite des Kristalls befindet sich ein Wellenlängenschieber, der die Aufgabe hat, das Spektrum des vom Kristall erzeugten Lichts so zu verschieben, dass es mit dem Maximum der spektralen Empfindlichkeit der verwendeten Fotodioden übereinstimmt. Die Fotodiode ist an der Seite des Wellenlängenschiebers aufgeklebt. Ladungsempfindliche Vorverstärker für die Fotodioden und Kabeltreiber befinden sich auf einer kleinen Leiterplatte direkt hinter dem Wellenlängenschieber. Am Ausgang werden Pulse mit einer Anstiegszeit von 10 - 15 μs und einer Abfallzeit von ca 100 μs erzeugt. Die Pulshöhe beträgt etwa 1,5 V/GeV. Das elektronische Rauschen von 250 keV erlaubt die Messung von Energiedepositionen bis herab zu 1 MeV. Die relative Energieauflösung für Photonen beträgt etwa $2,5\%/(E/\ \mathrm{GeV})^{1/4}$.
Zur Monitorierung der Elektronik dient u.a. ein Lichtpulsersystem, dessen Lichtsignale über Glasfaserkabel in die Wellenlängenschieber eingekoppelt werden. Die Elektronik ist in eine Aluminiumhülse eingeschlossen, die mit der Titanhülle vernietet ist. An dieser Hülse ist eine Gewindestange befestigt, mit der das Kristallmodul an der äußeren Haltestruktur des Detektors befestigt wird.

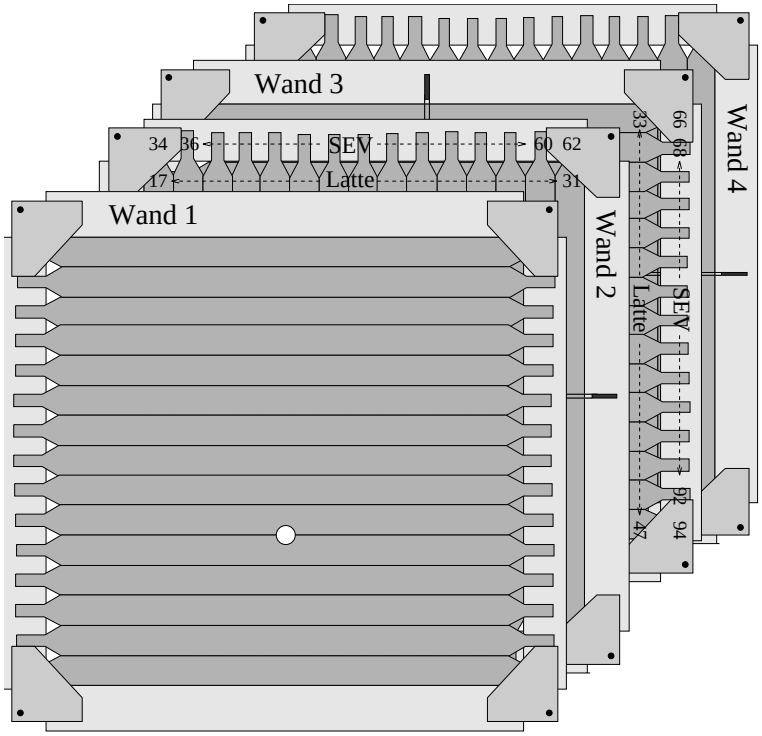


Abbildung 3.6: Aufbau der vier Time-of-Flight-Wände, die für das CB-ELSA-Experiment benutzt werden

3.3 Die Vorwärtsdetektoren

Für den Nachweis von in Vorwärtsrichtung fliegenden geladenen oder neutralen Teilchen ist es geplant, je nach den experimentellen Erfordernissen verschiedene Vorwärtsdetektoren einzusetzen. Im derzeitigen experimentellen Aufbau wird ein Flugzeitspektrometer zum Nachweis der Rückstoßprotonen eingesetzt, das aus vier Szintillatorwänden besteht.

3.3.1 Die TOF-Wand

Das Flugzeitspektrometer, das in Abbildung 3.6 dargestellt ist, besteht aus 4 Wänden mit jeweils 15 Szintillatorlatten. Die durch diese Wände abgedeckte Fläche beträgt $3 \times 3\ \mathrm{m}^2$. Die Szintillatoren haben eine Dicke von 5 cm und werden an beiden Enden von Photomultipliern ausgelesen. Damit ist es möglich, den Durchtrittsort eines geladenen Teilchens durch eine Szintillatorlatte durch die Bildung der Zeitdifferenz mit einer Auflösung von weniger als 5 cm zu bestimmen.
Die Flugzeit wird aus der Summe beider Zeitmessungen bestimmt, wobei die Zeitauflösung besser als 300 ps ist.
Die Szintillatorlatten der vier Wände sind jeweils um $90°$ gegeneinander verdreht. Damit ist eine Bestimmung des Durchstoßpunktes in x- und y-Richtung möglich.

3.3.2 Das Magnetspektrometer

Bei Reaktionen, die nahe einer Mesonenproduktionsschwelle ablaufen, werden die Rückstoßprotonen unter sehr kleinen Winkeln nach vorne emittiert. Zu ihrem Nachweis steht ein Magnetspektrometer zur Verfügung, das zwar einen sehr viel kleineren Akzeptanzwinkel als die TOF-Wand hat, dafür jedoch über eine wesentlich bessere Impulsauflösung verfügt.

Dieses Spektrometer besteht aus zwei Quadrupolmagneten, auf die ein impulsselektierender Dipolmagnet folgt. Zur Rekonstruktion der Teilchentrajektorien sind vor dem Dipolmagneten eine und hinter ihm vier weitere Drahtkammern angebracht. Die Gate-Signale für die Drahtkammern werden von vier Szintillatoren erzeugt, die mit den Kammern eine sandwichartige Detektoranordnung bilden.
Hinter diesem Detektorsystem befindet sich ein Gas-Schwellen-Cherenkovzähler, mit dem sich Protonen von niederenergetischem Untergrund und leichteren Teilchen trennen lassen. Das in diesem Zähler, in dem Frigen (CCl_2F_2) als Cherenkovgas benutzt wird, erzeugte Licht wird von zwei Hohlspiegeln auf Photomultiplier-Röhren fokussiert.

3.3.3 Der TAPS-Detektor

Ein weiterer Detektor, der zusammen mit dem Crystal Barrel in Bonn eingesetzt werden soll, ist das TAPS Spektrometer der Universität Giessen [No91]. Dabei handelt es sich um ein modulares System, das aus hexagonalen Bariumfluorid-Kristallen zusammengesetzt wird.
Abbildung 3.7 zeigt den Aufbau eines TAPS-Moduls. Jedes Modul enthält einem BaF_2-Kristall mit einer Länge von 250 mm, was 12 Strahlungslängen entspricht. Der Abstand zwischen einander gegenüberliegenden Seitenflächen beträgt 59 mm. Das mit dem Photomultiplier gekoppelte Ende des Kristalls hat eine zylindrische Gestalt und einen Durchmesser von 54 mm.
Vor jedem BaF_2-Kristall ist ein dünner Plastikszintillator angebracht, der eine Trennung von neutralen und geladenen Teilchen erlaubt. Diese Plastikszintillatoren werden ebenfalls über Lichtleiter von Photomultipliern ausgelesen. In die Stirnfläche eines solchen Plastikszintillators ist am äußeren Rand eine Rille eingefräst, in die ein Wellenlängenschieber in Gestalt einer Faser eingebettet ist, die den Szintillator praktisch umschließt. Die Auslese durch Photomultiplier erfolgt über eine an den Wellenlängenschieber angekoppelte Glasfaser.
Dieser Detektor ist in hervorragender Weise dazu geeignet, Photonen, die bei der Reaktion in Vorwärtsrichtung emittiert werden, nachzuweisen und stellt somit eine ideale Erweiterung zum Crystal Barrel dar. Abbildung 3.8 zeigt den geplanten Aufbau des TAPS-Detektors am CB-ELSA-Experiment. Dabei soll eine Wand aus 508 Kristallmodulen aufgebaut werden, die in Vorwärtsrichtung einen Polarwinkel von 30° abdeckt. Für den Primärstrahl wird in der Wand eine Öffnung mit einem Polarwinkel von 5° gelassen. Durch diese Öffnung besteht weiterhin die Möglichkeit, Protonen, die unter kleinen Öffnungswinkeln in Vorwärtsrichtung emittiert werden, hinter dem TAPS-Detektor mit der Flugzeitwand nachzuweisen.

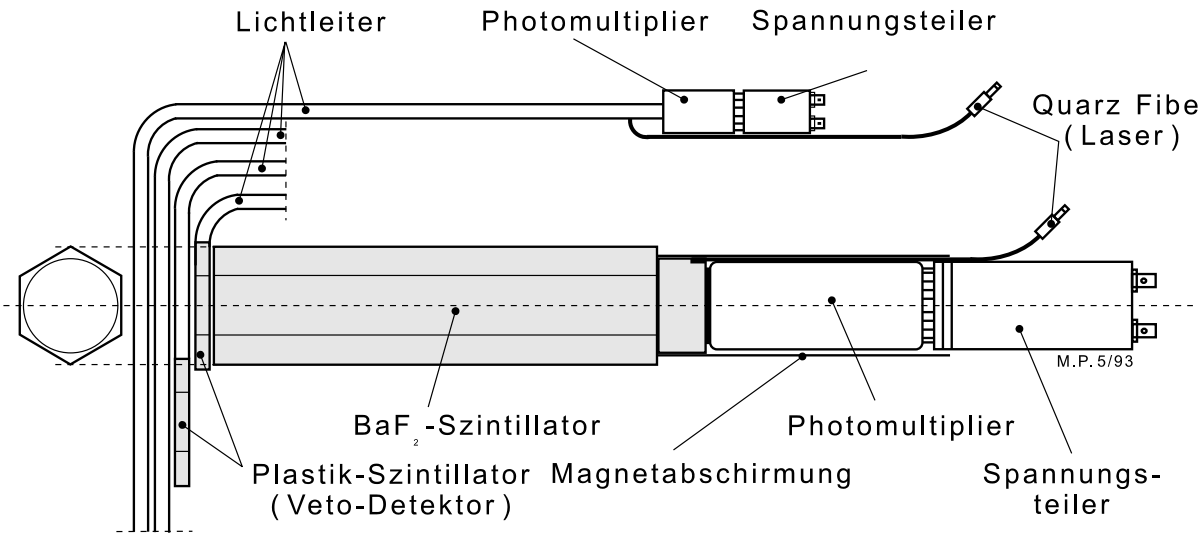


Abbildung 3.7: Ein Modul des TAPS-Detektors. Photonen werden mit einem hexagonalen BaF_2-Kristall detektiert, der von einem Photomultiplier ausgelesen wird. Als Veto-Detektor für geladene Teilchen dient ein dünner Plastik-Szintillator, der vor dem BaF_2-Kristall angebracht ist. Dieser wird über einen Wellenlängenschieber und eine Lichtleitfaser ebenfalls von einem Photomultiplier ausgelesen (Bildquelle: II. Phys. Inst. Uni Giessen)

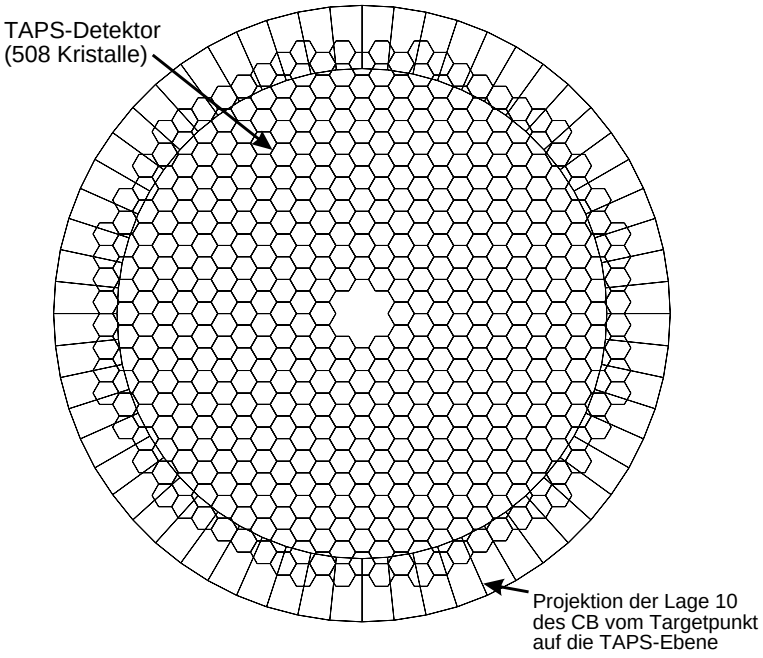


Abbildung 3.8: Schematische Darstellung der Anordnung der 508 BaF_2-Module des TAPS-Detektors zu einer Detektorwand. Am Rand ist zudem die Projektion der vom Targetort aus gesehenen äußersten Barrellage auf dem TAPS-Detektor eingezeichnet

Kapitel 4

Elektromagnetische Schauer im Kalorimeter

Die zentrale Komponente des CB-ELSA-Experimentes ist das elektromagnetische Kalorimeter, das dem Nachweis der überwiegend aus mesonischen Zerfällen stammenden Photonen durch die Registrierung elektromagnetischer Schauer dient. Durch die hohe Segmentierung des Kalorimeters ist es möglich, sowohl die Emissionsrichtung, als auch die Energie der Photonen mit hoher Genauigkeit zu bestimmen.
Die Wechselwirkung von Photonen mit den verwendeten Detektormaterialien erfolgt über die drei Prozesse Photoeffekt, Comptoneffekt und Paarbildung, wobei im Energiebereich, der bei CB-ELSA entscheident ist, die Paarbildung dominiert.
Elektronen und Positronen, die bei all diesen Prozessen freigesetzt werden, können auf ihrem Weg durch das Material mit den Hüllenelektronen der Atome in Wechselwirkung treten, wobei es zur Ionisation oder Anregung der Atome kommt. Bei Positronen kommt es zudem zur Anihilation, wobei zwei Photonen mit einer Energie von 511 keV ausgesendet werden.
Im Bereich hoher Energien dominiert jedoch die Erzeugung von Bremsstrahlung, die durch elektromagnetische Wechselwirkung der Elektronen und Positronen mit Atomkernen entsteht. Im Folgenden wird auf die einzelnen beteiligten Prozesse näher eingegangen. Dabei werden jedoch nur die wesentlichen Zusammenhänge kurz beschrieben. Eine detaillierte Darstellung findet sich z.B. in [Le94] und [Mu88].

4.1 Wechselwirkung von Photonen mit Materie

Trifft ein monoenergetischer Photonenstrahl mit der Intensität I_0 in Materie ein, so wird seine Abschwächung als Funktion der Eindringtiefe x durch die Gleichung

$$I(x) = I_0 e^{-\mu x} \tag{4.1}$$

beschrieben. Der Absorptionskoeffizient μ setzt sich, wie in Abbildung 4.1 zu ersehen ist, aus den Einzelabsorptionskoeffizienten für Photoeffekt, Comptoneffekt und Paarbildung zusammen.

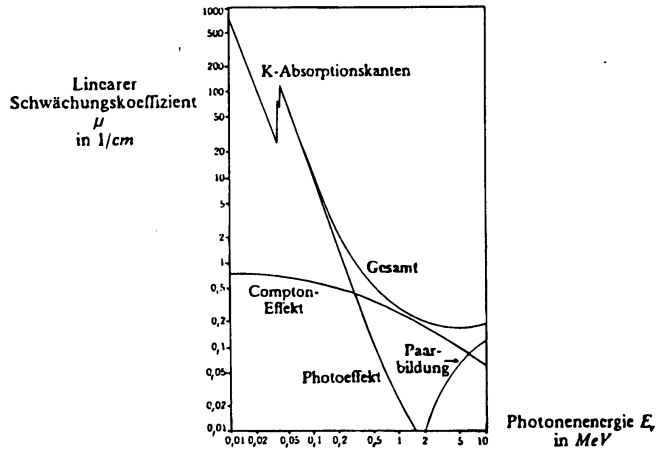


Abbildung 4.1: Verlauf des Abschwächungskoeffizienten μ. Gezeigt ist die Energieabhängigkeit der Abschwächungskoeffizienten für die Einzelprozesse und deren Summe.

4.1.1 Photoeffekt

Als Photoeffekt bezeichnet man die Absorption eines Photons in der Atomhülle unter Emission eines Elektrons. Die kinetische Energie des Elektrons ist die um die Bindungsenergie reduzierte Energie des Photons. Der Wirkungsquerschnitt ist im Wesentlichen abhängig von der Photonenenergie und von der Kernladungszahl Z des Atoms. Für den meist auftretenden Fall des Photoeffekts aus der K-Schale heraus, gilt, wenn E_B^K die Bindungsenergie des Elektrons ist:

$$\sigma_P \sim \begin{cases} \frac{Z^5}{E_\gamma^{\frac{7}{2}}} & \text{für } E_\gamma > E_B^K \\ \frac{Z^5}{E_\gamma} & \text{für } E_\gamma \gg E_B^K \end{cases} \tag{4.2}$$

4.1.2 Comptoneffekt

Beim Comptoneffekt handelt es sich um die Streuung eines Photons an einem freien Elektron, wobei ein Elektron als frei angenommen wird, wenn seine Bindungsenergie vernachlässigbar gegenüber der Energie des Photons ist. Der Energieübertrag ist durch die Kinematik des Prozesses festgelegt

$$E_e = E_\gamma \frac{(1 - \cos\vartheta)}{1 + \frac{E_\gamma}{m_e c^2}(1 - \cos\vartheta)} \tag{4.3}$$

und ist bei Rückwärtsstreuung des Photons ($\vartheta = 180°$) maximal. Aufgrund der DeBroglie-Beziehung verändert sich die Wellenlänge des Photons dabei um

$$\Delta\lambda = \lambda' - \lambda = \lambda_c (1 - \cos\vartheta) \tag{4.4}$$

mit der *Compton-Wellenlänge* des Elektrons $\lambda_c = 3,86 \cdot 10^{-13}$ m.

Der Wirkungsquerschnitt für Compton-Streuung wird durch die Klein-Nishina-Gleichung gegeben. Bei hohen Energien verhält er sich wie

$$\sigma_c \sim \frac{Z}{E_\gamma} \tag{4.5}$$

4.1.3 Paarerzeugung

Bei der Paarerzeugung wird das Photon unter Emission eines Elektron-Positronpaares im Feld eines Atomkerns absorbiert, wobei der Atomkern als Stoßpartner für die gleichzeitige Erhaltung von Energie und Impuls notwendig ist. Da für diesen Prozess zunächst die Ruheenergie von Elektron und Positron aufgebracht werden muss, gibt es für ihn eine Schwellenenergie von $2m_ec^2 = 1,02$ MeV. Im Energiebereich von 5 bis etwa 50 m_ec^2 wird der Wirkungsquerschnitt durch den folgenden Zusammenhang beschrieben:

$$\sigma_{Paar} \sim Z^2 \ln(E_\gamma/m_ec^2) \tag{4.6}$$

Danach steigt er nur noch langsam mit der Energie an und bleibt ab etwa 10^3 m_ec^2 konstant bei

$$\sigma_{Paar} \cong 12\alpha Z^2 {r_e}^2 \tag{4.7}$$

wobei α die Feinstrukturkonstante und r_e der klassische Elektronenradius sind.

4.2 Wechselwirkung von Elektronen und Positronen mit Materie

Elektronen und Positronen, die im elektrischen Feld eines Kerns abgebremst werden, senden Bremsstrahlung aus, wobei der Wirkungsquerschnitt für diesen Prozess nicht nur von der Energie der Elektronen, sondern auch vom Stoßparameter und von der Kernladungszahl Z abhängig ist. Eine detaillierte Betrachtung dieses Prozesses ist kompliziert, da die Hüllenelektronen das Feld des Kernes nach außen abschirmen. In der Theorie versucht man, diese Abschirmung mit Hilfe von Abschirmungsparametern zu berücksichtigen.
Grundsätzlich erhält man für den Energieverlust von Elektronen in Materie durch Bremsstrahlung den folgenden Zusammenhang:

$$\left(\frac{dE}{dx}\right)_{\text{rad}} \sim -E_eZ^2 \ln E_e \tag{4.8}$$

Bei hohen Energien gilt aufgrund der abschirmenden Wirkung der Hüllenelektronen nur noch

$$\left(\frac{dE}{dx}\right)_{\text{rad}} \sim -E_eZ^2 \tag{4.9}$$

Im Bereich hoher Energien dominiert der Energieverlust durch Bremsstrahlung, wogegen bei sehr kleinen Energien der Bremsstrahlungsverlust klein wird. Hier dominiert dann die durch elektromagnetische Wechselwirkung zwischen den Elektronen und Positronen mit den Hüllenelektronen hervorgerufene Ionisation. Das Verhältnis des Energieverlustes durch Strahlung zu dem durch Ionisation beträgt

$$\frac{(dE/dx)_{\text{rad}}}{(dE/dx)_{\text{ion}}} \cong \frac{ZE_e}{1600m_ec^2} \tag{4.10}$$

Dementsprechend definiert man als kritische Energie, bei der Strahlungs- und Ionisationsverluste gleich groß sind

$$E_{\text{krit}} \cong \frac{1600m_ec^2}{Z} \tag{4.11}$$

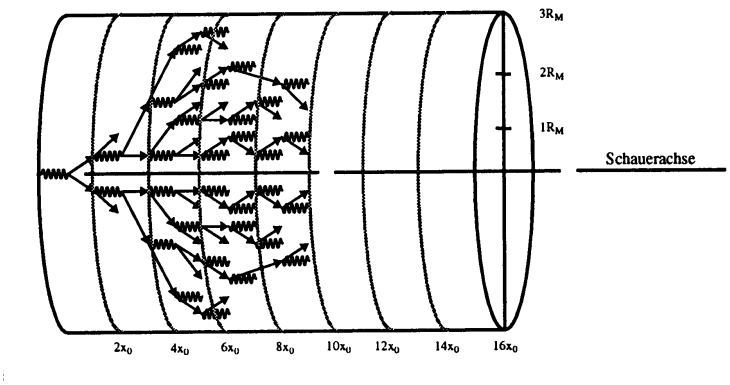


Abbildung 4.2: Stark vereinfachte Darstellung der Entwicklung eines photoninduzierten elektromagnetischen Schauers. Fast 99% der Energie des einfallenden Photons wird in einem Zylinder mit einem Radius von drei Moliere-Radien R_M und einer Tiefe von 16 Strahlungslängen X_0 deponiert. Das Schauermaximum liegt etwa in einer Tiefe von 4 bis 6 Strahlungslängen [Sc89]

4.3 Elektromagnetische Schauer

Wie sich aus den vorhergehenden Abschnitten ergibt, wechseln sich bei hohen Photonenenergien in Materie Paarbildungs- und Bremsstrahlungsprozesse ab. Auf diese Weise bildet sich ein elektromagnetischer Schauer aus, der aus Photonen, Elektronen und Positronen besteht (siehe Abbildung 4.2. Sobald die Energie der beteiligten Teilchen einen kritischen Energiewert E_{krit} unterschreitet, dominiert der Energieverlust durch Ionisation gegenüber dem durch Bremsstrahlung und die Schauerentwicklung kommt zum Erliegen. Die im Schauer enthaltenen Elektronen und Positronen verlieren ihre Restenergie durch Ionisation und Anregung, während die niederenergetischen Photonen vor allem durch Comptoneffekt und Photoeffekt wechselwirken.

4.3.1 Kenngrößen eines elektromagnetischen Schauers

Drei Kenngrößen charakterisieren Detektormaterialien, um deren Eignung als Schauerdetektoren zu beurteilen.

- Die bereits erwähnte *kritische Energie* E_{krit}. Sie gibt an, bei welcher Energie der Energieverlust durch Bremsstrahlung und der durch Ionisation gleich groß sind.
- Die *Strahlungslänge* X_0. Sie entspricht der Strecke im jeweiligen Material, auf der die Energie eines Elektrons durch die Abgabe von Bremsstrahlung sich um den Faktor $\frac{1}{e}$ verringert.
- Der *Moliere-Radius* R_M. Er ist ein Maß für die transversale Ausdehnung des Schauers.

Für Detektormaterialien, deren Kernladungszahl Z im Bereich von $13 < Z < 92$ liegt, gelten für die erwähnten Gößen folgende Werte:

$$E_{\text{krit}} = \frac{800}{Z} MeV \qquad (\pm 10\%)$$

$$X_0 = 180 \frac{A}{Z^2} \frac{g}{cm^2} \qquad (\pm 20\%)$$

$$R_M = \frac{E_{\text{Streu}}}{E_{\text{krit}}} = \frac{21 MeV}{E_{\text{krit}}} X_0 = 7 \frac{A}{Z} \frac{g}{cm^2} \qquad (\pm 10\%)$$

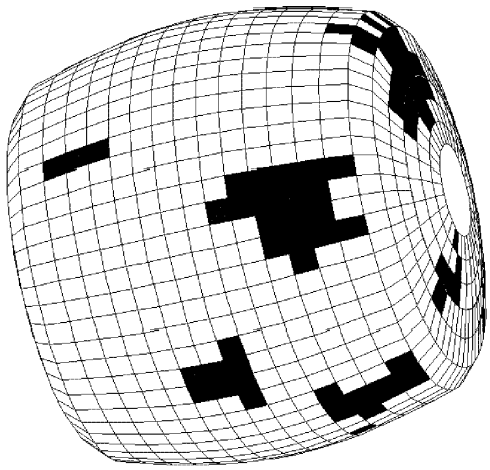


Abbildung 4.3: Ansicht des Crystal Barrel Detektors mit einem auf die Basisflächen der Kristalle projizierten Treffermuster. Auf der dem Betrachter zugewandten Seite sind sechs Cluster klar zu erkennen [Ar98]

Die Konstante E_{Streu} erhält man aus der Theorie der Vielfachstreuung am Kern. In einem Zylinder mit einer Länge von 16 Strahlungslängen und einem Radius von drei Moliere-Radien werden 99% der Schauerenergie deponiert [Sc89]. Aus diesen Beziehungen ergeben sich die beim Crystal-Barrel-Detektor gewählten Abmessungen der Kristalle.

4.4 Clusterbildung in elektromagnetischen Kalorimetern

Aufgrund der durch den Moliere-Radius gekennzeichneten transversalen Ausdehnung des elektromagnetischen Schauers im Detektormaterial wird bei einer genügend feinen Granularität des Kalorimeters die Energie des Schauers nicht nur in einem einzelnen Kristall deponiert. Dies führt dazu, dass bei Auftreffen eines hochenergetischen Photons im Kalorimeter im Allgemeinen nicht nur ein einzelner Kristall, sondern eine Gruppe von Kristallen anspricht.
Koinzidente Signale, die von einer Gruppe paarweise an den Begrenzungsflächen oder Kanten aneinander angrenzender Kristalle stammen, nennt man einen Cluster. Es spielt in diesem Zusammenhang keine Rolle, ob diese Signale von einem, durch ein einzelnes Photon ausgelösten Schauer oder von überlappenden Schauern stammen, die von mehreren unter kleinen Relativwinkeln emittierten Photonen erzeugt wurden.
In Abbildung 4.3 ist eine Darstellung des Crystal-Barrel-Detektors zu sehen, in der ein Treffermuster auf die Basisflächen der Kristalle projiziert wurde. Die Darstellung entspricht einem Ereignis, das so mit diesem Detektor am CERN beobachtet wurde. Die spätere Analyse ergab, dass bei diesem Ereignis 10 Photonen den Barrel getroffen haben. Deutlich sind auf der dem Betrachter zugewandten Seite 6 Cluster zu erkennen.
Aus der aus diesem Bild ersichtlichen Information lässt sich jedoch nicht festlegen, ob die darin identifizierten Cluster von einzelnen oder von mehreren Photonen stammen, die das Kalorimeter dicht beieinander getroffen haben. Dennoch kann die Kenntnis der Anzahl der Cluster hilfreich sein, um online eine Vorauswahl von Ereignissen bei hohen Ereignisraten zu treffen. Zu diesem Zweck wurde für das CB-ELSA-Experiment im Rahmen dieser Arbeit ein Trigger entwickelt und aufgebaut, der solche Cluster identifizieren und zählen kann. In den folgenden Kapiteln werden Aufbau und Funktion dieses Triggers beschrieben.

Kapitel 5

Zellularlogik

Die Arbeitsweise des in dieser Arbeit behandelten Multiplizitätstriggers beruht zu einem ganz wesentlichen Teil auf dem Funktionsprinzip einer Zellularlogik. Seinen historischen Ursprung hat der Begriff der Zellularlogik bzw. der allgemeiner gefasste Begriff des Zellularautomaten in einer theoretischen Arbeit von J. VON NEUMANN, die aus dem Jahr 1948 stammt [Ne66]. Er bezeichnet eine regelmäßige Anordnung hinsichtlich ihres Aufbaus und ihrer Funktionsweise identischer digitaler Schaltungen, den sogenannten Zellen. Die Ein- und Ausgänge jeder beliebig herausgegriffenen Zelle (sogenannte Zentralzelle) sind mit den Aus- und Eingängen genau P anderer Zellen verbunden. Die Zentralzelle bildet zusammen mit diesen Zellen die sogenannte Nachbarschaft.
Um das Konzept der Zellularlogik und damit auch die Funktionsweise des Triggers besser verstehen zu können, soll hier etwas näher auf die Theorie der Zellularautomaten eingegangen werden. Einen Überblick über die Theorie der Zellularautomaten gibt [Vo79].

5.1 Die Theorie der Zellularautomaten

Im Hinblick auf die spezielle Realisierung, die dieses Konzept in der vorliegenden Arbeit erfährt, wird hier ein zweidimensionaler Zellularautomat betrachtet. Seine identisch aufgebauten Zellen werden mit dem Symbol $C_{m,n}$ bezeichnet, wobei m und n den Gitterplatz in einem quadratischen Gitter indizieren. Jede dieser Zellen enthält J binäre Speicherelemente. Demnach lässt sich der innere Zustand einer Zelle durch einen Zustandsvektor $\vec{S}_{m,n}$ beschreiben:

$$\vec{S}_{m,n} = \left(S^1_{m,n}, S^2_{m,n}, \ldots, S^j_{m,n}, \ldots, S^{J-1}_{m,n}, S^J_{m,n}\right) \tag{5.1}$$

wobei $S^j_{m,n} \in \{0, 1\}$ den momentanen Zustand des Speicherelements mit dem Index j charakterisiert.
Jede Zelle verfügt über $K \leq J$ Ausgänge. Unter der Annahme, dass jedes ihrer Ausgangssignale Z^k nur vom momentanen inneren Zustand abhängt, gilt

$$Z^k_{m,n} = g^k\left(S^1_{m,n}, S^2_{m,n}, \ldots, S^j_{m,n}, \ldots, S^{J-1}_{m,n}, S^J_{m,n}\right) \tag{5.2}$$

$$= g^k\left(\vec{S}_{m,n}\right) \tag{5.3}$$

Gemeinsam bilden alle Ausgangssignale $Z^k_{m,n}$ der Zelle $C_{m,n}$ wieder den Ausgangsvektor $\vec{Z}_{m,n}$ dieser Zelle.

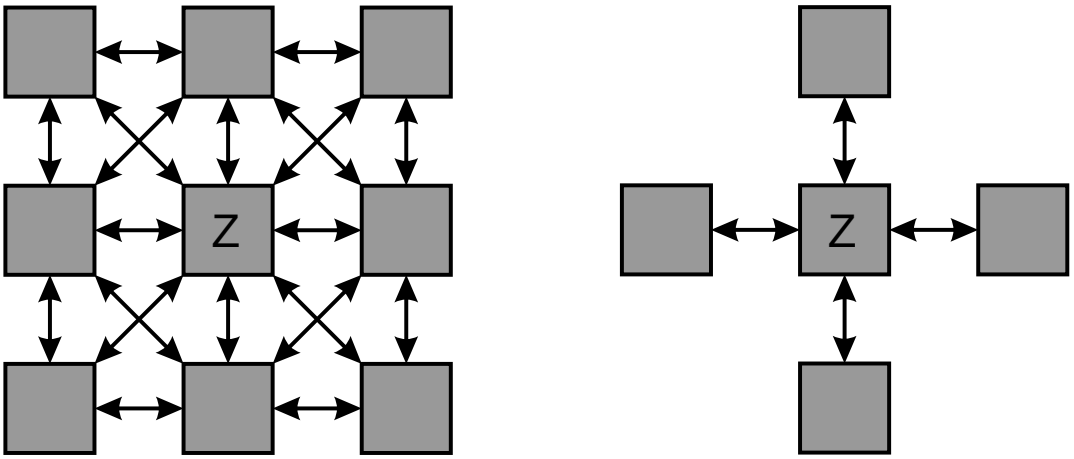


Abbildung 5.1: Schematische Darstellung der Moore- (links) und der von-Neumann-Nachbarschaft (rechts) der Größe 1

Charakteristisch für einen Zellularautomaten ist, dass jede Zelle $C_{m,n}$ mit genau P anderen Zellen verbunden ist. Deren Relativkoordinaten bezogen auf die Position von $C_{m,n}$ seien durch Vektoren $\vec{V}_1, \vec{V}_2, \ldots, \vec{V}_p, \ldots, \vec{V}_{P-1}, \vec{V}_P$ definiert, wobei $\vec{V}_p = (m_p, n_p)$ mit $(m_p, n_p) \in \mathcal{Z}^2$ und $(m_p, n_p) \neq (0, 0)$ gilt. Diese Vektoren kennzeichnen zusammen mit dem Nullvektor $\vec{V}_0 = (0, 0)$ die sogenannte Nachbarschaft V der Zelle $C_{m,n}$.
Im Prinzip lassen sich beliebige Nachbarschaften definieren. In der Theorie der Zellularautomaten werden jedoch vorwiegend zwei Arten von Nachbarschaften betrachtet, die als von-Neumann- und Moore-Nachbarschaft bezeichnet werden.
Gilt lediglich $m_p, n_p \in \{-a, \ldots, -1, 0, 1, \ldots, a\}$, so spricht man von einer Moore-Nachbarschaft der Größe a. Dabei ist $P = 4(a^2 + a)$. Gilt zusätzlich $|m_p| + |n_p| \leq a$, so spricht man von einer von-Neumann-Nachbarschaft der Größe a und es ist $P = 2(a^2 + a)$. Abbildung 5.1 zeigt als Beispiel dafür eine Moore- und eine von-Neumann-Nachbarschaft der Größe 1.
Aus dem zuvor Gesagten ergibt sich, dass jede Zelle über $K \cdot P$ Eingänge verfügt. Der Folgezustand eines Speicherelementes einer Zelle ergibt sich aus der für alle Zellen identischen Übergangsfunktion

$$S^{j,\text{Folge}}_{m,n} = f^j \left(\vec{S}_{m,n}, \vec{Z}_{m+m_p,n+n_p} \right) \tag{5.4}$$

wobei $\vec{Z}_{m+m_p,n+n_p}$ über alle Indizes $p \in 1 \ldots P$ läuft, sodass f^j eine Funktion der Ausgangssignale aller Nachbarn ist. Unter Anwendung von Gleichung 5.2 lässt sich diese Beziehung auch ausdrücken als

$$S^{j,\text{Folge}}_{m,n} = f'^j \left(\vec{S}_{m,n}, \vec{S}_{m+m_p,n+n_p} \right) \tag{5.5}$$

Aus diesen lokalen Übergangsregeln folgt eindeutig eine globale Übergangsregel, sodass der Zellularautomat durch die Angabe der lokalen Übergangsregel f'^j vollständig definiert ist.

Eine Erweiterung erfährt dieses Konzept, wenn man Signale zuläßt, die von außen an alle Logikzellen gemeinsam herangeführt werden. Solche Signale nennt man Broadcastsignale. Sie

können z.B. zur Initialisierung der Zellularlogik oder zum Einleiten bestimmter Arbeitsschritte dienen.

Es seien L Broadcastsignale vorhanden, deren Zustand durch einen Zustandsvektor $\vec{B} = (b_1, b_2, \ldots, b_l, \ldots, b_{L-1}, b_L)$ gegeben ist. Der Zustand der Zelle C_{mn} selbst hängt nicht vom Broadcastvektor $\vec{B}$ ab, wohl aber die Übergangsregel. Gleichung 5.4 wird nun zu

$$S_{m,n}^{j,\text{Folge}} = g^j \left(\vec{S}_{m,n}, \vec{Z}_{m+m_p,n+n_p}, \vec{B} \right) \tag{5.6}$$

oder unter Anwendung von Gleichung 5.2

$$S_{m,n}^{j,\text{Folge}} = g'^j \left(\vec{S}_{m,n}, \vec{S}_{m+m_p,n+n_p}, \vec{B} \right) \tag{5.7}$$

Auch hier gilt wieder, dass aus der lokalen Übergangsregel eindeutig eine globale Übergangsregel folgt, sodass auch der Zellularautomat mit Broadcastsignalen wieder eindeutig durch die Angabe seiner Übergangsregel g'^j definiert ist.

5.2 Clusteridentifikation mit Zellularautomaten

Die Struktur einer Zellularlogik ist offensichtlich in besonderer Weise geeignet, Daten, die von ein- oder mehrdimensionalen Detektorfeldern erzeugt werden, zu analysieren. Der hohe Grad an Parallelität führt dabei zu einer sehr hohen Verarbeitungsgeschwindigkeit, was den Einsatzbereich auch auf Online-Anwendungen und Trigger erweitert. Bei solchen Anwendungen wird jedem Detektorelement eine Zelle der Zellularlogik zugeordnet. Durch die entsprechende Wahl der Nachbarschaftsbeziehung kann auf sehr einfache Weise die Geometrie des Detektors auf die Struktur der Zellularlogik abgebildet werden.
Auf den im Rahmen dieser Arbeit realisierten Kalorimetertrigger für den Crystal-Barrel-Detektor angewandt heißt das, dass jedem Kristall eine Logikzelle zugeordnet wird und die Nachbarschaftsbeziehungen zwischen den Logikzellen so gewählt werden müssen, dass sie den Nachbarschaftsbeziehungen zwischen den Kristallen entsprechen. Die Struktur der Logikzellen ist natürlich abhängig von den Daten, die der Zellularlogik über die Energiedeposition in den Kristallen vorliegen. Es gibt grundsätzlich zwei Möglichkeiten:

- Der Zellularlogik steht für jeden Kristall lediglich die Information zur Verfügung, ob das Energiedeposit in einem Kristall über einer definierten Schwelle liegt.
- Der Zellularlogik stehen für jeden Kristall auch Daten über die Größe des Energiedeposits zur Verfügung.

Die letzte der beiden Möglichkeiten führt zu der sogenannten PED-Logik, die hier zuerst beschrieben werden soll.

5.2.1 Die PED-Logik

Wenn der Zellularlogik die Energiedeposits jedes Kristalls bekannt sind, besteht die Möglichkeit, nicht nur nach Clustern, sondern statt dessen nach lokalen Maxima der Energiedeposits,

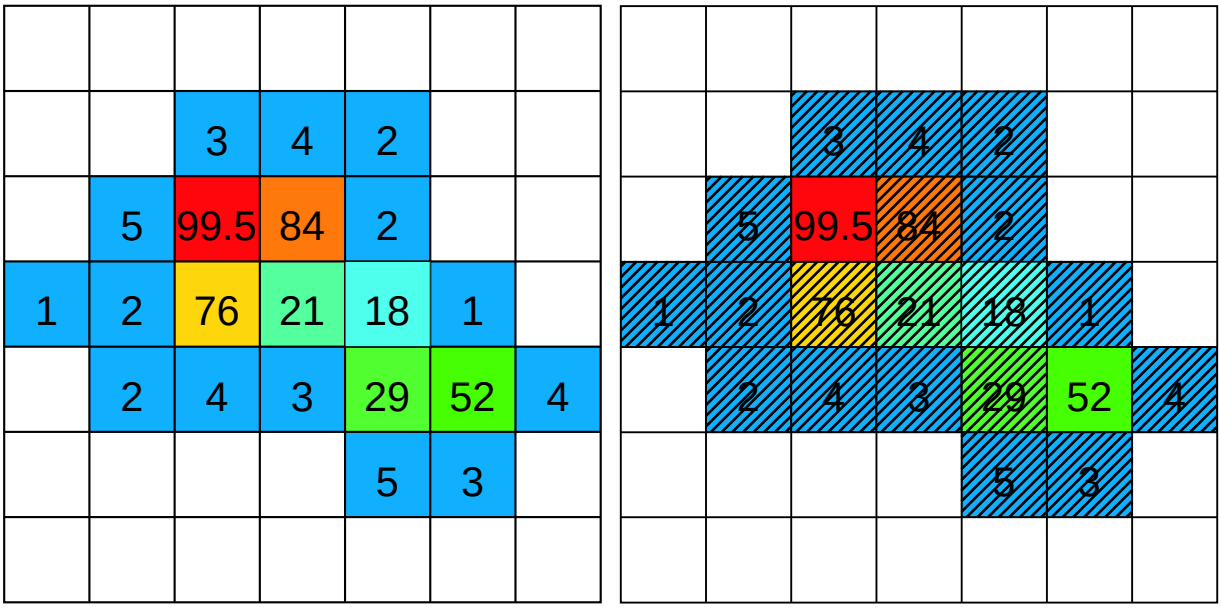


Abbildung 5.2: Arbeitsweise der PED-Logik. Links ist ein Ausschnitt der Zellularlogik zu erkennen. In jeder Zelle ist das Energiedeposit des entsprechenden Kristalls eingetragen. Rechts sind alle Zellen, die durch den Algorithmus gelöscht werden schraffiert. Übrig bleibt für jedes PED genau eine Zelle

auch PED[1] genannt, zu suchen. Diese Maxima bilden sich immer dort aus, wo das Zentrum eines elektromagnetischen Schauers liegt, somit also genau dort, wo das Kalorimeter von Photonen getroffen wurde. Auch dann, wenn sich die Schauer zweier dicht benachbarter Photonen überlappen, ist mit dieser Art von Logik i.a. die Trennung der Photonen möglich, was eine präzisere Bestimmung der Photonenanzahl gestattet als das mit einer reinen Clusteridentifikation möglich ist.
Ein lokales Maximum ist genau dann gegeben, wenn das Energiedeposit in einem Kristall größer ist, als die in allen Nachbarkristallen. Jede Logikzelle einer PED-Logik muss also in der Lage sein, das Energiedeposit in ihrem zugeordneten Kristall als binären Wert mit vorgegebener Wortbreite zu speichern und diesen Wert mit den in allen benachbarten Zellen gespeicherten Werten zu vergleichen.
Abbildung 5.2 demonstriert die Arbeitsweise der PED-Logik. In einem 7 mal 7 Zellen großen Ausschnitt aus der Matrix ist in jede Zelle der Wert des Energiedeposits des entsprechenden Kristalls eingetragen. Die PED-Logik arbeitet vollständig parallel. Simultan vergleichen alle Zellen der Matrix den Inhalt ihres Speichers mit dem aller benachbarten Zellen. Ist in mindestens einer benachbarten Zelle ein Wert gespeichert, der größer ist als der in der Zelle selbst gespeicherte, ist das Kriterium für ein lokales Maximum nicht mehr erfüllt. Die Zelle wird dann markiert, was rechts in Abbildung 5.2 durch eine Schraffur dargestellt ist. Als nicht markierte Zellen bleiben nach dem Markierungsvorgang nur solche Zellen übrig, bei denen der gespeicherte Wert des Energiedeposits ein lokales Maximum annimmt, das heißt i.a. für jedes PED genau eine Zelle. Die Einträge in den markierten Zellen werden gelöscht. Die nicht gelöschten Zellen können mit geeigneten Algorithmen sehr schnell gezählt werden.
Dies macht es für eine PED-Logik erforderlich, dass die in den Zellen gespeicherte Information

[1] **P**artial **E**nergy **D**eposit

zwischen allen benachbarten Zellen ausgetauscht werden kann. Bei einer großen Wortbreite der digitalisierten Energieinformation führt dies zu einer hohen Zahl von Verbindungen zwischen den Zellen. Deshalb wird sinnvollerweise dieser Informationsaustausch nicht parallel, sondern seriell, beginnend mit dem höchstwertigen Bit (MSB), durchgeführt. Somit ist zwischen zwei benachbarten Zellen nur noch eine Leitung für jede Datenrichtung erforderlich. Eine genaue Beschreibung des Ablaufs der PED-Identifizierung und des Aufbaus der Zellularlogik ist in [Ke95] zu finden.
Das zuvor beschriebene Verfahren ist allerdings als Funktionsprinzip eines Triggers nur dann anwendbar, wenn die Information über die Energiedeposite in den Kristallen ausreichend schnell verfügbar ist. Da beim CB-Experiment aus Kostengründen auf den Einsatz schneller ADCs verzichtet wurde, erwies sich seine Anwendung als nicht möglich. Statt dessen musste auf ein Verfahren zurückgegriffen werden, das mit der 1-Bit-Information aus den Diskriminatoren auskommt.

5.2.2 Die Clusterlogik

Im Gegensatz zu der im vorangegangenen Abschnitt beschriebenen PED-Logik, in der die in den Zellen gespeicherte Energieinformation ein lokales Kriterium darstellt, dass zur gezielten Identifikation und Auswahl einzelner Zellen genutzt werden kann, steht in einer Zellularlogikmatrix, die lediglich 1-Bit-Trefferinformationen enthält, kein solches Kriterium für die Clusteridentifikation zur Verfügung. Dies macht es erforderlich, zur Unterscheidung der Cluster ein globales Kriterium einzuführen, was strenggenommen die Symmetrie der Zellen in der Zellularlogik aufhebt und somit zwar der Definition einer Zellularlogik widerspricht, aber dennoch zu einem ausgesprochen leistungsfähigen Konzept für einen Multiplizitätstrigger führt.
Dieses globale Kriterium zur Unterscheidung der Cluster kann in der Zelle selbst in Form einer Zellnummerierung implementiert oder von außen vorgegeben werden. Beide Möglichkeiten wurden beim Entwurf des Triggers in Betracht gezogen und in konkrete Schaltungsentwürfe umgesetzt. Dabei ist die Realisierung des globalen Kriteriums in Form der Zellnummerierung technisch anspruchsvoller. Ähnlich wie bei der PED-Logik müssen innerhalb eines Clusters alle Zellen markiert werden, bis je Cluster genau eine einzige Zelle übrig bleibt. Als Auswahlkriterium dafür dient die Zellnummerierung. Um die Zelle mit der größten Zellnummer in einem Cluster zu ermitteln, müssen die Zellen ihre Zellnummer nicht nur mit denen ihrer direkten Nachbarn, sondern mit denen aller Zellen eines Clusters vergleichen. Dies entspricht einem globalen Vergleich innerhalb eines Clusters.
Um diesen Vergleich zu ermöglichen, wurde das Konzept eines selbstorganisierenden Bussystems entwickelt, das sich genau auf dem Gebiet eines Clusters bildet. Dieses Bussystem wird mit Schaltern realisiert, mit denen der Bus auf dem Gebiet eines Clusters vom Rest der Matrix isoliert wird. Dazu wird die Verbindung zwischen zwei Zellen aufgetrennt, wenn sich eine der beiden Zellen außerhalb eines Clusters befindet. Als Gate-Signal für diesen Schalter wird die UND-Verknüpfung zwischen den Treffersignalen der beiden benachbarten Zellen benutzt. Auf diesem Bus finden dann die Vergleichsoperationen statt, die wiederum, um Verbindungsleitungen einzusparen, sequentiell, beginnend mit dem höchstwertigsten Bit (MSB) erfolgen.
Der Vorteil dieser Technik besteht darin, dass die gesamte Clusteridentifizierung vollständig parallel abläuft. Aufgrund des hohen technischen Anspruchs dieses Konzeptes, das weit über die klassische Zellularlogik hinausgeht, wären für die Realisierung jedoch zeitaufwendige Entwick-

lungsarbeiten notwendig gewesen, die für den Aufbau des CB-ELSA-Triggers nicht möglich waren.
Letztlich fiel die Wahl daher auf die Vorgabe eines globalen Kriteriums mit zwei externen Prioritätsencodern. Dieses Verfahren wurde zuerst von H. MATTHÄY vorgeschlagen [Ma88]. Eine ähnliche Methode kam bereits kurz darauf bei dem am Fermilab (USA) durchgeführten Experiment E-731 [As90] zur Anwendung.
Die Identifizierung der Cluster im Kalorimeter wird in Abbildung 5.3 am Beispiel eines aus drei Clustern bestehenden und in einer auf 14×14 Zellen reduzierten Zellularlogikmatrix abgespeicherten Treffermusters demonstriert. Sie geschieht in mehreren Schritten.

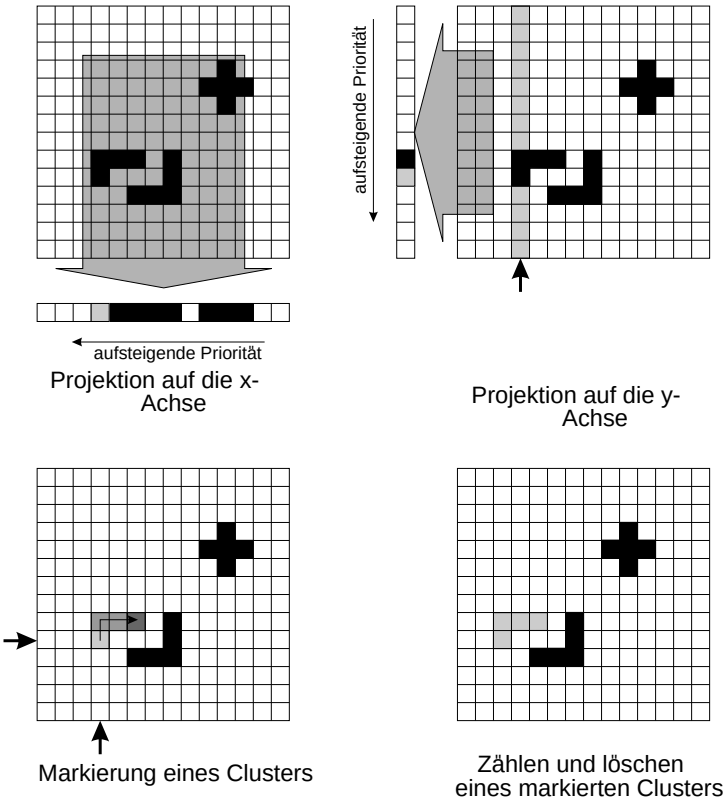


Abbildung 5.3: Schematische Darstellung des zeitlichen Ablaufs der einzelnen für die Identifizierung eines Clusters mit der Zellularlogik auszuführenden Schritte

1. Zuerst wird das gesamte Treffermuster unter Verwendung von ODER-Verknüpfungen zwischen den Ausgängen aller in einer Spalte angeordneten Zellen auf die in der Abbildung eingetragene x-Achse projiziert.

2. Ein mit den ODER-Ausgängen sämtlicher Spalten verbundener Prioritätsencoder identifiziert die am weitesten links liegende Spalte, die mindestens einen Treffer enthält (Spalte 4).

3. Alle in dieser Spalte enthaltenen Zellen werden simultan adressiert. Dadurch wird das in

ihnen abgespeicherte Treffermuster selektiv über ODER-Verbindungen, die zwischen den Ausgängen aller in einer Zeile angeordneten Zellen bestehen, auf die y-Achse projiziert.

4. Innerhalb der ausgewählten Spalte wird ebenfalls mit einem Prioritätsencoder die unterste Zelle identifiziert, die einen Treffer gespeichert hat (Zeile 5).

5. Diese Zelle wird selektiv adressiert und durch das Setzen eines speziellen Flip-Flop (Markierungs-FF) markiert.

6. In der Folge werden von dieser Zelle ausgehend automatisch alle Zellen markiert, die einen Treffer gespeichert und einen bereits markierten Nachbarn haben. Dieser Vorgang wird automatisch beendet, sobald der ganze Cluster markiert ist.

7. Anschließend wird ein Zähler, in dem die Anzahl der identifizierten Cluster gezählt wird, inkrementiert. Zusätzlich kann aber auch eine von der Zellularlogik autonom generierte Liste aller zum markierten Cluster gehörenden Kristalladressen erzeugt werden.

8. Danach werden die Speicher in allen markierten Zellen gelöscht. Damit ist die Identifizierung des ersten Clusters abgeschlossen.

9. Mit dem ersten Schritt beginnend wird diese Sequenz so lange wiederholt, bis alle Cluster identifiziert und damit die Speicherinhalte aller Zellen, die einen Treffer gespeichert haben, gelöscht sind. Der Zählerstand entspricht dann der Gesamtzahl aller identifizierter Cluster.

Für die Erzeugung der unter Punkt 7 erwähnten Liste aller zum markierten Cluster gehörenden Kristalladressen sind weitere Schritte durchzuführen. Dabei wird davon ausgegangen, dass in der Zellularlogik ein komplett markierter Cluster vorliegt. Dann kann wie folgt die Liste der zu diesem Cluster gehörenden Kristalladressen erzeugt werden.

1. Zuerst wird der *markierte* Cluster unter Verwendung der ODER-Verknüpfungen zwischen den Ausgängen aller in einer Spalte angeordneten Zellen auf die in der Abbildung eingetragene x-Achse projiziert.

2. Der mit den ODER-Ausgängen sämtlicher Spalten verbundene Prioritätsencoder identifiziert die am weitesten links liegende Spalte, die mindestens einen Treffer *dieses Clusters* enthält (Spalte 4).

3. Alle in dieser Spalte enthaltenen Zellen werden simultan adressiert. Dadurch wird das in ihnen abgespeicherte Treffermuster des markierten Clusters selektiv über ODER-Verbindungen, die zwischen den Ausgängen aller in einer Zeile angeordneten Zellen bestehen, auf die y-Achse projiziert.

4. Innerhalb der ausgewählten Spalte wird ebenfalls mit dem Prioritätsencoder die unterste Zelle identifiziert, die eine markierte Zelle enthält (Zeile 5).

5. Die auf diese Weise ausgeählte Zelle wird selektiv gelöscht, nachdem die von den Prioritätsencodern erzeugte Adresse in einem FIFO-Speicher abgespeichert wurde.

6. Mit dem ersten Schritt beginnend wird der gesamte Cluster ausgelesen, bis alle markierten Zellen gelöscht sind.

Es ist evident, dass diese Art der Clusteridentifizierung gegenüber herkömmlichen Methoden, die auf teilweiser oder vollständiger Projektion des Treffermusters in eine oder mehrere Richtungen beruhen, eine Reihe von wesentlichen Vorteilen bietet:

- Indem man jedem der 1380 Kristalle des Detektors eine Logikzelle zuordnet und durch die entsprechende Wahl der Nachbarschaftsbeziehungen der Zellularlogik die geometrischen Nachbarschaftsbeziehungen zwischen den Kristallen im Kalorimeter genau wiederspiegelt, lässt sich die Topologie des Detektors auf sehr einfache Weise exakt auf die Struktur der Zellularlogik abbilden.

- Die Clusteridentifizierung mit einer Zellularlogik unterliegt keinerlei topologischen Einschränkungen, d.h. der Trigger identifiziert alle gleichzeitig in einem Ereignis registrierten Cluster unabhängig von deren Anzahl, Gestalt und Position.

- Aufgrund der autonomen und überwiegend parallelen Arbeitsweise des Zellularlogik-Triggers sind sehr kurze Reaktionszeiten erreichbar.

Kapitel 6

Der Entwurf des Zellularlogik-ASICs

Für die Realisierung der im Kapitel 5 beschriebenen Zellularlogik bieten sich aufgrund der hohen Komplexität entweder die Verwendung von programmierbaren digitalen Schaltkreisen, z.B. sogenannten FPGAs[1] oder die Entwicklung eines voll anwenderspezifischen integrierten Schaltkreises, eines sogenannten ASICs[2] an. Da die zweite Möglichkeit beim Entwurf eine weitaus größere Flexibilität bietet, wurde hier davon Gebrauch gemacht.
Für den Entwurf anwenderspezifischer integrierter Schaltkreise steht in Bochum am Institut für Experimentalphysik I eine Hewlett Packard Workstation mit dem Halbleiter-CAD-Softwarepaket Design Framework II von der Firma Cadence zur Verfügung. Dieses Paket enthält neben Schaltplan- und Layout-Editor auch mehrere Schaltungssimulatoren sowie einige Hilfsprogramme, um die Schaltung, bzw. das Layout auf Fehler zu überprüfen. Um einen bestimmten Herstellungsprozess verwenden zu können, ist zudem der entsprechende Design-Kit erforderlich, der in einer für die CAD-Software lesbaren Form Informationen über die Design-Regeln dieses Prozesses enthält. Solche Designregeln schreiben zum Beispiel minimal einzuhaltene Abstände oder Strukturgrößen vor. In Bochum stehen solche Design-Kits für mehrere Prozesse der Firmen Alcatel-Mitec und AMS[3] zur Verfügung.
Die Herstellung der Chips erfolgt über EUROPRACTICE, eine Organisation der Europäischen Union, die Universitäten und anderen Forschungseinrichtungen kostengünstig die Produktion von ASICs auch in kleinen Stückzahlen ermöglicht. Dabei wird auf sogenannte Multi-Projekt-Wafer zurückgegriffen, bei denen auf einem Halbleiterwafer verschiedene Halbleiterchips mehrerer Anwender gemeinsam produziert werden.

6.1 CMOS-Technik

Die Auswahl unter den in Bochum zur Verfügung stehenden Herstellungsprozessen fiel auf einen CMOS-Prozess der Firma AMS mit einer minimalen Strukturgröße von 0,8 μm. Wesentliches Schaltungselement bei allen MOS-Prozessen ist der MOS-Transistor, der sowohl mit einem p-leitendem Kanal (pMOS), als auch mit einem n-leitendem Kanal (nMOS) realisiert werden kann. In der CMOS-Technik werden die logischen Funktionen durch Parallel- und Reihenschaltungen

[1] **F**ield **P**rogramable **G**ate **A**rrays
[2] **A**pplication **S**pecific **I**ntegrated **C**ircuit
[3] **A**ustria **M**ikro **S**ysteme AG

von nMOS- und pMOS-Transistoren realisiert. Daher ist es erforderlich, dass prozesstechnisch MOS-Transistoren mit beiden Kanalpolaritäten zur Verfügung stehen.

6.1.1 MOS-Transistoren

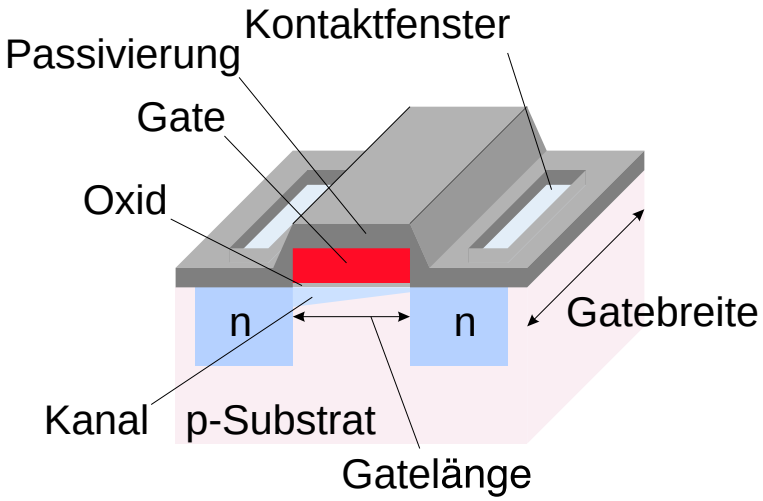


Abbildung 6.1: Darstellung eines nMOS-Transistors. Als Drain und Source Kontakte sind zwei n-Zonen in ein p-Substrat eindotiert. Dazwischen befindet sich auf der Oberfläche, vom Substrat durch eine dünne Oxidschicht isoliert, das Gate. Die ganze Halbleiterstruktur ist durch eine Nitrid- oder Oxidschicht gegen Umwelteinflüsse passiviert

Abbildung 6.1 zeigt eine Prinzipdarstellung eines nMOS-Transistors. Als Grundmaterial wird ein p-dotierter Halbleiter benutzt, in dem zwei n-Zonen eindotiert werden, die als Source- und Drain-Elektrode dienen. Der Bereich zwischen den beiden Elektroden wird von einer dünnen Oxidschicht bedeckt, auf deren Oberfäche ein leitender Belag aus Metall oder hochdotiertem Polysilizium aufgedampft ist. Dieser Belag stellt die Gate-Elektrode dar.
Im Betrieb wird zwischen der Source- und der Drain-Elektrode die Spannung U_{ds} angelegt, wobei der Drainanschluss mit dem höheren Potential verbunden wird. Das Substrat wird mit dem Sourceanschluss verbunden. Dadurch werden die Dioden, die durch den Übergang zwischen den n-dotierten Elektroden und dem p-Substrat gebildet werden, in Sperrrichtung vorgespannt. Die Funktion des nMOS-Transistors beruht darauf, dass zwischen Source und Drain durch Ladungsinversion ein leitender n-Kanal entsteht, der eine Verbindung zwischen diesen beiden Elektroden herstellt. Diese Ladungsinversion wird durch ein elektrisches Feld hervorgerufen, das von einer zwischen Gate und Source angelegten Spannung U_{gs} erzeugt wird. Dieses Feld führt mit zunehmender Feldstärke zunächst zu einer Depletierung des p-Substrats unterhalb des Gate-Oxids und dann zu einer Ladungsinversion, d.h. einer Anreicherung mit Elektronen. Die Inversion setzt erst dann ein, wenn die Anzahldichte der Elektronen in der Schicht unterhalb des Oxids den gleichen Wert erreicht hat, wie die Dotierstoffkonzentration N_A des p-dotierten Substrats. Dazu muss eine definierte Schwellenspannung U_t überschritten werden, bevor der Transistor leitet. Für diese Schwellenspannung gilt nach [We94]

$$U_t = U_{t-mos} + U_{fb} \tag{6.1}$$

Für U_{t-mos} gilt:

$$U_{t-mos} = 2\phi_b - \frac{Q_b}{C_{ox}} \tag{6.2}$$

mit

$$\phi_b = \frac{kT}{q} \ln\left(\frac{N_A}{N_i}\right), \tag{6.3}$$

und dem sogenannten Volumenladungsterm

$$Q_b = \sqrt{2\epsilon_0 \epsilon_{Si} N_A 2\phi_b} \tag{6.4}$$

Die auf die Flächeneinheit bezogene Oxid-Kapazität C_{ox} berechnet sich nach

$$C_{ox} = \frac{\epsilon_0 \epsilon_{Si}}{t_{ox}} \tag{6.5}$$

Die in den Gleichungen 6.3 bis 6.5 eingeführten Größen sind die intrinsische Ladungsträgerkonzentration von Silizium N_i, die Feldkonstante ϵ_0, die relative Dielektrizitätszahl von Silizium ϵ_{Si}, sowie die Dicke des Gateoxids t_{ox}.
Die in Gleichung 6.1 auftauchende Flachbandspannung U_{fb} wiederum ist gegeben durch

$$U_{fb} = \phi_{ms} - \frac{Q_{fc}}{C_{ox}} \tag{6.6}$$

Der Term ϕ_{ms} ist die Differenz der Austrittsarbeiten von Gatematerial und Substrat. Für ein Gate aus n^+ Polysilizium und ein n-Substrat gilt

$$\phi_{ms} = -\left(\frac{E_g}{2} + \phi_b\right) \tag{6.7}$$

Dabei ist E_g der Abstand zwischen Valenz- und Leitungsband, der bei Silizium 1,1 eV beträgt. Q_{fc} ist die Flächenladungsdichte der festen Ladungen an der Grenze zwischen Oxid und Substrat, die auf nicht abgesättigte Bindungen zurückzuführen sind.
Aufgrund des Spannungsabfalls entlang der Source-Drain-Strecke ist die Gate-Substratspannung und damit auch die Stärke des für die Inversion verantwortlichen elektrischen Feldes abhängig vom Ort entlang der Source-Drain-Strecke. Sobald die Source-Drain-Spannung den Wert $U_{gs} - U_t$ überschreitet, schnürt sich der n-Kanal an der Grenze zwischen Substrat und Drain ab. Mit zunehmender Spannung U_{ds} bewegt sich der Punkt, an dem es zur Abschnürung kommt, auf die Source-Elektrode zu. In diesem Fall muss der Source-Drain-Strom I_{ds} einen depletierten Bereich durchqueren. Dies führt dazu, dass auch bei einem weiteren Ansteigen der Source-Drain-Spannung der Strom nicht weiter ansteigen kann. Diesen Arbeitsbereich nennt man daher Sättigungsbereich. Erstreckt sich hingegen der n-Kanal über die gesamte Source-Drain-Strecke, spricht man vom ungesättigten oder linearen Arbeitsbereich. Liegt die Gate-Spannung auf der gesamten Strecke zwischen Source und Drain unterhalb der Schwellenspannung, spricht man von der Cutoff-Region.
In erster Näherung kann man für den Source-Drain-Strom in diesen drei Arbeitsbereichen die folgenden Beziehungen angeben:

$$I_{ds} = 0 \quad \text{für} \quad U_{gs} \leq U_t \tag{6.8}$$

$$I_{ds} = \beta\left[(U_{gs} - U_t)U_{ds} - \frac{U_{ds}^2}{2}\right] \quad \text{für} \quad 0 < U_{ds} < U_{gs} - U_t \tag{6.9}$$

$$I_{ds} = \beta\frac{(U_{gs} - U_t)^2}{2} \quad \text{für} \quad 0 < U_{gs} - U_t < U_{ds} \tag{6.10}$$

Dabei ist β der MOS-Transistor Verstärkungsfaktor, der im Wesentlichen von der Geometrie des Transistors und der Ladungsträgerbeweglichkeit abhängt. Es gilt

$$\beta = \frac{\mu\epsilon}{t_{ox}} \left(\frac{W}{L}\right) \tag{6.11}$$

Dabei ist μ die effektive Ladungsträgerbeweglichkeit, ϵ die Dielektrizitätskonstante des Gateoxids mit $\epsilon = \epsilon_0 \cdot \epsilon_{ox}$. Die Geometrie des Transistors steckt in der Dicke des Gateoxids t_{ox}, der Gatebreite W und der Gatelänge L.
Beim Design einer integrierten Schaltung besteht für den Anwender nur die Möglichkeit, die Eigenschaften des Transistors über die Variation von Gatelänge und -breite zu beeinflussen, alle anderen Parameter sind durch den Herstellungsprozess fest vorgegeben. Die Gleichungen 6.10 und 6.11 geben Aufschluss darüber, welcher maximale Strom durch den Transistor fließen kann. Dies ist insbesondere für das Design von Treiber- und Ausgangsstufen wichtig.
An dieser Stelle soll nicht tiefer auf die Theorie der MOS-Transistoren eingegangen werden. Eine genauere Beschreibung der MOS-Theorie, die auch Effekte höherer Ordnung berücksichtigt, findet sich in [We94].
Analog zum nMOS-Transitor arbeitet der pMOS-Transistor. Es sind lediglich die p und n Dotierungen zu vertauschen. Zu beachten ist, dass die Ladungsträgerbeweglichkeit von Löchern im allgemeinen kleiner ist, als die von Elektronen. Ein nMOS-Transistor kann somit einen höheren Strom aufbringen, als ein pMOS-Transistor mit gleichen Abmessungen. Um dies auszugleichen wird in der CMOS-Technik bei pMOS-Transistoren meist etwa die doppelte Kanalbreite der nMOS-Transistoren gewählt.

6.1.2 CMOS-Prozesstechnik

Als Ausgangsmaterial für die Herstellung von Integrierten Schaltkreisen in CMOS-Technik dient im Allgemeinen ein schwach p-dotierter Silizium-Wafer. Zur Herstellung dieser Wafer wird unter einer Argonatmosphäre aus einer Schmelze von Reinstsilizium, der die Dotierstoffe in der benötigten Konzentration zugesetzt werden, ein Silizium-Einkristall gezogen. Dieser Einkristall wird in zylindrische Form gebracht und dann in etwa 300 μm dicke Scheiben geschnitten, die anschließend plan poliert werden, um eine minimale Rautiefe der Waferoberfläche zu erreichen. Um die Oberfläche des Wafers zu schützen, wird diese thermisch oxidiert.
Die Durchmesser dieser Wafer reichen von 3 Zoll, entsprechend ca. 75 mm bis zu 10 Zoll, entsprechend ca. 250 mm. Bei größeren Waferflächen lassen sich mehr Schaltungen auf einem Wafer unterbringen, was bei der Massenproduktion zu geringeren Kosten führt. Durch die Vergrößerung erhöhen sich aber auch die Ansprüche an die Waferproduktion und die nachfolgende Prozesstechnik.
Die Herstellung der Halbleiterstrukturen auf dem Wafer erfolgt in einer Reihe von Prozessschritten, die in festgelegter Folge zur Anwendung kommen. Die einzelnen Prozessschritte lassen sich kurz folgendermaßen charakterisieren:

Epitaxie und Abscheidung werden benutzt, um Material auf der Oberfläche des Halbleiters aufzubringen. Mittels Epitaxie kann man einkristalline Schichten aufbringen, sofern die Oberfläche, auf die das Material aufgebracht wird, selbst einkristallin ist.

Eine andere Möglichkeit, Material auf der Oberfläche aufzubringen, besteht darin, diese Stoffe physikalisch oder chemisch auf der Oberfläche abzuscheiden.

Ionenimplantation Dies ist ein Prozess, bei dem der Wafer mit Ionen im Energiebereich zwischen einigen 10 keVbis etwa 1 MeVbeschossen wird. Ihre Eindringtiefe ist durch die Energie bestimmt. Dadurch ist eine gezielte Dotierung möglich. Da bei der Abbremsung der Ionen die Kristallstruktur des Halbleiters lokal gestört werden kann, muss nach der Ionenimplantation der Wafer thermisch ausgeheilt werden.

Diffusion Dies ist der andere Prozess, mit dem ein Halbleiter dotiert werden kann. Damit sich die Dotierungsionen entlang des Konzentrationsgradienten bewegen können, muss der Halbleiter bis auf ca 200° C unterhalb der Schmelztemperatur aufgeheizt werden. Die Dotierstoffionen selbst können dabei vorher mittels Implantation in den Kristall eingebracht worden sein oder aus einer entsprechenden Gasatmosphäre auf der Waferoberfläche abgeschieden werden. Im Gegensatz zur Ionenimplantation ist der Prozess der Diffusion nicht gerichtet, sondern läuft isotrop ab.

Ätzen wird als Prozessschritt angewendet, um gezielt Material von der Halbleiteroberfläche zu entfernen. Dabei werden sowohl chemische Ätzverfahren unter Verwendung chemisch aktiver Gase oder Flüssigkeiten, als auch physikalische Ätzverfahren, die mit Ionenbeschuss arbeiten (Plasmaätzen), verwendet. Die chemischen Ätzverfahren haben den Vorteil, dass sie materialsensitiv sind, wogegen die physikalischen Verfahren geometrisch gezielt eingesetzt werden können.

Oxidation Die Oxidation von Silizium zu Siliziumdioxid (SiO_2) erfolgt bei Temperaturen zwischen 800 und 1200° C. Als Prozessgas findet dabei entweder Wasserdampf (sogenannte feuchte Oxidation) oder Sauerstoff (sogenannte trockene Oxidation) Verwendung. Da die feuchte Oxidation schneller verläuft, kommt sie vorwiegend bei der Erzeugung dicker Oxidschichten zum Einsatz, die zur Passivierung von Halbleiterstrukturen oder als Masken dienen, die bei der Dotierung durch Diffusion und Ionenimplantation genutzt werden. Für die Herstellung des Gateoxids von MOS-Transistoren kommt hingegen nur die trockene Oxidation zur Anwendung.

Entscheidend bei allen Prozessschritten ist, dass die Waferoberfläche vor ihrer Anwendung so präpariert wird, dass der jeweilige Prozess nur an bestimmten Stellen auf dem Wafer wirksam wird. Dazu wird üblicherweise ein lichtempfindlicher Lack auf den Wafer aufgetragen, auf den die Struktur einer sogenannten *Maske* durch Belichtung mit UV-Licht übertragen wird. Mit einem geeigneten Lösungsmittel lassen sich dann, je nach Art des Lacks, die belichteten oder die unbelichteten Stellen leicht entfernen. An den Stellen, an denen sich danach noch eine Lackschicht befindet, ist der Halbleiter vor dem folgenden Prozessschritt geschützt. Diesen Prozess nennt man *Lithographie*.
Für die minimale Ausdehnung, bis zu der herab sich noch Strukturen auf dem Halbleiter realisieren lassen, ist im Wesentlichen das Lithographieverfahren verantwortlich. Üblicherweise werden heute die Strukturen 10-fach vergrößert mit einer Schicht aus Chrom- oder Eisenoxid auf einen

Quarzglasträger aufgebracht. Die minimal erreichbare Strukturgröße hängt hier natürlich ganz wesentlich von der Wellenlänge des verwendeten Lichtes ab. Man ist daher bemüht, möglichst kurzwelliges Licht zu verwenden, was jedoch größere Anforderungen an die Abbildungsoptik stellt.
Ein anderes Verfahren zur Lithographie ist die sogenannte *Elektronenstrahllithographie*. Bei diesem Verfahren werden die Strukturen mit einen 50-keV-Elektronenstrahl direkt auf den Fotolack geschrieben. Der Vorteil dieses Verfahrens besteht darin, dass, wegen der kleinen DeBroglie-Wellenlänge der Elektronen, sehr kleine Strukturgrößen erreichbar sind. Andererseits ist die Anwendung dieses Verfahrens so zeitaufwendig, dass es sich kaum für eine Massenfertigung eignet. Daher beschänkt sich die Anwendung im Wesentlichen auf die Maskenherstellung.

6.1.3 Der verwendete AMS-0,8-μ-CMOS-Prozess

Prozessname	CYE
Prozesstyp	Mixed Signal
Geometrische Kanallänge	0,8 μm
Betriebsspannung	2,5 - 5,5 V
Anzahl der Masken	13
Anzahl der Metalllayer	2
Anzahl der Polysiliziumlayer	2
Substrattyp	p-EPI
Diffusionspitch[4]	3,8 μm
Metall 1/2 Pitch	2,4 μm / 2,8 μm
Metall 1/2 + Via Pitch	2,5 μm / 2,9 μm
Poly1 Pitch	1,8 μm
N/PMOS effektive Kanallänge	0,66 μm / 0,79 μm
N/PMOS Sättigungsstrom[5]	400 μA/μm / 195 μA/μm
Poly2 Kapazität	1,8 fF/μm^2

Tabelle 6.1: Die wichtigsten Eigenschaften des verwendeten AMS-CMOS-Prozesses nach [Am00]

Der für die Herstellung des in dieser Arbeit beschriebenen Zellularlogik-ASICs verwendete CMOS-Prozess der Firma AMS ist ein sogenannter N-Wannen-Prozess, der auf einem leicht p-dotierten Grundmaterial basiert. Um auf einem p-dotierten Wafer pMOS-Transistoren realisieren zu können, werden sogenannte n-Wannen in das Halbleitermaterial eindiffundiert. Dabei handelt es sich um tiefe Diffusionszonen, in denen eine leichte n-Leitfähigkeit erreicht wird. Die wichtigsten Charakteristika dieses Prozesses finden sich in Tabelle 6.1.

[4] Als Pitch wird der einzuhaltende Mindestabstand zwischen den Mittellinien zweier Leiterbahnen minimaler Breite des gleichen Layers bezeichnet.

[5] Bezogen auf die Einheit der Kanalbreite bei minimaler Kanallänge

6.2 Die Entwurfstechnik eines CMOS-ASICs mit der verwendeten CAD-Software

Der erste Schritt des ASIC-Entwurfs bestand in der Erstellung des Schaltplans. Dazu steht in dem in Bochum verwendeten Software-Paket DFII von Cadence ein komfortabler Schaltungseditor zur Verfügung, in dem die Bauelemente, auf die in dem verwendeten Herstellungsprozess zurückgegriffen werden kann, als Makros in den Schaltplan eingefügt werden und dann miteinander verbunden werden können. Zudem besteht die Möglichkeit, auf Schaltungsblöcke zurückzugreifen, die vorher selbst aus kleineren Blöcken zusammengesetzt wurden, sodass ein hierachischer Aufbau einer Schaltung möglich ist.
Der Schaltplaneditor ist in der Lage, aus dem erstellten Schaltplan alle verwendeten Bauelemente mit deren Eigenschaften, sowie alle Verbindungsnetze zu extrahieren. Diese Listen können von verschiedenen Simulationspaketen benutzt werden, um das Verhalten der entworfenen Schaltung auf Transistorebene zu simulieren. Dabei können verschiedene Spannungs- und Signalquellen im Schaltplan eingefügt werden, um die Spannungsversorgung und die Eingangssignale der Schaltung zu simulieren. Ausgangsbelastungen können durch Einfügen zusätzlicher Widerstände und Kondensatoren realistisch erfasst werden. Das CAD-Paket erlaubt es nach einer erfolgten Simulation den zeitlichen Verlauf von Strom und Spannung an allen Schaltungspunkten zu visualisieren. Aus diesen Simulationsergebnissen kann der Entwickler ersehen, ob die Schaltung bereits den Anforderungen entspricht oder weitere Änderungen erforderlich sind.
Sobald die Schaltung den gesetzten Anforderungen entspricht, beginnt der eigentliche Pro-

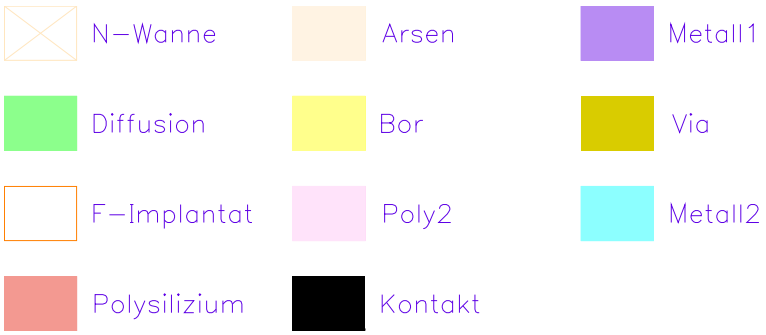


Abbildung 6.2: Die unterschiedlichen Layer, die im verwendeten 0,8-μm-CMOS-Prozess von AMS zur Verfügung stehen

zess des Layoutentwurfs, für den die CAD-Software DFII einen separaten Layout-Editor zur Verfügung stellt. Mit Hilfe dieses Layout-Editors kann der Entwickler in verschiedenen Layern Leiterbahnen, Rechtecke oder Polygone zeichnen. Abbildung 6.2 zeigt die Farben, die zur Kennzeichnung der verschiedenen von dem Prozess zur Verfügung gestellten Layer benutzt werden. Es handelt sich dabei um die folgenden Layer:

N-Wanne gibt die Bereiche an, der n-dotiert werden sollen, um dort pMOS-Transistoren einzubetten

Diffusion ist der aktive Bereich, der einen Transistor von der Umgebung abgrenzt

F-Implantat muss deckungsgleich mit dem Layer N-Wanne sein.

Polysilizium dient sowohl als Gate-Material, als auch als unterste Verdrahtungsebene.

Arsen kennzeichnet den Bereich, in dem ein aktiver (Diffusions) Bereich p-dotiert statt n-dotiert wird. Hiermit werden pMOS-Transitoren realisiert.

Bor Dieser Layer muss deckungsgleich mit dem Arsen-Layer sein.

Poly2 ist ein zweiter Polysilizium-Layer, der vornehmlich für die Herstellung von Kapazitäten verwendet wird.

Kontakt dient dazu, Kontaktfenster zur Kontaktierung des Substrates und aller darin eindiffundierter Strukturen, also auch von Source- und Drainanschlüssen sowie von Poly1- und Poly2-Flächen, mit dem ersten Metalllayer herzustellen.

Metall1 definiert die Leiterbahnen des ersten Metalllayers.

Via dient zur Kontaktierung von Metall1 mit Metall2.

Metall2 ist der zweite Metalllayer.

Die in diesem Layout-Editor verwendeten Layer stehen in einer direkten Beziehung zu den bei der Chipherstellung benötigten Masken.
Die Aufgabe des Layoutens besteht darin, aus geeigneten Strukturen der verschiedenen Diffusionslayer die benötigten Bauelemente auf dem Chip zu realisieren und diese dann mit den Poly- und Metalllayern in der richtigen Art und Weise zu verbinden. Dabei ist eine Reihe von vom Hersteller vorgegebenen Regeln einzuhalten, in denen Mindestabstände, Mindestbreiten etc. festgelegt sind. Um automatisch zu überprüfen, ob diese Regeln an allen Stellen der Halbleiterstruktur eingehalten wurden, kann ein spezielles Programmpaket des CAD-Systems, der *Design Rule Check*, verwendet werden. Dieser Teil der CAD-Software erhält die Informationen über den verwendeten Herstellungsprozess aus einem sogenannten *Design-Kit*, in dem in maschienenlesbarer Form alle Design-Regeln aufgeführt sind.
Wurde das Layout mit dem Design-Rule-Check überprüft, besteht noch die Möglichkeit, aus dem Layout ebenso wie aus dem Schaltplan Netzlisten zu erzeugen. Diese Netzlisten können zum einen dazu dienen, die Schaltung zusätzlich mit parasitären Bauelementen zu simulieren, die durch die physikalische Realisierung des Layouts entstehen. Sie können zum anderen dazu benutzt werden, um die Netzlisten, die aus Schaltplan und Layout erzeugt wurden, miteinander zu vergleichen. Dies wird von einem Programmpaket erledigt, das sich *Layout versus Schematic Check* nennt.
Ist auch diese Überprüfung erfolgreich verlaufen, ist das Layout fertig und kann entweder zum Chiphersteller gegeben werden, um daraus die Masken für die Chipherstellung zu generieren oder als Teil eines größeren Layouts weiterverwendet werden.

6.3 Anforderungen an die Funktion der Logikzelle

Aufgrund der im vorangegangenen Kapitel beschriebenen Abläufe bei der Identifizierung von Clustern in einer zweidimensionalen Zellularlogikmatrix ergeben sich eine Reihe von Anforderungen an die elektronische Schaltung, mit der diese realisiert werden soll. Das Basiselement dieser Schaltung ist eine einzelne Logikzelle. Ausgehend von diesem Basiselement wird die gesamte Schaltung aufgebaut. Daher muss zunächst einmal auf die Anforderungen eingegangen werden, die für den Entwurf der Logikzelle ausschlaggebend waren. Ausgehend von dem in Abschnitt 5.2 beschriebenen Ablauf der Clusteridentifikation ergeben sich folgende Anforderungen an die Logikzelle:

- Die Zelle muss ein Speicherelement enthalten, in dem ein Treffersignal des zugeordneten Kristalls gespeichert werden kann (Trefferflipflop). Dieses Speicherelement muss von außen gesetzt werden können.
- Ein weiteres Speicherelement in der Zelle muss den Markierungszustand speichern (Markierungsflipflop). Dieses Speicherelement muss nicht von außen beschrieben werden können.
- Zur Initialisierung der Zelle wird ein Reset-Signal benötigt, mit dem alle Speicherelemente gelöscht werden können. Zusätzlich ist ein weiteres Reset-Signal erforderlich, mit dem nur die Markierungsflipflops gelöscht werden.
- Die Zelle muss getrennt sowohl über Spalten- als auch über Zeilenselektionsleitungen adressiert werden können.
- Es ist ein weiteres Speicherelement in der Zelle vorzusehen, mit dem Trefferdaten temporär ausgeblendet werden können (Maskierungsflipflop). Dieses Flipflop muss über ein Steuersignal gesetzt werden, wenn das Markierungsflipflop gesetzt ist. Über ein weiteres Steuersignal muss dieses Flipflop gelöscht werden können.
- Der Ausgang des nicht maskierten Trefferflipflops muss über eine geeignete Projektionslogik mit denen aller anderen Zellen der gleichen Spalte ODER-verknüpft werden, um die Projektion auf die x-Achse zu realisieren.
- Wenn die Spalte, in der sich die Zelle befindet, selektiert wurde, muss der Ausgang des nicht maskierten Trefferflipflops über eine zweite geeignete Projektionslogik mit denen aller anderen Zellen der gleichen *Zeile* ODER-verknüpft werden, um die Projektion auf die y-Achse zu realisieren.
- Wahlweise muss die Projektionslogik statt dessen auch die UND-verknüpften Signale aus Markierungs- und Trefferflipflop für die Projektion verwenden können.
- Das Markierungsflipflop muss dann, wenn der Markierungsprozess eingeleitet wird, selektiv gesetzt werden können. Der Markierungsprozess wird über ein besonderes Steuersignal $Mark$ eingeleitet. Die Selektion der Startzelle geschieht über die Spalten- und Zeilenselektion.

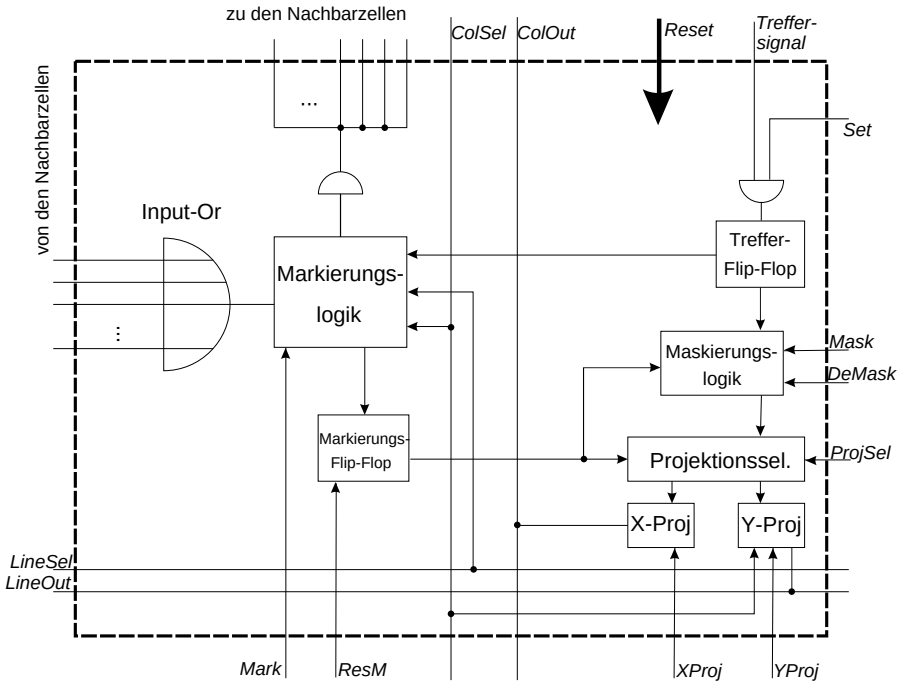


Abbildung 6.3: Vereinfachtes Blockschaltbild der Logikzelle für die Zellularlogik zur Clusteridentifikation

- Wenn in einer Zelle das Trefferflipflop und in einer Nachbarzelle das Markierungsflipflop gesetzt ist, muss während des Markierungsprozesses das eigene Markierungsflipflop ebenfalls gesetzt werden.
- Wenn das Signal $Mark$ aktiv ist, muss das Ausgangssignal des Markierungsflipflops an alle Nachbarzellen weitergeleitet werden.
- Über ein weiteres Steuersignal $ResT$ muss es möglich sein, selektiv über Spalten- und Zeilenselektionsleitungen das Trefferflipflop dieser Zelle zu löschen.

6.4 Der Entwurf der Logikzelle

Aus dem in Abschnitt 6.3 aufgelisteten Anforderungskatalog ergab sich das Grundkonzept für den Aufbau einer Logikzelle, das zu dem in Abbildung 6.3 dargestellten vereinfachten Blockschaltbild führt. Das Trefferflipflop wird gesetzt, wenn simultan ein Signal am Eingang *Treffersignal* und am Steueingang Set anliegt. Der Inhalt des Trefferflipflops wird sowohl an die Markierungs-, als auch an die Maskierungslogik weitergeleitet.
Mit der Maskierungslogik verbunden ist die Projektionslogik, mit der zunächst ausgewählt wird, welche Information projiziert werden soll. Abhängig vom Zustand des Signals $ProjSel$ wird entweder das Ausgangssignal des Markierungsflipflops oder das des Trefferflipflops weitergeleitet. Die eigentliche Projektion geschieht in zwei unabhängigen Einheiten für die x- und y-Projektion, die jeweils ein eigenes Steuersignal $XProj$, bzw. $YProj$ erhalten. Die Y-Projektionseinheit ist zudem mit dem Spaltenselektionssignal $ColSel$ verbunden, da die Projektion auf die Y-Achse nur selektiv in einer Spalte geschehen soll. Die Projektion selbst geschieht über Wired-Or-Verküpfungen auf den Ausgangsleitungen $ColOut$, bzw. $LineOut$, auf die später noch genauer

eingegangen wird.
Die Markierungsinformation wird im Markierungsflipflop gespeichert, das im Anfangszustand gelöscht ist. Wenn mit dem Steuersignal $Mark$ der Markierungsprozess eingeleitet ist, gibt es zwei Bedingungen, die dazu führen, dass dieses Flipflop von der Markierungslogik gesetzt wird. Die eine besteht darin, dass durch die externe Prioritätsentscheidung diese Zelle ausgewählt wurde und nun selektiv angesprochen wird. Dazu wurden die beiden Selektionssignale $ColSel$ und $LineSel$ mit der Markierungslogik verbunden. Die andere Bedingung besteht darin, dass in der Zelle selber das Trefferflipflop und in einer der Nachbarzellen das Markierungsflipflop gesetzt ist. Diese Information erhält die Zelle von den acht Nachbarzellen über getrennte Verbindungsleitungen, die mit einem 8-fach-ODER-Gatter verknüpft werden. Zudem muss die Markierungslogik mit dem Ausgang des Trefferflipflops verbunden sein. Das Setzsignal für das Markierungsflipflop wird über entsprechende Treiberstufen an die acht Nachbarn weitergeleitet. Mit dem Signal $ResM$ kann das Markierungsflipflop wieder in den Anfangszustand zurückgesetzt werden. Wenn die Zelle markiert wurde, kann mit der Maskierungslogik temporär das Trefferflipflop ausgeblendet werden. Dazu erhält die Maskierungslogik von außen das Steuersignal $Mask$. Das in der Maskierungslogik enthaltene Maskierungsfliplop wird dann gesetzt, wenn dieses Steuersignal anliegt und das Markierungsflipflop aktiv ist. Wenn nach der Clusterzählung der Ursprungszustand der Trefferflipflops wieder hergestellt werden soll, kann mit dem Steuersignal $DeMask$ das Maskierungsflipflop wieder gelöscht werden, sodass der Zustand des Trefferflipflops wieder transparent zur Verfügung steht.
Besonderes Augenmerk musste beim Design der Zelle auf die Größe der Projektionstransistoren und der Transistoren in den Treiberstufen gelegt werden, die zur Weiterleitung des Markierungssignals an die Nachbarzellen dienen. Für die Projektionstransistoren wurde eine Gatebreite von $50\,\mu\mathrm{m}$ bei der minimalen Gatelänge von $0{,}8\,\mu\mathrm{m}$ gewählt, womit die Transistoren nach den Angaben des Chip-Herstellers in Tabelle 6.1 einen Strom von maximal $20\ \mathrm{mA}$ zulassen. Für die Treiberstufen wurden Inverter mit pMOS-Transistoren mit $16\,\mu\mathrm{m}$ Gatebreite und nMOS-Transistoren mit $8\ \mu\mathrm{m}$ Gatebreite gewählt.
Abbildung 6.4 zeigt das Layout der Logikzelle. Dabei ist oben rechts das Treffer- und das externe Markierungsflipflop mit der dazugehörigen Ansteuerlogik angeordnet, links daneben befindet sich die Maskierungslogik. Im unteren Bereich der Zelle befindet sich links die Markierungslogik, in der deutlich das achtfach-Oder-Gatter und die vier Treiber für die Weiterleitung des Treffersignals zu den Nachbarzellen zu sehen sind. Rechts ist die Projektionslogik zu sehen. Hier fallen besonders die beiden Projektionstransistoren ins Auge. Diese beiden Transistoren sind von je zwei Guardringen umgeben, wobei der innere Ring aus einer p^+-Zone besteht, die mit der Masse verbunden ist, während der äußere Ring aus einer n^+-Zone besteht, die sich in einer n-Wanne befindet und mit $+5\ \mathrm{V}$ verbunden ist. Diese beiden Ringe dienen zum Schutz des Chips vor sogenannten Latch-Up-Effekten. Beim Latch-Up kommt es zum Zünden eines parasitären Thyristors, der aus den Diffusionszonen benachbarter n- und pMOS-Transistoren besteht [We94]. Dies kann z.B. dann auftreten, wenn die Source- oder Drain-Potentiale im Betrieb außerhalb der Versorgungsspannungspotentiale geraten. Da diese Gefahr besonders bei solchen Transistoren gegeben ist, die direkt mit Ausgängen des Chips verbunden sind, wie dies bei den Projektionstransistoren der Fall ist, müssen hier besondere Schutzmaßnahmen vorgesehen werden. Durch die Guardringe werden die Ladungsträger abgesaugt, die aus den Diffusionszonen in das Substrat eindringen. Dadurch werden benachbarte Diffusionszonen elektrisch voneinander getrennt.

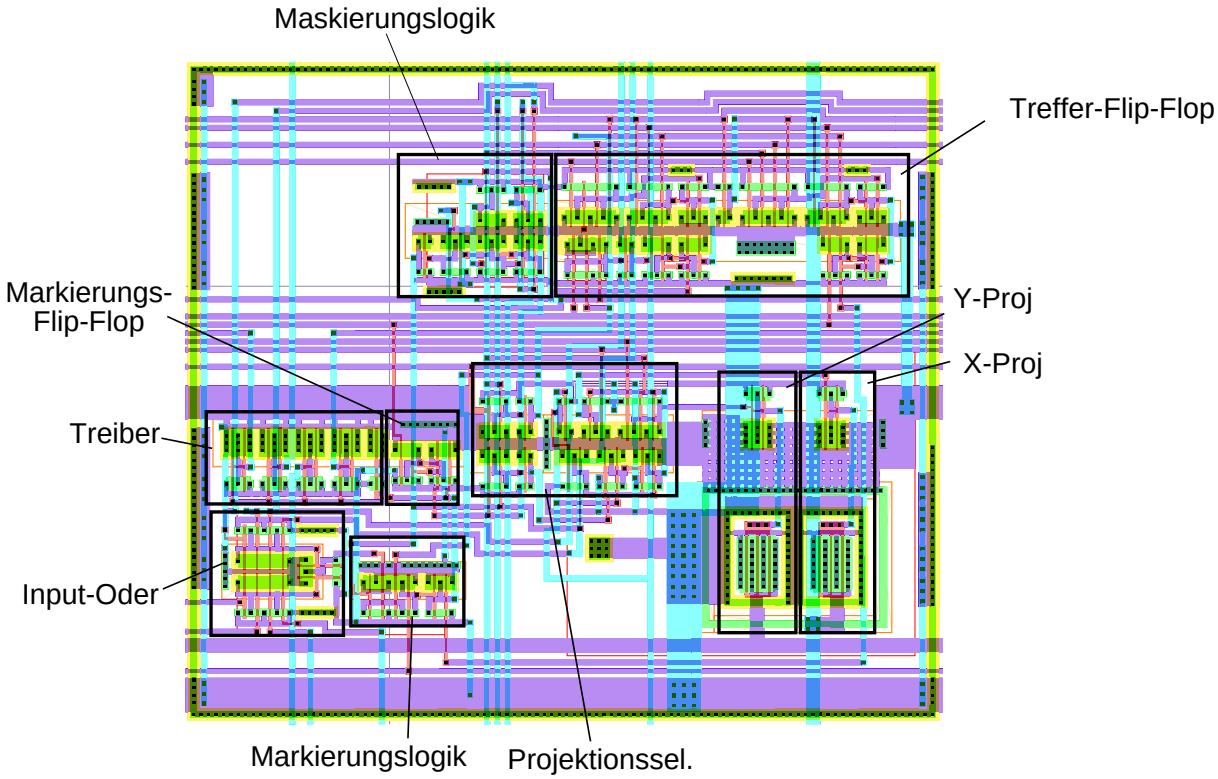


Abbildung 6.4: Das Layout der Logikzelle. Die Zelle hat eine Abmessung von 220 mal 195 mm^2

Um die gesamte Zelle herum liegt ebenfalls ein n^+-Guardring, um die einzelnen Zellen voneinander zu trennen. Die Zelle hat Außenabmessungen von etwa 220 μm mal 195 μm.
Bei der Festlegung der Anzahl von Zellen, die auf einen Chip platziert werden können, sind eine Reihe von Randbedingungen zu beachten. Mit den Chips soll eine Matrix von 60 mal 27 Zellen ausgefüllt werden. In Zeilenrichtung müssen die beiden Enden der Matrix miteinander verbunden werden, da der Barrel eine geschlossene Struktur besitzt. Somit müssen die Ränder der Matrix genau mit den Rändern der äußersten Chips übereinstimmen, was zur Folge hat, dass die Anzahl der Zellen pro Chip in dieser Richtung ein Teiler von 60 sein muss. Für die andere Richtung gibt es keine entsprechende Bedingung, da hier Randzellen unbenutzt bleiben können. Die Fläche des Chips sollte optimal ausgenutzt werden. Bei einer großen Anzahl von Ein- und Ausgängen wird jedoch die benötigte Chipfläche im Wesentlichen vom Umfang des durch die Bondingpads gebildeten Kranzes bestimmt, der die Zellen einschließt. Dabei ist zu berücksichtigen, dass die Anzahl der Bondingpads überproportional mit der Anzahl der auf dem Chip untergebrachten Spalten und Zeilen der Logikmatrix steigt. Die Ein- und Ausgangsleitungen lassen sich dabei in mehrere Kategorien einteilen:

- Spannungsversorgung:
  Es sind mindestens zwei Vcc- und zwei GND-Verbindungen vorgesehen.

- Steuersignale:
  Die Anzahl der Steuersignale ist unabhängig von der Anzahl der Zellen. Die Steuersignale

sind: $Reset$, $ResT$, $ResM$, $Mark$, $Mask$, $DeMask$, $XProj$, $YProj$, $ProjSel$, $SetSel$, Set und $TestSet$. Es werden somit 12 Eingänge benötigt.

- Selektionseingänge:
  Für die Spaltenselektion werden i Eingänge, für die Zeilenselektion j Eingänge benötigt.
- Projektionsausgänge:
  Für die Spaltenprojektion werden i und für die Zeilenprojektion j Ausgänge benötigt.
- Dateneingänge:
  Es werden $i \cdot j$ Eingänge für die Trefferdaten benötigt.
- Kaskadierungspins:
  Um die Kommunikation benachbarter Zellen, die auf unterschiedlichen Chips liegen, zu ermöglichen, müssen besondere Ein-/Ausgänge für diesen Zweck auf dem Chip untergebracht werden. Die genaue Arbeitsweise dieser IO-Ports wird weiter unten erläutert. Hier soll lediglich die Anzahl N_K der benötigten IO-Pins angegeben werden. Sie beträgt

$$N_K = 2(i-1) + 2(j-1) + 4$$

Somit beträgt die Anzahl der benötigten Bondingpads N_P

$$N_P = i \cdot j + 4i + 4j + 16 \tag{6.12}$$

Aufgrund dieser Überlegungen fiel die Entscheidung auf eine Matrix von vier mal vier Zellen pro Chip, womit nach Gleichung 6.12 die Anzahl der Pins 64 beträgt.
Das Layout der Zellen ist so aufgebaut, dass alle Steuer-, Selektions- und Ausgangsleitungen mit Leiterbahnen quer durch die Zelle weitergeleitet werden. Auf diese Weise können durch Verlängerung dieser Leiterbahnen die Zellen leicht nebeneinander angeordnet werden. Lediglich die Kommunikationsleitungen zum Austausch der Markierungsdaten erfordern eine spezielle Leiterbahnführung.
Für die Versorgung der Zellen mit den Steuersignalen sind ebenso wie für die Selektionsleitungen entsprechende Treiberstufen auf dem Chip untergebracht. Zudem wurden für die Selektionsleitungen noch zwei Adressdekoder auf dem Chip implementiert, die getrennt für die Spalten- und Zeilenselektionsleitungen arbeiten. Die Abbildung 6.5 zeigt das Prinzip der Adressdekodierung. Jeder Adressdekoder erhält zwei Adresssignale und ein Selektions-Signal. Wenn der Dekoder vom Selektionssignal freigeschaltet wird, aktiviert er abhängig vom Zustand der Adressleitungen die entsprechende Zeilen- oder Spaltenselektionsleitung. Damit können die Spalten- und Zeilenselektion völlig unabhängig voneinander aktiviert werden.
Da mit dieser Methode für die Zeilen und Spalten jeweils nur drei Eingänge für die Adressierung benötigt werden, verringert sich die Zahl der benötigten Bondingpads gegenüber der nach Gleichung 6.12 berechneten um zwei. Diese zwei Pads werden statt dessen zusätzlich als Masse-Pins benutzt.

6.5 Ein- und Ausgangsstrukturen

Die Verbindung der auf dem Chip integrierten Elektronik zu den anderen Zellularlogikchips und der Steuerelektronik geschieht über eine Reihe unterschiedlicher Ein- und Ausgabestrukturen,

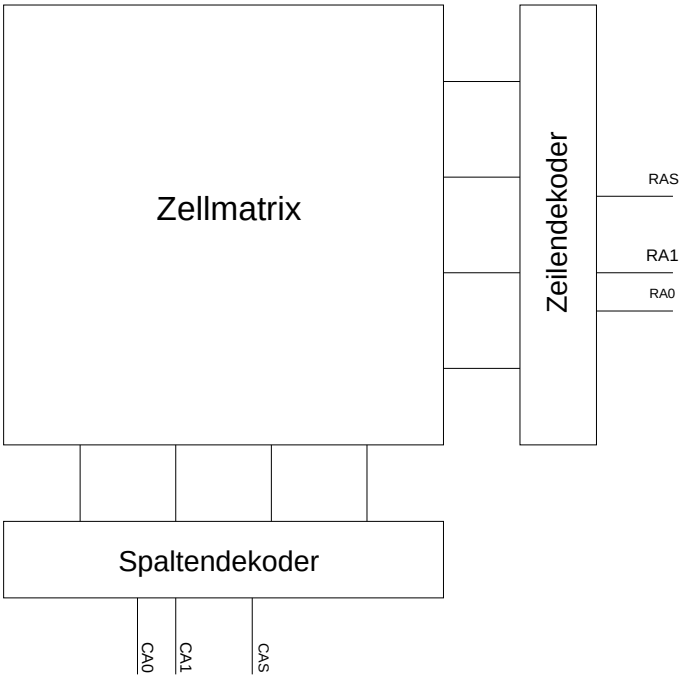


Abbildung 6.5: Prinzip der Adressdekoder zur Ansteuerung der Matrix

deren wesentlicher Bestandteil jeweils das Bondingpad ist. Dabei handelt es sich um eine etwa 100 mal 100 $\mu\mathrm{m}^2$ große Metallfläche, über der ein Fenster in die Nitridpassivierung des Chips geätzt ist. Für die Verbindung nach außen wird ein dünner Golddraht auf diese Fläche aufgepresst und mit Ultraschall festgeschweißt.
An die Verbindungsstrukturen werden eine Reihe von besonderen Anforderungen gestellt. Zum einen sind besondere Schutzmaßnahmen erforderlich, um die Elektronik auf dem Halbleiterchip vor Überspannungen, die von außen herangeführt werden können, zu schützen. Solche Effekte können besonders an den Eingängen zur Zerstörung der Eingangstransistoren führen, da schon geringe Ladungsmengen bei den kleinen Gate-Kapazitäten zu sehr hohen Spannungen führen können, die dann Durchschläge durch das Gateoxid verursachen. Deshalb werden hier Schutzdioden eingebaut, die die Ladung bei Überspannungen sofort nach Masse oder Vcc abführen. Um diese Dioden ihrerseits nicht zu überlasten, wurden zudem zwischen Bondingpads und Schutzdioden Widerstände eingefügt, die den Strom begrenzen.
An Ausgängen können Überspannungen zu den bereits erwähnten Latch-Up-Effekten führen, die durch Guardringe aus hochdotierten p- und n-Schichten vermieden werden, die die Ausgangstransistoren vollständig umgeben.
An den Ausgängen müssen die Transistoren wesentlich größere kapazitive und ohmsche Lasten treiben, als dies innerhalb des Chips notwendig ist. Daher kommen hier erheblich größere Transistoren zum Einsatz, die ihrerseits wieder größere Treiber erfordern. Dies führt an den Ausgangspads zu Treiberketten, die, angefangen bei der minimalen Transistorgröße, stufenweise höhere Treiberleistungen zur Verfügung stellen. Der Ausgangstransistor selber sollte in der Lage sein, einen Strom in der Größenordnung von $10\,\mathrm{mA}$ zu liefern. Dazu sind nach Tabelle 6.1 Gatebreiten von 25 $\mu\mathrm{m}$ bei nMOS-Transistoren, bzw. 50 $\mu\mathrm{m}$ bei pMOS-Transistoren erforderlich.
Für den Zellularlogikchip sind neben den Pads für die Spannungsversorgung drei verschiedene Arten von IO-Strukturen erfoderlich. Es handelt sich dabei um TTL-kompatible Eingangspads,

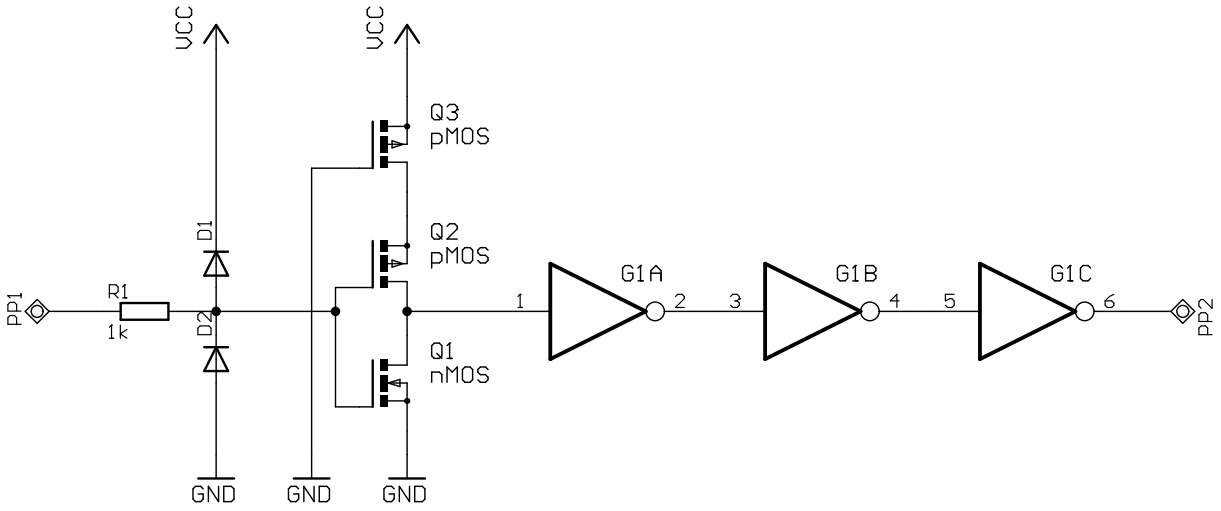


Abbildung 6.6: Schaltbild einer TTL-Eingangsstufe. Das Signal gelangt über das Bondingpad $PP1$ und den Schutzwiderstand zu der CMOS-Eingangsstufe. Um die Schaltschwelle auf TTL-Pegel anzupassen, wird mit dem Transistor Q3 die Versorgungsspannung künstlich heruntergesetzt. Danach folgt eine Kette aus drei Invertern

die Pads für die Projektionsausgänge und die Pads für die Kaskadierungsausgänge. Auf diese drei Strukturen soll etwas näher eingegangen werden.

6.5.1 TTL-Eingänge

Eingangsstufen bestehen im Wesentlichen aus der bereits erwähnten Schutzschaltung, die die nachfolgende Elektronik vor statischen Aufladungen schützt, und einer Kette von Invertern, die die Flanken des Eingangssignals regenerieren sollen. Um Eingänge zu realisieren, die kompatibel zu den TTL-Pegeln von maximal 0,8 V für Low-Pegel und minimal 2,4 V für High-Pegel sind, muss zudem die Schaltschwelle, die bei CMOS-Stufen genau auf der Hälfte der Versorgungsspannung liegt, heruntergesetzt werden. Dies erreicht man, wie in Abbildung 6.6 zu sehen, dadurch, dass man die Versorgungsspannung des ersten Inverters nach dem Bondingpad künstlich heruntersetzt, wozu in der Praxis ein als Widerstand benutzter pMOS-Transistor in der Zuleitung der Versorgungsspannung des ersten Inverters eingefügt wird.

In der Abbildung 6.7 ist die Realisation dieser Eingangsstufe als Layout für den verwendeten AMS-CMOS-Prozess zu sehen. Dominiert wird das Layout durch das große Bondingpad. Das Eingangssignal gelangt von diesem Bondingpad zu den Schutzwiderständen. Es gibt im verwendeten CMOS-Prozess mehrere Möglichkeiten, Widerstände zu realisieren. Im Widerstandsbereich bis etwa 1 $\mathrm{k\Omega}$ bietet sich die Verwendung von Polysilizium an. Dieses Material hat einen höheren spezifischen Widerstand als Metall. Er wird zudem vom Hersteller spezifiziert, sodass durch eine entsprechende Wahl der Abmessungen der gewünschte Widerstandswert genau erreicht werden kann.

In diesem Layout ist der Widerstand aufgeteilt in zwei Teilwiderstände mit jeweils etwa 500 Ω

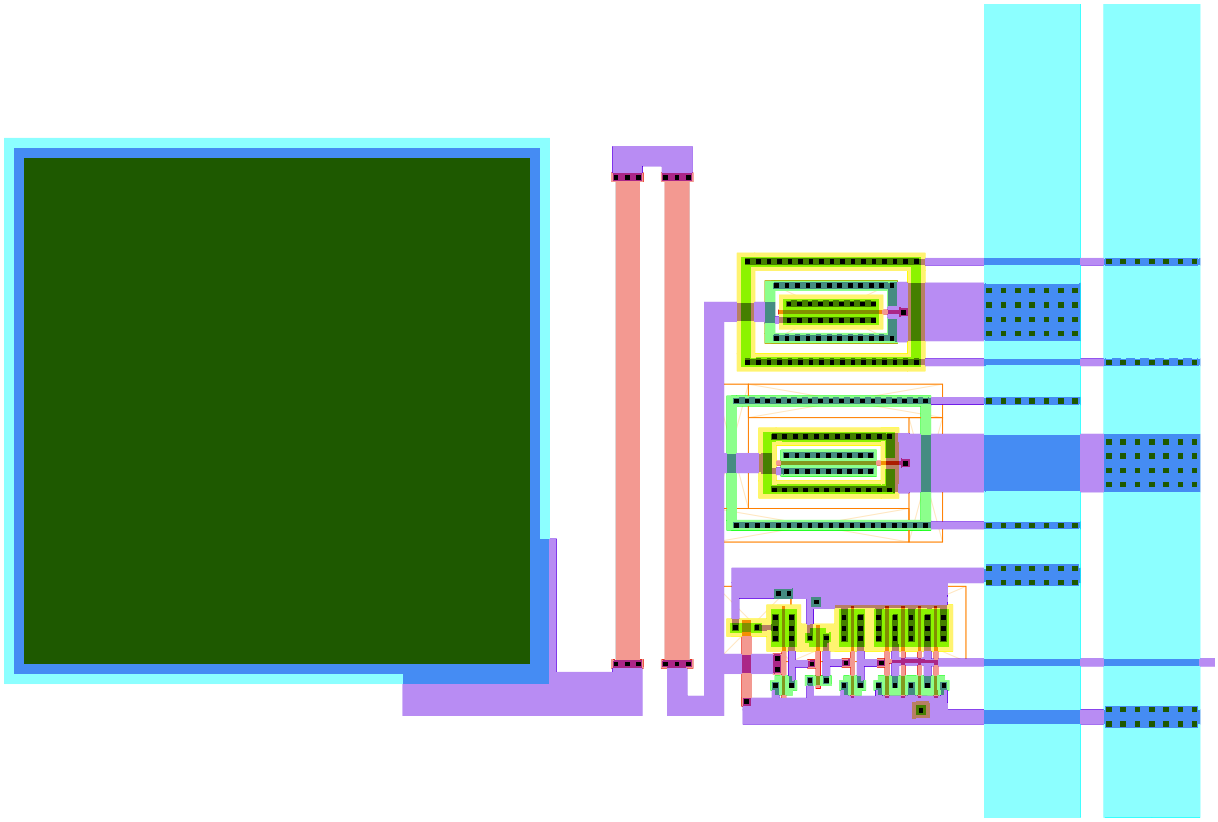


Abbildung 6.7: Layout der TTL-Eingangsstufe. Links sind wieder das große Bondingpad und direkt daneben die Schutzwiderstände zu erkennen. Daneben befinden sich im oberen Bereich die beiden Schutzdioden und darunter die Eingangselektronik

(exakt 470 Ω).
Um die Schutzdioden zu realisieren, wurden zwei große MOS-Transistoren, je einer mit p- und n-Kanal, verwendet, wobei die Kanal-Substrat-Strecke als Diode benutzt wird. Das Gate dieser Transistoren wird auf ein festes Potential gelegt.
Diese Dioden müssen wieder mit Guardringen, die gut in Abbildung 6.7 zu erkennen sind, gegen Latch-Up-Effekte geschützt werden.
Die Eingangselektronik belegt nur wenig Platz und befindet sich unterhalb der Schutzdioden. Sie besteht aus dem Inverter mit heruntergesetzter Betriebsspannung, der von drei Invertern gefolgt wird, deren Gatebreiten stufenweise ansteigt.
Rechts in der Abbildung sind die breiten Leiterbahnen für die Spannungsversorgung zu erkennen, die in allen Ein- Ausgabe-Stufen so angeordnet sind, dass die Stufen direkt nebeneinander platziert werden können und damit automatisch eine durchgehende Spannungsversorgung der Ein- und Ausgabestrukturen gegeben ist.

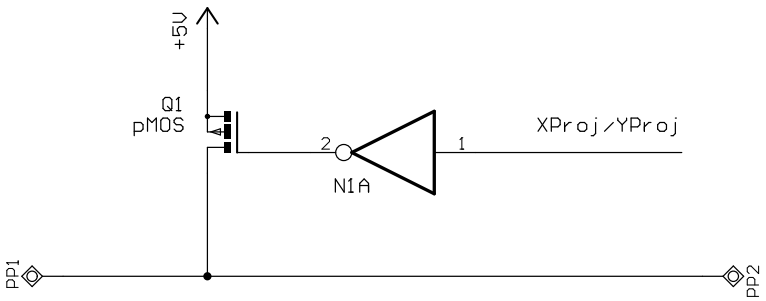


Abbildung 6.8: Prinzipschaltung der Projektionsausgänge. Um einen schnellen Wechsel zurück in den Ruhezustand zu ermöglichen, wird ein pMOS-Transistor als Bypass verwendet. Angesteuert wird dieser Transitor mit dem $Xproj$-, bzw. $YProj$-Signal

6.5.2 Projektionsausgänge

Die Schaltfunktion für die Projektionen wird bereits von den Projektionstransistoren in den Zellen wahrgenommen, sodass die Bondingpads lediglich mit diesen Transistoren verbunden werden müssen. Mit einem Pull-Up-Widerstand in der externen Beschaltung muss dafür gesorgt werden, dass sich die Projektionsausgänge im Ruhezustand auf dem High-Pegel befinden.
Voruntersuchungen, die u.a. mit einem CMOS-Testchip durchgeführt wurden, haben gezeigt, dass die Auswahl dieses Pull-Up-Widerstandes recht schwierig ist. Ein kleiner Pull-Up-Widerstand führt dazu, dass hohe Ströme fließen, was wiederum größere Projektionstransistoren bedingt. Größere Widerstände kommen mit kleineren Transistoren aus. Die kleinen Ströme führen jedoch auch dazu, dass wesentlich mehr Zeit benötigt wird, um vom Low-Zustand wieder zurück in den High-Zustand zu wechseln.
Dieses Problem wurde dadurch gelöst, dass in den Projektionsausgängen, wie in Abbildung 6.8 zu sehen, ein zusätzlicher pMOS-Transistor eingebaut wurde, der nach Beendigung der Projektion durchgeschaltet wird, um den Ausgang schnell wieder in den Ruhezustand auf High-Pegel zu bringen. Der externe Pull-Up-Widerstand hat jetzt nur noch die Aufgabe, die nicht aktiven Projektionsausgänge während der Projektion auf High-Pegel zu halten. Zusätzliche Schutzdioden, um Überspanungen abzuleiten, sind hier nicht notwendig, da die Kanal-Substratstrecken der verwendeten Ausgangstransistoren als Dioden wirken. Die Projektionsausgänge sind somit gegen Überspannungen geschützt.

6.5.3 Die Kaskadierungspads

Um die Kommunikation zwischen zwei Zellen herzustellen, die auf getrennten Chips untergebracht sind, müssen die Markierungssignale über Ausgänge aus dem Chip herausgeführt und auf dem benachbarten Chip über Eingänge wieder hineingeführt werden.
Dies erfordert eine hohe Anzahl von Ein- und Ausgängen. Um deren Anzahl zu reduzieren, wurden eine Reihe von besonderen Techniken entwickelt. Die erste Methode besteht, wie in Abbildung 6.9 dargestellt, darin, dort wo es möglich ist, mehrere Kommunikationsleitungen zusammenzufassen. Die Markierungssignale zweier benachbarter Randzellen eines Chips werden zunächst mit einem ODER-Gatter verknüpft und dann gemeinsam zum Nachbarchip übertragen. Dort wird dieses gemeinsame Signal aufgeteilt und zu den Zellen weitergeleitet, die den beiden Ursprungszellen gegenüberliegen. Damit ersetzt diese eine Signalleitung zwei horizontale und zwei Diagonalleitungen, die notwendig wären, wenn nach dem gleichen Verbindungssche-

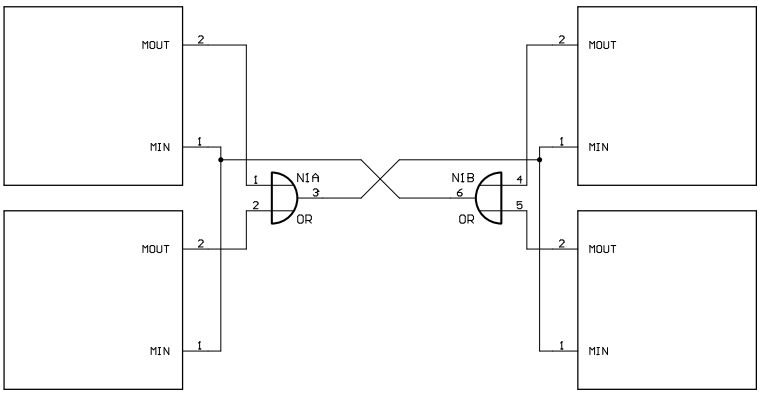


Abbildung 6.9: Die erste Methode um die Anzahl der Kaskadierungspins zu reduzieren. Die Markierungssignale zweier benachbarter Randzellen werden über ein ODER-Gatter miteinander verknüpft und über eine gemeinsame Leitung übertragen

ma wie innerhalb des Chips verfahren würde.
Zwischen den gleichen Zellen gibt es jetzt nur noch je eine Verbindung in beide Richtungen. Die Zahl der Verbindungsleitungen lässt sich also noch weiter reduzieren, wenn man von der Möglichkeit Gebrauch macht, eine Leitung in beide Richtungen zu benutzen.Um dies zu ermöglichen, wurden bidirektionale Kaskadierungspads entwickelt.
Abbildung 6.10 zeigt ein Prinzipschaltbild dieser bidirektionalen Kaskadierungspins. Im Ruhe-

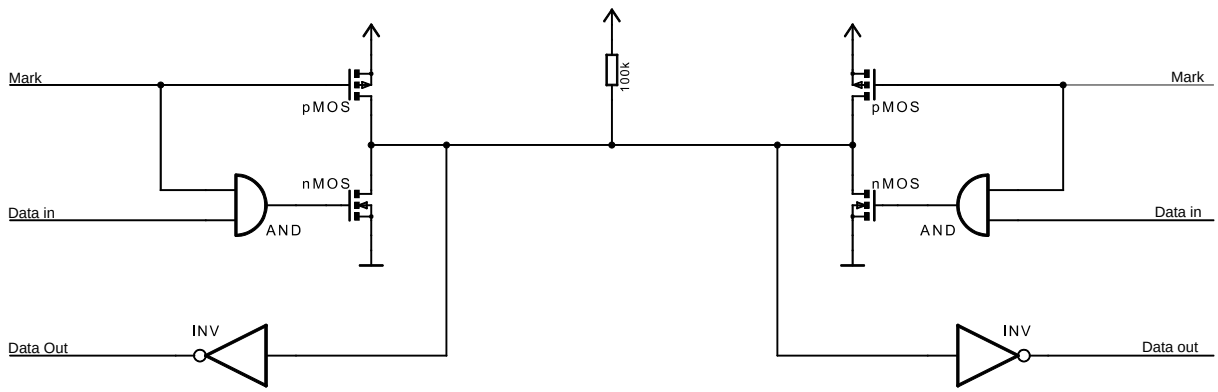


Abbildung 6.10: Prinzipschaltbild der bidirektionalen Kaskadierungspins

zustand, wenn das $Mark$-Signal nicht aktiv ist, wird die Kaskadierungsleitung mit dem pMOS-Transistor der Ausgangsstufe auf High-Pegel gehalten. Der nMOS-Transistor ist durch das AND-Gatter, das wegen des Low-Pegels auf der $Mark$-Leitung gesperrt ist, deaktiviert.
Wenn die Markierung durch eine Aktivierung des $Mark$-Signals eingeleitet wird, wird der pMOS-Transistor gesperrt. Solange der nMOS-Transistor nicht leitend wird, hält der externe Pull-Up-Widerstand jedoch die Kaskadierungsleitung weiterhin auf dem High-Pegel. Erst wenn aus einer der beiden Zellen ein Markierungssignal kommt, wird der nMOS-Transistor aktiviert und die Kaskadierungsleitung auf Low-Pegel gezogen. Da die Ausgänge an beiden Seiten der Kaskadierungsleitung eine Art Wired-Or-Schaltung bilden, ist es egal, aus welcher Richtung das Markierungssignal kommt, die Leitung kann also bidirektional genutzt werden.
Die Anwendung dieser Technik führt dazu, dass das Markierungssignal auch auf den Nach-

bar auf dem gleichen Chip und sogar auf die Zelle selbst zurück gelangt. Dies führt jedoch zu keinen weiteren Problemen. Die Nachbarzelle erhält bereits über eine separate Verbindung das Markierungssignal und die Zelle, die ein Markierungssignal aussendet, ist trivialerweise bereits markiert, sodass der zusätzliche Empfang ihres eigenen Markierungssignals keine Wirkung mehr hat.

6.6 Der Zellularlogikchip

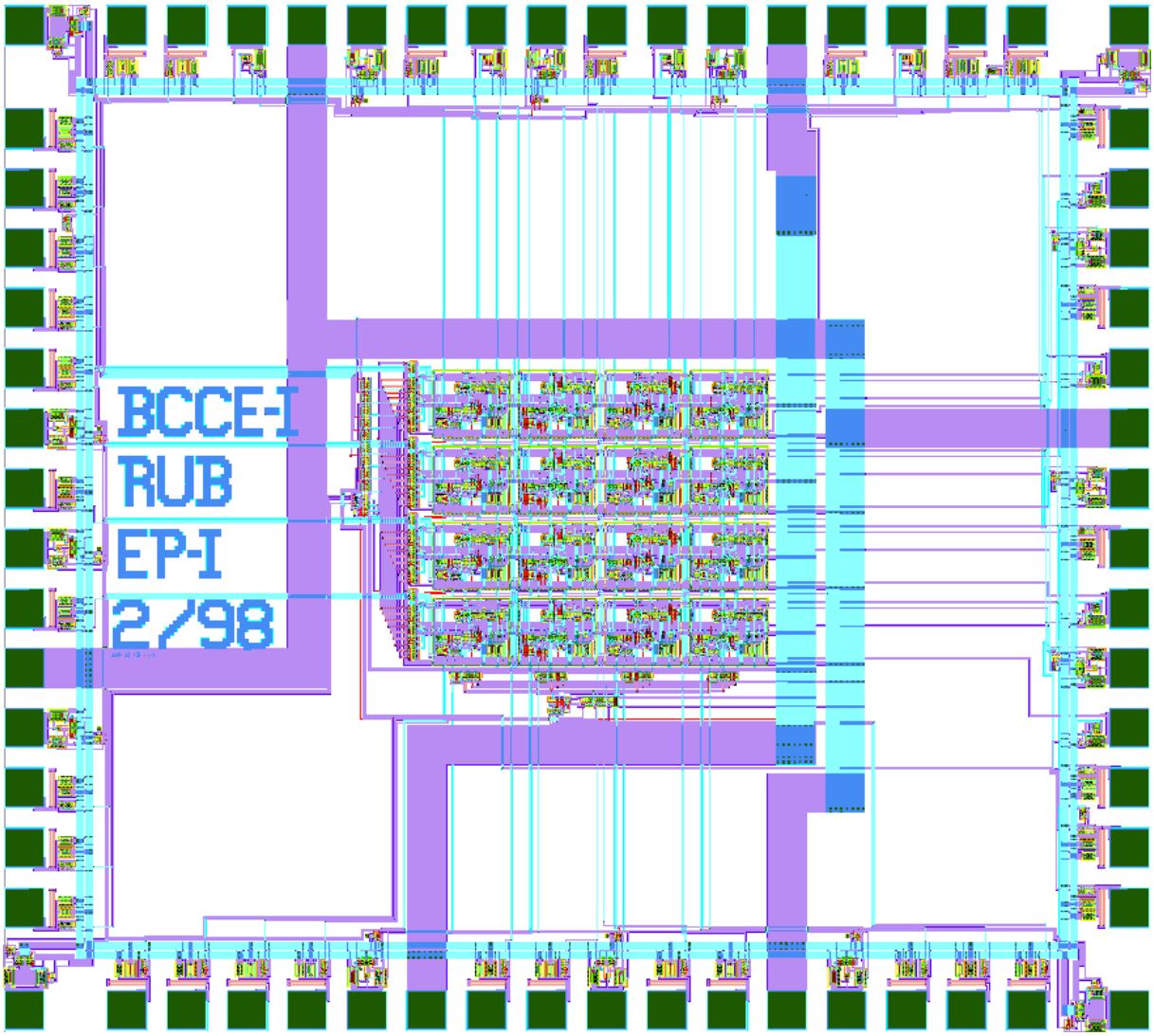


Abbildung 6.11: Das Layout des gesamten Chips

Abbildung 6.11 zeigt das Layout des gesamten Chips. Im Zentrum sind die in einer viermal-vier-Matrix angeordneten 16 Logikzellen zu sehen. Am linken und unteren Rand der Matrix befinden sich aufwendige Treiberstrukturen, um die Logikzellen mit den Steuersignalen zu versorgen. Die Spannungsversorgung geschieht über 100 μm breite Leiterbahnen, die von den

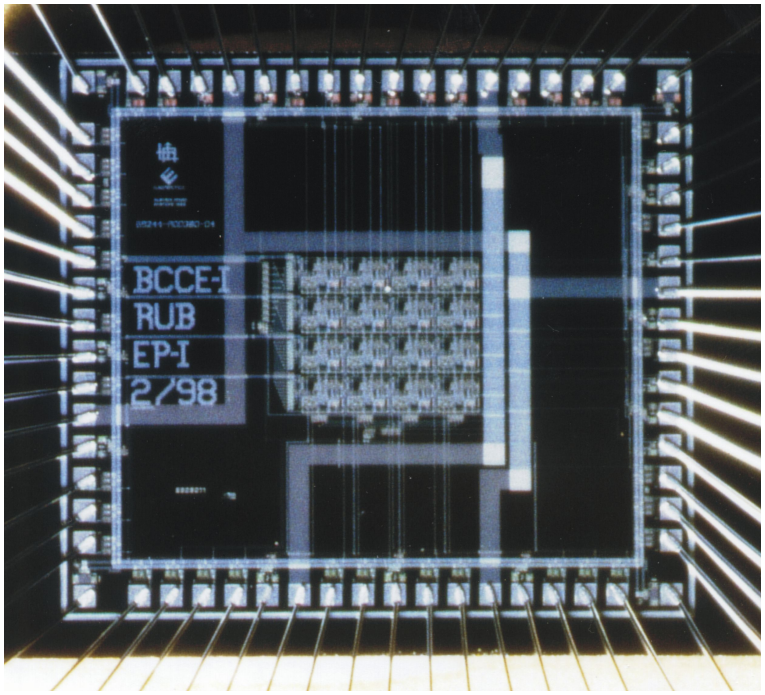


Abbildung 6.12: Mikrofotografie des gesamten Chips

Versorgungspads zu der Logikmatrix geführt werden. Damit ist sichergestellt, dass es bei Lastschwankungen durch das gleichzeitige Auftreten vieler Schaltvorgänge auf dem Chip nicht zu Rückwirkungen auf die Spannungsversorgung kommt.
Die 64 Bondingpads mit der zu ihnen gehörenden Interfaceelektronik bilden einen Kranz um dieses Zentrum herum und schliessen den Chip nach außen ab. Die Ausmaße des Chips betragen etwas mehr als 3 mal 3 mm^2. Eine genaue Beschreibung aller Schaltungs- und Layoutdetails des ASICs findet sich im Anhang B.
Das Design dieses Chips, der den Namen **B**ochum **C**ellular **C**luster **E**ncoder Nr. 1 erhalten hat, wurde im Februar 1998 als Datensatz an Europractice geschickt. Von dort wurde die Firma AMS damit beauftragt, den Chip im Rahmen eines Multi-Projekt-Wavers zu produzieren. Zunächst wurden von diesem Chip 10 Prototypen in LCC-68 Gehäusen gebondet und an die Ruhr-Universität ausgeliefert. Abbildung 6.12 zeigt den ins Gehäuse eingeklebten und gebondeten Chip.
Diese Prototypen wurden zunächst umfangreichen Tests unterzogen, um die Funktion der Logik in allen Punkten zu überprüfen und das dynamische Verhalten zu untersuchen. Anschließend wurde das Zusammenspiel mehrerer ASICs untereinander und mit der benötigten externen Elektronik in einer kleinen Testmatrix untersucht.

6.7 Die Tests der einzelnen Chips

Die Tests der einzelnen Chips hatte zwei Ziele. Zum einen sollte damit getestet werden, ob das Schaltungsdesign der Zellularlogikchips in allen Punkten korrekt ist und die daran gestellten Anforderungen erfüllt. Zum anderen sollte anhand dieser Stichprobe von 10 Prototypen abgeschätzt werden, wie hoch die zu erwartende Ausbeute an funktionierenden Chips sein wird.
Die Tests teilten sich dabei in zwei Kategorien auf: Zum einen rein logische Tests, die sich auf

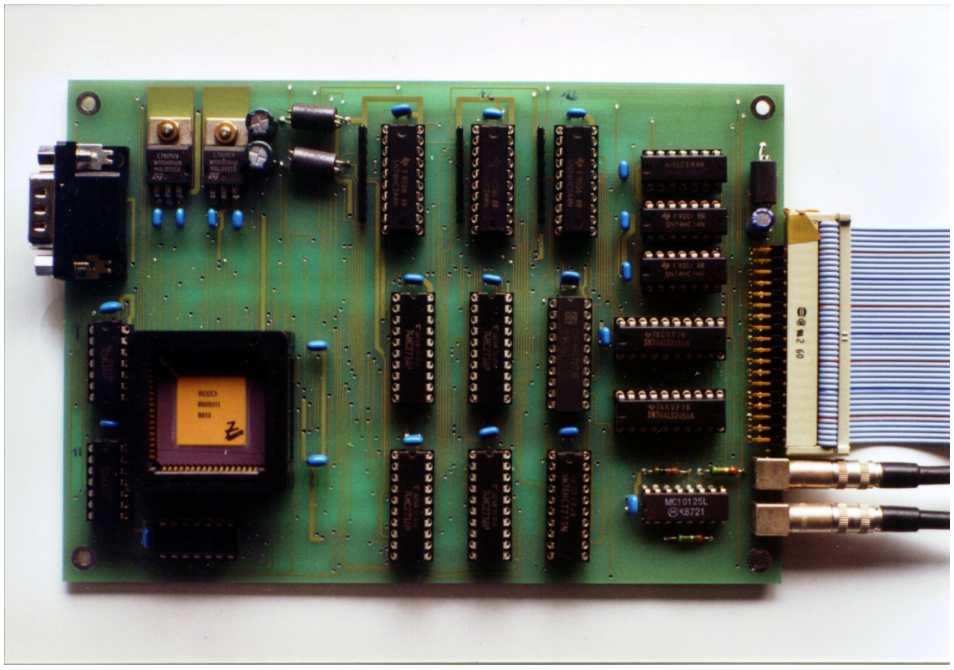


Abbildung 6.13: Die Testkarte für die Einzelchip-Tests

die logische Funktion der Chips beschränken, zum anderen dynamische Tests, die sich auf das Zeitverhalten der Chips beziehen. Für beide Testkategorien wurde ein gemeinsamer Testaufbau entworfen.

6.7.1 Der Testaufbau

Der Testaufbau, der in Abbildung 6.13 zu sehen ist, besteht im Wesentlichen aus einer Platine im Europa-Karten-Format, die einen Testsockel für den Zellularlogik-ASIC besitzt. Zur Ansteuerung des Chips befinden sich auf dieser Platine mehrere Register, über die an die Eingänge des Chips unabhängig voneinander beliebige Pegelkombinationen angelegt werden können. Von einem PC können die Register über ein CAMAC-Bussystem beschrieben werden, in dem eine spezielle Interface-Karte steckt, die mit der Testplatine über ein Flachbandkabel verbunden ist. Zudem können alle Ausgänge des Chips über dieses System ausgelesen werden. Die Kaskadierungspins des Chips sind so beschaltet, dass sie sowohl über das CAMAC System ausgelesen, als auch über Treiber mit Open-Kollektor-Ausgängen extern auf Low-Pegel gelegt werden können. Dadurch wird ein Test in beiden Signalflussrichtungen möglich.
Um die Geschwindigkeit zu messen, mit der der Markierungsprozess erfolgt, wird das Mark-Signal nicht direkt über ein Register, sondern von einem ECL-Pulser erzeugt, der von der Testkarte ein Triggersignal erhält, und dann über einen ECL-TTL-Wandler auf den ASIC gegeben. Gesteuert wird der Testaufbau von einem PC aus über ein kleines C++ Programm. Dieses Programm führt automatisch nacheinander eine Reihe von Tests durch, die im Folgenden beschrieben werden.

6.7.2 Der Single-Vector-Test

Beim Single-Vector-Test geht es darum, den Teil der Schaltung zu testen, der über die 16 Dateneingänge parallel die Trefferflipflops und die externen Markierungsflipflops setzt. Dazu wird bei diesem Test, der sich in zwei Teile gliedert, zunächst nacheinander ein High-Signal an jeden der 16 Dateneingänge angelegt und mit einem Set-Signal das entsprechende Trefferflipflop gesetzt. Dies wird dann mit dem Projektionsmechanismus überprüft.
Anschließend wird der gleiche Test mit den externen Markierungsflipflops durchgeführt, wobei zur Projektion die Markierungsflipflops genutzt werden, die von den Ausgängen der externen Markierungsflipflops automatisch mitgesetzt werden.
Bei diesem Test wird die Funktion der folgenden Baugruppen überprüft:

- Der Teil der Speicherlogik, der Treffer und externes Markierungsflipflop mit den parallelen Eingangsdaten setzt
- Das Markierungsflipflop und das NAND-Gatter, über das das Signal des externen Markierungsflipflops zugeführt wird
- Die komplette Projektionslogik

6.7.3 Der Single-Address-Test

Der Single-Address-Test entspricht weitgehend dem Single-Vector-Test. Allerdings werden bei diesem Test die Daten nicht über die 16 parallelen Eingänge in den Chip geschrieben, sondern die Flipflops werden selektiv über eine an den Spalten- und Zeilenadressdekoder angelegte Adresse mit dem $TestSet$-Signal gesetzt.
Die Baugruppen, die mit diesem Test überprüft werden, sind zum Teil die gleichen, die auch beim Single-Vector-Test geprüft werden. Getestet werden:

- Der Teil der Speicherlogik, der Treffer- und externes Markierungsflipflop selektiv mit dem $Testset$-Signal setzt
- Das Markierungsflipflop in der Markierungslogik mit dem NAND-Gatter, über das das Signal des externen Markierungsflipflops zugeführt wird
- Die komplette Projektionslogik
- Die Adressdekoder mit der gesamten Zeilen- und Spaltenselektion

6.7.4 Der Markierungstest

Beim Markierungstest werden nacheinander 10 verschiedene Testmuster in den Chip geladen, die voneinander getrennte horizontale, vertikale und diagonale Streifen enthalten. In diesen Testmustern werden nacheinander selektiv die beiden Randzellen jedes Streifens markiert, dann der Markierungsvorgang gestartet und schließlich das Ergebnis mit dem Projektionsmechanismus ausgelesen. Die Muster sind so gewählt, dass bei der Markierung jede der auf dem Chip vorhandenen Verbindungen zwischen zwei benachbarten Zellen benutzt wird, sodass bei diesem Test die gesamte Markierungslogik inklusive aller Nachbarschaftsverbindungen getestet wird.

6.7.5 Der Kaskadierungstest

Der Test der Kaskadierungsein- und -ausgänge erfolgt in zwei Teilen, wobei zunächst die Ausgangsfunktionen der Kaskadierungspins und dann die Eingangsfunktion getestet wird.
Beim Test der Ausgangsfunktion werden nacheinander alle Randzellen des Chips gesetzt. Diese Randzellen werden dann adressiert und der Markierungsvorgang gestartet, wobei der Zustand der Kaskadierungspins ausgelesen wird. Dieser Zustand wird dann mit einem Sollbitmuster verglichen, das sich aus der Verbindung der Kaskadierungspins mit der gesetzten Randzelle ergibt. Weiterhin werden nacheinander alle 16 Kaskadierungspins einzeln von außen auf den Low-Pegel gesetzt. Für jeden von ihnen werden dann alle Randzellen des Chips nacheinander gesetzt und das Markierungssignal aktiviert, ohne eine Zelle zu adressieren. Ist die gesetzte Randzelle mit dem aktivierten Kaskadierungspin verbunden, muss sie markiert werden, andernfalls muss sie unmarkiert bleiben. Dies wird durch die Auslese des Chips über den Projektionsmechanismus überprüft.

Die beschriebenen Tests wurden bei allen 10 gelieferten Prototypen durchgeführt. Dabei trat kein Fehler auf. Damit wurde gezeigt, dass zum einen das logische Design der Chips korrekt war, d.h. die Chips keine Schaltungsfehler aufweisen, zum anderen aber auch, dass mit einer hinreichend großen Ausbeute für die erforderliche Kleinserienproduktion zu rechnen war. Um Aussagen über einige dynamische Eigenschaften des Chips zu machen, wurden noch einige weitere Messungen durchgeführt.

6.7.6 Messung der Projektionsgeschwindigkeit

Der Zeitbedarf für den Projektionsvorgang setzt sich aus zwei Zeiten zusammen, der Verzögerungszeit, die zwischen dem Aktivieren des entsprechenden Projektionssignales $XProj$ oder $YProj$ und der Reaktion an den Projektionsausgängen liegt, und der Zeit, die für den Wechsel der Projektionsausgänge von High- auf Low-Pegel benötigt wird.
Um diese Zeiten zu messen, wird ein einfaches Muster in den Chip geschrieben, dann werden periodisch die Projektionssignale aktiviert, um dann die Verzögerungszeiten und die Flankensteilheit mit einem Oszilloskop messen zu können. In den Abbildungen 6.14 und 6.15 sind die abfallende und ansteigende Flanke des Signals an einem der Projektionsausgänge zu sehen. Die Abfall- bzw. Anstiegszeit beträgt 4,3 ns, bzw. 6,3 ns. Diese Werte werden im Wesentlichen durch die Größe der Projektionstransistoren, die für die abfallende Flanke verantwortlich sind, und der Bypasstransistoren in den Projektionsausgängen, die für die ansteigende Flanke verantwortlich sind, beeinflusst. Die gemessenen Zeiten sind hinreichend klein, sodass das Design auch in diesem Punkt den Anforderungen genügt.
Um die Verzögerungszeit zu ermitteln, muss mit dem Oszilloskop noch zusätzlich das Steuersignal für die Projektion dargestellt werden. Führt man dies durch, erhält man eine Verzögerungszeit von 4,9 ns für die x- und 5,6 ns für die y-Projektion.

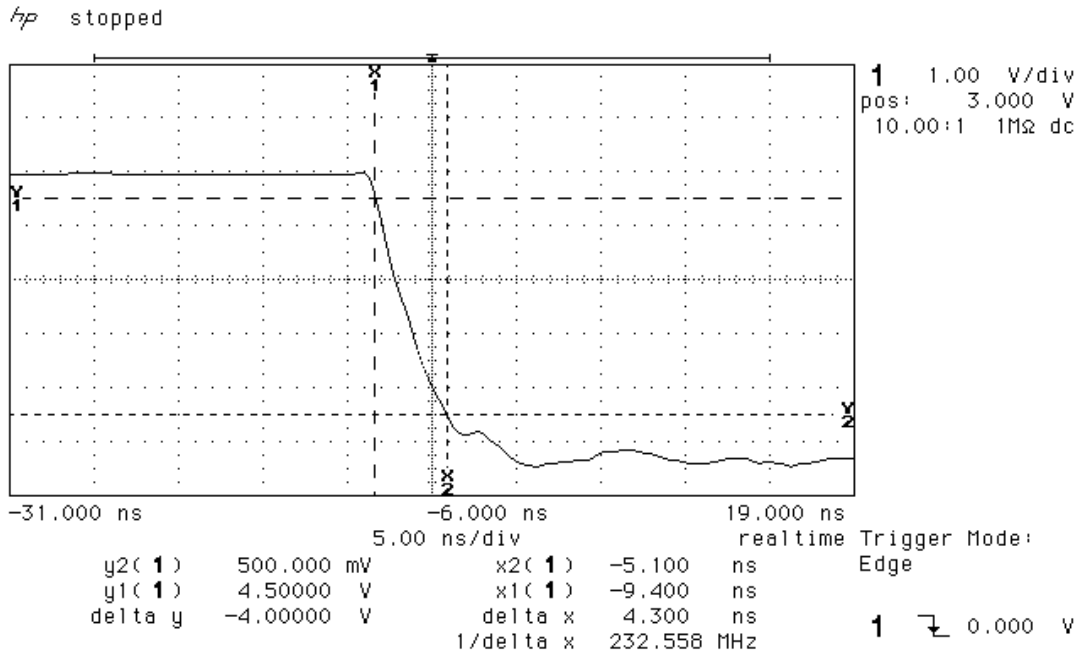


Abbildung 6.14: Oszillographische Darstellung der abfallenden Flanke des Projektionssignals

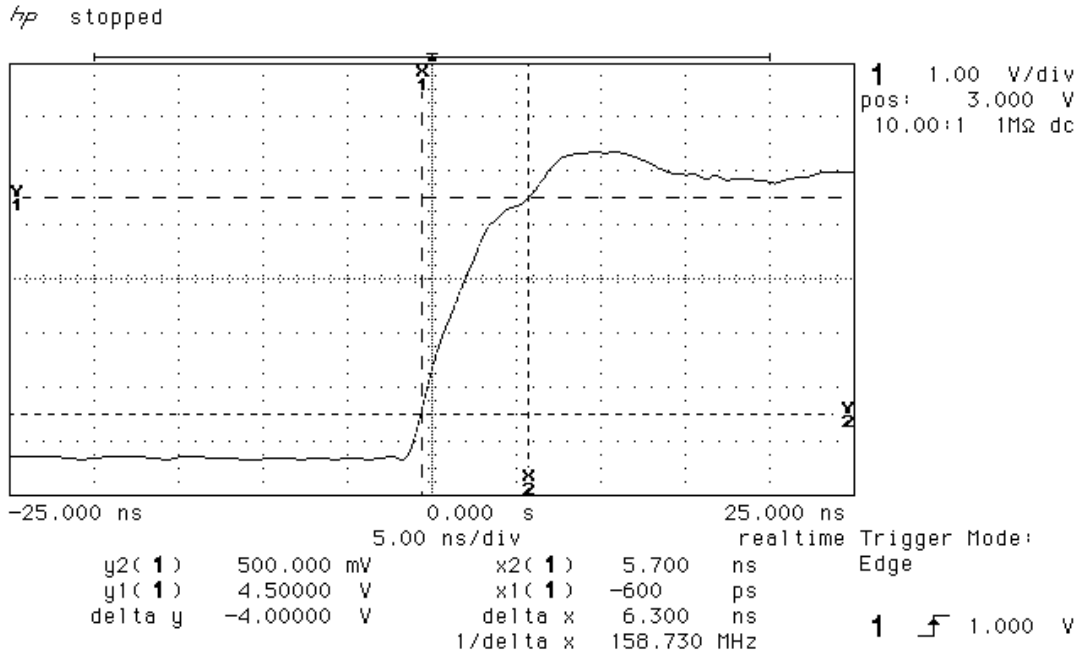


Abbildung 6.15: Oszillographische Darstellung der ansteigenden Flanke des Projektionssignals

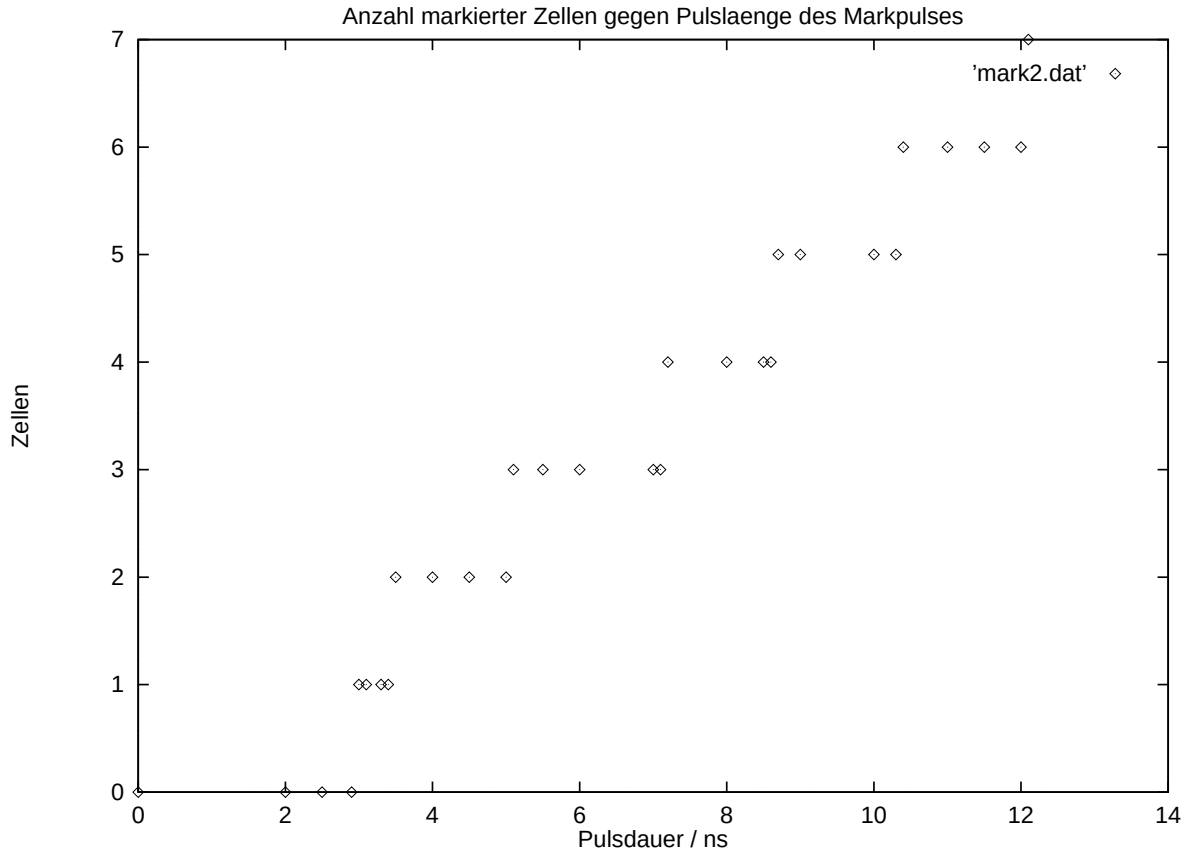


Abbildung 6.16: Anzahl der markierten Zellen in Abhängigkeit von der Länge des Markierungsimpulses

6.7.7 Messung der Markierungsgeschwindigkeit

Zur Messung der Markierungsgeschwindigkeit wird ein U-förmiges Muster in den Chip geschrieben, das 7 gesetzte Zellen enthält, die in einer fortlaufenden Reihe angeordnet sind, sodass, abgesehen von den beiden Endzellen, jede Zelle genau zwei gesetzte Nachbarn hat. Für den Test wird eine der beiden Endzellen selektiert und dann ein $Mark$-Impuls auf den Chip gegeben. Die Markierungslogik in den Chips arbeitet so, dass eine Markierung einer Zelle nur stattfindet, wenn das $Mark$-Signal aktiv ist, wodurch mittels einer Variation der Länge des $Mark$-Impulses die Zeit gemessen werden kann, die für die Markierung unterschiedlich vieler Zellen benötigt wird. In Abbildung 6.16 ist das Ergebnis dieser Messung an einem der 10 Prototypen zu sehen. Man sieht deutlich die Schrittstruktur, die entsteht, wenn in bestimmten Zeitabständen eine weitere Zelle markiert wird. Aus der Grafik lässt sich entnehmen, dass es etwa 3 ns in Anspruch nimmt, bis die erste Zelle markiert wird. Das Markieren jeder weiteren Zelle dauert dann etwa 2 ns. In einer weiteren Messung wurde untersucht, wie sich die Markierungszeit verlängert, wenn ein Cluster die Grenzen eines Chips überschreitet. Dazu wurde die Reihe der gesetzten Zellen unterbrochen und zwischen diesen beiden Teilketten eine Verbindung über zwei Kaskadierungspins hergestellt. Dabei zeigte sich, dass dort, wo das Markierungssignal über die Kaskadierungsleitung geführt wird, eine zusätzliche Signallaufzeit von ca 3 ns auftritt.

6.8 Test mit mehreren Chips

Nachdem alle 10 Protoypen einzeln die im vorigen Abschnitt beschriebenen Tests durchlaufen haben, wurde das Zusammenspiel dieser Chips in einer größeren Matrix getestet. Dazu wurde eine Testmatrix von 5 mal 2 Chips aufgebaut, in der bereits alle externen Komponenten der Zellularlogik implementiert wurden. Somit konnte diese Testmatrix bereits zur Erprobung des Gesamtkonzeptes der Zellularlogik, der Ansteuerung dieser Logik und der verwendeten Software benutzt werden.
Ausgehend von dieser Testmatrix wurde dann der vollständige Zellularlogiktrigger, wie er im nächsten Kapitel beschrieben wird, entwickelt.

6.9 Produktion und Test der ASIC-Kleinserie

Nachdem mit den in den vorherigen Abschnitten beschriebenen umfangreichen Tests gezeigt wurde, dass das Design der ASICs den Anforderungen genügt, wurden wiederum bei Europractice 213 *Dies*, dies sind ungebondete Halbleiterchips, die noch aus dem ersten Produktionsrun vorhanden waren, in PLCC-68-Gehäuse montiert und gebondet.
Im normalen Produktionsweg werden diese Dies vor dem Aussägen aus dem Wafer getestet, um das unnötige Weiterverarbeiten defekter Halbleiterchips zu vermeiden. Da dieser Test jedoch zur Kontaktierung der Halbleiterchips eine Nadelkarte erfordert, die relativ kostenintensiv als Einzelanfertigung gebaut werden müsste, wurde auf diesen Test verzichtet. Statt dessen wurden alle 213 Dies zunächst gebondet und dann einzeln mit dem vorher beschriebenen Testaufbau getestet. Dabei stellte sich heraus, dass etwa 10 % der 213 Zellularlogik-ASICs der Kleinserie defekt waren. Somit stehen mit knapp 200 ASICs deutlich mehr als die für den Trigger erforderlichen 105 ASICs zur Verfügung.

Kapitel 7

Die Realisierung des Zellularlogiktriggers

Bei der Realisierung des Gesamtaufbaus für den Kalorimetertrigger auf der Basis der Zellularlogikchips waren eine Reihe von Gesichtspunkten, die seinen elektronischen Aufbau und eine Reihe von Randbedingungen, die seinen mechanischen Aufbau betreffen, gemeinsam zu berücksichtigen. Um alle zuvor beschriebenen Funktionen ausführen zu können, muss die Zellularlogikmatrix durch einige zusätzliche Schaltungsteile ergänzt werden:

1. Die Prioritätslogik, deren Aufgabe darin besteht, bei einem vorgegebenen Treffermuster jeweils die Spalte bzw. Zeile mit der höchsten Priorität zu identifizieren und sie dann selektiv zu adressieren.

2. Ein Sequencer, der die für den in Abschnitt 5.2.2 beschriebenen Ablauf der Clusteridentifizierung und -auslese erforderliche Folge von Steuersignalen für die Zellularlogikmatrix und die Prioritätslogik erzeugt und die jeweilige Sequenz beendet, sobald alle Cluster identifiziert bzw. die zugehörigen Adressen ausgelesen sind.

3. Ein Interface zu der globalen Experimentsteuerung und -auslese des CB-ELSA-Experimentes, über das der Trigger am Beginn eines Experimentes initialisiert wird und über das computergenerierte Testmuster in den Trigger geladen werden können. Es muss zudem ein von der Experimentkontrolle auslesbares Statusregister enthalten, in dem die gesamte Information über den Triggerstatus zusammengefasst ist.

Alle genannten Elektronikkomponenten müssen zusammen mit der Zellularlogikmatrix in einem mechanischen Rahmen untergebracht werden, der einerseits genügend Platz für die gesamte Triggerelektronik bietet und andererseits die Zuführung der 1380 Barrelsignale zu den einzelnen Zellularlogikchips ermöglicht. In Hinblick darauf, dass es eine einfache Zuführung der benötigten Betriebsspannungen ermöglicht und bereits auf seiner Backplane über einen Bus verfügt, der zum Informationsaustausch genutzt werden kann, erschien es sinnvoll, hierbei auf ein Standardcrate zurückzugreifen. Deshalb wurde als mechanischer Rahmen ein VNX-9-Crate gewählt. Dabei handelt es sich um eine Erweiterung des VMEbus-Industriestandards auf Einschübe mit 9 Höheneinheiten. Damit steht eine Platinenfläche von 400×386 mm^2 zur Verfügung. Aufgrund der Verwendung des VNX-9-Crates bot es sich natürlich an, das Interface als VMEbus-Interface auszulegen. Da beim Bonner CB-ELSA-Experiment ohnehin bereits eine Reihe von Komponenten über VMEbus angesteuert wurde, ließ sich der Trigger somit leicht in die Experimentsteuerung integrieren.

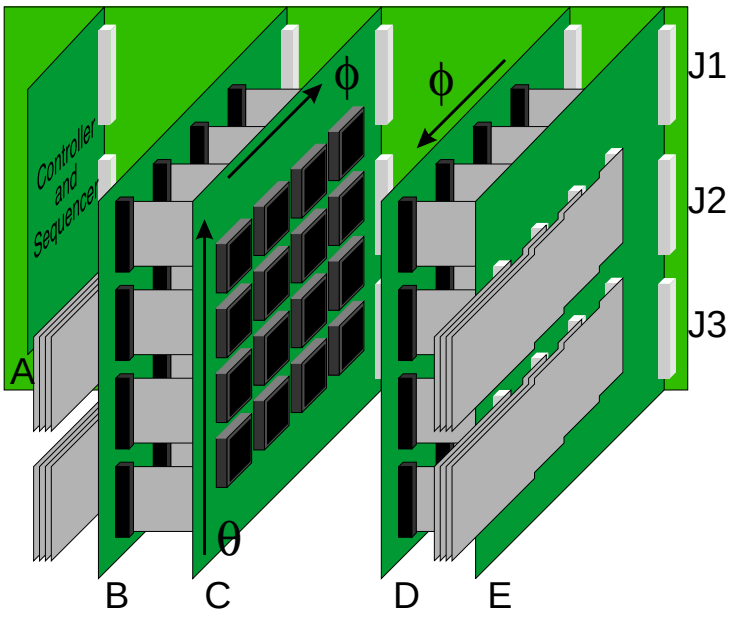


Abbildung 7.1: Schematische Darstellung des mechanischen Aufbaus der Triggerelektronik

7.1 Die mechanische Konzeption des Triggeraufbaus

Die Zellularlogikmatrix besteht aus 60 mal 27 Zellen, wovon 60 mal 26 Zellen für den Barreldetektor und 60 Zellen für die geplante Erweiterung auf dem Randbereich des TAPS Detektors benutzt werden. Bei dem gewählten ASIC-Design mit vier mal vier Zellen pro Chip sind dazu 15 mal 7, also 105 Chips notwendig. Durch die Verwendung der VNX-9 Platinen wurde es möglich, die gesamte Logik auf zwei Platinen unterzubringen. Abbildung 7.1 veranschaulicht das Konzept für den mechanischen Aufbau in einer schematischen Darstellung. Es erwies sich als sinnvoll, die Matrix in zwei ϕ-Sektoren aufzuteilen. Aus Abbildung 7.1 lässt sich auch ersehen, auf welche Weise die Positionen der Chips auf den beiden Platinen dem für das Kalorimeter definierten Polarwinkel θ und dem Azimutwinkel ϕ entsprechen. Um die geschlossene Ringstruktur des Barrels auf die beiden Triggerplatinen abbilden zu können, müssen diese über spezielle Flachbandkabel beidseitig miteinander verbunden werden.
Das größte mechanische Problem beim Aufbau des Triggers bestand in der Zuführung der 1380 Treffersignale vom Barrel. Dies geschieht über einhundertundzwanzig 40-adrige Flachbandkabel, die von den Latchmodulen kommen. Um die Triggerplatinen von den für deren Kontaktierung erforderlichen Steckverbindern zu entlasten, wurden die Signalzuführung und die Steckverbinder auf zwei separaten Platinen (B und E) untergebracht. Die Verbindung zwischen diesen zusätzlichen Platinen und den Haupttriggerplatinen geschieht über High-Density-Flachbandkabel, auf die Pfostenstecker mit einem Rastermaß von 1,27 mm aufgequetscht sind. Über jedes dieser Kabel werden die 16 Datensignale für genau einen Chip übertragen.
Die Prioritätslogik ist aufgrund der engen Verbindung mit der Zellularlgogikmatrix zusammen mit dieser auf den Platinen C und D untergebracht. Der Sequencer und das VMEbus-Interface wurden dagegen zu einem Controller-Modul zusammengefasst und auf einer weiteren Platine (A) aufgebaut. Für die Zuführung der Steuersignale vom Sequencer zur Zellularlogik wird ein Teil der Anschlüsse vom Stecker J2 (siehe Abbildung 7.1) benutzt, der nach den VMEbus-Spezifikationen [Vi87] dem Anwender zur freien Verfügung steht. Dazu wurde für die Verbindung über diesen Stecker ein spezielles Bussystem definiert. Die Steckerbelegung von J2 für

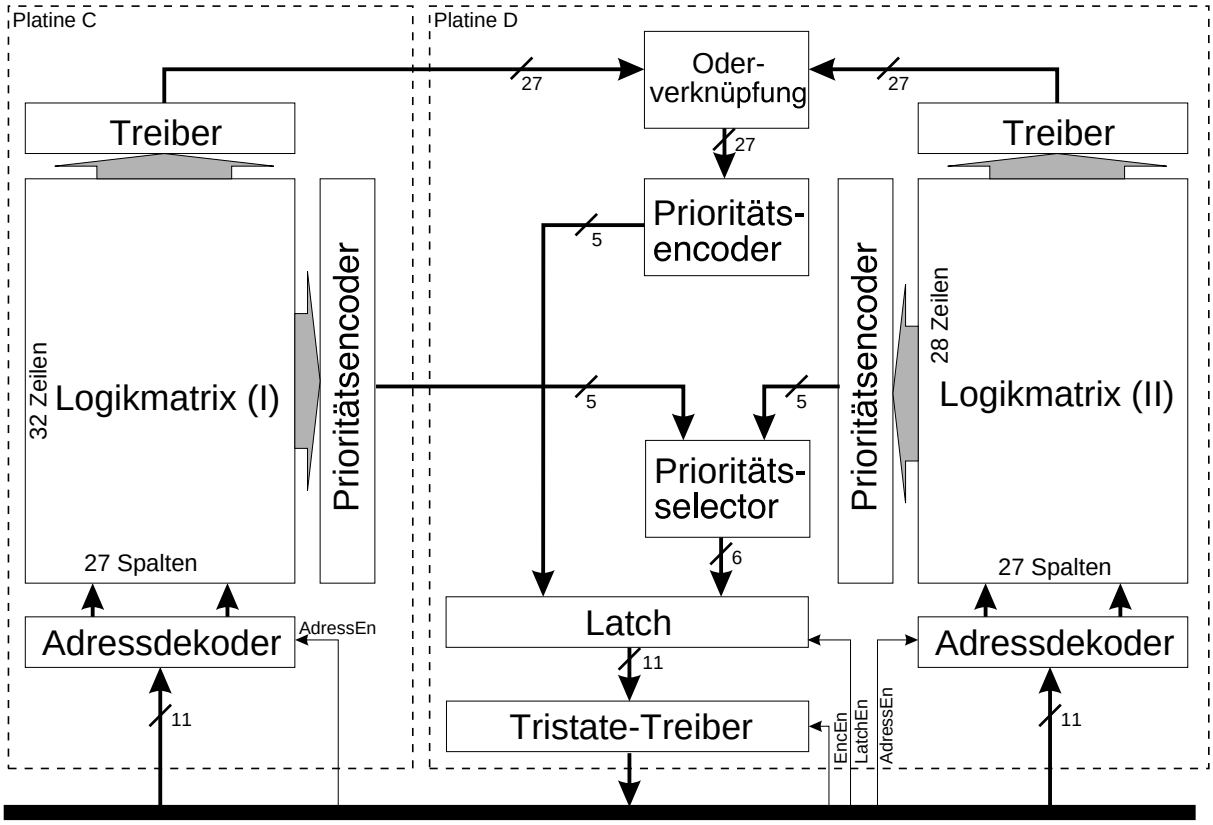


Abbildung 7.2: Blockschaltbild der beiden Haupttriggerplatinen

diesen Zellularlogikbus ist zusammen mit der Beschreibung des Controller-Moduls in Anhang D zu finden.

7.2 Die Haupttriggerplatinen

Auf den beiden Haupttriggerplatinen (C und D) musste neben den 105 Zellularlogik-ASICs auch die Prioritätslogik für die Spalten- und die Zeilenadressierung untergebracht werden. Dabei erforderte die Aufteilung der Logik in zwei Teilmatrizen auf getrennten Platinen zusätzliche Elektronikkomponenten für die Realisierung des Prioritätsencoders.
Wie in Abbildung 7.2 zu erkennen ist, werden die Spaltenprojektionssignale beider Teilmatrizen zunächst mit ODER-Gattern verknüpft, bevor sie auf den gemeinsamen Prioritätsencoder gegeben werden. Die Zeilenprojektionssignale werden hingegen zunächst von getrennten Prioritätsencodern verarbeitet. Im Prioritätsselector wird überprüft, ob am Encoder auf Platine D, der die höhere Priorität hat, ein Signal anliegt. Ist dies der Fall, werden die Daten dieses Encoders übernommen, andernfalls werden die Daten des Prioritätsencoders der Platine C weitergeleitet. Jeder der beiden Encoder erzeugt ein 5 bit breites Datenwort. Das sechste Bit, das für die Adressierung von 60 Zeilen notwendig ist, liefert der Prioritätsselector, und zwar abhängig davon, ob die Daten von der Platine C oder der Platine D stammen.
Die aus den Ausgängen von Spalten- und Zeilenprioritätsencoder zusammengesetzte Zelladresse passiert ein Latch, in dem sie bei Bedarf zwischengespeichert werden kann, und wird dann über

Tristatetreiber auf den Zelladressbus gelegt. Die Latches und die Tristate-Treiber werden von den Steuersignalen $LatchEn$ und $EncEn$ angesteuert.
Zur Selektion einer adressierten Zelle dienen wieder Adressdekoder, die unabhängig voneinander die adressierte Chipzeile oder -spalte auswählen, wenn das Steuersignal $AdressEn$ aktiv ist. Von den Zeilendekodern auf den beiden Platinen wird, in Abhängigkeit davon, welchen Zustand das höchste Bit der Zeilenadresse hat, jeweils nur einer aktiv.
Alle Steuersignale werden über durchgehende abgeschlossene Leitungen mit definiertem Wellenwiderstand zu den ASICs geführt. Diese aufwendige Zuführung der Treibersignale erlaubt es, auch sehr schnelle Pulsfolgen mit sehr steilen Signalflanken zu den Steuereingängen der ASICs zu übertragen.

7.2.1 Der Entwurf der Haupttriggerplatinen

Mit dem Entwurf der Haupttriggerplatinen wurde nicht, wie sonst üblich, auf der Schaltplanebene begonnen, weil bei der großen Anzahl von ASIC-Bausteinen, die individuell miteinander verbunden werden mussten, die Gefahr, dass dabei Fehler gemacht würden, zu groß erschien. Statt dessen bot es sich aufgrund der regelmäßigen Struktur der Schaltung an, von einem Rechnerprogramm Bauteil- und Netzlisten erzeugen zu lassen, die anschließend in das Platinenlayoutprogramm EAGLE eingelesen werden können. Da beide Platinen, aus denen sich der Trigger zusammensetzt, unterschiedliche Baugruppen enthalten, wurde eigens für diesen Entwurf ein Programm entwickelt, das solche Listen erzeugt und über einen Grad von Flexibilität verfügt, der es auch für andere Anwendungsfälle verwendbar macht. Dieser Netzlistengenerator wird in Anhang C näher beschreiben.
Dazu wird eine Datei vom Netzlistengenerator eingelesen, die die verwendeten Baugruppen sowie die Verbindungen zwischen diesen Baugruppen definiert. Der Netzlistengenerator erzeugt daraus Bauteil- und Netzlisten für die gesamte Schaltung und schreibt diese zusammen in eine EAGLE-Scriptdatei. Diese Scriptdatei enthält alle Befehle zur Platzierung der Bauelemente und zur Realisierung von Verbindungen zwischen Schaltungspunkten.
Beim eigentlichen Layoutprozess wurden zuerst die Leiterbahnen für die Steuersignale ‘von Hand’ verlegt und die Entflechtung der restlichen Leiterbahnen dem Autorouter überlassen. Die beiden Innenlagen der vierschichtigen Multilayerplatine wurden, von wenigen Durchgangspunkten abgesehen, auf Masse, bzw. +5 Volt gelegt. Die beiden Platinen sind beidseitig mit SMD-Bauteilen bestückt. Deren Verbindung mit der Platine erfolgt über insgesamt ca. 15000 Lötstellen. Die Fertigung dieser Platinen wurde an die Firma Straschu Industrie Elektronik GmbH übergeben, die die komplette Fertigung einschließlich Herstellung der Platinen, Bauteilbeschaffung und Automatenbestückung übernahm und von beiden Platinen je zwei Exemplare gefertigt hat. Die Abbildungen 7.3 und 7.4 zeigen die Oberseiten von Platine C und D. Die Abbildungen 7.5 und 7.6 zeigen Ausschnitte dieser Platinen im Detail.

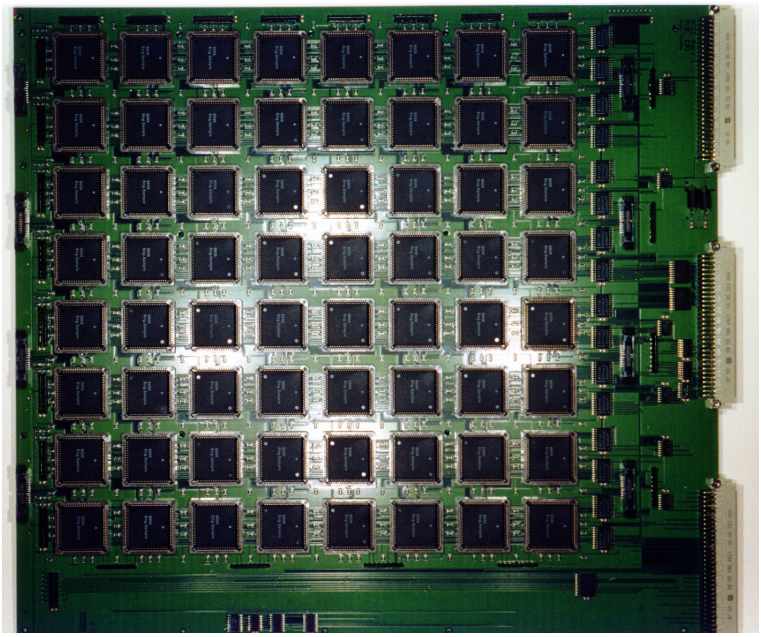


Abbildung 7.3: Die Oberseite der Triggerplatine C. Zu sehen ist die Matrix der Zellularlogik-ASICs

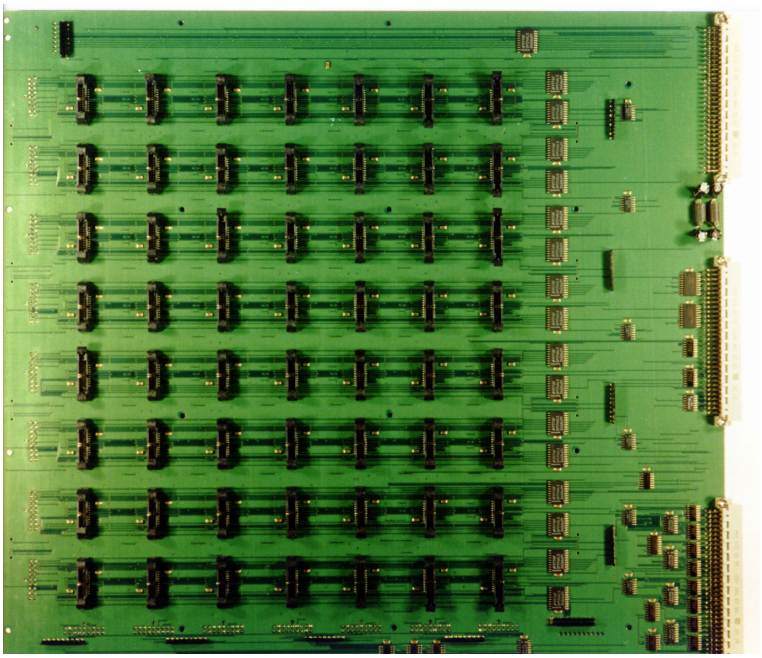


Abbildung 7.4: Die Oberseite der Triggerplatine D. Aufgrund der symmetrischen Anordnung beider Platinen im Crate sind hier die Steckverbinder zur Datenzuführung zu sehen

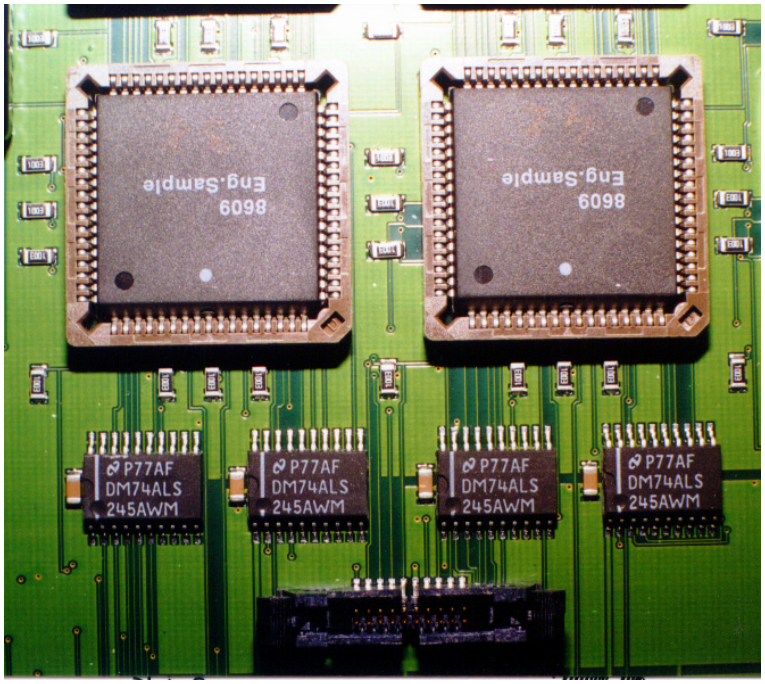


Abbildung 7.5: Zwei benachbarte Zellularlogik-ASICs mit Treibern. Zu sehen sind die ASICs in ihren PLCC-Sockeln, die von SMD-Widerständen umgeben werden, die als Pull-Up-Widerstände für die Kaskadierungsleitungen dienen. Darunter befinden sich die Treiber-ICs für die Steuersignale der Chips und ein 20-Pol-Fine-Pitch-Stecker für die Verbindung mit der zweiten Logikplatine

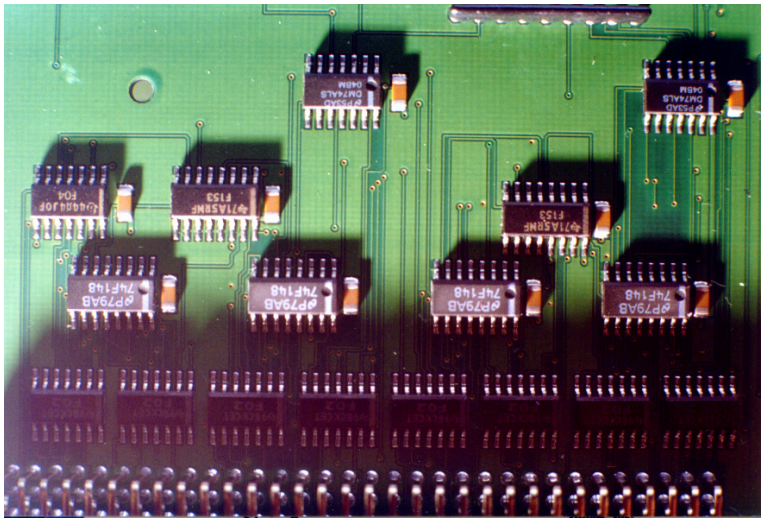


Abbildung 7.6: Oder-Verknüpfung und Prioritätsencoder auf Platine D. Über die VG-Leiste (unten) kommen die Projektionssignale von Platine C und werden über ODER-Gatter mit denen der Platine D verknüpft. Darüber befindet sich der Prioritätsencoder für die Spalten

7.3 Die Signalzuführung

Das Layout für die beiden übrigen Platinen B und E, über die die Signalzuführung für die eigentlichen Triggerplatinen erfolgt, wurde ebenfalls unter Verwendung des Netzlistengenerators und des EAGLE-Autorouters erstellt.
Die Terminierung der von den Latchmodulen kommenden Signale erfolgt mit einem Widerstandsnetzwerk (330 Ω nach $+5$ V und 4,7 kΩ nach Masse) auf kleinen Platinen, die auf die Signalzuführungsplatinen aufgesteckt sind. Zur Aufbereitung der Signale werden Signaltreiber vom Typ 74ACT244 eingesetzt. Auf den beiden Trägerplatinen werden die Signale dann umgruppiert, um sie den jeweiligen Zellularlogik-ASICs zuzuordnen.
Die Herstellung der Träger-, und der Aufsteckplatinen wurde der Firma Beta-Layout GmbH übergeben. Die Aufsteckplatinen wurden von den Lebenshilfe-Werkstätten in Oberhausen bestückt und im Lötbad gelötet, wohingegen die beiden Trägerplatinen im Labor von Hand bestückt und gelötet wurden.

7.4 Das Controller-Modul

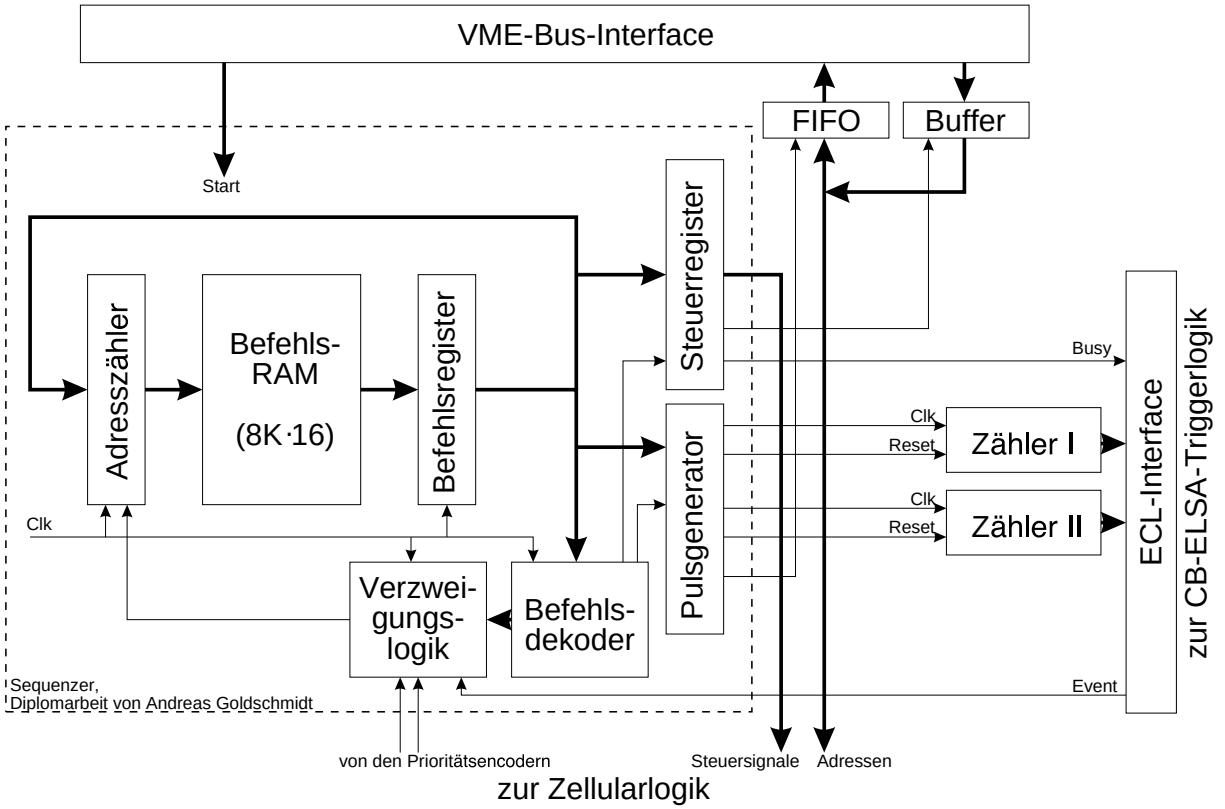


Abbildung 7.7: Blockschaltbild des gesamten Controller-Moduls. Als Kernstück ist der Sequencer zu erkennen

Zur Ablaufsteuerung der Triggerelektronik dient ein Controller-Modul, das als VME-Einheit realisiert wurde. Kernstück dieser Einheit ist ein frei programmierbarer Sequencer, für den A. GOLDSCHMIDT im Rahmen seiner Diplomarbeit [Go98] einen Prototyp entwickelt hat. Dieser Sequencer enthält ein Befehls-RAM mit einer Speicherkapazität von 8K $\times$ 16 bit für den Programmcode, der beim Einschalten des Sequencers automatisch aus einem EPROM in das RAM geladen wird. Ein Adresszähler adressiert, angefangen von der jeweiligen Startadresse des abzuarbeitenden Programmes, das RAM, das darauf ein Befehlswort an das Befehlsregister übergibt. Dieses Wort wird vom Befehlsdekoder dekodiert, der dann die entsprechenden Steuerregister setzt, um die Steuersignale für die Elektronik in Form von Pegelsignalen und Impulsen zu erzeugen.
Der Sequencer ist zudem in der Lage, bedingte und unbedingte Sprünge auszuführen. Dies wird von der Verzweigungslogik erledigt, die wiederum vom Befehlsdekoder angesprochen wird. In der Abbildung 7.7 ist der Sequencer als Hauptkomponente im Blockschaltbild der Controllerkarte zu erkennen.
Der Sequencer ist vollständig in Fast-TTL-Technik aufgebaut und in der Lage, mit einer Taktfrequenz von mehr als 20 MHz zu arbeiten, wobei jeder Befehl in einem einzigen Taktzyklus

Befehl \ Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
unbedingter Sprung	0	0	0	0	Sprungadresse											
bedingter Sprung	0	Konditionscode			Sprungadresse											
Ausgabe des Steuerworts für die Zellularlogik	1	0	0	0	Steuerwort											
Ausgabe eines Steuersignals an ein Bauteil	1	0	0	1	D	Adresse eines Ausgangs			X	X	X	X	X	X	X	X
Ausgabe eines Impulses an ein Bauteil	1	0	1	0	X	Adresse eines Ausgangs			X	X	X	X	X	X	X	X
Programm stoppen	1	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X
No Operation	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X

Tabelle 7.1: Bitstruktur der Sequencerbefehle. Das *X* steht für einen beliebigen Eintrag 0 oder 1 und *D* für einen beliebigen aber definierten Eintrag 0 oder 1[Go98]

ausgeführt wird. Die Tabelle 7.1 gibt eine Übersicht über alle Befehle, die vom Sequencer ausgeführt werden können.

Außer dem Sequencer sind auf dem Controller-Board zwei Binärzähler, mit denen die Cluster gezählt werden können, und ein FIFO-Speicher implementiert, in dem, nach Zugehörigkeit zu den identifizierten Clustern sortiert, die Adressen aller gesetzten Zellen gespeichert werden können. Dieser FIFO-Speicher kann über das mit dem Sequencer zusammen auf der Controller-Karte implementierte VME-Interface von einer Steuer-CPU ausgelesen werden.

7.4.1 Das VME-Interface

Abbildung 7.8 zeigt das Blockschaltbild des VME-Interfaces für das Controller-Modul. Die gesamte Buszugriffslogik ist in zwei programmierbaren Logikbausteinen der Firma Lattice vom Typ ispLSI 1016 implementiert, wobei der eine Baustein für Datentransfer-Buszyklen und der andere für die Interrupt-Behandlung benutzt wird.

Ein Buszyklus wird nur dann begonnen, wenn die anliegende Adresse im Adressbereich des Controller-Mdouls liegt. Um dies sicherzustellen, enthält das VME-Interface DIP-Schalter zur Einstellung der Basisadresse und einen Adresskomparator. Der Ausgang dieses Komparators ist mit dem Logikbaustein verbunden, der für Datenzyklen zuständig ist.

Die einzelnen Register auf dem Controller-Modul werden vom VME-Interface über die anliegende VMEbus-Adresse angesprochen. Gleichzeitig wird der Datenbuffer des VME-Interfaces aktiviert.

Tabelle 7.2 gibt eine Übersicht über alle Register, die in die Controllerkarte implementiert sind. Von entscheidener Bedeutung ist das Statusregister. Es enthält sowohl eine Reihe von Statusbits, die Auskunft über den Zustand des Controllers geben, als auch Steuerbits, mit denen der Controller beeinflusst werden kann. Die folgende Tabelle zeigt den Aufbau des Steuerwortes

Die Bits 12 bis 15 steuern die Interruptbehandlung des Controllers. Mit den Interrupt-Enable-

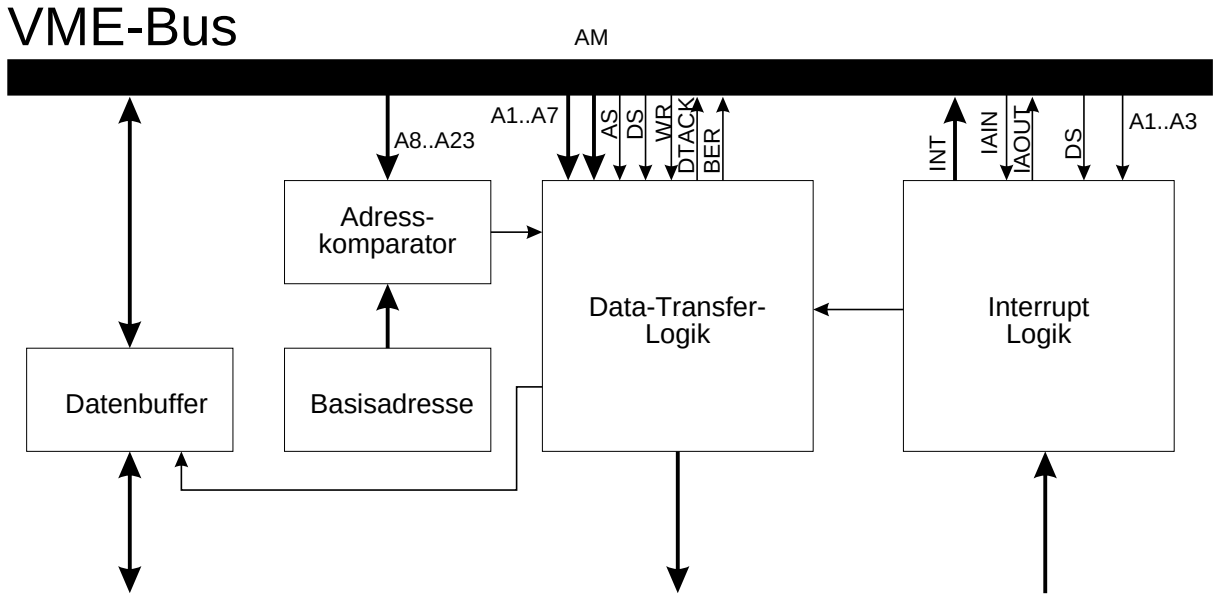


Abbildung 7.8: Blockschaltbild des VME-Interfaces

Bit IEn werden Interrupts generell freigegeben, bei gelöschtem Bit sind sie dagegen komplett gesperrt.

Bit	15	14	13	12	11	10	9	8
	IPen	IRes	IEn	IMask	$\overline{\text{Res}}$	Startcon	Stop	Start
	R	RW	RW	RW	RW	RW	RW	RW
Bit	7	6	5	4	3	2	1	0
				Timeout	Running	Full	Empty	Busy
				R	R	R	R	R

Wird auf dem Controller-Modul eine Interruptbedingung erfüllt, wird dies, sofern das *IEn*-Bit gesetzt ist, zunächst im sogenannten Interrupt-Pending-Flipflop gespeichert. Der Ausgang dieses Fliflops kann über das Statusbit *IPen* abgefragt werden. Das Auslösen des Interrputs kann jedoch noch vom Interrupt-Maskierungs-Bit *IMask* verhindert werden. Ist dieses Bit gelöscht, werden keine Interrupts ausgelöst. Dies geschieht erst, sobald *IMask* gesetzt wird. Mit dem Interrupt-Reset-Bit *IRes* kann das Interrupt-Pending-Flipflop jederzeit gelöscht werden.
Mit dem $\overline{Res}$-Bit wird die gesamte Controllerkarte initialisiert. Dazu gehört auch das Laden des Sequencer-RAMs mit den Daten aus dem EPROM, wobei zu beachten ist, dass dies einige Zeit in Anspruch nimmt. Wird für den Sequencer der interne Quarzoszillator verwendet, beträgt die Gesamtzeit für den Ladevorgang etwa 2,7 ms.
Mit dem $Startcon$-Bit kann das Startsignal für den Sequencer, das im Normalbetrieb von der Haupt-Triggerlogik des CB-ELSA-Experimentes kommt, auch softwaremäßig ausgelöst werden. Mit den Steuerbits $Start$ und $Stop$ lässt sich der Sequencer starten und anhalten.
Bei allen Steuerbits ist zu beachten, dass diese Bits nach dem Aktivieren von der Software wieder zurückgesetzt werden müssen, da dies nicht automatisch geschieht.

Register	Breite	Richtung	Adress-Offset
Status	Wort	R/W	0
Latch-Control	Wort	W	2
Startadresse	Wort	W	4
Zelladresse	Wort	W	6
FIFO Ausgabe	Wort	R	8
Zähler I	Byte	R	10
Zähler II	Byte	R	11
Zellzähler	Wort	R	12

Tabelle 7.2: Die Register der Controller-Karte, die über den VME-Bus angesprochen werden können. Breite gibt die Größe des Registers (Byte oder Wort) und Richtung die Datenrichtung, in der Zugriffe auf dieses Register erlaubt sind (R = Lesen, W = Schreiben), an

Im unteren Byte des Wortes befinden sich die Status-Bits. $Timeout$ ist gesetzt, wenn die Controller-Karte eine Timeout-Bedingung erkannt hat. Es gibt mehrere Bedingungen, die einen Timeout auslösen können: Wenn der Trigger mehr als 17 μs für die Clusterzählung benötigt, angezeigt durch ein gesetztes Busy-Signal und ein fehlendes Data-Valid-Signal, liegt eine Timeoutbedingung ebenso vor, wie bei einer Gesamtbearbeitungszeit für ein Hit-Pattern von mehr als 34 μs. Eine dritte Bedingung, die ein Timeout auslöst, ist ein gesetztes Busy-Bit während der Sequencer nicht läuft. Allen drei Bedingungen ist gemein, dass ihr Auftreten darauf hindeutet, dass sich der Sequencer nicht mehr im korrekten Betriebszustand befindet.
Das $Running$-Bit zeigt an, dass der Sequencer läuft. $Full$ und $Empty$ beziehen sich auf den Füllzustand des FIFO-Speichers. $Busy$ ist ein Statussignal, das von der Software des Sequencers gesetzt werden kann und anzeigt, wenn die Clustererkennung läuft.
Das Latch-Control-Register ist direkt mit drei Ausgängen verbunden, die den Testmustergenerator der Latchmodule steuern. Die folgende Tabelle zeigt den Aufbau dieses Registers.

Bit	7	6	5	4	3	2	1	0
						DataIn	TestClk	TestEn

Mit $TestEn$ werden die Latchmodule in den Testmodus umgeschaltet, d.h. die Daten des Testmustergenerators werden auf die Ausgänge geschaltet. Mit einem Impuls an $TestClk$ wird das Datenbit, das an $DataIn$ anliegt, in das Schieberegister des Testmustergenerators übernommen. Dabei ist wieder darauf zu achten, dass das TestClk-Bit von der Software nach dem Aktivieren wieder deaktiviert wird. Alle anderen Bits dieses Registers werden derzeit noch nicht verwendet und stehen noch für spätere Erweiterungen zur Verfügung.
Im Startadressenregister wird die Adresse abgespeichert, ab der der Sequencer beim nächsten Start beginnen soll. Eine Veränderung dieses Registers bei laufendem Sequencer hat keinen Einfluss auf dessen Betrieb. Im Register Zelladresse kann eine Adresse einer Zelle der Logikmatrix abgespeichert werden. Die Software des Sequencers hat dann die Möglichkeit, diese Adresse auf den Zelladressbus zu geben, um damit eine von außen vorgegebene Zelle selektiv anzusprechen. Dies kann z.B. benutzt werden, um von der VME-CPU gesteuert Testmuster in der Zellularlogik zu setzen.
Der lesende Zugriff auf Zelladressen, die bei der Clusteridentifikation erzeugt und in den FIFO-

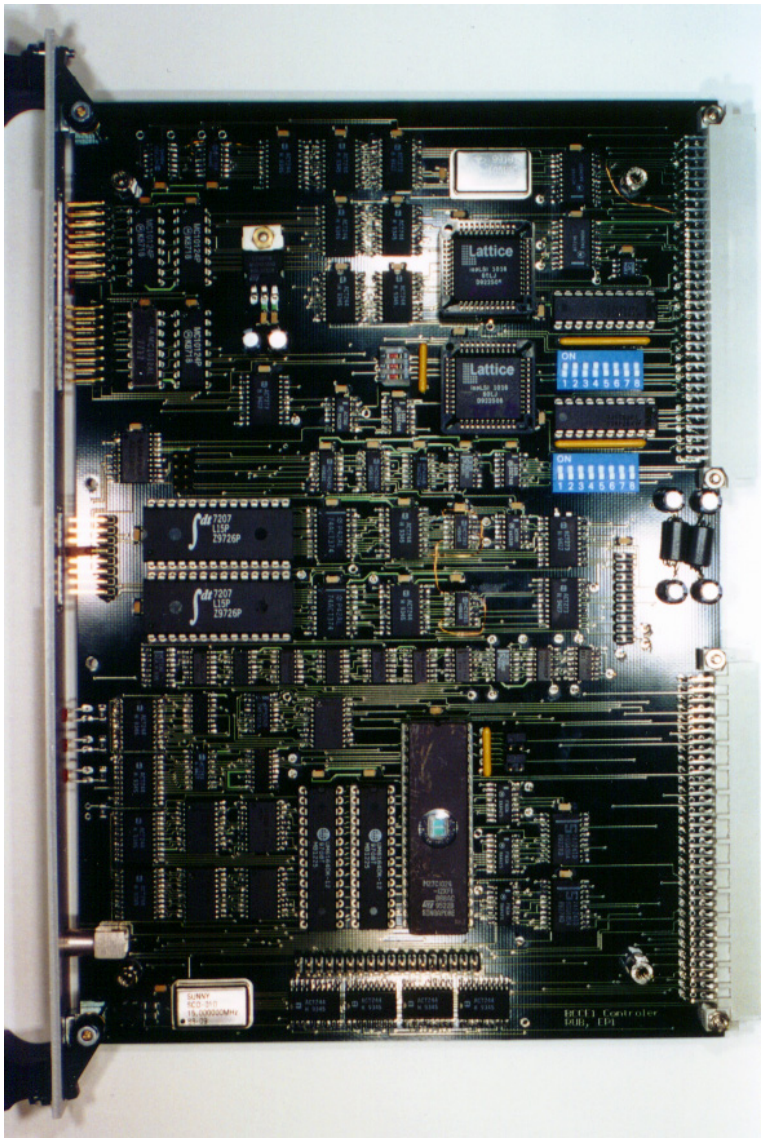


Abbildung 7.9: Der Aufbau der Controller-Platine. Unten befindet sich der Sequencer, wo deutlich das EPROM und daneben das Befehls-RAM zu erkennen ist. In der Mitte links sind die beiden FIFO-Speicherbausteine zu sehen, oben rechts ist das VME-Interface angeordnet. Deutlich sind die DIP-Schalter zur Einstellung der Basis-Adresse zu erkennen

Speicher geschrieben werden, erfolgt über das FIFO-Leseregister. Ein Lesezugriff auf diese Adresse ergibt die erste zur Verfügung stehende Zelladresse, die im FIFO gespeichert wurde. Mit dem Zugriff wird diese Adresse aus dem FIFO gelöscht und es liegt die nächste Adresse im FIFO an. Somit müssen die Daten, falls sie von der Software weiterverwendet werden sollen, im Speicher der CPU gesichert werden.
Die Zählerstände der beiden Clusterzähler können an den Adressen 10 und 11 als Byte-Werte abgefragt werden. Es ist selbstverständlich auch möglich, beide Zählerstände gemeinsam als Wort aus dem Controllermodul auszulesen. Das letzte Register ist der Zellzähler. In diesem Zähler wird während der Clusteridentifikation mitgezählt, wie viele Zelladressen in den FIFO-Speicher geschrieben wurden. Somit ist es möglich, den FIFO im Block auszulesen, ohne nach jedem Lesezugriff auf den FIFO das $Empty$-Bit überprüfen zu müssen.
Der gesamte Controller wurde auf einer 6-HE-VME-Platine in Multilayertechnik mit vier Ebenen aufgebaut. Die Platine ist, von wenigen ICs abgesehen, in SMD-Technik bestückt. Die Herstellung der Platine übernahm wieder die Firma Beta-Layout, die Bestückung erfolgte im Labor

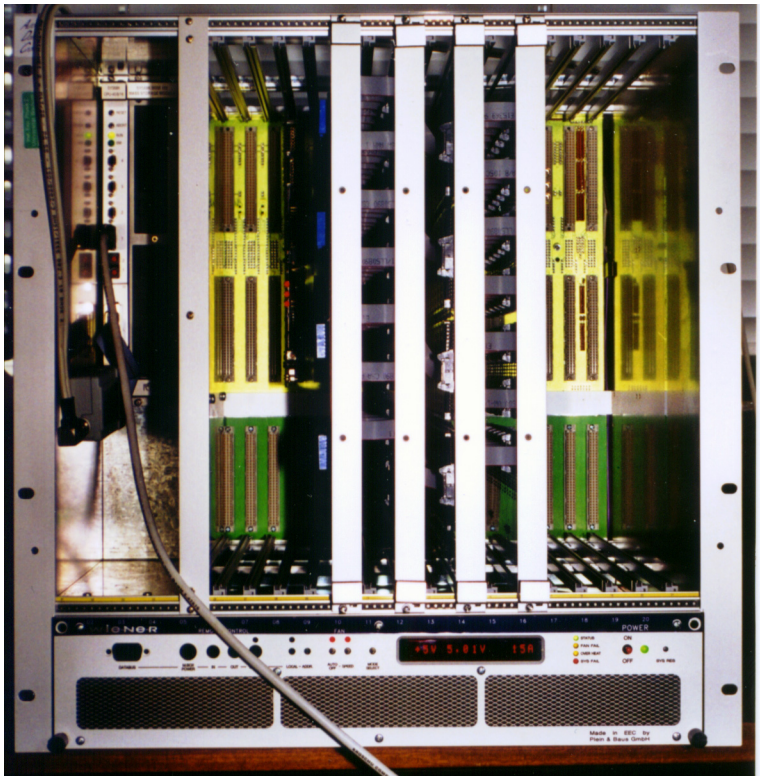


Abbildung 7.10: Das Triggercrate beim Labortest in Bochum mit den beiden Haupttriggerplatinen in der Mitte, umgeben von den Signalzuführungsplatinen. Zwischen Trigger- und Signalzuführungsplatine sind die Flachbandkabel zu sehen. Ganz links befindet sich die VME-CPU mit den Zuleitungen für Netzwerk und Terminal

von Hand. Abbildung 7.9 zeigt das Controller-Board. Auf dem Bild sind einige Konfigurationsmöglichkeiten, so z.B. zur Einstellung der VME-Basisadresse u.a. zu sehen. Diese Einstellungen werden in einer detaillierten Beschreibung der Controllerschaltung in Anhang D erläutert.

7.5 Die ersten Tests des Triggers

Nachdem der Aufbau der Einzelkomponenten abgeschlossen war, wurde der gesamte Trigger, bestehend aus den beiden Haupttriggerplatinen, den beiden Signalzuführungsplatinen und dem Controller-Modul, in das VNX-9-Crate eingebaut. Zur Ansteuerung des Controllers über den VMEbus diente eine Force-040-CPU. Der gesamte Aufbau ist in Abbildung 7.10 zu sehen. Der Trigger ließ sich bereits beim ersten Einschalten ohne größere Probleme in Betrieb nehmen. Diese Konfiguration des Triggers erlaubt bereits einen vollständigen Test mit Testmustern, die von der VME-CPU erzeugt und über den Controller Zelle für Zelle in die Logikmatrix geschrieben werden. Dazu wurden solche Testmuster ausgewählt, die es gestatten, sämtliche Verbindungen zwischen den Zellen selektiv zu testen. Insgesamt fiel die Wahl auf 10 Muster, die alle Horizontalen, Vertikalen und Diagonalen enthalten, sowie zwei weitere Muster, nämlich eine komplett gefüllte Matrix sowie eine Kette maximaler Länge. Diese Testmuster werden von der CPU in die Logikmatrix geladen und dann autonom vom Trigger analysiert. Das Ergebnis dieser Analyse wird wiederum von der CPU ausgelesen und mit einem von der Software erzeugten Vergleichsdatensatz verglichen. Dieser Test wurde mit jedem der 12 beschriebenen Testmuster 100000 mal durchgeführt, ohne dass eine Fehlfunktion beobachtet wurde.
Darüberhinaus wurden mit einem Zufallsalgorithmus Muster generiert, die eine Clusterstruktur besitzen und somit eine große Ähnlichkeit mit den später im Triggerbetrieb zu erwartenden Mu-

stern aufweisen. Abbildung 7.11 zeigt ein typisches Muster, das von diesem Generator erzeugt wurde. Der Trigger wurde mit mehr als einhunderttausend dieser Muster getestet. Auch dabei trat keine Fehlfunktion auf.

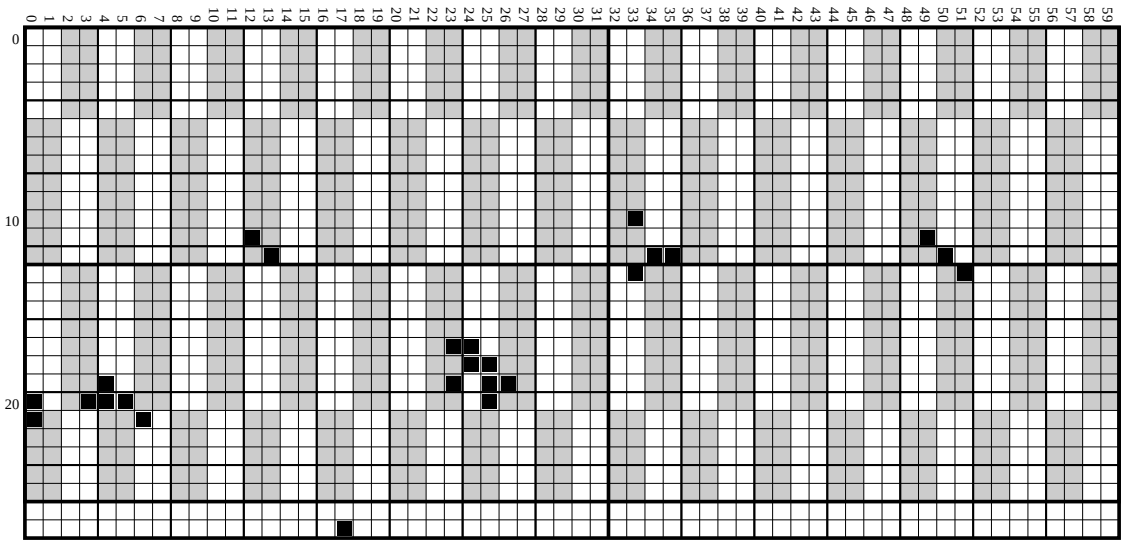


Abbildung 7.11: Typisches Zufallsmuster zum Test der Zellularlogik

Anschließend wurden mit Hilfe eines Logicanalyzers[1] die Zeitabläufe im Trigger genau untersucht. Abbildung 7.12 zeigt einen Plot, der mit dem Logicanalyzer bei der Analyse eines Zufallstestmusters mit vier Clustern aufgenommen wurde. Deutlich sind vier sich wiederholende Signalfolgen zu erkennen. Der Plot in Abbildung 7.13 zeigt eine dieser Sequenzen mit einer schnelleren Zeitbasis. In diesem Bild ist der Ablauf der Clusteranalyse deutlich zu erkennen. Sie wird mit einem kurzen Puls auf der $ResM$-Leitung (Cntl 01) begonnen, um ggf. vorher gesetzte Markierungsflipflops zu löschen. Gleichzeitig wird die $XProj$-Leitung (Cntl 06) aktiviert. Mit nur wenigen Nanosekunden Verzögerung reagiert die Zellularlogik darauf, indem die $XEMPTY$-Leitung in den Low-Zustand übergeht. Dem Sequencer wird damit angezeigt, dass es noch gesetzte Zellen gibt, die bei der X-Projektion vom Prioritätsencoder erkannt wurden. Fast zeitgleich stellt sich eine neue Adresse auf dem Zelladressbus ein. Nach einer Wartezeit von etwa 200 ns, die dazu dient, einen stabilen Zustand nach der X-Projektion zu erreichen, wird die $YProj$-Leitung (Cntl 07) aktiviert. Die Zellularlogik reagiert darauf, indem $YEmpty$ auf Low-Pegel wechselt. Zudem ändert sich erneut die Adresse auf dem Zelladressbus. Nach etwas mehr als 100 ns wird das $Mark$-Signal (Cntl 03) aktiviert und dadurch die Markierung aller Zellen des ausgewählten Clusters eingeleitet. Im nächsten Schritt werden diese mit dem $Mask$-Signal (Cntl 04) maskiert. Gleichzeitig werden $XProj$ und $YProj$ wieder deaktivert, was dazu führt, dass $XEmpty$ und $YEmpty$ in den High-Zustand wechseln, während alle Leitungen des Zelladressbusses in den Low-Zustand übergehen. Die Zellularlogik ist nun für die Identifikation des nächsten Clusters bereit, die damit beginnt, dass der am Rande des Plots in Abbildung 7.13 noch sichtbare $ResM$-Puls erscheint.

[1] Logicanalyzer Typ HP 1652B von Hewlett Packard

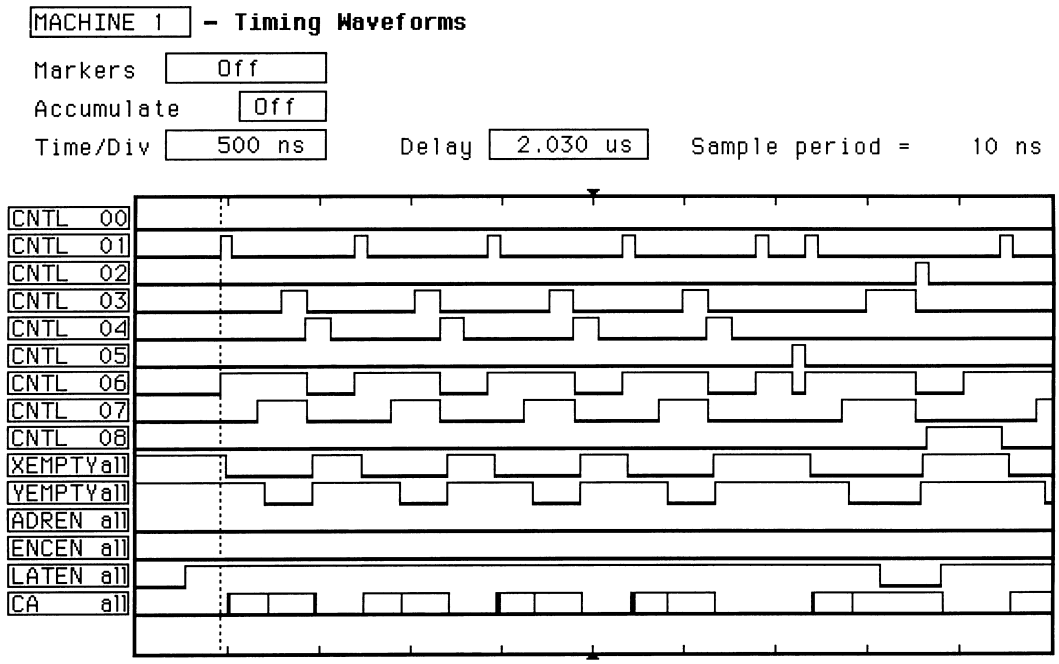


Abbildung 7.12: Messplot 1 des Logicanalyzers. Die mit CNTL00 bis CNTL08 bezeichneten Signale sind in folgender Reihenfolge die Steuersignale $Reset$, $ResM$, $ResT$, $Mark$, $Mask$, $DeMask$, $XProj$, $YProj$ und $ProjSel$

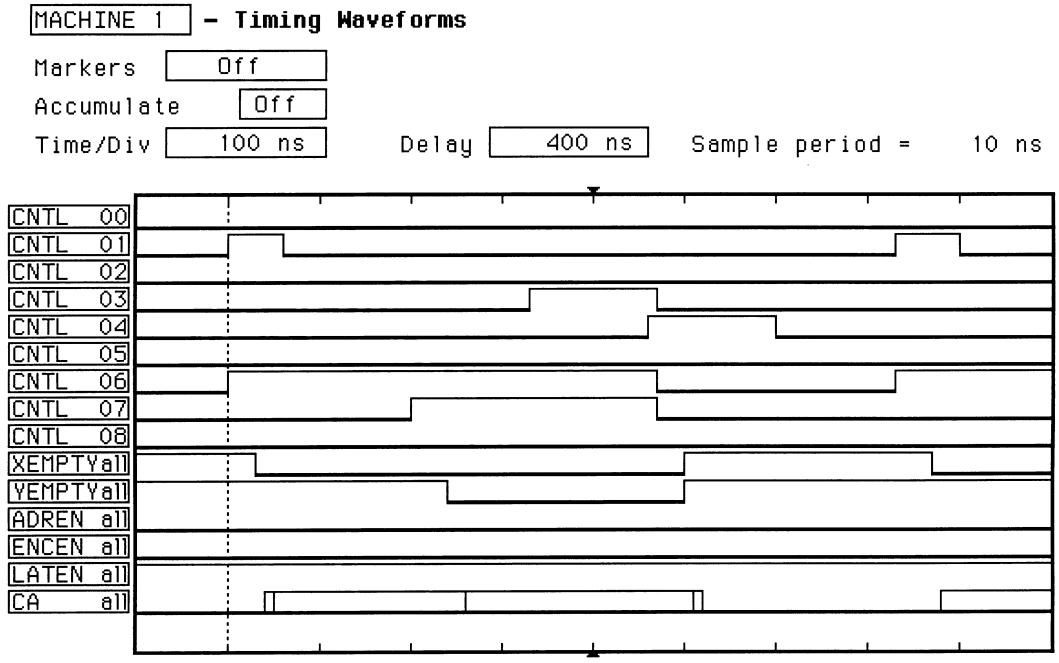


Abbildung 7.13: Ausschnitt aus dem Logicanalyzer-Messplott 1 mit einer schnelleren Zeitbasis von 100 ns/Div.

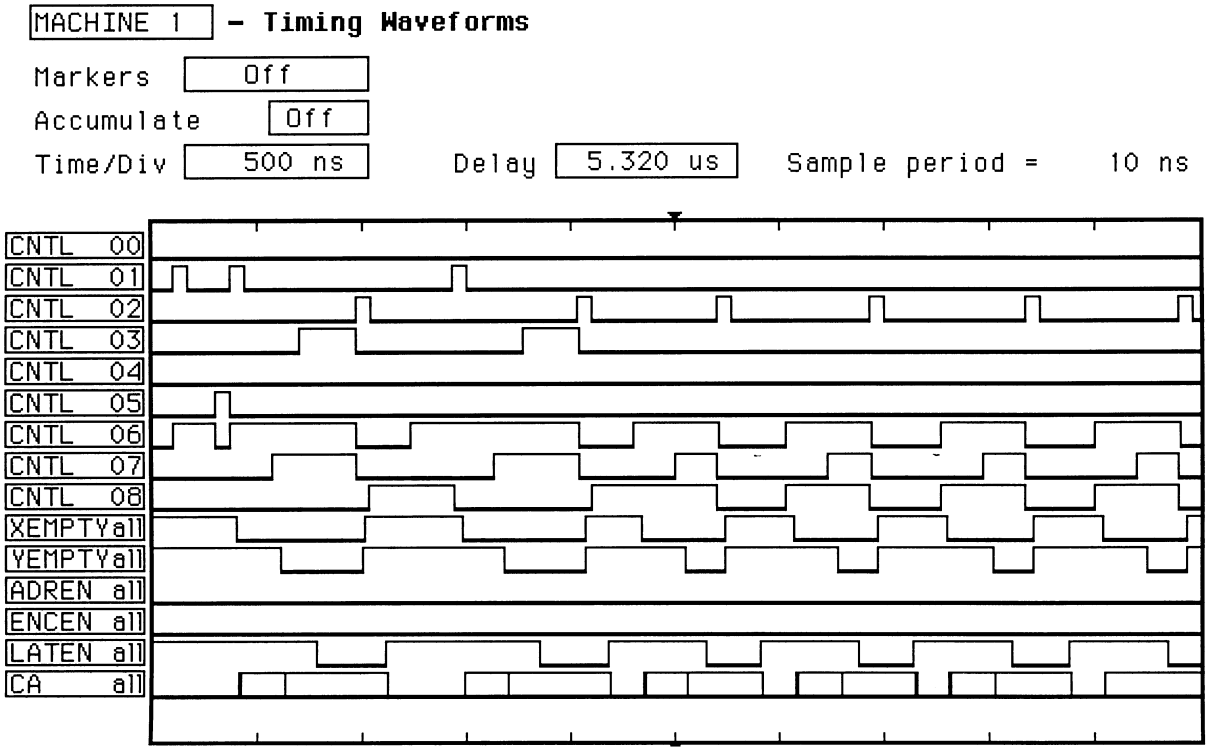


Abbildung 7.14: Messplot 3 des Logicanalyzers. Dargestellt ist die Auslese der zum ersten Cluster gehörenden Zellen

In der Abbildung 7.14 ist der Signalverlauf bei der Auslese der Zellen eines Clusters, die sich nach der Clusterzählung anschließt, zu sehen. Die Auslese beginnt damit, dass mit einem $DeMask$-Signal (Cntl 05) die Maskierung der zuvor identifizierten Cluster aufgehoben wird. Danach wird zunächst wieder ein Cluster markiert. Dieser Zyklus, der in Abbildung 7.15 dargestellt ist, beginnt damit, dass zur Initialisierung zunächst alle noch aus vorherigen Arbeitsschritten gesetzten Markierungsflipflops gelöscht werden. Anschließend wird wieder mittels Projektion eine Zelle ausgewählt und mit dieser Zelle als Ausgangspunkt ein Cluster markiert. Abweichend vom Clusterzählalgorithmus wird jetzt jedoch das $LatchEn$-Signal auf Low gesetzt, womit die anliegende Zelladresse im Adresslatch gespeichert wird. Die Projektionssignale und das Markierungssignal können jetzt deaktiviert werden. Mit einem $ResT$-Signal (Cntl 02) wird selektiv in der ausgewählten Zelle das Trefferflipflop gelöscht. Gleichzeitig wird der Controller die im Latch gespeicherte Zelladresse in seinen FIFO-Speicher schreiben.
Auf die gleiche Weise werden nun die Adressen aller zum gleichen Cluster gehörenden Zellen nacheinander in absteigender Prioritätsfolge identifiziert und in den FIFO-Speicher übertragen. Dazu muss am Ende jedes einzelnen Auslesevorgangs das entsprechende Trefferflipflop zurückgesetzt werden.

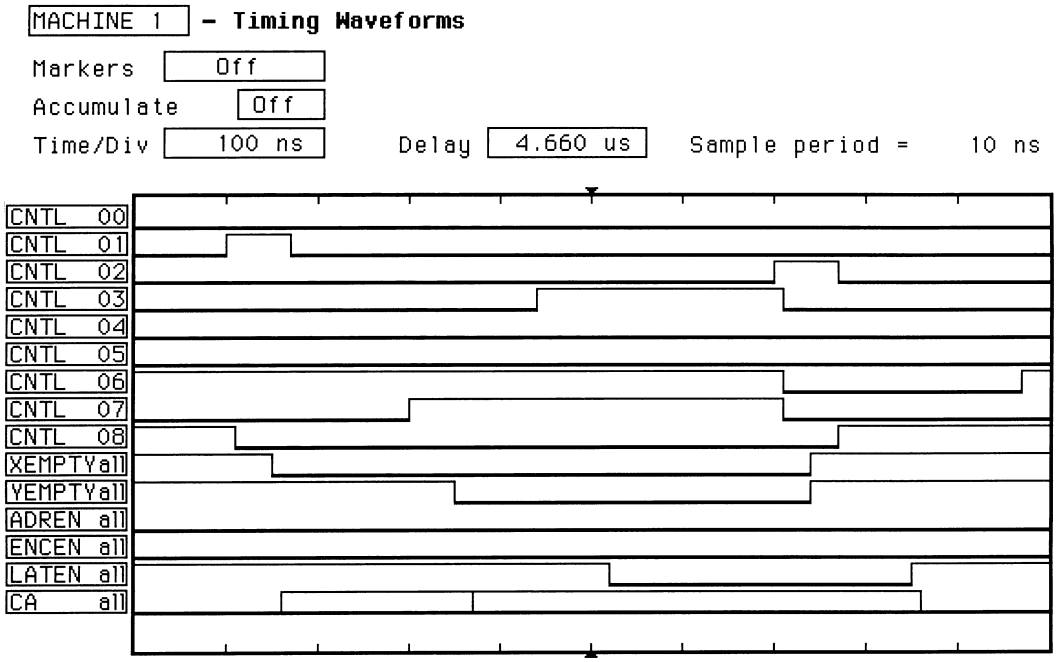


Abbildung 7.15: Messplot 4 des Logicanalyzers. Hier ist der Ausschnitt aus Messplot 3 zu sehen, in dem die Auslese der ersten Zelle des Clusters dargestellt wird

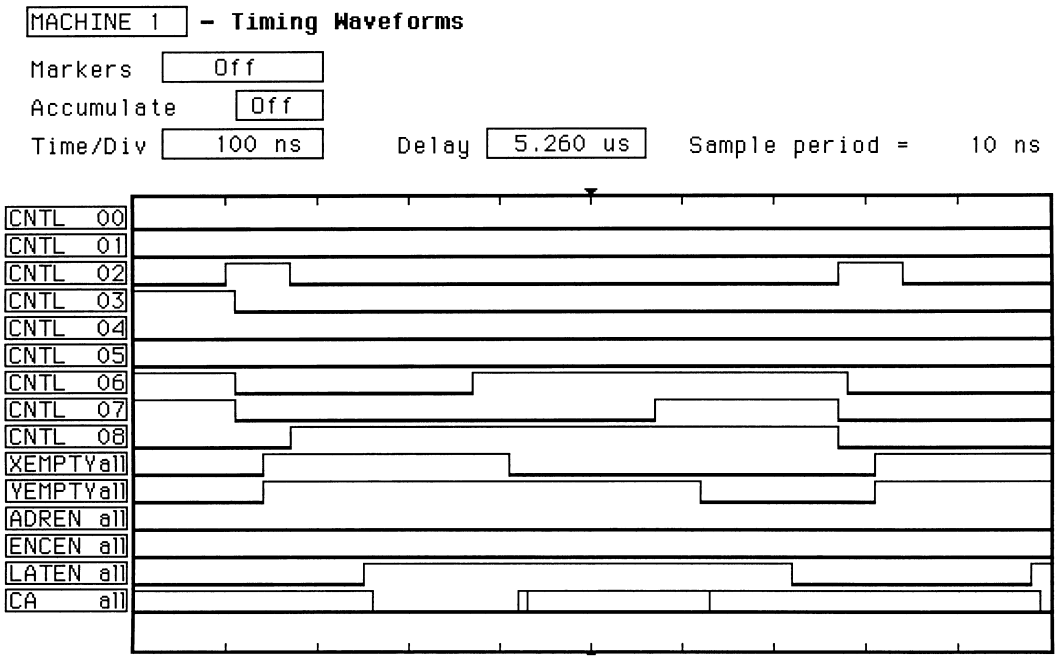


Abbildung 7.16: Messplot 5 des Logicanalyzers. Zu sehen ist ein Ausschnitt aus Messplot 3, in dem eine weitere Zelle ausgelesen wird

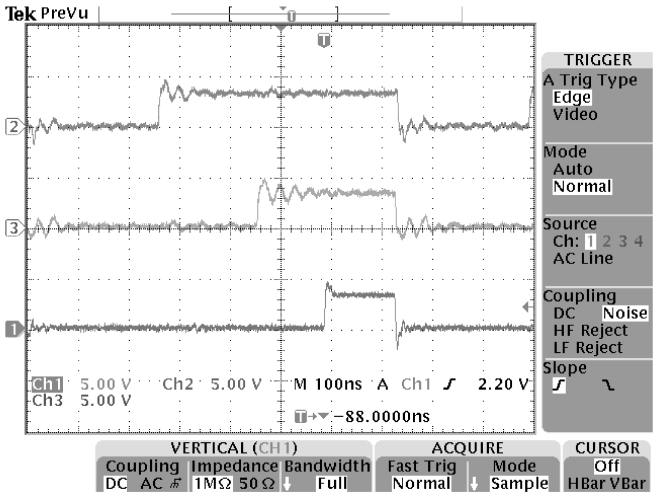


Abbildung 7.17: Signalverlauf auf der $XProj$, $YProj$ und der $Mark$-Leitung bei Clusteridentifikation

Um die Güte der Signalübertragung auf der Triggerplatine zu überprüfen, wurden die Steuersignale an den Abschlusswiderständen mit einem Oszilloskop[2] untersucht. Abbildung 7.17 zeigt ein Oszillogramm, das den Verlauf der beiden Projektions- und des Markierungssignals darstellt. Das Oszillogramm zeigt den sauberen Verlauf der Digitalsignale. Die Anstiegs- und Abfallzeiten betragen weniger als 2 ns, ohne dass übermäßiges Überschwingen zu beobachten wäre. Zwischen den Signalen ist kein Übersprechen zu erkennen.

Um die Geschwindigkeit der Zellularlogik messen zu können, wurde die Zeit, die die Zellularlogik für die Projektionen benötigt, mit einem Oszilloskop dargestellt. Dazu wurde das Oszilloskop auf dem $XProj$- bzw. $YProj$-Signal getriggert und auf dem zweiten Kanal eine der Zelladressleitungen dargestellt, die durch die Projektion von Low auf High übergeht.
Aus den beiden Oszillogrammen in Abbildung 7.18 sind die Verzögerungszeiten von 46,4 ns für die Projektion auf die x-Achse und 55,2 ns für die Projektion auf die y-Achse zu entnehmen.

[2]Tektronix Modell TDS3054

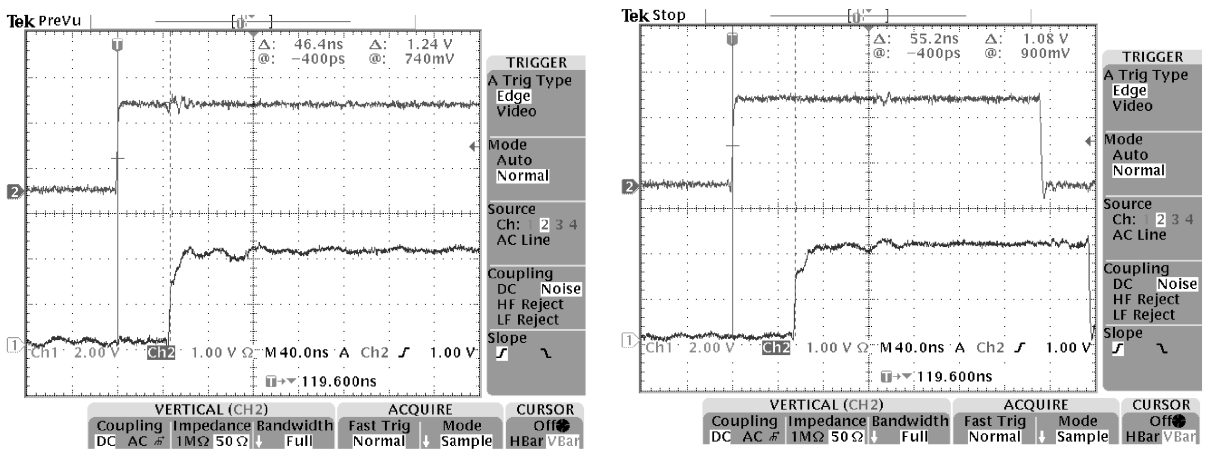


Abbildung 7.18: Reaktionen der Zelladressbus-Leitungen auf X- bzw. YProjektionssignale

Kapitel 8

Einbau des Triggers und erste experimentelle Ergebnisse

Nach Abschluss der Aufbau- und Testphase in Bochum wurde der Trigger im Frühjahr 2000 nach Bonn an die Elektronenbeschleunigeranlage ELSA gebracht und in das CB-Experiment integriert. Der Einbau des Kalorimetertriggers unterteilte sich in drei Abschnitte:

1. Seine Verbindung mit der übrigen Experimentelektronik
2. Seine Einbindung in den globalen Trigger des Gesamtexperimentes
3. Seine Einbindung in die Experimentsteuerung und -auslese

Angesichts der großen Anzahl von Signalen, die auf den Triggerplatinen zusammengeführt werden müssen, erwies sich die Verkabelung des Triggers als eine komplexe Aufgabe, die umfangreiche Vorüberlegungen erforderte. Die Zuordnungsschemata, die sich aus diesen Überlegungen ergaben, sind in Anhang E wiedergegeben.
Nach dem Einbau des Triggers wurde seine Funktionsfähigkeit als Komponente des CB-ELSA-Experimentes in mehreren Tests demonstriert. Dabei wurden die Funktion sämtlicher 1380 Kanäle, über die dem Trigger die benötigte Information zugeführt wird, die korrekte Kristallzuordnung und die richtige Übernahme der Clusteridentifikationsergebnisse in den CB-ELSA-Trigger überprüft.

8.1 Die Ausleseelektronik des Crystal-Barrel-Kalorimeters

Das erste Glied in der Ausleseelektronik des CB-Kalorimeters sind die in den Kristallmodulen integrierten Vorverstärker. Ihre Ausgangssignale gelangen über 60 m lange Kabel auf die Receiver/Shaper-Module, die sich zusammen mit der übrigen Experimentelektronik in der Phoenics-Halle der Bonner Beschleunigeranlage befinden. Als Kabel wurden *Singel-Shielded-Twisted-Pair*-Kabel ausgewählt, bei denen jeweils 8 Paare zu einem Rundkabel gebündelt sind. Jedes der verwendeten Receiver/Shaper-Module, die für das alte Crystal-Barrel-Experiment am LEAR von der Universität Zürich gebaut wurden, enthält 8 unabhängige Kanäle, sodass die Ausgänge der Kristallmodule jedes Doppelsektors auf drei Receiver/Shapermodule aufgeteilt werden müssen.

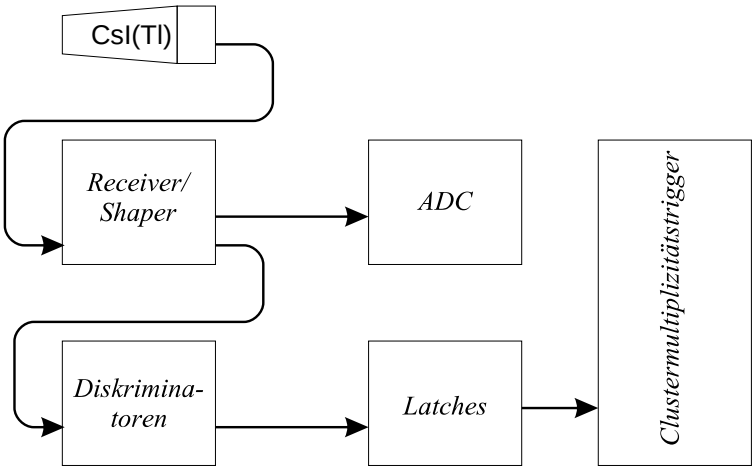


Abbildung 8.1: Schematische Darstellung des Signalverlaufs der Kalorimeterauslese

Für den gesamten Barrel werden 180 dieser Module benötigt. Die Shaper formen die Barrelsignale so, dass an ihren Ausgängen Signale mit einer Impulsdauer von ca 6 μs und einer zum Energiedeposit proportionalen Fläche zur Verfügung stehen [Eh00]. Die Shaper besitzen drei Ausgänge, von denen beim CB-ELSA-Experiment nur noch zwei benutzt werden. Die Signale des einen Ausgangs werden zu den ADCs weitergeleitet, die des anderen zu den weiter unten beschriebenen Diskriminatoren.
Bei den Analog-Digital-Konvertern handelt es sich um Fastbus-ADCs vom Typ 1885F von LeCroy. Jedes ADC-Modul enthält 96 Kanäle, sodass insgesamt 15 ADC-Module notwendig sind. Die 96 Signale werden aber nicht parallel, sondern nacheinander in einem ADC konvertiert. Daher ist die Konversionszeit relativ groß und beträgt 265 μs. Der ADC enthält einen Low- und einen High-Range-Kanal, wodurch der ADC bei einer Auflösung von 12 bit eine Dynamik von 15 bit erreicht.
Die ADC-Module werden mit einem $Gate$-Signal getriggert, die Konversion aber erst nach einer einstellbaren Verzögerungszeit von bis zu 300 μs gestartet. Während dieser Zeit, die bei CB-ELSA auf 10 μs eingestellt ist, kann die Konversion jederzeit mit einem $FastClear$-Signal abgebrochen werden. Dies ermöglicht es, mit einem 2nd-Level-Trigger unter Benutzung der langsamen Barrelsignale bei unerwünschten Ereignissen ein *FastClear*-Signal zu erzeugen und somit durch die Vermeidung eines unnötigen Auslesezyklusses die Totzeit zu minimieren.
Die Signale aus den zweiten Ausgängen der Receiver/Shaper-Module werden über einzelne Koaxialkabel auf die Diskriminatoren geführt. Diese ebenfalls von der Universität Zürich hergestellten Diskriminatoren enthalten wieder 8 Kanäle pro Modul. Wenn das Eingangssignal eines Diskriminators eine definierte Schwelle überschreitet, die von außen über einen separaten Eingang für alle Diskriminatoren gemeinsam vorgegeben wird, erzeugt er auf dem entsprechenden Differential-ECL-Ausgang einen kurzen Impuls. Diese Ausgangssignale werden über *Twisted-Pair*-Flachbandkabel an die in der Signalkette folgenden Latchmodule weitergeleitet.

8.1.1 Die Latchmodule

Diese Module, von denen jedes einen Doppelsektor einer Barrelhälfte abdeckt, sodass für den gesamten Barrel 60 Module benötigt werden, wurden im Rahmen dieser Arbeit neu entwickelt. Sie haben im Wesentlichen drei Aufgaben:

1. Die Differential-ECL-Signale, die von den Diskriminatoren geliefert werden, müssen auf die von der folgenden Triggerelektronik benötigten TTL-Pegel umgesetzt werden.

2. Die koinzident mit dem $Gate$-Signal eintreffenden ECL-Signale werden gespeichert, um die Übertragung kurzer TTL-Impulse auf den folgenden, mehrere Meter langen Flachbandkabeln zu vermeiden.

3. Die Diskriminatorsignale der Kristalltypen 11, 12 und 13 werden in den Latches auf jeweils zwei Ausgänge aufgeteilt. Diese bisher unerwähnte Zuordnung von einem Kristall auf zwei Logikzellen ist notwendig, um im Randbereich des Barrels, wo von den Kristallen ein Azimutwinkel von 12° statt von 6° abgedeckt wird, die korrekten Nachbarschaftsbeziehungen einzuhalten.

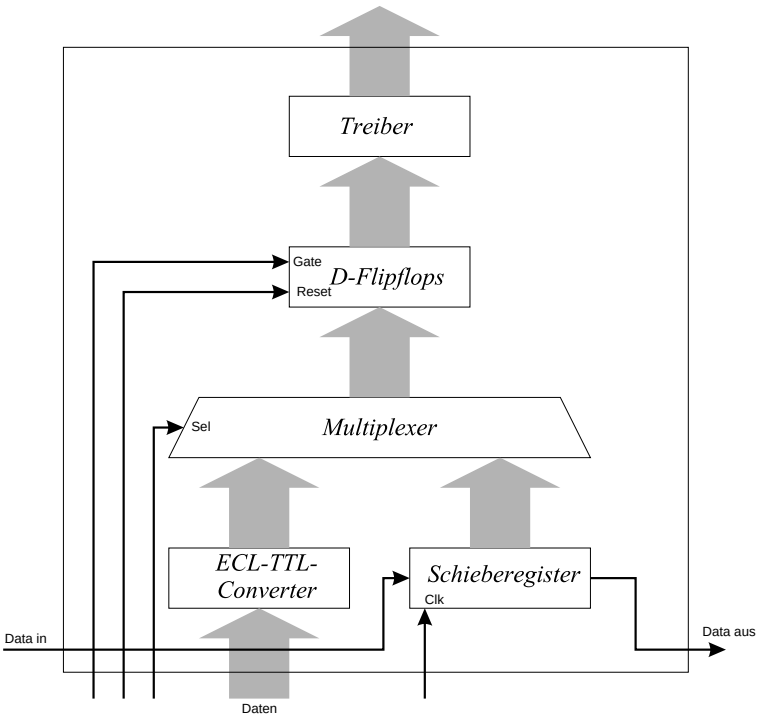


Abbildung 8.2: Blockschaltbild der für CB-ELSA neu konstruierten Latchmodule

Weiterhin können die Latches dazu benutzt werden, Testmuster für die nachfolgende Clusteridentifikation zu erzeugen. Damit kann der gesamte Signalweg von den Speichern in den Latches bis zur Zellularlogik überprüft werden.
Die Abbildung 8.2 zeigt schematisch den Aufbau der Latchmodule anhand eines Blockschaltbildes. Die Daten gelangen von den Eingängen über ECL-TTL-Pegelwandler zu einem Multiplexer. Mit diesem Multiplexer wird die Auswahl getroffen, ob Experimentdaten zum Multiplizitätstrigger geleitet werden oder Testdaten, die aus einem Schieberegister stammen.
Die ausgewählten Daten werden synchron mit einem $Gate$-Signal in D-Flipflops gespeichert und über eine Treiberstufe auf die Ausgänge gegeben. Die Trefferdaten, die als statische TTL-Signale an den Ausgängen anliegen, können wahlweise über zwei 40-polige Fine-Pitch-Pfostenstecker auf der Frontseite oder über eine 64-polige VG-Steckerleiste auf der Rückseite des Latchmoduls ausgelesen werden. Eine ausführliche Beschreibung der Schaltung und der Funktionsweise der

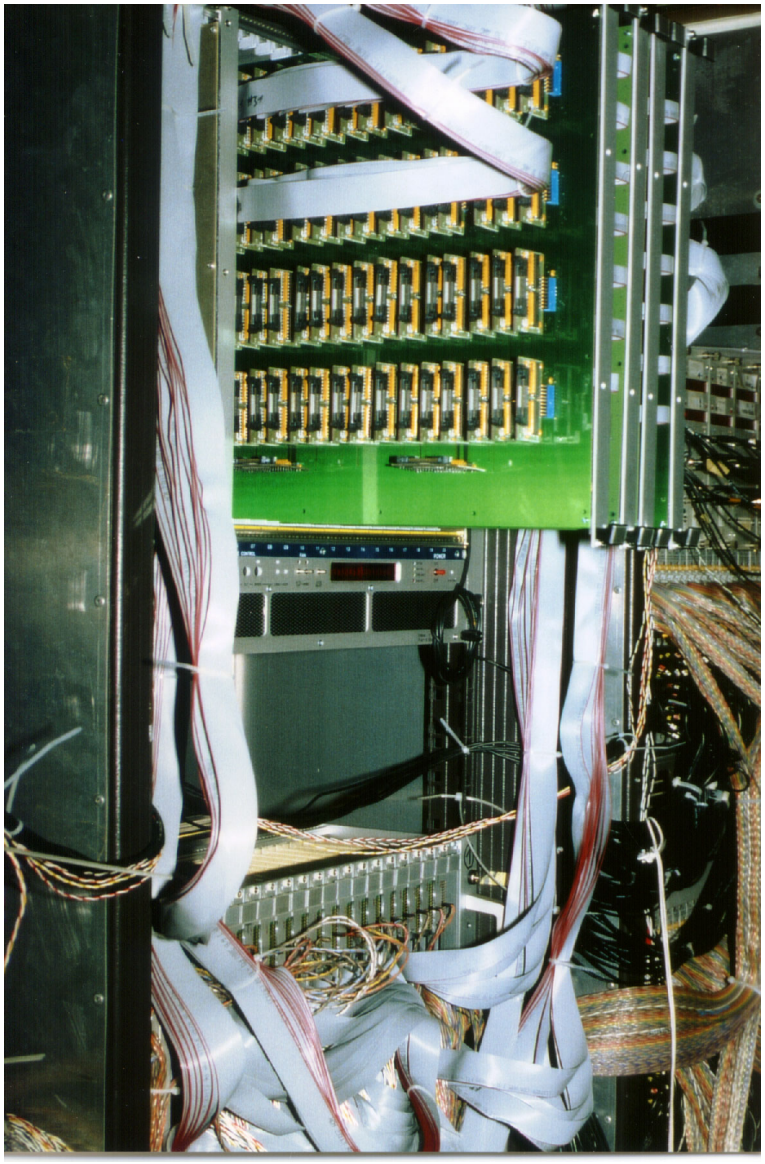


Abbildung 8.3: Das Triggercrate in der Elektronikhalle am Bonner Elektronenbeschleuniger ELSA bei der Verkabelung

Latches findet sich im Anhang A.

Die drei VME-Crates, die die Latchmodule beherbergen, wurden zusammen mit dem VNX-9-Crate, das die Triggerelektronik enthält, in einem Rack eingebaut. Abbildung 8.3 zeigt oben das Triggercrate und darunter eines der Latchcrates während der Verkabelung. Die Platinen sind weit aus dem Crate herausgezogen, um den Zugang zu den Steckplatinen zu ermöglichen.
Um die Testmustergeneratoren der Latchmodule anzusteuern, steckt in jedem der drei Latchcrates in Slot 21 eine einfache Interfacekarte, über die die drei Steuersignale $TestEn$, $TestClk$ und $DataIn$ auf die Backplane des Crates gelangen. Diese Interfacekarten tragen auf der Frontseite einen Stecker, über den die aus dem Triggercontroller kommenden Steuersignale mit einem zusätzlichen Flachbandkabel eingespeist werden.
Nach einigen Anfangsschwierigkeiten, die sich durch zusätzliche Abblockungs- und Potential-

ausgleichsmaßnahmen leicht beheben ließen, konnte der Trigger in Betrieb genommen und mit Testdaten aus den Testmustergeneratoren der Latchmodule getestet werden.

8.2 Einbindung in DAQ und Trigger des Gesamtexperimentes

Der nächste Schritt bestand darin, den Clustertrigger in den globalen Trigger des Gesamtexperimentes sowie in die DAQ einzubinden.

8.2.1 Einbindung in den Trigger

Im globalen Trigger des Crystal-Barrel-Experimentes müssen Signale von allen Teildetektoren zusammengeführt werden. Ein Problem dabei ist, dass die für den Trigger nutzbaren Signale der verschiedenen Teildetektoren erst zu sehr unterschiedlichen Zeitpunkten zur Verfügung stehen. Dementsprechend ist der Trigger in zwei Ebenen aufgebaut, wie aus der schematischen Darstellung in Abbildung 8.4 ersichtlich ist. Jede dieser Ebenen wurde unter Verwendung von PLUs (Programable Logic Unit) vom Typ LeCroy, LRS4508 realisiert. Diese Einheiten enthalten einen schnellen Speicher mit einer Kapazität von 256×8 bit, der über die acht Eingangssignale adressiert wird. Auf diese Weise lassen sich mit jeder dieser Einheiten acht unabhängige logische Funktionen mit acht Eingangsvariablen realisieren.
In der ersten Stufe werden die schnellen Triggersignale aus Innendetektor und Flugzeitwänden, die in vorgeschalteten Einheiten zu je zwei Triggersignalen zusammengefasst wurden, zusammen mit dem Tagger-Oder und dem γ-Veto-Signal auf eine PLU gegeben. Diese PLU arbeitet freilaufend, wobei ein Triggerereignis erkannt wird, wenn die entsprechenden Signale einen Überlapp von mehr als 5 ns aufweisen. In einer zweiten, parallel arbeitenden PLU werden weitere Signale, die als Testtrigger dienen, erfasst. Aus jeder der beiden PLUs werden 3 Signale an das Triggerregister weitergeleitet.
Das Triggerregister dient dazu, den Zustand dieser 6 Signale zu speichern. Dazu wird im Triggerregister mit einer ODER-Verknüpfung von zwei Eingangssignalen, die von der PLU3 bzw. PLU4 kommen, ein Strobe-Signal erzeugt. Das im Triggerregister erzeugte *Strobe*-Signal wird als *OrOut* nach außen geführt und dient als *Gate*-Signal für alle folgenden Stufen des Triggers und für die Experimentauslese.
Die acht Ausgangssignale des Triggerregisters werden einer weiteren PLU zugeführt. Sie arbeitet nicht mehr im kontinuierlichen Betrieb, sondern wird vom oben erwähnten *Gate*-Signal getriggert. Diese PLU dient dazu, die acht Signale an den Ausgängen des Triggerregisters in eine 3-Bit-Information umzucodieren. Diese Information wird der nächsten PLU zugeführt, die die Eingangsstufe der zweiten Triggerebene bildet.
Diese PLU ist auch über zwei Leitungen mit dem Clustertrigger verbunden. Um dem Clustertrigger die Trefferinformationen zuzuführen, werden alle Barrelsignale, die in einem Zeitfenster von $3\,\mu$s nach dem *Gate*-Signal eintreffen, in den Latchmodulen gespeichert. Nach Ablauf dieser Zeit wird die Trefferinformation in die Trefferflipflops des Clustertriggers übernommen und die Clusteridentifikation gestartet. Wenn alle Cluster identifiziert sind, meldet der Clustertrigger dies der Triggerlogik mit einem $DataValid$-Signal, das als Strobe für eine eigens dem Clustertrigger zugeordnete PLU (PLU8) dient, in der die acht Ausgangsbits des Clusterzählers gespeichert werden. Durch die zwei Ausgangssignale der Stufe kann beispielsweise angezeigt werden, ob

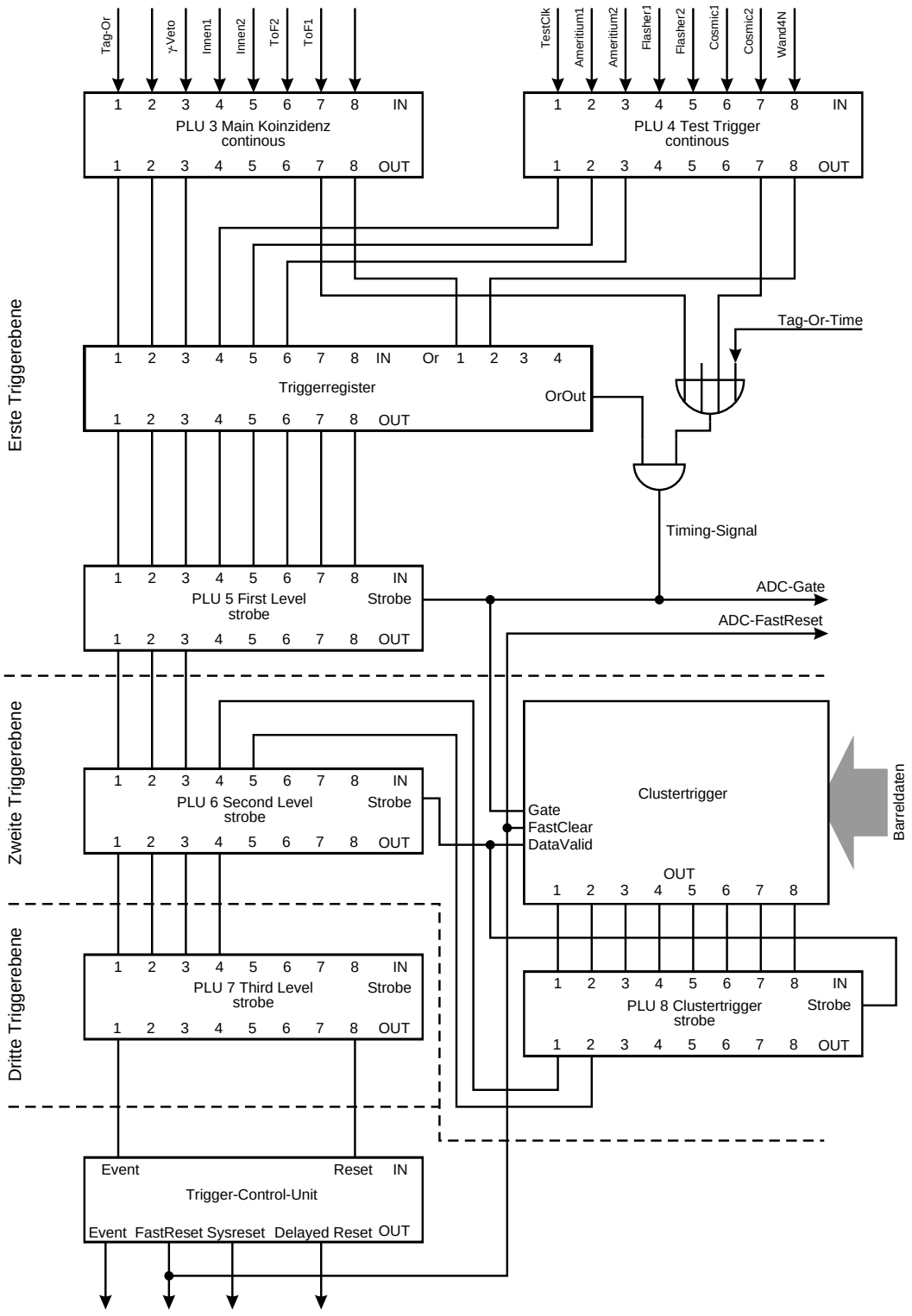


Abbildung 8.4: Der Zentraltrigger des CB-ELSA-Experimentes nach [Eh00]

die Anzahl der identifizierten Cluster größer als eine vorgegebene Mindestzahl und kleiner als eine Maximalzahl ist.
Mit der PLU7 können im Prinzip weitere Signale in einer dritten Triggerebene verarbeitet werden. Derzeit wird davon jedoch kein Gebrauch gemacht. Mit dieser PLU wird die Entscheidung getroffen, ob ein Ereignis aufgezeichnet werden soll oder nicht. Wenn das Ereignis verworfen wird, müssen alle ADCs und TDCs, die bereits durch die erste Triggerebene das $Gate$-Signal erhalten haben, mit einem $FastReset$-Signal gestoppt werden. Auch der Clustertrigger erhält dieses $FastReset$-Signal, um die Auslese der gesetzten Zellen zu stoppen und wieder in die Bereitschaftsphase überzugehen.
Durch die Verwendung programmierbarer Look-Up-Table gewinnt das Triggersystem eine hohe Flexibilität. Eine genaue Beschreibung dieser Triggerelektronik befindet sich in [Eh00].

8.2.2 Einbindung in die DAQ

Der Clustertrigger muss zu Beginn eines Experimentes initialisiert und nach jedem Ereignis ausgelesen werden. Dazu mussten entsprechende Programmteile in der DAQ-Software des CB-ELSA-Experimentes implementiert werden. Der dazu in der Programmiersprache C++ geschriebene Programmcode findet sich in Anhang F. Er besteht im Sinne dieser Programmiersprache im Wesentlichen aus drei Funktionen:

- Die Funktion *initFace*[1] initialisiert die Ablaufsteuerung des Clustertriggers. Dazu wird zunächst ein Reset auf der Karte erzeugt und anschließend mehrere Millisekunden gewartet, damit der Sequencer in dieser Zeit sein Programm aus dem EPROM ins RAM laden kann. Danach wird die Startadresse des verwendeten Sequencerprogramms in das entsprechende Register geschrieben und der Sequencer gestartet. Der Clustertrigger läuft dann autonom weiter.

- Mit *readFace* kann der Clustertrigger nach einem Ereignis ausgelesen werden. Zunächst wird der Zählerstand der Clusterzähler ausgelesen und in die erste Speicherzelle einer dafür vorgesehenen Speicherbank geschrieben. Danach wird der Zellzähler, mit dem alle Einträge in den FIFO-Speicher gezählt werden, ausgelesen und in einer *for*-Schleife entsprechend oft auf den FIFO-Speicher des Controller-Moduls des Clustertriggers zugegriffen. Sämtliche Worte werden nacheinander in die eben erwähnte Speicherbank geschrieben.
  Vor der Auslese der Zellularlogik-Controllerkarte wird überprüft, ob das Busy-Flag zurückgesetzt ist und somit der Sequencer die Matrixauslese abgeschlossen hat. Wenn dies nicht der Fall ist, geht die Auslesesoftware in eine Warteschleife, die jedoch nach 100 Durchläufen abgebrochen wird, um zu verhindern, dass bei einer Fehlfunktion des Clustertriggers die DAQ-Software in eine Endlosschleife gerät. In diesem Fall wird von der DAQ-Software eine Fehlermeldung ausgegeben.

- Mit *stopFace* wird der Sequencer nach Beendigung eines Runs wieder angehalten.

[1] Die Bezeichnung FACE für **Fa**st **C**luster **E**ncoder entstammt dem Crystal-Barrel-Experiment am LEAR und bezeichnete den dort verwendeten Clustertrigger

8.3 Funktionstest des Triggers an CB-ELSA

8.3.1 Test der Datenzuführung

Nach dem Einbau des Cluster-Triggers in das CB-ELSA-Experiment bestand erstmalig die Möglichkeit, die Clusteridentifikation mit realen Ereignissen aus dem Crystal-Barrel-Detektor zu testen. Zunächst wurden dazu Ereignisse aus der Höhenstrahlung analysiert, was den Vorteil hat, dass diese Untersuchungen unabhängig von der Verfügbarkeit des Elektronenbeschleunigers möglich sind. Bei diesen Messungen wurde der Innendetektor als Triggerdetektor benutzt, wobei gefordert wurde, dass mindestens eine Lage angesprochen hat. Bei einer Messzeit von ca 12 Stunden wurden etwa 300 000 Ereignisse aufgezeichnet.

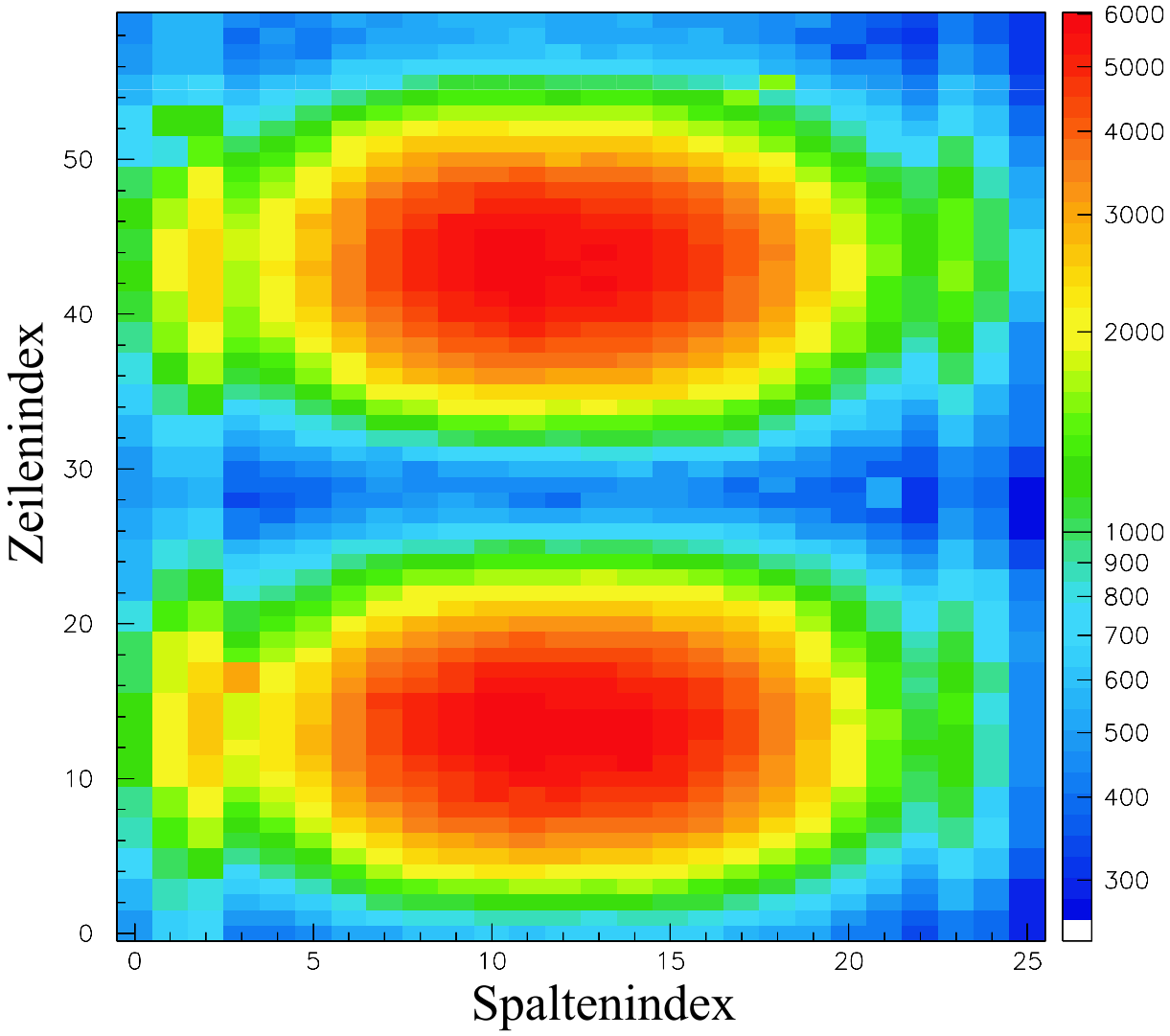


Abbildung 8.5: Verteilung der Ansprechhäufigkeit in der Zellmatrix. Die obere bzw. untere Bildhälfte bezieht sich auf die obere bzw. untere Halbschale des Barrels

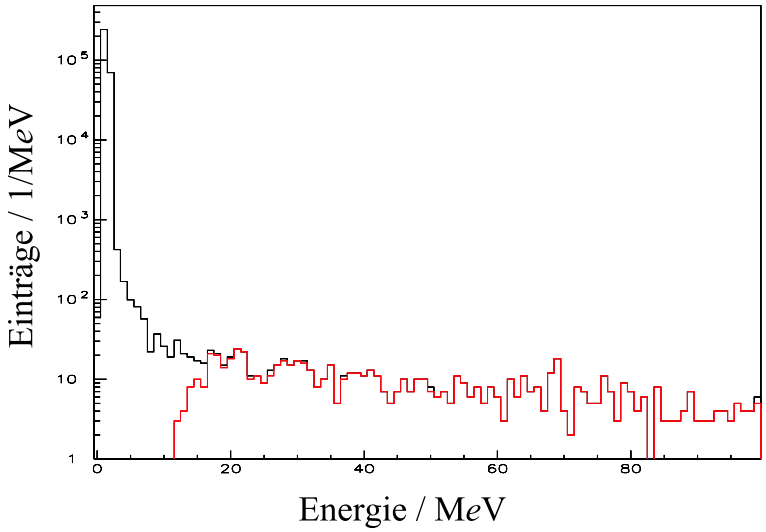


Abbildung 8.6: Cosmic-Spektrum eines Barrelkristalls. Der rote Plot zeigt nur solche Ereignisse, bei denen auch die entsprechende Zelle in der Logikmatrix gesetzt war

Ein erster Test bestand darin, für jede Zelle die Anzahl der Ereignisse zu zählen, bei denen das Signal aus dem zugeordneten Kristall über der Diskriminatorschwelle lag. Abbildung 8.5 zeigt ein Histogramm, in dem diese Anzahl farblich für jede Zelle der Matrix dargestellt ist. Die beiden sichtbaren Strukturen entstehen durch die kosinusförmige Winkelabhängigkeit der Höhenstrahlung auf der Erdoberfläche. Defekte Kanäle, bei denen keine Verbindung zwischen Kristall und Logikzelle besteht oder bei denen die Zelle aufgrund von Rauschen oder Oszillationen dauernd gesetzt ist, wären in diesem Histogramm bereits deutlich zu erkennen. Unterbrochene Verbindungen wären als weiße Felder sichtbar, wogegen dauernd gesetzte Zellen die Skalierung des Histogramms derart verändern würden, dass nur noch sie als isolierte rote Felder zu sehen wären. Solche Effekte sind jedoch nicht zu sehen, sodass derartige Fehler bereits mit diesem einfachen Test ausgeschlossen werden können.
Ein empfindlicherer Test, aus dem auch Vertauschungen in der Signalzuführung zwischen benachbarten Kanälen zu erkennen sind, bestand darin, jeweils nach Abschluss der Konversion in einem beliebigen ADC-Kanal zu prüfen, ob das zugehörige Trefferflipflop in der Zellularlogik gesetzt war. Dazu zeigt Abbildung 8.6 das Energiespektrum der Höhenstrahlung in einem willkürlich ausgewählten Kristall. Die schwarze Kurve zeigt das gemessene Energiespektrum. Der rot markierte Teil des Spektrums stammt von den Ereignissen, bei denen auch in der dem Kristall zugeordneten Logikzelle ein Treffersignal registriert wurde. Deutlich erkennt man hier den Einfluss der Diskriminatorschwelle, die bei etwa 15 MeV liegt. Oberhalb der Schwelle liegen von minimalen Abweichungen abgesehen die rote und die schwarze Kurve aufeinander.
Offenbar ist der ausgewählte Kanal voll funktionsfähig. Um diese Überprüfung gleichzeitig für alle Kanäle durchführen zu können, wurde eine andere Art der Darstellung ausgewählt. Dazu wurde für jedes Energiebin im dargestellten Spektrum die Anzahl der Einträge, die der roten Kurve entsprechen, durch die Anzahl der der schwarzen Kurve entsprechenden Einträge dividiert. Der so ermittelte Quotient ist in Abbildung 8.7 in einem zweidimensionalen Plot (sog. Diagnoseplot) als Funktion der Energie und des Kristallindizes für 130 Kristalle farblich dargestellt. Der Bereich unterhalb der Schwelle, wo der Quotient den Wert 0 hat, ist blau dargestellt, der Bereich oberhalb der Schwelle hingegen, wo der Quotient bei 1 liegt, rot markiert ist. Der Bereich dazwischen, in dem der Quotient von 0 auf 1 übergeht, ist durch den Wechsel der Farben deutlich sichtbar. Dieser Bereich, in dem die Diskriminatorschwelle liegt, fluktuiert leicht von

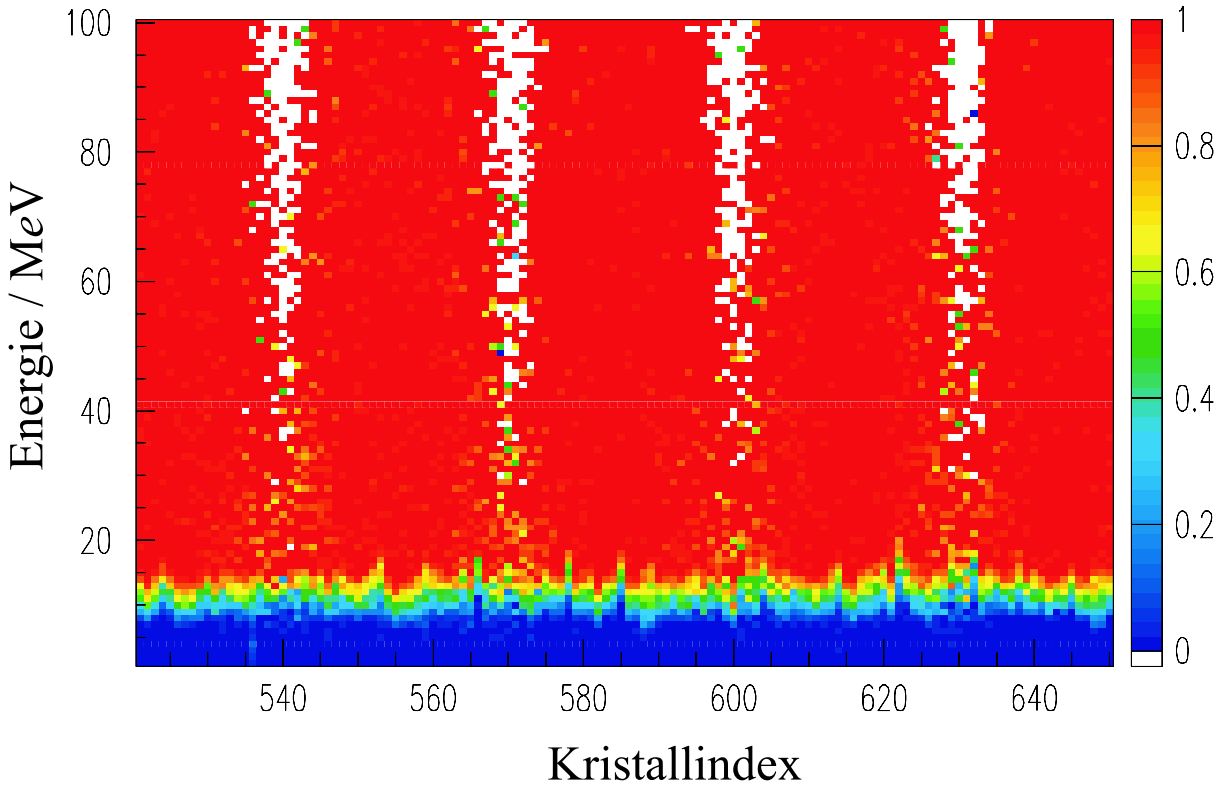


Abbildung 8.7: Diagnoseplot für die Kristalle 520 bis 650. Die Darstellungsweise wird im Text erläutert

Kristall zu Kristall. Er liegt bei etwa 10 MeV bis 15 MeV.
Der Kristallindex ist in Azimutrichtung des Barrels durchgezählt. Somit wird mit einer Periode, die einer Anzahl von 30 Kristallen entspricht, ein Bereich durchschritten, der von der Höhenstrahlung aufgrund der Winkelverteilung schlechter ausgeleuchtet wird. Deshalb liegen dort vor allem im Bereich höherer Energien deutlich weniger Ereignisse vor, was das Auftreten der weißen keilförmigen Zonen erklärt.
Defekte oder vertauschte Signalzuführungen würden sich in diesem Histogramm sofort als einfarbige rote oder blaue vertikale Linien zeigen. Entsprechende Histogramme wurden für alle 1380 Kristalle des Barrels angelegt. Damit steht ein ausgesprochen sensitives Diagnosewerkzeug für die Funktion der Signalzuführung und des Clustertriggers zur Verfügung. Es konnte gezeigt werden, dass, nachdem einige Kabelvertauschungen beseitigt wurden, die Signalzuführung korrekt arbeitet. Diese Messungen werden in regelmäßigen Zeitabständen während der Datennahme wiederholt.

8.3.2 Übernahme des Clusterzählwertes in die Triggerelektronik

Der Zählerstand des Clusterzählers wird in Form von ECL-Signalen über Flachbandkabel zu der bereits in Abschnitt 8.2.1 eingeführten PLU8 geführt. Dort wird der Zählerstand nach erfolgter Clusterzählung gespeichert, wobei das $DataValid$-Signal des Triggers als Strobe-Signal dient. Wenn das *DataValid*-Signal nach einer definierten Zeit nicht erzeugt wird, bricht die Trigger-

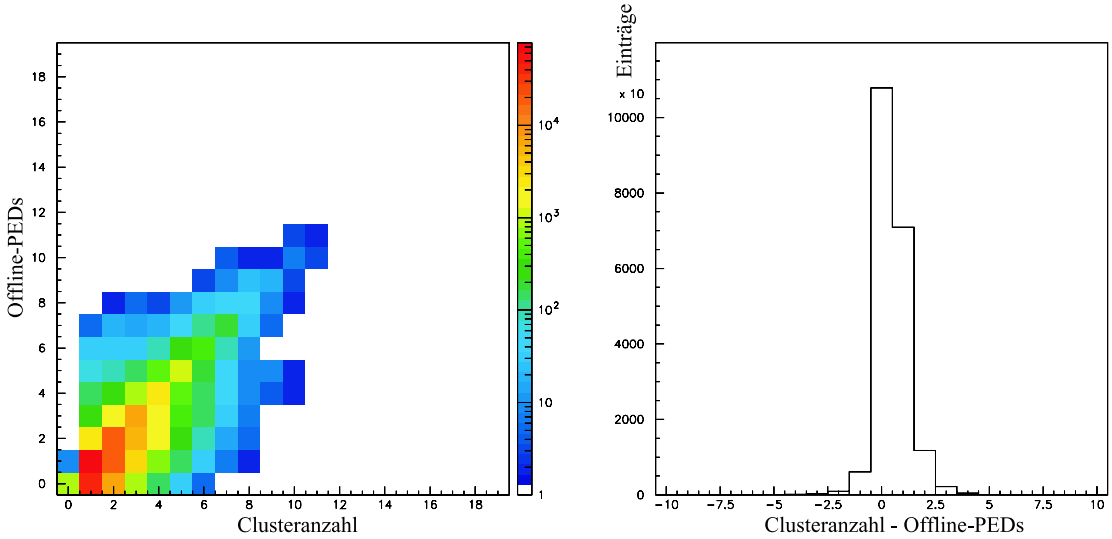


Abbildung 8.8: Die Korrelation zwischen der vom Trigger ermittelten Clusteranzahl und der Anzahl der PEDs.

elektronik den Vorgang ab und der momentane Zählerstand wird in die PLU übernommen. Diese Timeout-Zeit entspricht in etwa der Zeit, die der Clustertrigger für die Identifizierung von 16 Clustern benötigt.
Die DAQ liest während des Experimentes sowohl den Inhalt dieser PLU über CAMAC, als auch den Clusterzähler in der Triggereinheit über das VMEbus-Interface aus und speichert diese Daten mit den übrigen Experimentdaten ab. Bei einer korrekten Funktion der Datenübernahme sollten bei Ereignissen mit 16 oder weniger Clustern die Einträge in der PLU und im Clustertrigger exakt übereinstimmen. Bei Ereignissen mit mehr als 16 Clustern sollte dagegen in der PLU wegen des Timeouts stets 16 eingetragen sein. In der Tat konnte in den Experimentdaten genau dieses Verhalten beobachtet werden. Somit arbeitet auch dieser Teil der Elektronik korrekt.

8.3.3 Korrelation zwischen Cluster- und Photonenanzahl

Um zu überprüfen, inwieweit das Ergebnis der Clusteranalyse mit der Anzahl der Photonen korreliert, die das Barrel-Kalorimeter getroffen haben und über die Analyse der von den ADCs gelieferten Daten als PEDs identifiziert werden, wurde die vom Clustertrigger gezählte Clusteranzahl mit der Anzahl der PEDs verglichen, die aus der Offline-Rekonstruktion ermittelt wurde. Aus diesem Vergleich ergeben sich die beiden Histogramme, die in Abbildung 8.8 zu sehen sind. Im linken Histogramm ist die Anzahl der Ereignisse als Funktion der mit dem Clustertrigger gezählten Cluster und der Anzahl der von der Offline-Rekonstruktion für das gleiche Ereignis gefundenen PEDs dargestellt. In dem Histogramm auf der rechten Seite wurde dagegen die Anzahl der Ereignisse über der Differenz aus diesen beiden Zahlen aufgetragen. Die Anzahl der aus einem gegebenen Ereignis ermittelten PEDs und Cluster hängt von verschiedenen Schwellen ab. In der Anzahl der identifizierten Cluster wirkt sich lediglich die eingestellte Diskriminatorschwelle aus, wobei zu beachten ist, dass diese zwischen den Kristallen leicht fluktuiert. In der Software zur Offline-Rekonstruktion der PEDs werden hingegen mehrere Schwellen benutzt, die

die deponierte Mindestenergie in einem Kristall, in einem Cluster und in einem PED festlegen. Wenn mehrere PEDs einen gemeinsamen Cluster bilden, führt dies dazu, dass die Zahl der PEDs die der Cluster übersteigt. Andererseits kann es aber auch sein, dass die Clusteranzahl größer ist, als die der PEDs. Dies tritt dann auf, wenn in der Rekonstruktionssoftware eine Schwelle verwendet wird, die höher ist als die Diskriminatorschwelle. Dies ist z.B. für die Mindestenergie eines PEDs (≥ 20 MeV) der Fall. In den Histogrammen in Abbildung 8.8 sieht man, dass in der überwiegenden Anzahl der Ereignisse Cluster- und Offline-PED-Anzahl gleich sind. Die Verteilung ist jedoch nicht symmetrisch. Dies ist darauf zurückzuführen, dass die Diskriminatorschwelle im Vergleich zu den Softwareschwellen sehr niedrig angesetzt wurde, um den Trigger möglichst effizient zu machen.

Kapitel 9

Erste physikalische Ergebnisse

Seit der Inbetriebnahme des CB-ELSA-Experimentes im Januar 2000 wurden bereits bei verschiedenen Primärstrahlenergien und Triggerkonfigurationen Daten aufgenommen. Auch der Clustertrigger wurde bei einem großen Teil der Messungen zur Anreicherung gewünschter Reaktionskanäle eingesetzt. Die Auswertung der dabei gewonnenen Daten steht erst am Anfang, jedoch lässt sich bereits jetzt erkennen, dass die mit dem Crystal-Barrel-Detektor gewonnenen Daten eine hervorragende Qualität besitzen.
Die in den Abbildungen 9.1 bis 9.4 gezeigten Spektren basieren auf einem Teildatensatz des CB-ELSA-Experimentes, der etwa 22 Mio. Ereignisse bei einer Primärenergie von 1400 MeV und 111 Mio. Ereignisse bei einer Primärenergie von 2600 MeV umfasst. Alle Ereignisse bei 1400 MeV und etwa die Hälfte der Ereignisse bei 2600 MeV Primärstrahlenergie wurden unter Anwendung des Clustertriggers aufgenommen, wobei eine Clusteranzahl von mindestens zwei Clustern gefordert wurde.
In den beiden in Abbildung 9.1 gezeigten Spektren ist die Anzahl der beobachteten Ereignisse im Reaktionskanal γp $\rightarrow$ p$\gamma\gamma$ über der invarianten $\gamma\gamma$-Masse aufgetragen. Man erkennt deutlich die Peaks, die den drei neutralen Mesonen π^0, η und η', die jeweils einen Zerfallskanal in zwei Photonen besitzen, zuzuordnen sind. Bei einer Primärenergie von 1400 MeV kann das η' noch nicht erzeugt werden und ist daher im linken Spektrum nicht zu sehen. Die Abweichungen der mit einem einfachen Gaußfit ermittelten Massen dieser Mesonen von den Literaturwerten (π^0: 135,0 MeV/c^2, η: 547,3 MeV/c^2und η': 957,8 MeV/c^2[Ca98]) liegen durchweg unter einem Prozent. Auch die gemessenen Breiten der Resonanzen zeigen, dass das Detektorsystem eine hervorragende Auflösung besitzt. Zudem weisen die Spektren nur einen minimalen Untergrund auf. Die Überhöhung im rechten Spektrum bei 780 MeV/c^2 stammt aus Fehlinterpretationen der Offline-Rekonstruktion, die auftreten, wenn bei Zerfällen des ω in $\pi^0\gamma$ ein niederenergetisches γ aus dem π^0-Zerfall nicht erkannt wird.
Aus den Vierervektoren der so identifizierten Mesonen und des ebenfalls nachgewiesenen Protons lässt sich für jedes Ereignis die Gesamtenergie im Schwerpunktsystem berechnen. In Abbildung 9.2 ist die Anzahl der rekonstruierten γp $\rightarrow$ pπ^0-Ereignisse als Funktion dieser Gesamtenergie aufgetragen. Abbildung 9.3 zeigt die entsprechenden Spektren für den Kanal γp $\rightarrow$ pη. In allen Spektren sind deutlich Resonanzstrukturen zu erkennen. So ist im pπ^0-Kanal im linken Spektrum bei 1280 MeV die Δ-Resonanz zu erkennen. Die zu große Masse ist vermutlich darauf zurückzuführen, dass ein Teil der Resonanzstruktur durch den eingeschränkten Akzeptanzbereich des Taggers bei kleinen Energien abgeschnitten ist. Aus dem gleichen Grund ist

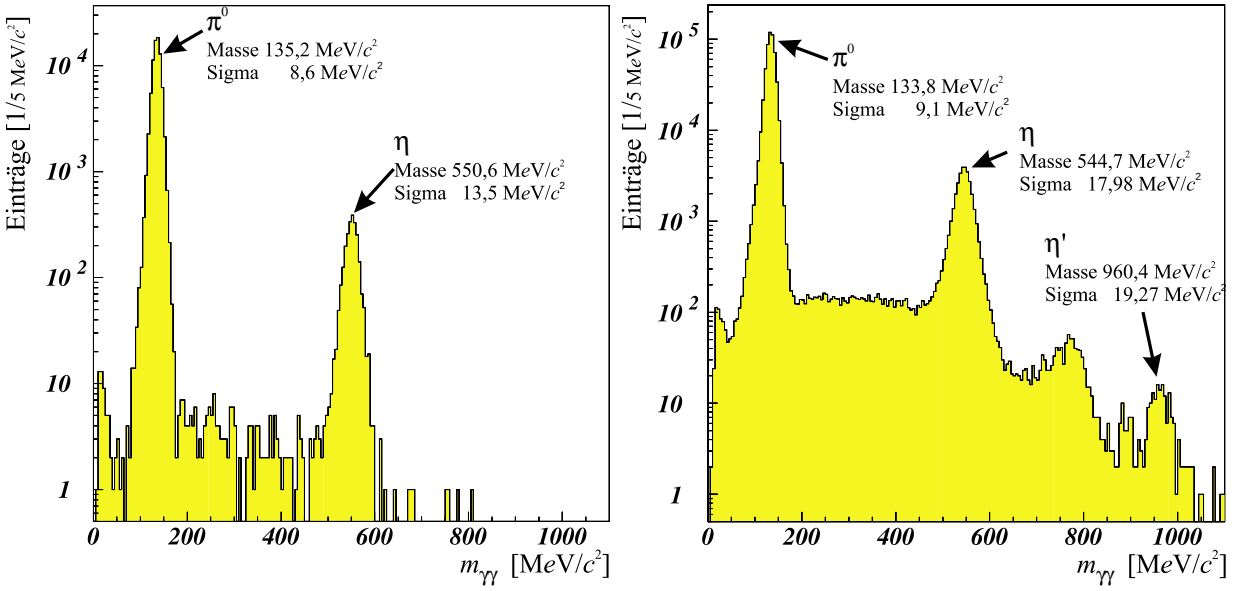


Abbildung 9.1: Anzahl der Ereignisse des Kanals $\gamma p \rightarrow p\gamma\gamma$, aufgetragen über der invarianten $\gamma\gamma$-Masse, links bei einer Primärstrahlenergie von 1,4 GeV und rechts bei 2,6 GeV

diese Resonanz bei einer Primärenergie von 2,6 GeV gar nicht mehr zu sehen. Oberhalb der Δ-Resonanz sind in beiden Spektren noch zwei weitere Resonanzen zu erkennen.
Im pη-Kanal wird das Spektrum durch eine Resonanz bei 1570 MeV dominiert. Auch hier sind im Verlauf des Spektrums noch weitere Resonanzen zu sehen. Genauere Aussagen hierüber bedürfen jedoch noch weiterer Untersuchungen, die z.B. Partialwellenanalysen einschliessen. Solche Analysen befinden sich jedoch derzeit erst im Anfangsstadium, sodass in Zukunft noch eine Reihe von interessanten Ergebnissen von CB-ELSA zu erwarten ist.
Bislang wurden keine Messungen mit dem CB-ELSA-Experiment durchgeführt, bei denen Kanäle hoher γ-Multiplizität durch gezielte Vorgaben des Clustertriggers angereichert wurden. Ein Beispiel dafür, welche Möglichkeiten solch eine Anreicherung böte, ist die Beobachtung der η- und η'-Mesonen im 6γ-Ausgangskanal, wobei das η-Meson in $3\pi^0$ und das η'-Meson in $2\pi^0\eta$ zerfällt. Abbildung 9.4 zeigt die Spektren, die sich ergeben, wenn die Anzahl der rekonstruierten Ereignisse dieser beiden Kanäle über der invarianten 6-γ-Masse aufgetragen wird. Durch eine gezielte Auswahl von 7-Cluster-Ereignissen ließen sich diese Kanäle schon bei der Messung stark anreichern und damit die Statistik zukünftiger Messungen mit CB-ELSA noch deutlich verbessern.

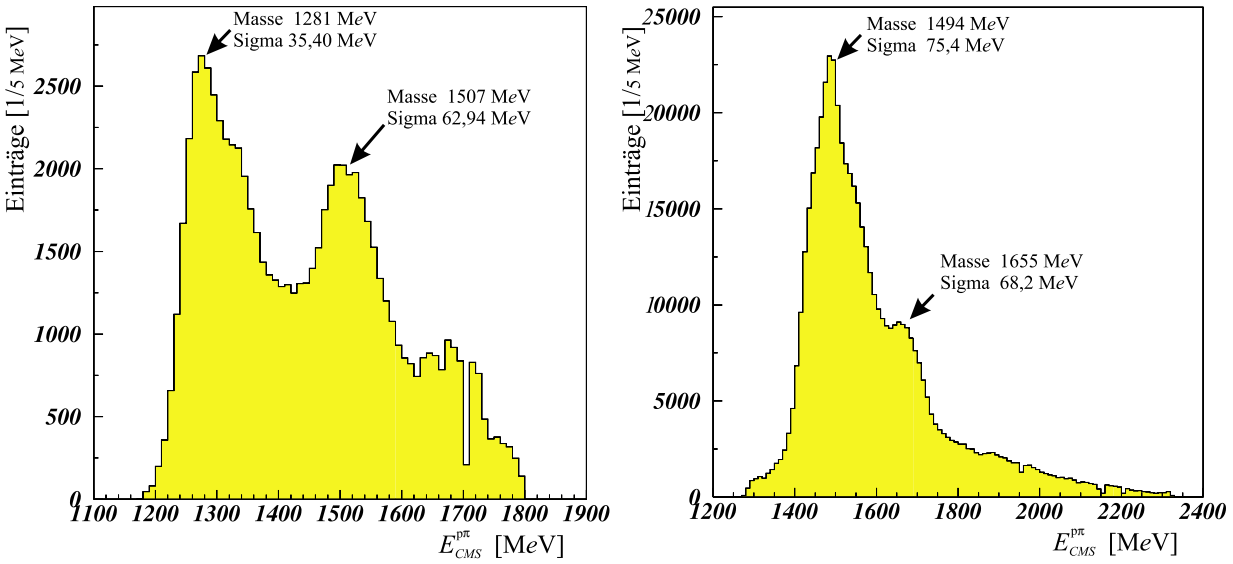


Abbildung 9.2: Anzahl der rekonstruierten Ereignisse des Kanals $\gamma p \rightarrow p\pi^0$, aufgetragen über der Gesamtenergie im Schwerpunktssystem von $p\pi^0$. Das linke Spektrum stammt aus einer Messung bei 1,4 GeV Primärstrahlenergie, das rechte Spektrum aus einer Messung bei 2,6 GeV

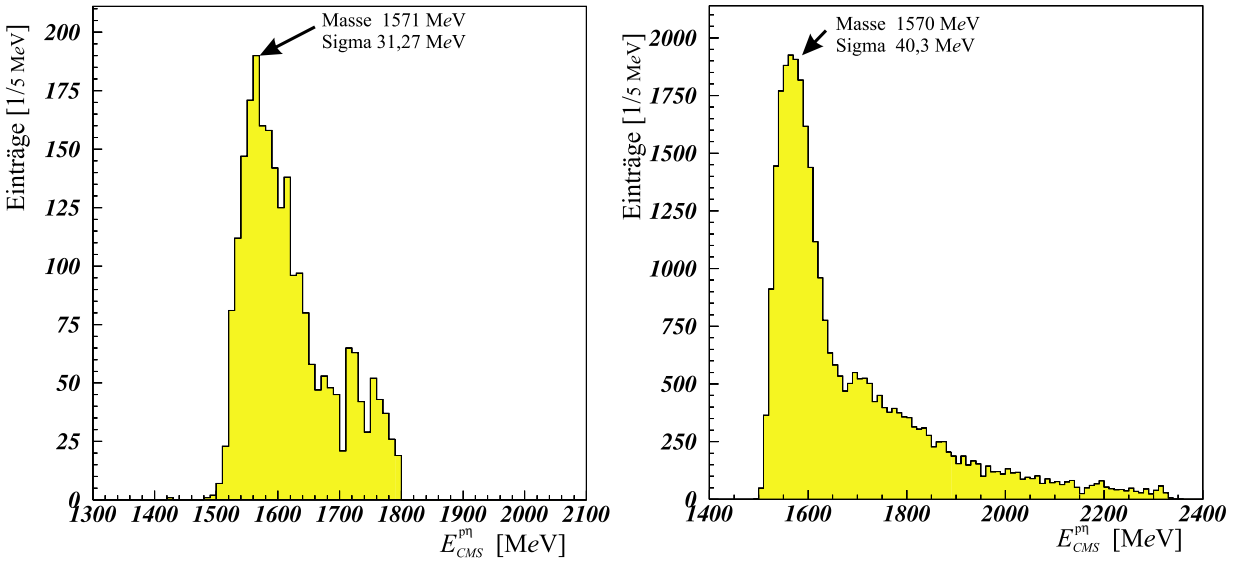


Abbildung 9.3: Anzahl der rekonstruierten Ereignisse des Kanals $\gamma p \rightarrow p\eta$, aufgetragen über der Gesamtenergie im Schwerpunktssystem von $p\eta$. Das linke Spektrum stammt aus einer Messung bei 1,4 GeV Primärstrahlenergie, das rechte Spektrum aus einer Messung bei 2,6 GeV

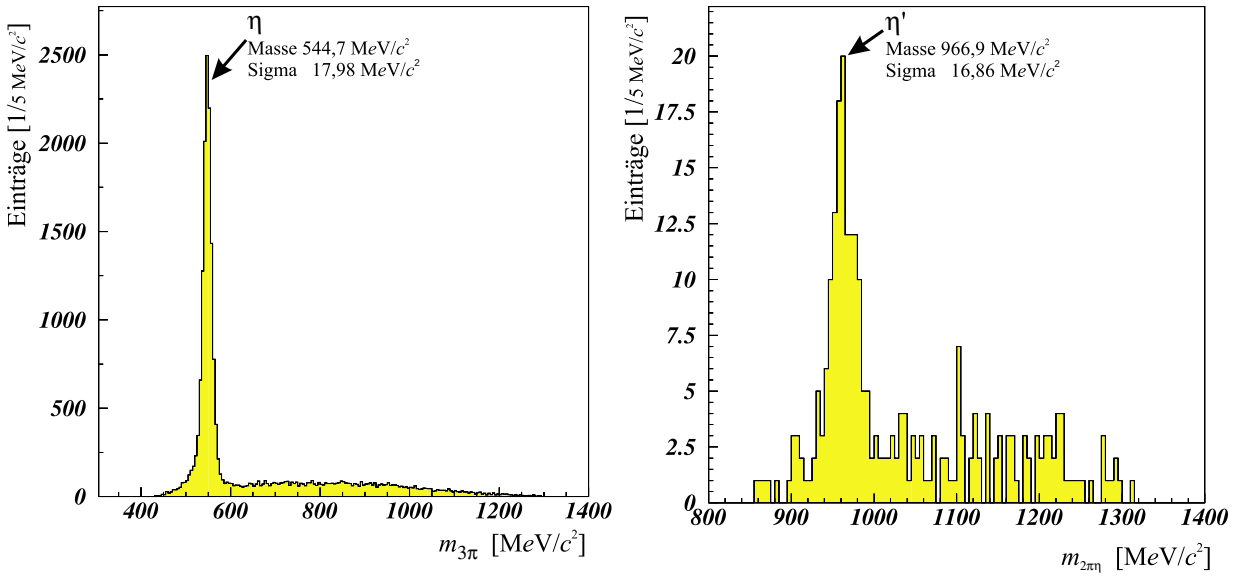


Abbildung 9.4: Anzahl der rekonstruierten Ereignisse der Kanäle $\gamma p \rightarrow p3\pi^0$(links) und $\gamma p \rightarrow p2\pi^0\eta$ über der invarianten 6-γ-Masse bei einer Primärstrahlenergie von 2,6 GeV

Anhang

Anhang A

Beschreibung der neuen Latches für CB-ELSA

A.1 Schaltungsbeschreibung der Latches

A.1.1 Die Eingangsstufe

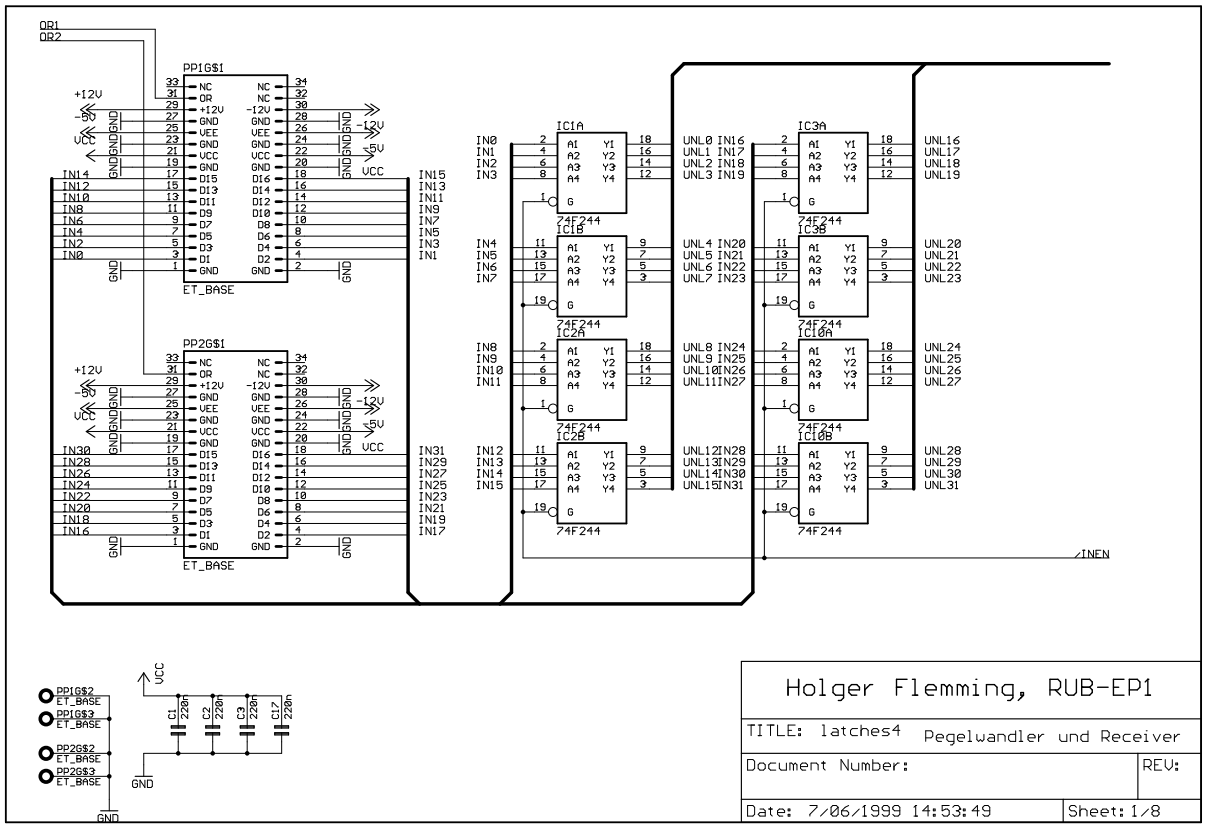


Abbildung A.1: Die Eingangsstufe der Latches

Abbildung A.1 zeigt die Schaltung für die Eingangsstufe der Latches. Die Pegelwandlung von ECL auf TTL übernehmen zwei Piggy-Pack-Platinen, die auf die Latches aufgesteckt werden. Auf diesen Piggy-Packs befinden sich eine 34-polige Pfostenleiste für 16 Differential-ECL-Eingangssignale und die notwendigen Pegelwandler-ICs. Die TTL-Signale werden über 34-polige Pfostenstecker, über die die Piggy-Packs auch mit den erforderlichen Betriebsspannungen versorgt werden, zur Hauptplatine geführt.
Die TTL-Signale von den Piggy-Packs gelangen auf der Hauptplatine zunächst zu TTL-Linereceivern mit Tri-State-Ausgängen (IC1 - IC3). Mit der Steuerleitung $\overline{InEn}$ können die Ausgänge dieser Receiver deaktiviert werden, sodass hinter den Receivern Testsignale in den Signalweg eingespeist werden können.
Die Receiverausgänge werden zu einem Bus mit dem Namen UNL zusammengefasst.

A.1.2 Die Speicher

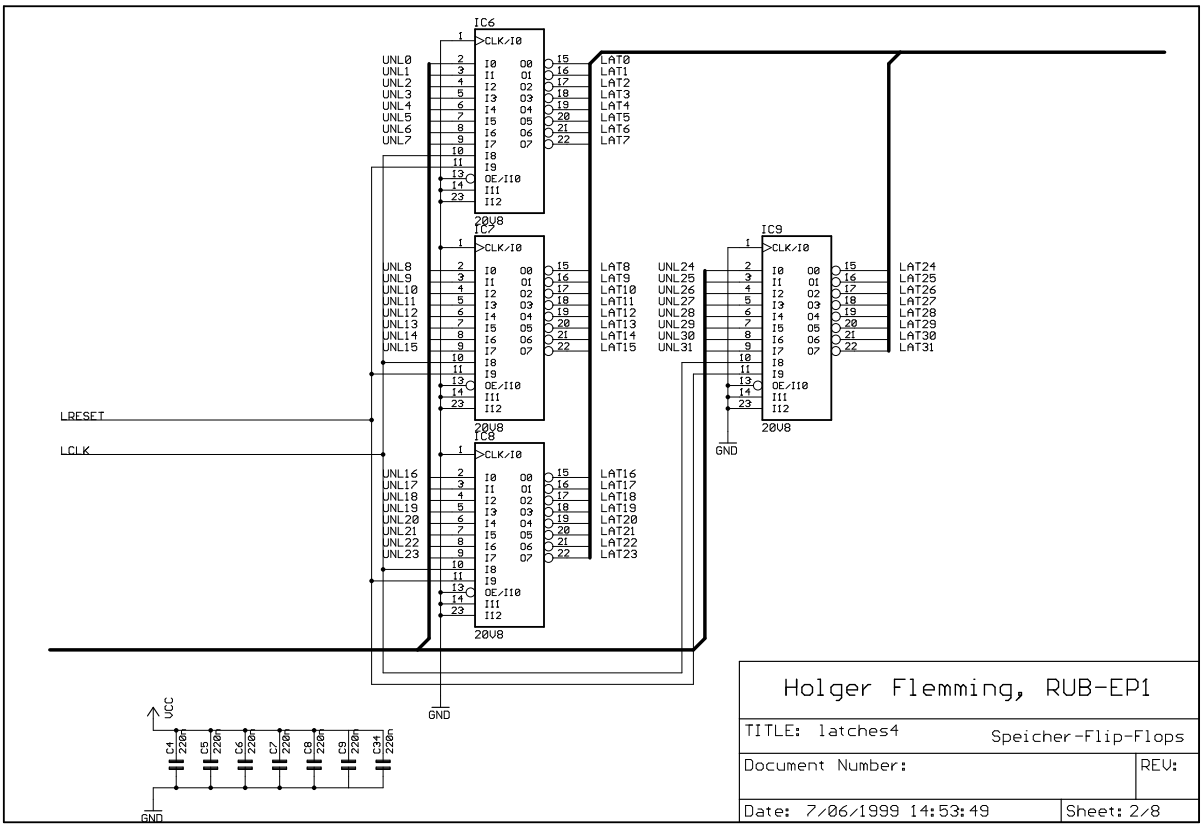


Abbildung A.2: Die Speicherbausteine der Latchmodule

Die Schaltung für die Speicherbausteine der Latches sind in Abbildung A.2 dargestellt. Es handelt sich hierbei um programmierbare Logikbausteine vom Typ GAL 20V8.
Sie sind so programmiert, dass im Ruhezustand die Ausgänge auf Low-Pegel liegen. Die Ausgänge wechseln auf High-Pegel, wenn am Clock-Eingang *und* am entsprechenden Daten-Eingang

ein High-Pegel anliegt. Sind die Ausgänge auf High-Pegel, wechseln sie erst wieder auf Low-Pegel, wenn ein Reset-Signal erfolgt. Dieses Verhalten lässt sich durch die folgende Logikgleichung beschreiben:

$$OUT_n = \overline{Reset} \cdot Out_n + Clk \cdot D_n$$

Die Gals erhalten ihre Clock-Signale aus der $LClk$- und ihre Reset-Signale aus der $LReset$-Leitung. Der UNL-Datenbus ist mit den Eingängen der Latches verbunden, die Ausgänge werden zu dem Bus mit Namen LAT zusammengefasst.

A.1.3 Die Signalverteilung

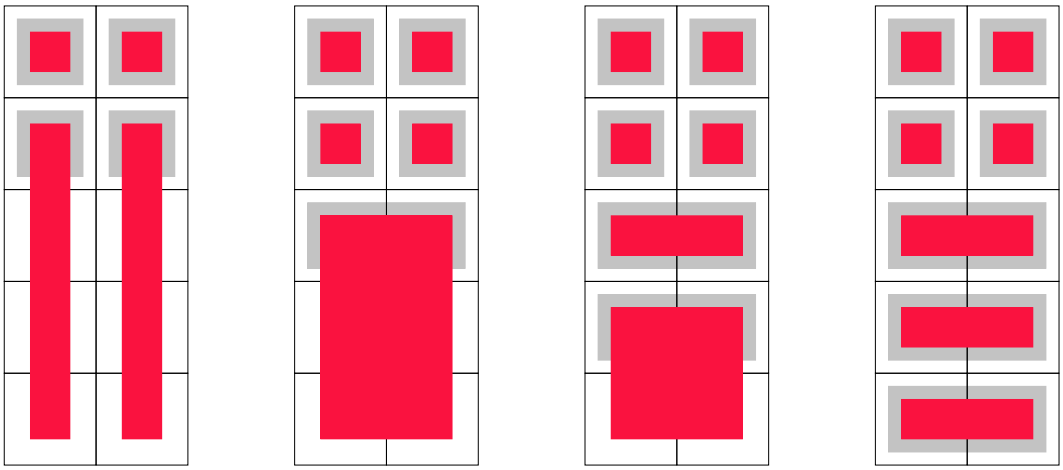


Abbildung A.3: Die Signalverteilung, wenn keine Randkristalle eingebaut sind, bzw. bei einer, bei zwei und drei Lagen von Randkristallen. Die schwarz umrandeten Felder stellen die Logikzellen dar, die grau unterlegten Felder die Kristalle. Die Zuordnungen von den Kristallen zu den Zellen sind rot dargestellt

Die drei Randlagen des Barrels decken, im Gegensatz zu allen anderen Kristallagen, die einen ϕ-Winkel von 6° abdecken, einen Winkel von 12° ab. Dies führt dazu, dass sich an der Schnittstelle zwischen dritter und vierter Lage die Nachbarschaftsbeziehungen ändern. Um an dieser Stelle die korrekten Nachbarschaftzbeziehungen im Zellularlogik-Trigger wiederzugeben, müssen diese 12°-Randkristalle auf zwei benachbarte Logikzellen abgebildet werden.
Zusätzliche Komplikationen ergeben sich dadurch, dass für verschiedene Experimente verschiedene Barrelkonfigurationen mit unterschiedlicher Anzahl von Randlagen geplant sind. Will man trotzdem eine Anbindungsmöglichkeit zu einem Vorwärtsdetektor über die letzte Spalte der Zellularlogik schaffen, so bedeutet dies, dass, abhängig von der Anzahl der Randlagen, vier verschiedene Zuordnungen von den Kristallen auf die Logikzellen im Randbereich realisiert werden müssen. Diese vier Zuordnungen sind in Abbildung A.3 dargestellt. Realisiert werden sie von zwei Gals, die im Schaltbild in Abbildung A.4 zu sehen sind. Die oben beschriebenen Zuordnungen müssen zweimal realisiert werden, da die Reihenfolge der eingehenden Signale gespiegelt ist, abhängig davon ob es sich um Signale der up- oder downstream-Hälfte handelt. Daher ist jeweils nur eines der beiden Gals IC11 oder IC12 aktiv. Das andere Gals gibt die Signale so

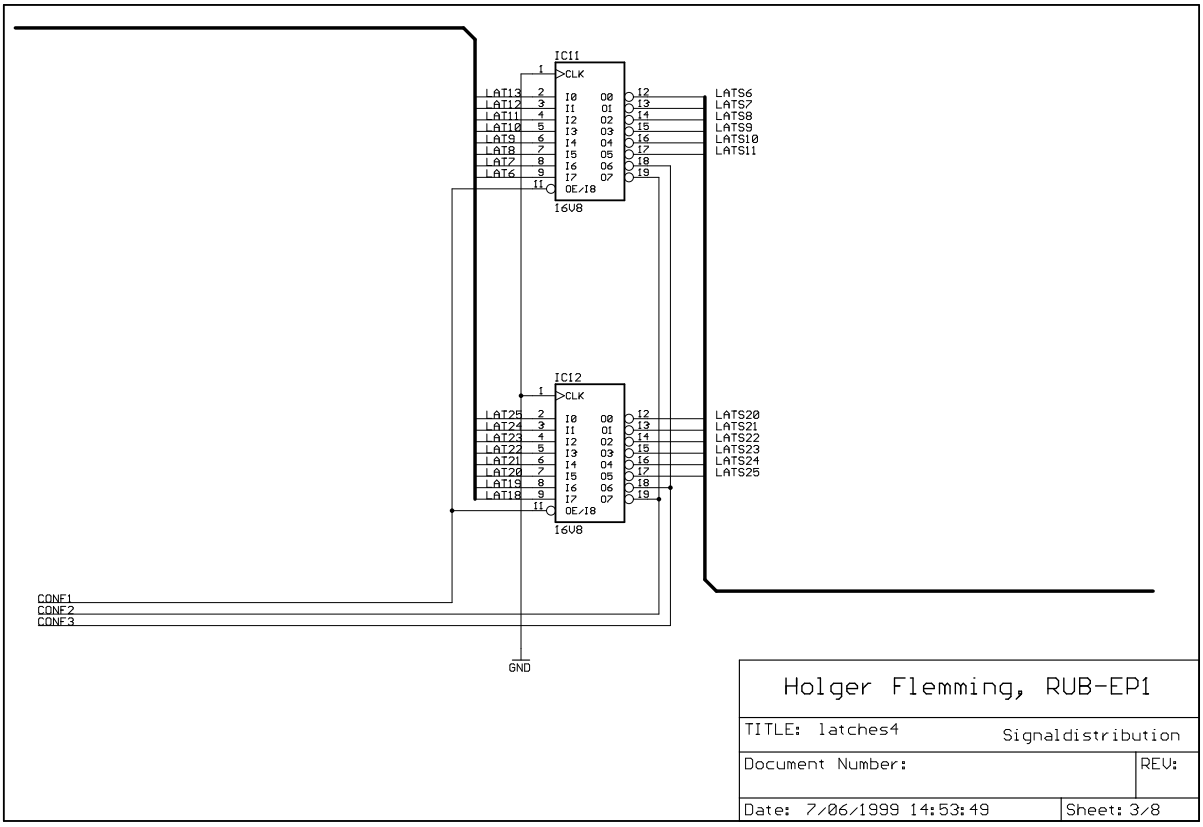


Abbildung A.4: Die Signalverteilung zur Anpassung an unterschiedliche Barrelkonfigurationen

weiter, wie sie an den Eingängen anliegen. Die drei Signale $Conf1$, $Conf2$ und $Conf3$ geben an, welches der beiden Gals aktiv werden muss und wie viele Randlagen vorhanden sind.

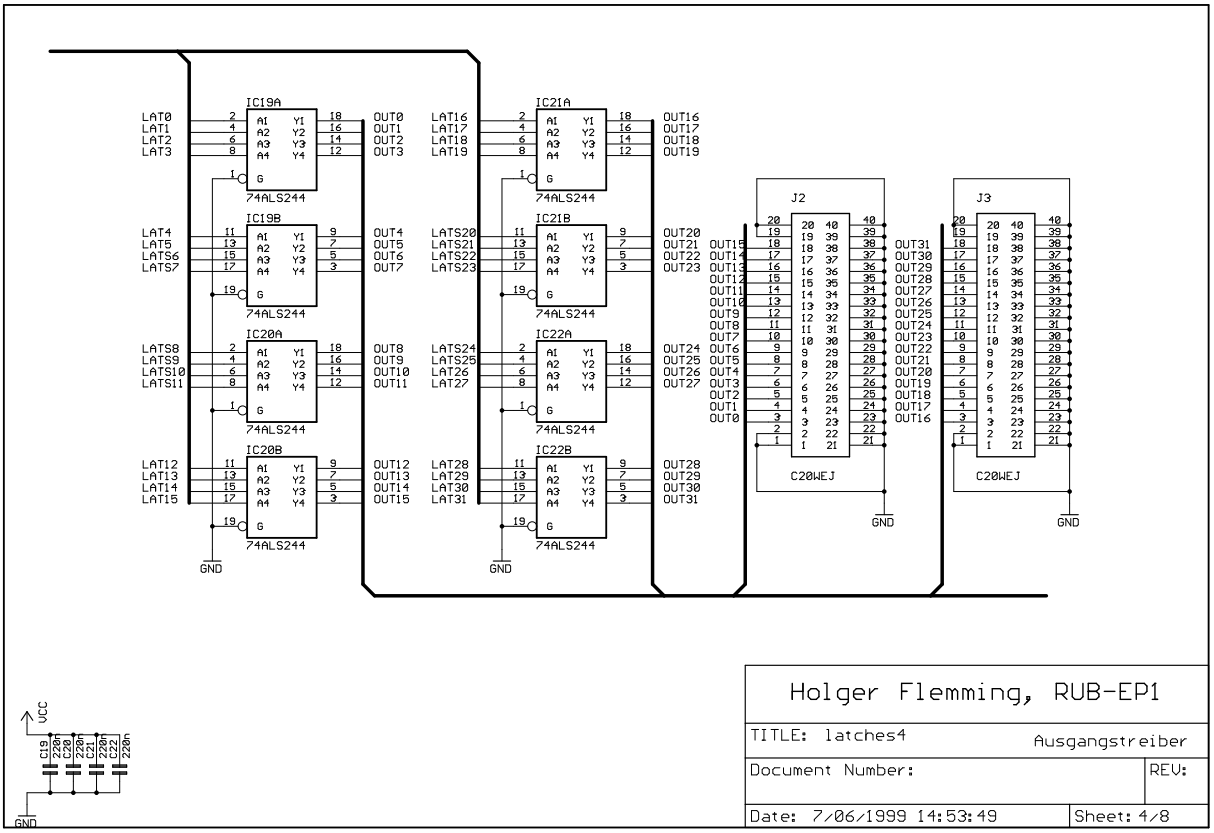


Abbildung A.5: Die Ausgangstreiber

A.1.4 Die Ausgangstreiber

Im Ausgangstreiber werden die Bussignale LAT, bzw. $LATS$ für die Signale, die über die Gals laufen, mit den Leitungstreibern IC19 .. IC22 auf den Ausgangsbus OUT gegeben. Um eine hinreichende Treiberkapazität zu gewährleisten, werden Treiber vom Typ 74ALS244 verwendet. Der $\overline{Enable}$-Eingang liegt fest auf Low-Pegel, sodass die Ausgangstreiber ständig aktiv sind.
Die Ausgangsignale sind an den Steckern $J2$ und $J3$ (zweireihige Fine-Pitch-Stecker mit 2 mal 20 Pinnen) abgreifbar. Um auch von der Backplane auf diese Signale zugreifen zu können, liegen sie zusätzlich an P2 des VME-Businterfaces an.

A.1.5 Der Testmustergenerator

Der auf den Latch-Modulen realisierte Testmustergenerator soll dazu dienen, einzelne Bits des Latches setzen zu können, sodass mit 32 nacheinander durchgeführten Tests das gesamte Latchmodul und die nachfolgende Elektronik überprüft werden können. Die Testmustergeneratoren mehrerer Latches können im Daisy-Chain-Betrieb miteinander gekoppelt werden.
Der Testmustergenerator besteht aus den vier 8-Bit-Schieberegistern IC4, IC23 - IC24 und IC13, die zusammen ein 32-Bit-Schieberegister mit seriellem Eingang und parallelen Ausgängen bilden. Das Schieberegister kann mit dem $\overline{TReset}$-Signal initialisiert werden. Tin ist der seriel-

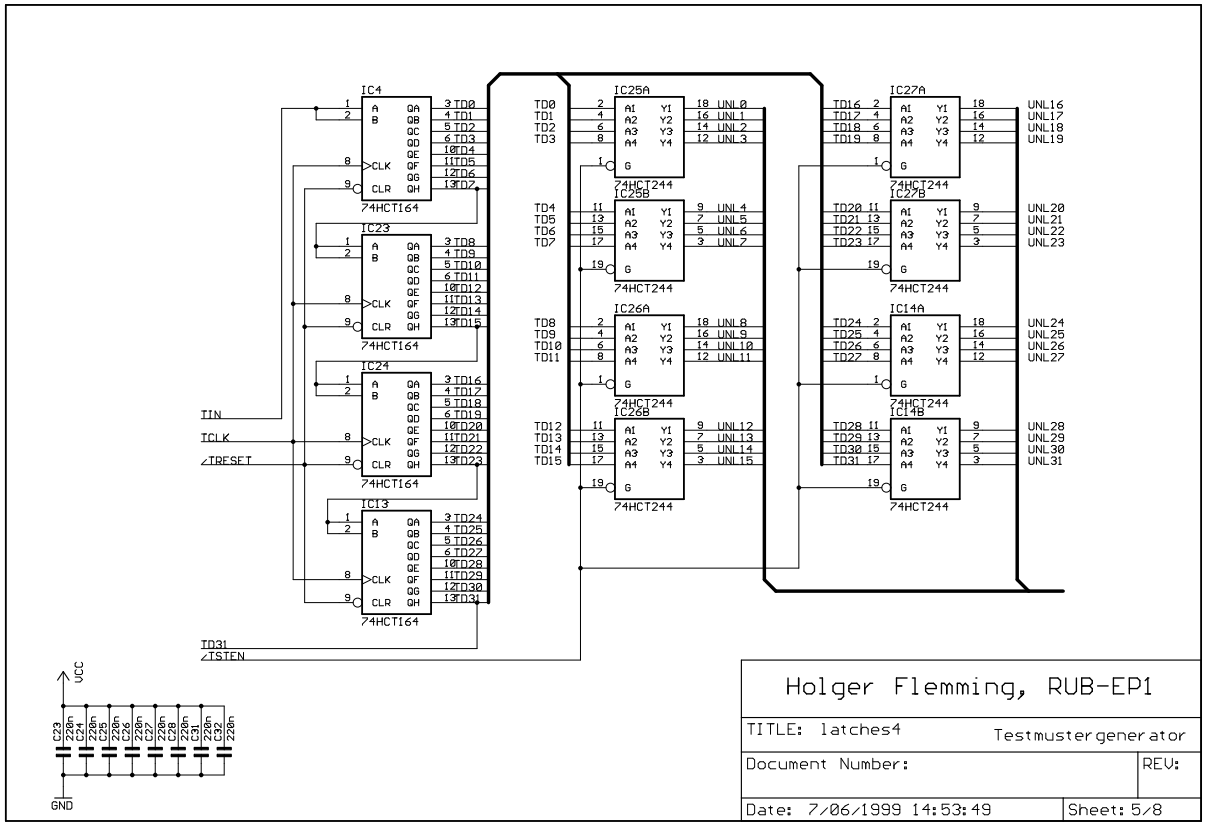


Abbildung A.6: Der Testmustergenerator

le Eingang des Schieberegisters, $TD31$ der serielle Ausgang, um den Daisy-Chain-Betrieb zu ermöglichen. Mit der positiven Flanke des *TClk*-Signals wird der Inhalt des Schieberegisters um ein Bit weiter in Richtung Bit 31 geschoben.
Die Ausgänge des Schieberegisters sind mit den Tri-State-Treibern IC25 - IC27 und IC14 verbunden. Werden diese Treiber durch das $\overline{TstEn}$-Signal aktiviert, speisen sie das Testmuster direkt hinter den Eingangsreceivern (Abbildung A.1) auf den Bus UNL ein.

A.1.6 Die Steuerlogik

Die gesamte Ablaufsteuerung des Latchmoduls erfolgt über die beiden ICs IC5 und IC16. Es handelt sich dabei um programmierbare Logikbausteine vom Typ GAL 16V8. Das Gal IC5 entscheidet darüber, in welchem Betriebsmodus sich das Modul befindet. Als Eingänge liegen die Signale $Test2$ und $TestClk$ an, die über den Leitungstreiber IC32 zum Gal geleitet werden.
Abhängig vom Zustand des $Test2$-Signals werden über die Enablesignale $\overline{InEn}$ und $\overline{TstEn}$ die Signale der Receiver oder des Testmustergenerators aktiviert.
Bei einem Wechsel des Signals $Test2$ von Low auf High erzeugt IC5 einen Lowimpuls auf der Leitung $\overline{TReset}$, wodurch beim Einschalten des Testmodus der Testmustergenerator durch einen Reset-Impuls initialisiert wird. Ebenso erkennt IC5 eine steigende Flanke auf der $TestClk$-Leitung. Eine solche Flanke löst am Ausgang (Pin 17) von IC5 einen Puls aus, der an IC16 wei-

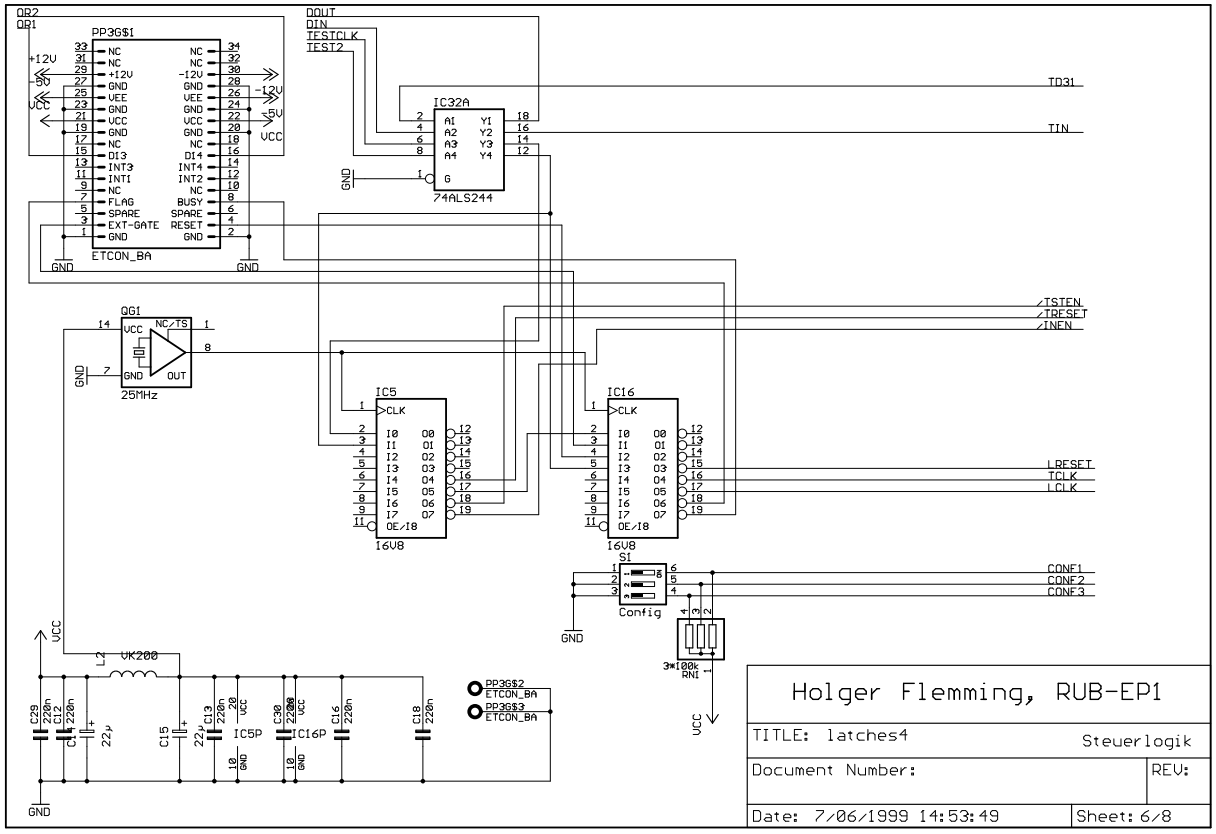


Abbildung A.7: Die Steuerlogik

tergeleitet wird.
IC16 erzeugt die Signale $LReset$, $TClk$ und $LClk$. Dazu werden im Beriebsmodus die Signale $ExtGate$ und $Reset$ benutzt, die von außen über den Piggy-Pack-ECL-TTL-Wander $PP3$ zugeführt werden. Dies sind die zentralen Steuersignale des Latches im Betriebsmodus. Bei Auftreten eines Signals auf diesen Leitungen werden sofort die Steuerleitungen $LReset$, bzw. $LClk$ aktiviert, sofern sich das Latchmodul nicht im Testmodus befindet.
Im Testmodus führt ein Signal am Eingang Pin 2, der von IC5 bei einem $TestClk$-Signal aktiviert wird, dazu, dass eine Signalfolge von Pulsen auf den Leitungen $LReset$, $TClk$ und $LClk$ (in dieser Reihenfolge) erzeugt wird.
Sobald Daten im Latchmodul gespeichert sind, aktivert das Gal IC16 das Ausgangssignal $Flag$, das über den Piggy-Pack-ECL-TTL-Wandler nach außen geführt wird. Das $Busy$-Signal wird aktiviert, sobald das Latchmodul in den Testmodus übergeht. Damit wird der Master-Triggerlogik signalisiert, dass die Latches zur Zeit nicht verfügbar sind.
Um definierte Impulsfolgen erzeugen zu können, werden die beiden GALs IC5 und IC16 mit einem Taktsignal versorgt. Dieses Taktsignal wird von dem Modul selbst mit einem 25-MHz-Quarzoszillator erzeugt. Erwähnenswert ist hierbei, dass die Spannungsversorgung für den Quarzoszillator und die beiden GALs zusätzlich mit der Drossel L2 und den Kondensatoren C14 und C15 abgeblockt wird.

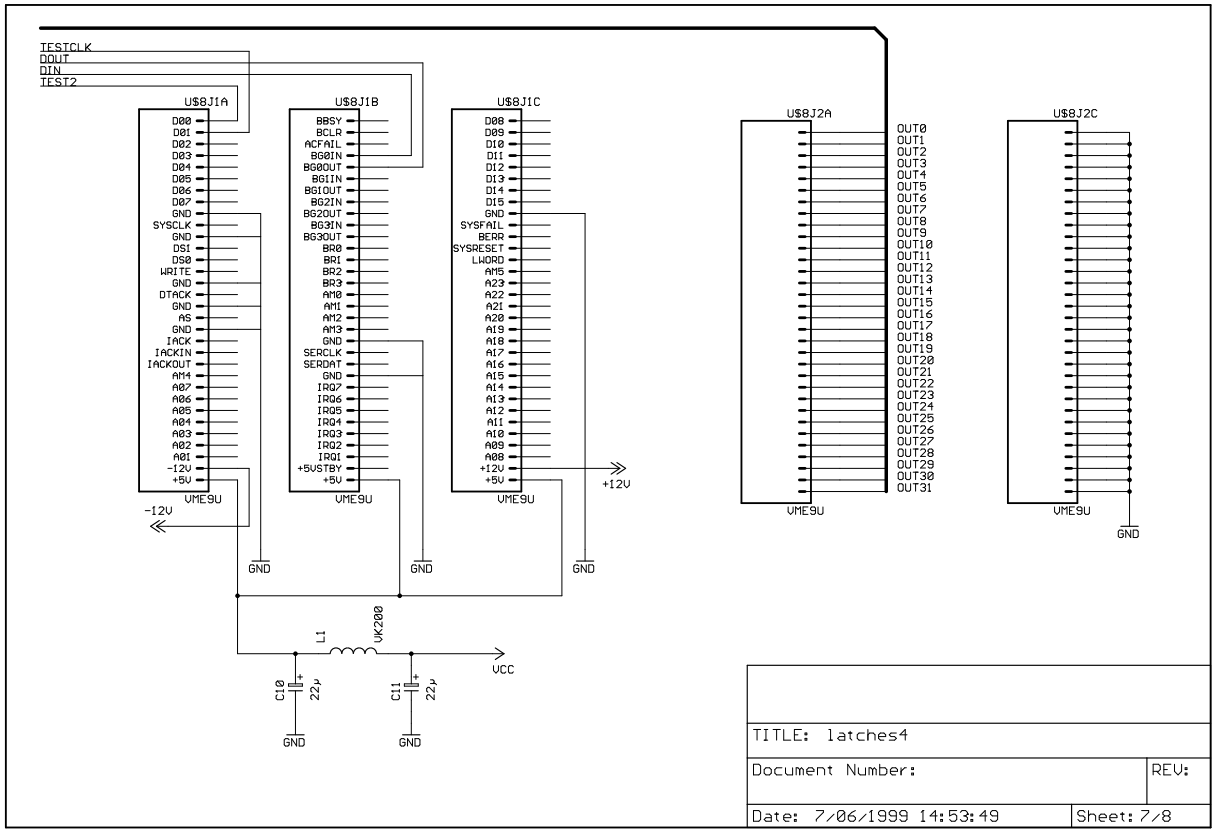


Abbildung A.8: Der Busanschluss

A.1.7 Der Busanschluss

Der VME-Busanschluss wird im Wesentlichen nur zur Spannungsversorgung benutzt. Die $+5$-V-Versorgung wird über eine Drossel L1 und die Kondensatoren C10 und C11 abgeblockt. Die $+12$-V- und -12-V-Versorgung wird direkt den Piggy-Pack-Karten zugeführt.
Die Steuersignale für den Testmustergenerator erreichen das Modul ebenfalls über J1 des VME-Busanschlusses. Diese Belegung entspricht nicht dem VME-Bus-Standard. Hinweise hierzu folgen in Abschnitt A.3.2. Mit J2A des VME-Busanschlusses sind die Datensignale $OUT0$ bis $OUT31$ verbunden.

A.1.8 Die ECL-Spannungsversorgung

Die drei Piggy-Pack-Module benötigen für die Versorgung der ECL-ICs eine Spannung von $-5,2$ V. Im Normalbetrieb in den dafür vorgesehenen Crates der Fa. Struck wird diese Spannung über einen Siemens-Hochstromstecker auf der Rückseite zwischen J1 und J2 des VME-Anschlusses zugeführt und gelangt dann über den Jumper JP1 zu den Piggy-Pack-Modulen.
Der Spannungsregler IC15 wird benötigt, falls das Modul zu Testzwecken in einem anderen Crate oder sogar außerhab eines Crates betrieben werden soll. Die -12-V-Versorgung, die über den VME-Anschluss erfogt, wird mit der Drossel L3 und den Kondensatoren C35 und C36 ab-

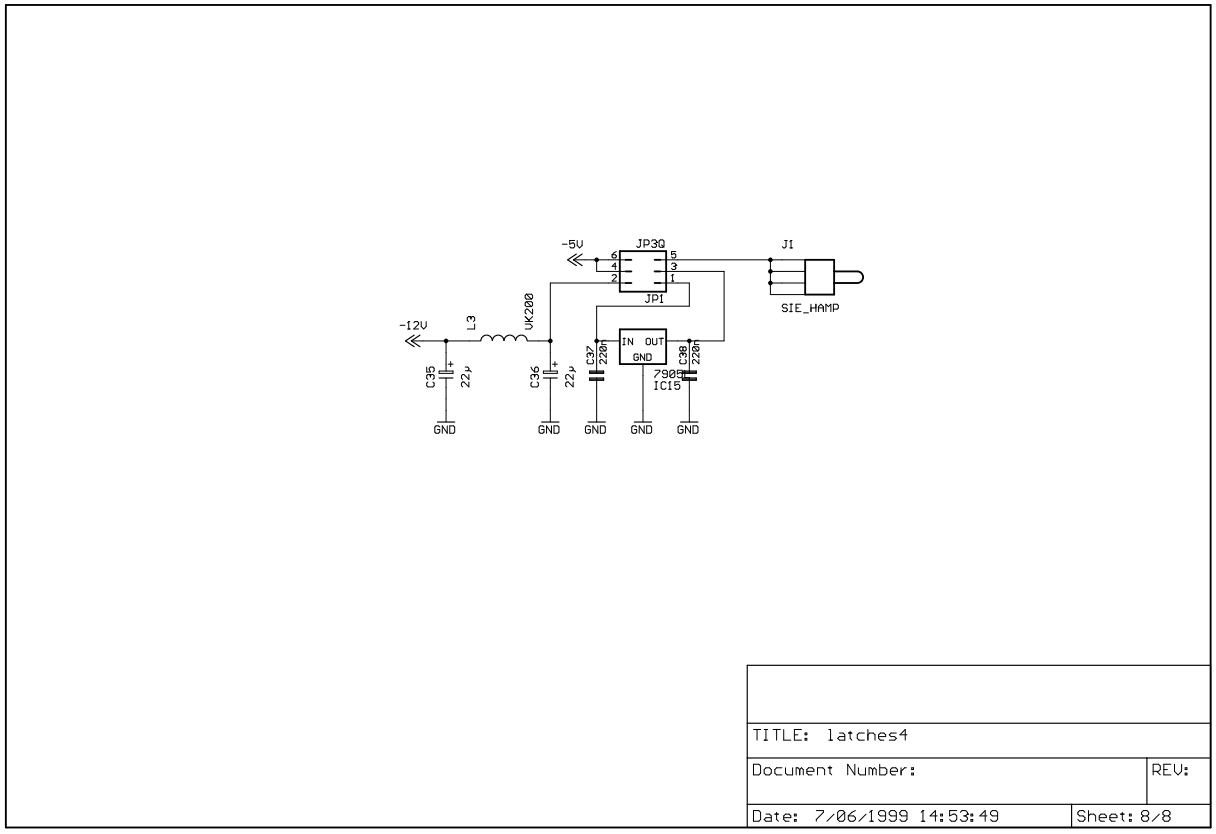


Abbildung A.9: Die ECL-Spannungsversorgung der Piggy-Packs

geblockt und ist über den Jumper JP1 mit dem Eingang des Spannungsreglers verbunden. Die -5 V, die am Ausgang des Spannungsreglers anliegen, können dann, alternativ zu der externen ECL-Versorgung, über den Jumper JP1 zu den Piggy-Packs geführt werden. Es sollten niemals beide -5-V-Versorgungen gleichzeitig aktiviert werden!

A.2 Das Layout der Latchmodule

Durch die Verwendung als VME-Modul ist die Bauform der Latches bereits vorgegeben. Sie werden auf einer Platine von 160 mal 233 mm^2 untergebracht. Das Layout wurde mit der Leiterplatten-CAD-Software EAGLE von der Firma CadSoft erstellt. Die Leiterplattenentflechtung konnte weitgehend mit dem Autoroutermodul vorgenommen werden.
Die Abbildungen A.10, A.11 und A.12 zeigen den Bestückungsplan, sowie das Layout von Bestückungs- und Lötseite. In Abbildung A.13 ist der Aufbau der Platine zu sehen.

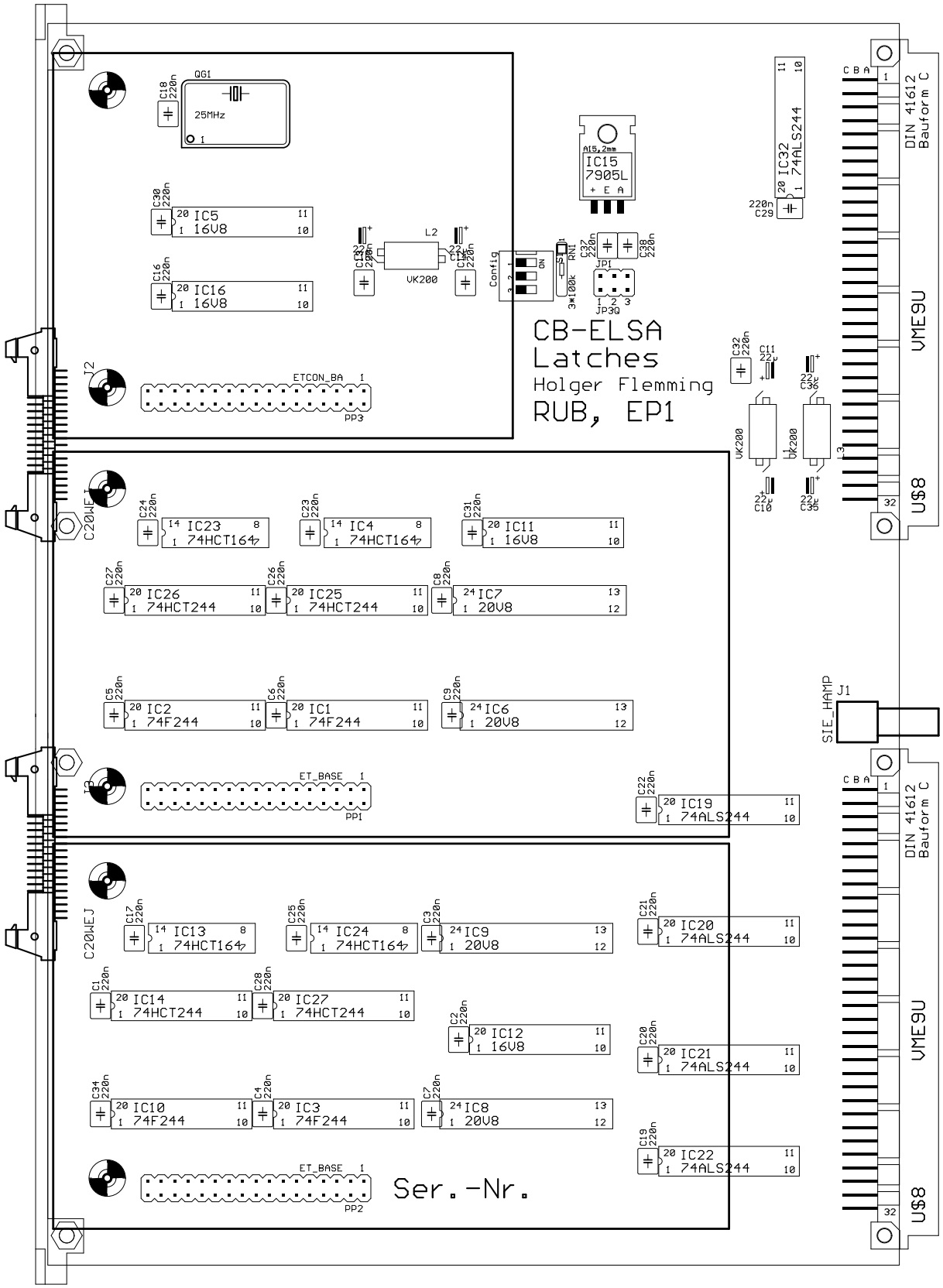


Abbildung A.10: Bestückungsplan des Latchmoduls

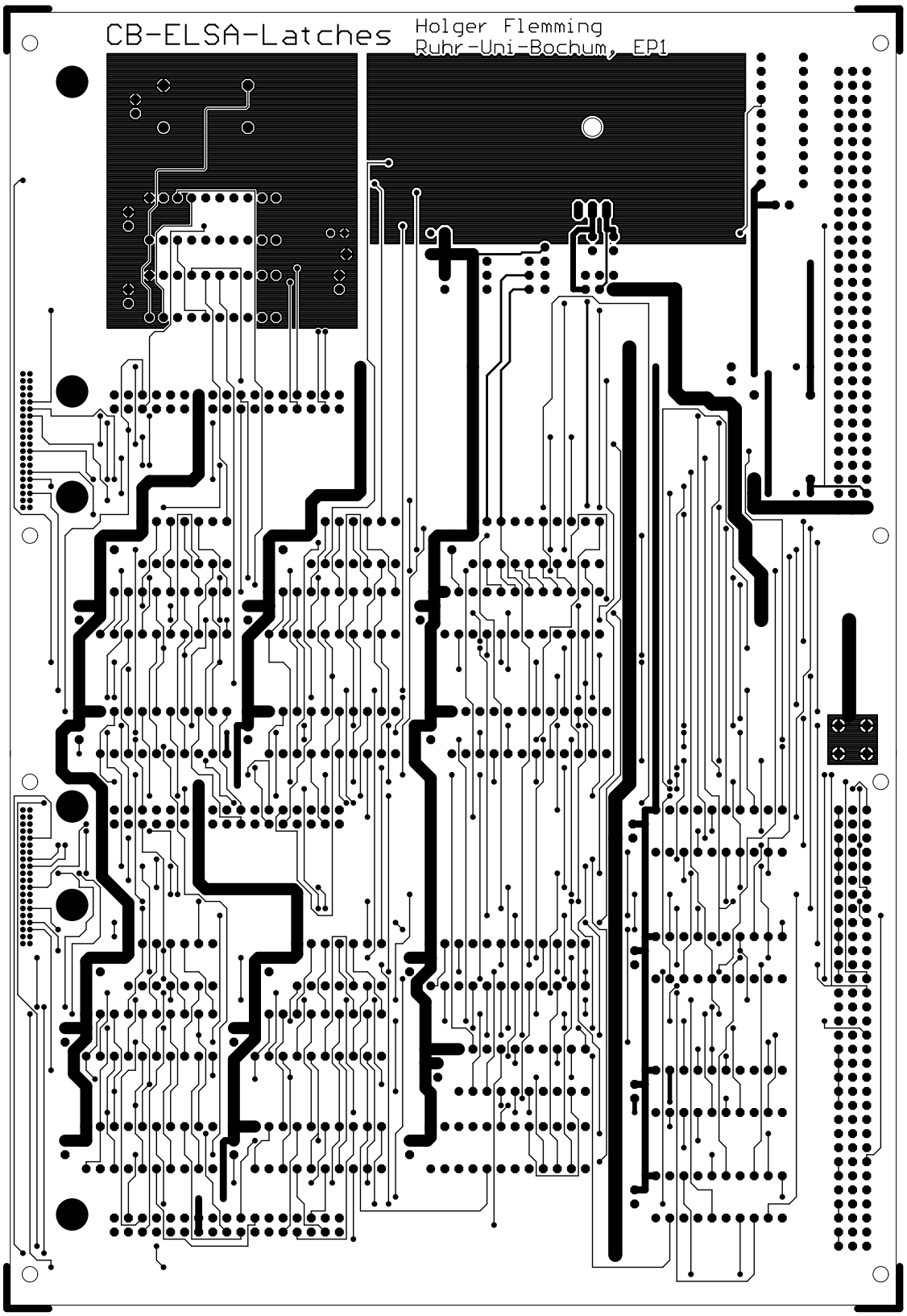


Abbildung A.11: Layout der Bestückungsseite

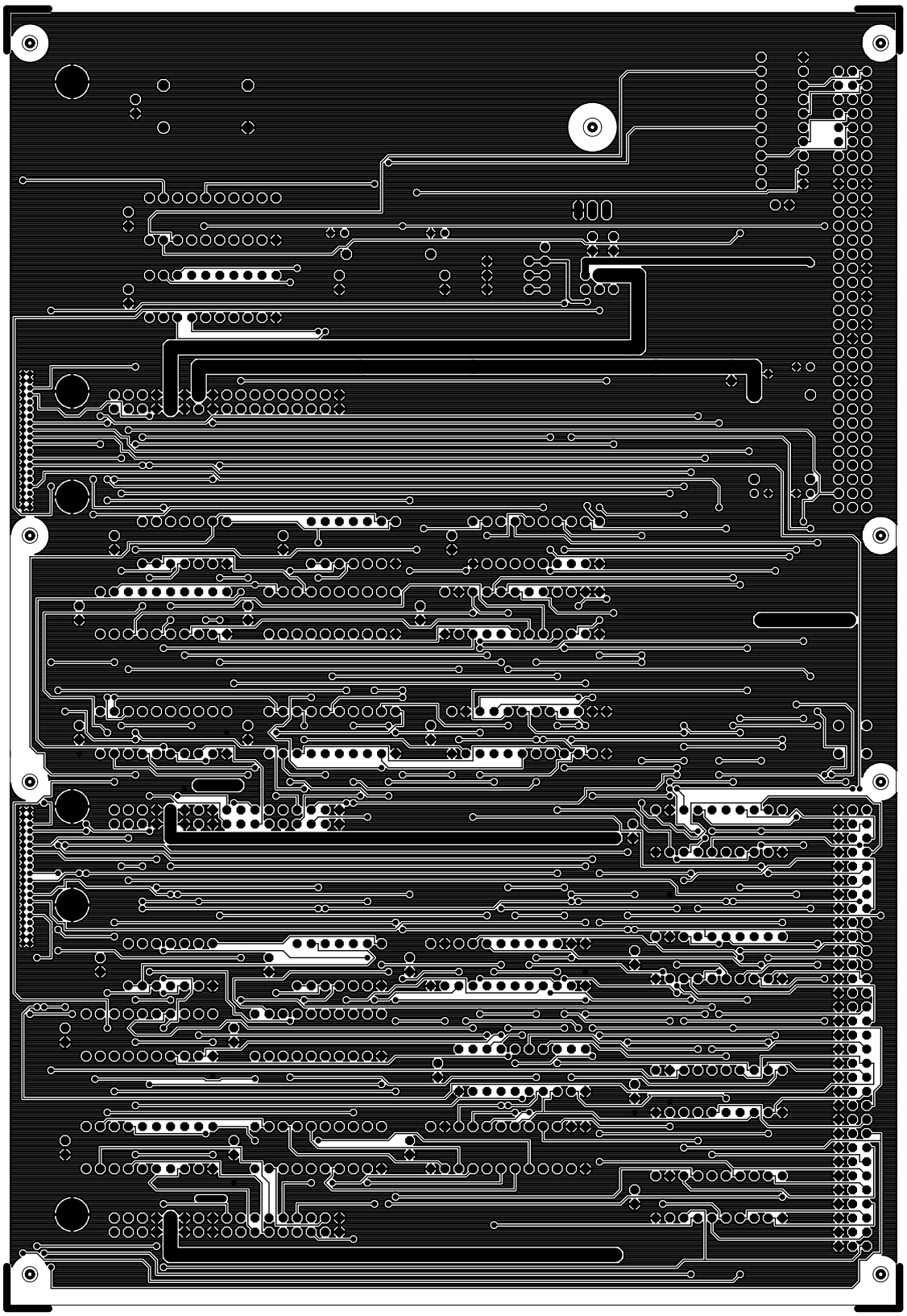


Abbildung A.12: Layout der Lötseite

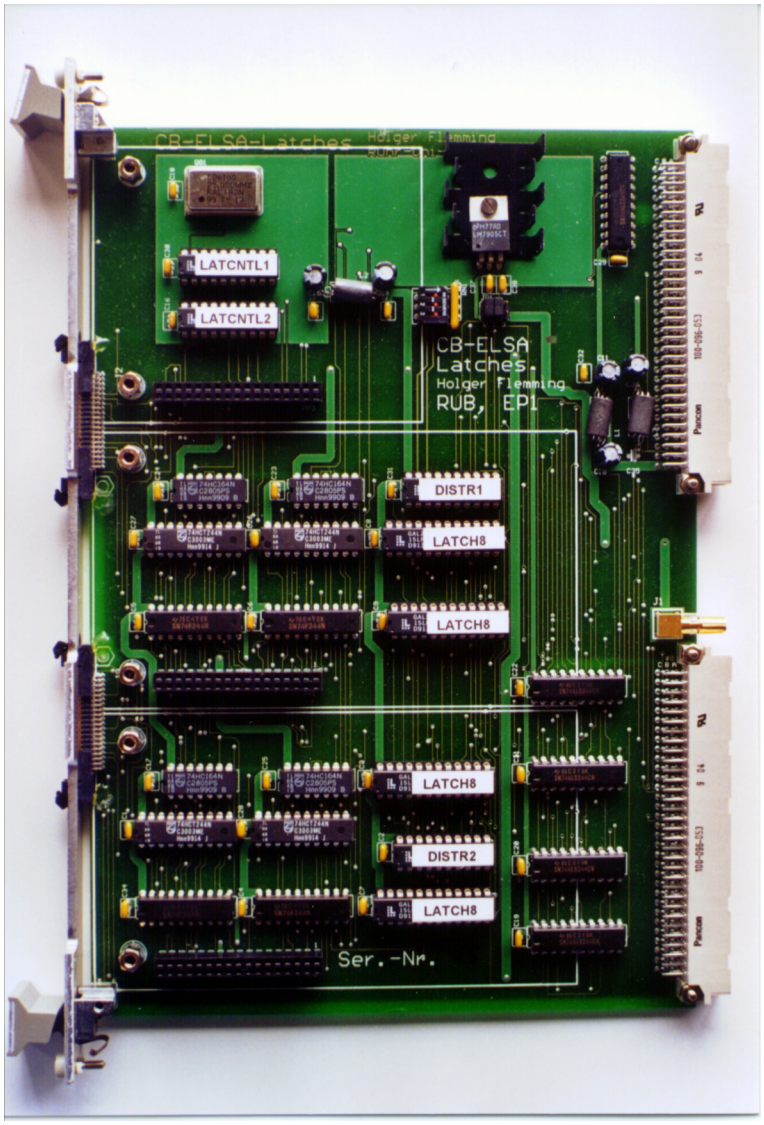


Abbildung A.13: Das Latchmodul von der Bestückungsseite gesehen

A.3 Der Betrieb der Latchmodule

Für den Betrieb der Latches ist ein spezielles VME-Crate notwendig, das zusätzlich zu den Steckern J1 und J2 eine $-5,2$-V-Versorgung über die passenden Siemens-Hochstromstecker zur Verfügung stellt. Die Latchmodule benötigen keinen VME-Controller, lediglich für den Betrieb des Testmustergenerators ist eine spezielle Masterkarte erforderlich, die die dazu benötigten Steuersignale erzeugt.

A.3.1 Die Konfiguration der Latchmodule

Um die korrekte Funktion der Latchmodule zu gewährleisten, müssen auf der Karte einige Einstellungen zur Spannungsversorgung und zur Barrel-Konfiguration vorgenommen werden. Dazu gibt es auf der Karte sechs Pfostenstecker, sowie einen dreifach DIP-Schalter.

Die Betriebsspannungsauswahl

Der Pfostenstecker dient zur Auswahl der $-5,2$-V-Betriebsspannung. Er ist auf der Platine (siehe Bestückungsplan, Abbildung A.10) oben rechts unter dem Spannungsregler angeordnet. Die Steckpositionen sind mit den Ziffern 1 bis 3 gekennzeichnet.
Mit ihm kann zwischen einer direkten Versorgung über die Backplane und einer internen Erzeugung dieser Betriebsspannung aus der -12-V-Versorgung des VME-Bus entschieden werden.
Im Normalfall, wenn die Latchmodule in den dafür vorgesehenen Crates mit einer zusätzlichen $-5,2$-V-Versorgung über die Backplane betrieben werden, sollte diese Spannung auch für die ECL-TTL-Wandler verwendet werden. Dazu ist an dem Pfostenstecker JP1 lediglich auf die Position 3 eine Kurzschlussbrücke zu stecken. Die Positionen 1 und 2 sind unbedingt frei zu halten.
Soll das Modul zu Testzwecken in einem anderen Crate betrieben werden, das nicht über eine passende $-5,2$-V-Versorgung verfügt, kann diese Spannung auch intern von einem Spannungsregler erzeugt werden. Dazu muss er über einen Jumper auf Steckposition 1 mit den $-12\ \mathrm{V}$ des VME-Bus verbunden werden. Mit einen zweiten Jumper auf Position 2 wird die Ausgangsspannung des Spanungsreglers auf die interne $-5,2$-V-Verteilung gegeben. Auf Position 3 sollte zur Sicherheit kein Jumper gesteckt sein.
Bei Betrieb des Latches mit allen drei ECL-TTL-Wandlermodulen kommt es am Spannungsregler zu einer erheblichen Wärmeentwicklung, sodass der Betrieb nur in einem ausreichenden Luftstrom erfolgen sollte.

Die Einstellung der richtigen Signalverteilung

Direkt links neben dem Pfostensteckfeld befindet sich ein dreifach-Dip-Schalter, über den die unterschiedlichen Signalverteilungen ausgewählt werden können, die sich aus einer unterschiedlichen Anzahl von Barrellagen oder dem Unterschied von Up- und Downstreamhälfte ergeben. Die einzelnen Schalter sind auch hier mit den Ziffern 1 bis 3 gekennzeichnet. Mit den Schaltern 2 und 3 legt man die Anzahl der Barrellagen fest. Dabei gibt es folgende Einstellmöglichkeiten:

Anzahl Randlagen	Sw2	Sw3
0	On	On
1	Off	On
2	On	Off
3	Off	Off

Mit dem Schalter 1 kann man die Bitposition einstellen, an der sich die Signale der Randkristalle befinden. Abhängig von der Schalterstellung erfolgt die Signalverteilung bei den Bits 6 bis 13 oder 18 bis 25. Bei Schalterstellung Off werden die Bits 6 bis 13 bearbeitet.

In der derzeit vorgesehenen Triggerverkabelung müssen die beiden Schalterstellungen wie folgt für die angegebenen Barrelsignale angewendet werden:

- **Schalter 1 Off**
  Forwardhälfte, Sektoren 2 bis 33 und Backwardhälfte, Sektoren 34 bis 1
- **Schalter 1 On**
  Backwardhälfte, Sektoren 2 bis 33 und Forwardhälfte, Sektoren 34 bis 1

A.3.2 Betrieb des Testmustergenerators

Tabelle A.1 zeigt die Belegung des VME-Bussteckers J1 des Latchmoduls. Neben den Anschlüssen für die Spannungsversorgung werden vier Leitungen mit Signalen für den Testmustergenerator belegt. **Diese Belegung ist nicht kompatibel zum VME-Standard.** Es ist jedoch ohne Probleme möglich, ein Standard-VME-Crate mit den üblichen Busterminierungen zu benutzen. Voraussetzung ist lediglich, dass sich keine anderen VME-Module in diesem Crate befinden, insbesondere keine VME-Controller. Die Belegung des Steckers ist so gewählt, dass bei einem gemeinsamen Betrieb mit anderen VME-Modulen keine Beschädigungen, weder am Latchmodul, noch an den anderen Modulen auftreten sollten. **Die korrekte Funktion der Latchmodule ist dann jedoch nicht mehr gewährleistet.**

Pin	a	b	c
1	$TestEn$		
2	$TestClk$		
3			
4		D_{In}	
5		D_{Out}	
6			
7			
8			
9	Gnd		Gnd
10			
11	Gnd		
12			
13			
14			
15	Gnd		
16			
17	Gnd		
18			
19	Gnd		
20		Gnd	
21			
22			
23		Gnd	
24			
25			
26			
27			
28			
29			
30			
31	-12 V		+12 V
32	+5 V	+5 V	+5 V

Tabelle A.1: Die Pinbelegung des VME-Bussteckers J1 an den Latchmodulen

Der Testmustergenerator wird mit dem Steuersignal $TestEn$ aktiviert. Im Normalbetrieb *muss* sich diese Leitung auf Low-Pegel befinden. Wenn beim Betrieb der Latches auf den Testmustergenerator verzichtet werden kann, reicht es also aus, sie statisch auf Low zu setzen.
Durch ein High-Signal am $TestEn$-Eingang wird der Testmustergenerator aktiviert. Der Testmustergenerator besteht aus einem 32-Bit-Schieberegister, in das seriell ein Testmuster hineingeschrieben werden kann. Dazu dient das Steuersignal $TestClk$ zusammen mit dem Dateneingang D_{In}. Bei einer positiven Flanke an $TestClk$ werden die Daten des Schieberegisters um ein Bit

in Richtung zu Bit 31 hin weitergeschoben und der Zustand von D_{In} wird in Bit 0 übernommen. Anschliessend wird der Zustand des Gesamtschieberegisters automatisch vom Latch gespeichert und liegt somit statisch an den Datenausgängen des Latchmoduls an.
Um eine Kaskadierung mehrerer Latchmodule zu ermöglichen, wird der Zustand des letzten Bits des Schieberegisters über den Ausgang D_{Out} nach außen geführt. Die Pinbelegung auf dem VME-Busstecker ist so gewählt, dass eine VME-Daisy-Chain-Leitung für das D_{In} und D_{Out} Signal benutzt wird. Somit werden die Schieberegister nebeneinander platzierter Latchmodule automatisch miteinander verkettet. Es ist darauf zu achten, dass die Daisy-Chain-Jumper für das BG0-Signal auf der VME-Backplane nicht gesetzt sind.

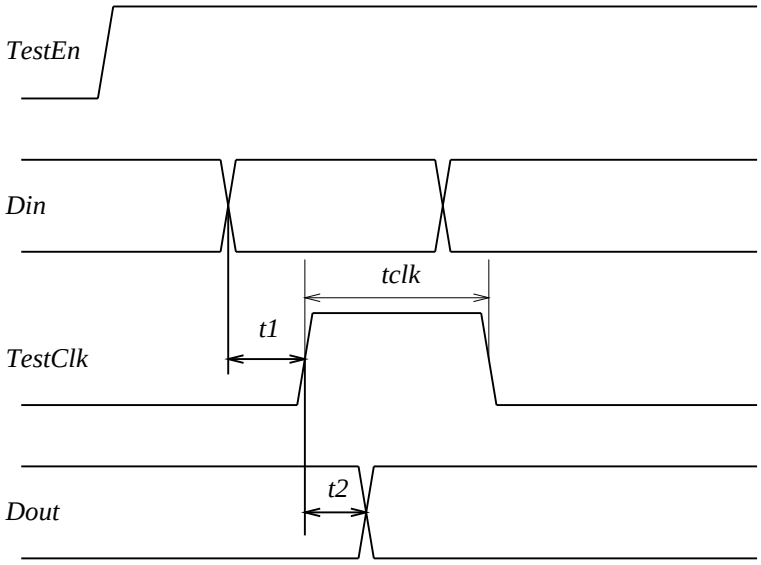


Abbildung A.14: Das Timing der Steuersignale für den Testmustergenerator

Abbildung A.14 stellt das Timing der Steuersignale für den Testmustergenerator graphisch dar. Mit der steigenden Flanke des $TestClk$-Signals wird ein neues Testmuster erzeugt. Die Dauer dieses Signals ist t_{Clk}. Damit das Datenbit von D_{In} korrekt übernommen werden kann, muss es vorher die Zeit t_1 lang stabil sein. Nach der Zeit t_2 liegt das neue Datenbit D_{Out} stabil am Ausgang an. Die Werte für die Zeiten t_{Clk}, t_1 und t_2 können aus der folgenden Tabelle entnommen werden. Alle Angaben sind in ns.

Zeit	min	typ	max
t_{Clk}	160		
t_1	40		
t_2			40

A.3.3 Die Ansteuerung über das ECL-Interface

Im Normalbetrieb wird das Latchmodul über einen ECL-Steuerport angesteuert. Die Tabelle A.2 zeigt die Belegung des Steuerports. Die beiden Signale $Gate$ und $Reset$ sind die Hauptsteuersignale des Latch-Moduls in diesem Betriebsmodus. Mit $Reset$ können alle Speicher gelöscht werden. Das Latchmodul befindet sich danach im Grundzustand. Während das $Gate$-Signal aktiv ist, werden die an den Eingängen anliegenden Daten gespeichert.

Reihe	Signal	
1	Gate	Eingang
2	Reset	Eingang
3		
4		
5	Flag	Ausgang
6	Busy	Ausgang
7		
8		
9	OR	Ausgang
10		
11		
12		
13		
14		
15		
16		
17		

Tabelle A.2: Belegung des ECL-Steuerports

Um den momentanen Zustand des Moduls kontrollieren zu können, stehen die Signale $Flag$ und $Busy$ zur Verfügung. $Busy$ zeigt an, ob das Modul betriebsbereit ist. Ist dieser Ausgang aktiv, dann befindet sich das Modul zur Zeit im Testmodus und kann nicht angesprochen werden. Erst wenn der Ausgang inaktiv ist, ist das Modul betriebsbereit.
Der $Flag$-Ausgang zeigt an, ob Daten im Modul gespeichert sind. Ist der Ausgang aktiv, sind Daten im Modul gespeichert.
Der Or-Ausgang liefert eine Oder-Verküpfung aller 32 Dateneingänge.

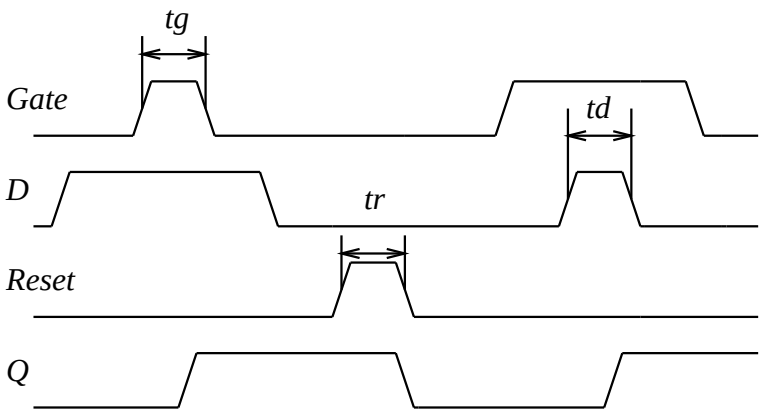


Abbildung A.15: Das Timing der ECL-Steuersignale für die Latches

Abbildung A.15 zeigt das Timing der ECL-Steuersignale. Die eingezeichneten Zeitparame-

ter wurden an einem Prototypen gemessen. Die Messergebnisse sind in der folgenden Tabelle dargestellt:

Zeit	min	typ	max
t_g	7,21		
t_d	7,3		
t_r	6,6		

Alle Angaben sind in ns.

A.4 Die Logikgleichungen für die verwendeten Gals

A.4.1 Die Speicherbausteine LATCH8

```
Projekt : Neue Latches fr den CB-ELSA-Trigger

Die hiermit beschriebenen GALS bernehmen die Aufgabe der Speicherelemente.


%ID

        LAT8_10

%TYP

        gal20v8

%PINS

  CLK D0 D1 D2 D3 D4 D5 D6 D7 LAT CLR
  OE  NC Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7  NC

%LOGIC
        Q0.OE = VCC;
        Q0 = !CLR * Q0 + LAT * D0;

        Q1.OE = VCC;
        Q1 = !CLR * Q1 + LAT * D1;

        Q2.OE = VCC;
        Q2 = !CLR * Q2 + LAT * D2;

        Q3.OE = VCC;
        Q3 = !CLR * Q3 + LAT * D3;

        Q4.OE = VCC;
        Q4 = !CLR * Q4 + LAT * D4;

        Q5.OE = VCC;
        Q5 = !CLR * Q5 + LAT * D5;

        Q6.OE = VCC;
```

```
        Q6 = !CLR * Q6 + LAT * D6;

        Q7.OE = VCC;
        Q7 = !CLR * Q7 + LAT * D7;

%END
```

A.4.2 Die Signalverteilung DISTR1

```
Projekt : Neue Latches fr den CB-ELSA-Trigger

Das Distributionsgal 1 ist aktiv, wenn der Datenkanal 1 ( Kristalltyp I )
an Latchkanal 31 anliegt. Erzeugt werden mu dann die richtige Zuordnung
fuer die Kanle 6 bis 11. Zur Verfgung stehen dafr die Kanle 13 und 12
als die beiden zu Kristall

      Eingangskanle       Ausgangskanle

      +-----+              +--+--+
      |  9  |              | 6| 7|
      +-----+              +--+--+
      | 10  |              | 8| 9|
      +-----+     ==>      +--+--+
      | 11  |              |10|11|
      +--+--+              +--+--+
      |12|13|              |12|13|
      +--+--+              +--+--+

Die Mglichen Konfigurationen werden dargestellt, indem die Matrix der
Ausgangskanle gezeigt wird und dort die Nummern der Eingangskanle eingetragen
werden:

     3 Randlagen       2 Randlagen     1 Randlage       keine Randlage

          3                 2               1                0

       +--+--+           +--+--+         +--+--+          +--+--+
       | 9| 9|           |10|10|         |11|11|          |12|13|
       +--+--+           +--+--+         +--+--+          +--+--+
       |10|10|           |10|10|         |11|11|          |12|13|
       +--+--+           +--+--+         +--+--+          +--+--+
       |11|11|           |11|11|         |11|11|          |12|13|
       +--+--+           +--+--+         +--+--+          +--+--+

Wenn Conf1 low ist, dann werden die DAtensignale einfach 1:1 durchgeschleift.

%ID
        DIST1_10

%TYP

        gal16v8

%PINS
```

```
        NC    D13 D12 D11 D10 D9  D8  D7    D6
        CONF1 Q6  Q7  Q8  Q9  Q10 Q11 CONF3 CONF2

%LOGIC


        Q6 = D6 * !CONF1
           + D9 * CONF1 * CONF2 * CONF3
           + D10 * CONF1 * !CONF2 * CONF3
           + D11 * CONF1 * CONF2 * !CONF3
           + D12 * CONF1 * !CONF2 * !CONF3;

        Q7 = D7 * !CONF1
           + D9 * CONF1 * CONF2 * CONF3
           + D10 * CONF1 * !CONF2 * CONF3
           + D11 * CONF1 * CONF2 * !CONF3
           + D13 * CONF1 * !CONF2 * !CONF3;


        Q8 = D8 * !CONF1
           + D10 * CONF1 * CONF3
           + D11 * CONF1 * CONF2 * !CONF3
           + D12 * CONF1 * !CONF2 * !CONF3;

        Q9 = D9 * !CONF1
           + D10 * CONF1 * CONF3
           + D11 * CONF1 * CONF2 * !CONF3
           + D13 * CONF1 * !CONF2 * !CONF3;

       Q10 = D10 * !CONF1
           + D11 * CONF1 * CONF2
           + D11 * CONF1 * CONF3
           + D12 * CONF1 * !CONF2 * !CONF3;

       Q11 = D11 * !CONF1
           + D11 * CONF1 * CONF2
           + D11 * CONF1 * CONF3
           + D13 * CONF1 * !CONF2 * !CONF3;

%END
```

A.4.3 Die Signalverteilung DISTR2

```
Projekt : Neue Latches fr den CB-ELSA-Trigger

Das Distributionsgal 1 ist aktiv, wenn der Datenkanal 1 ( Kristalltyp I )
an Latchkanal 31 anliegt. Erzeugt werden mu dann die richtige Zuordnung
fuer die Kanle 6 bis 11. Zur Verfgung stehen dafr die Kanle 13 und 12
als die beiden zu Kristall

      Eingangskanle       Ausgangskanle

      +--+--+              +--+--+
```

```
|18|19|              |18|19|
+--+--+              +--+--+
| 20  |              |20|21|
+-----+     ==>      +--+--+
| 21  |              |22|23|
+-----+              +--+--+
| 22  |              |24|25|
+-----+              +--+--+
```

Die Mglichen Konfigurationen werden dargestellt, indem die Matrix der Ausgangskanle gezeigt wird und dort die Nummern der Eingangskanle eingetragen werden:

```
     3 Randlagen       2 Randlagen     1 Randlage       keine Randlage

          3                 2               1                0

       +--+--+           +--+--+         +--+--+          +--+--+
       |20|20|           |20|20|         |20|20|          |18|19|
       +--+--+           +--+--+         +--+--+          +--+--+
       |21|21|           |21|21|         |20|20|          |18|19|
       +--+--+           +--+--+         +--+--+          +--+--+
       |22|22|           |21|21|         |20|20|          |18|19|
       +--+--+           +--+--+         +--+--+          +--+--+
```

Wenn Conf1 high ist, dann werden die Datensignale einfach 1:1 durchgeschleift.

```
%ID
        DIST1_10

%TYP

        gal16v8

%PINS

        NC    D25 D24 D23 D22 D21 D20 D19   D18
        CONF1 Q20 Q21 Q22 Q23 Q24 Q25 CONF3 CONF2

%LOGIC

        Q20 = D20 * CONF1
            + D20 * !CONF1 * CONF2
            + D20 * !CONF1 * CONF3
            + D18 * !CONF1 * !CONF2 * !CONF3;

        Q21 = D21 * CONF1
            + D20 * !CONF1 * CONF2
            + D20 * !CONF1 * CONF3
            + D19 * !CONF1 * !CONF2 * !CONF3;

        Q22 = D22 * CONF1
            + D21 * !CONF1 * CONF3
            + D20 * !CONF1 * CONF2 * !CONF3
```

```
            + D18 * !CONF1 * !CONF2 * !CONF3;

        Q23 = D23 * CONF1
            + D21 * !CONF1 * CONF3
            + D20 * !CONF1 * CONF2 * !CONF3
            + D19 * !CONF1 * !CONF2 * !CONF3;

        Q24 = D24 * CONF1
            + D22 * !CONF1 * CONF2 * CONF3
            + D21 * !CONF1 * !CONF2 * CONF3
            + D20 * !CONF1 * CONF2 * !CONF3
            + D18 * !CONF1 * !CONF2 * !CONF3;

        Q25 = D25 * CONF1
            + D22 * !CONF1 * CONF2 * CONF3
            + D21 * !CONF1 * !CONF2 * CONF3
            + D20 * !CONF1 * CONF2 * !CONF3
            + D19 * !CONF1 * !CONF2 * !CONF3;


%END
```

A.4.4 Die Ablaufsteuerung LATCNTL1

```
Projekt : Neue Latches fr den CB-ELSA-Trigger

Steuerlogik


%ID

        L_CNTL1

%TYP

        gal16v8

%PINS

        CLK TSTCLK TST2 NC NC NC      NC     NC      NC
        NC  F1     F2   F3 F4 !TRESET CPULSE !TESTEN !INEN

%LOGIC

        F1 := TST2;

        F2 := F1;

        F3 := TSTCLK;

        F4 := F3;

        TRESET = F1 * !F2;
```

```
        CPULSE = F3 * !F4;

        TESTEN = TST2;

        INEN = !TST2;

%END
```

A.4.5 Die Ablaufsteuerung LATCNTL2

```
Projekt : Neue Latches fr den CB-ELSA-Trigger

Steuerlogik


%ID

        L_CNTL2

%TYP

        gal16v8

%PINS

        CLK CPULSE GATE RESET TST2   NC   NC   NC   NC
        NC  C0     C1   C2    LRESET TCLK LCLK FLAG BUSY

%LOGIC

        C0 := CPULSE
            + !C0 * C1
            + !C0 * C2;

        C1 := !C1 * C0
            + C1 * !C0;

        C2 := !C2 * C0 * C1
            + C2 * !C0
            + C2 * !C1;

        LRESET = RESET + C0 * !C1 * !C2;

        TCLK = C0 * C1 * !C2;

        LCLK = GATE + C0 * !C1 * C2;

        FLAG = FLAG * !LRESET + LCLK;

        BUSY = TST2;

%END
```

A.5 Korrektur

Während der Testphase des Clustertriggers an CB-ELSA wurde ein Fehler in der Schaltung der Latchmodule erkannt, der dazu führt, dass die Daisy-Chain-Funktion der Testmustergeneratoren nicht korrekt arbeitet. Da die 25-MHz-Oszillatoren auf den Modulen nicht synchron laufen, kann es vorkommen, dass die positiven Flanken auf dem internen Taktsignal für die Schieberegister um bis zu ± 40 ns auseinander liegen. Dies wiederum kann dazu führen, dass der Zustand am D_{out}-Pin des einen Moduls sich bereits geändert hat, bevor am nächsten Modul das Takt-Signal erzeugt wird, was wiederum dazu führt, dass ein falsches Bit in das nächste Modul übernommen wird.
Um diesen Fehler zu beseitigen müssen die folgenden Änderungen an allen Latchmodulen durchgeführt werden:

1. Eine zusätzliche Verbindung zwischen IC5 Pin 2 und IC16 Pin 6 muss hergestellt werden.

2. Im Quellcode für das GAL *Latcntl2* muss der zusätzliche Eingangspin $TSTCLK$ definiert werden. Der Eintrag für Pins lautet jetzt:

```
%PINS

        CLK CPULSE GATE RESET TST2   TSTCLK NC   NC   NC
        NC  C0     C1   C2    LRESET TCLK   LCLK FLAG BUSY
```

3. Die Logikgleichung für den Pin $TCLK$ im Quellfile für das GAL *Latcntl2* muss geändert werden. Sie muss lauten:

```
        TCLK = TSTCLK;
```

Anhang B

Schaltungs- und Layoutbeschreibung des Zellularlogik-ASICs

B.1 Die Logikzelle

Der Zellularlogikchip enthält 16 Zellen. Jede Zelle wiederum ist aus mehreren Funktionsblöcken aufgebaut:

1. Speicherlogik
2. Markierungslogik
3. Maskierungslogik
4. Projektionslogik

B.1.1 Die Speicherlogik

Die Schaltung

Abbildung B.1 zeigt das Schaltbild der Speicherlogik. Die beiden RS-Flipflops, I24 und I29, speichern die Informationen über Treffer (I24) und über externe Markierungen. Jedes dieser Flipflops kann auf zwei Weisen gesetzt werden, entwerder über das Set-Signal, bei dem alle Zellen parallel gesetzt werden oder über das $Testset$-Signal, bei dem nur eine Zelle selektiv gesetzt wird. Welches Signal gesetzt wird, hängt vom Zustand des $SetSel$-Signals ab. Bei $SetSel = 1$ wird das Trefferflipflop, bei $SetSel = 0$ das externe Markierungsflipflop gesetzt. Die Löschung geschieht bei beiden Flipflops unterschiedlich.
Die Setzlogik des Trefferflipflops besteht aus drei NAND-Gattern, die in zwei Stufen aufgebaut sind. In der ersten Stufe werden mit zwei NAND-Gattern die beiden Bedingungen realisiert, unter denen das Treffer-Flipflop mit den Set- oder mit dem $TestSet$-Signal gesetzt werden kann. In einem dreifach NAND-Gatter werden dazu das $TestSet$-Signal, das $CellSel$-Signal, das anzeigt, ob diese Zelle selektiv angesprochen wird, und das $SetSel$-Signal NAND-verknüpft. Für die zweite Bedingung befindet sich auf gleicher Ebene ein weiteres NAND Gatter mit vier

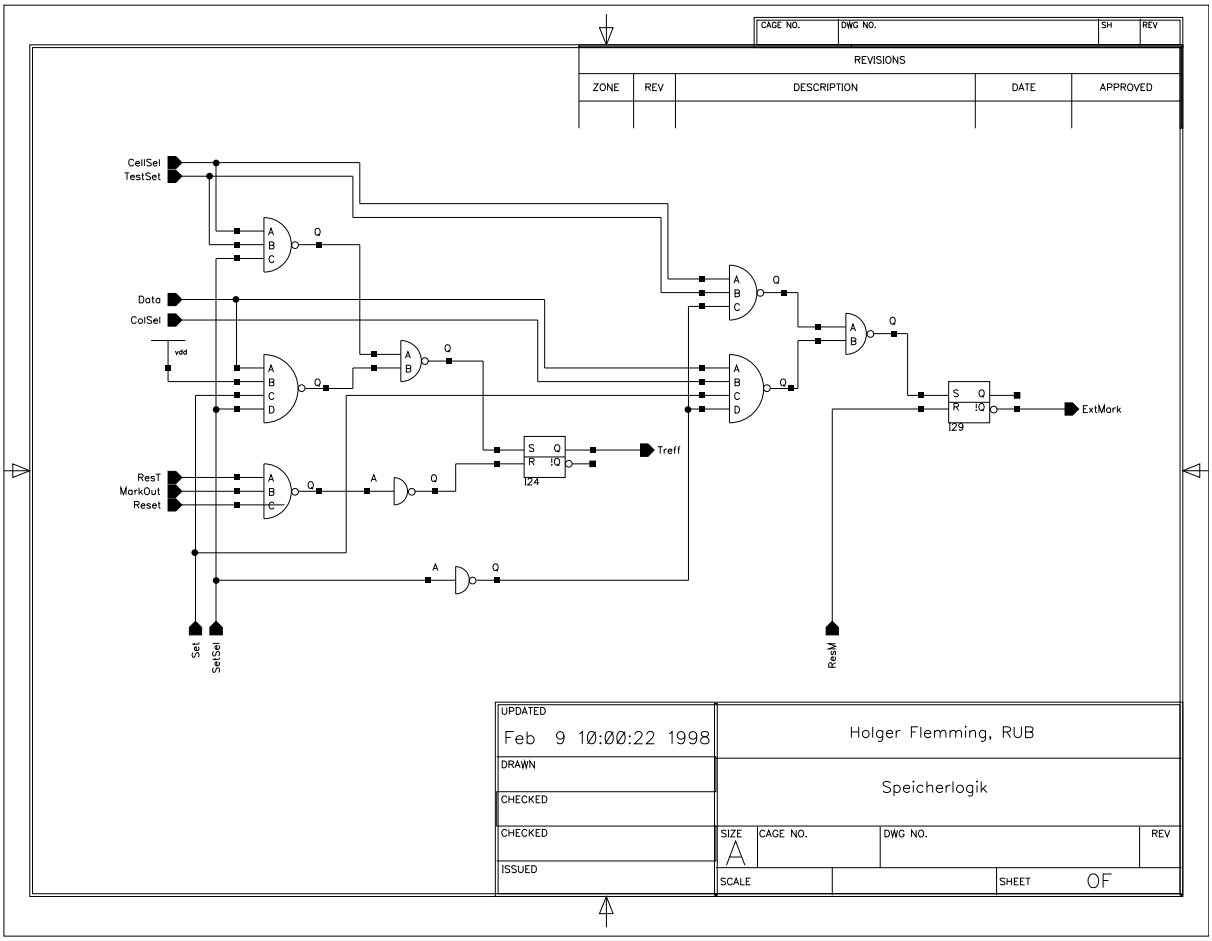


Abbildung B.1: Die Speicherlogik. In den beiden Flipflops werden Trefferinformation und externe Markierungsinformationen gespeichert

Eingängen, von denen jedoch einer fest auf High-Pegel gelegt ist[1]. Mit den verbleibenden drei Eingängen werden das $Data$-Signal, das Set-Signal und das $SetSel$-Signal NAND-verknüpft. Diese beiden Signale werden dann in der zweiten Stufe mit dem folgenden NAND-Gatter in negativer Logik NOR-verknüpft und auf den S-Eingang des Flipflops gegeben, sodass der S-Eingang aktiv wird, wenn eine der beiden Bedingungen erfüllt ist.
Für den Lösch-Eingang wurde ein spezielles Gatter aufgebaut, das die Eingänge $ResT$, $MarkOut$ und $Reset$ zu der Booleschen Gleichung

$$OUT = \overline{ResT \cdot MarkOut + Reset}$$

verknüpft. Dieses Signal wird invertiert und auf den R-Eingang des Flipflops gegeben.
Für das externe Markierungsflipflop wird zunächst das $SetSel$-Signal invertiert. Damit steht ein

[1]Hierfür gibt es keine besonderen technischen Gründe. Vielmehr wurde die Zelle im Laufe der Entwicklung verändert, wodurch der vierte Eingang nicht mehr benötigt wurde. Um größere Änderungen im Layout zu vermeiden, wurde der nicht mehr benötigte Eingang auf High gelegt.

Signal zur Verfügung, das bei $SetSel = 0$ aktiv wird. Die Verarbeitung der Setz-Bedingungen verläuft wieder in zwei Ebenen. In einem dreifach-NAND werden wieder $CellSel$, $TestSet$ und nun das invertierte $SetSel$-Signal NAND-verknüpft. Parallel dazu wird in einem vierfach-NAND das $Data$-Signal, das Set-Signal, das invertierte $SetSel$-Signal und das $ColSel$-Signal verknüpft, wobei hier das $ColSel$-Signal für die Verknüpfung herangezogen wird, da die externen Markierungsflipflops nur selektiv in einer Spalte gesetzt werden dürfen. Der R-Eingang des Flipflops wird direkt mit dem $ResM$-Signal verbunden. Mit der Speicherlogik werden somit die folgenden Booleschen Gleichungen realisiert:

$$Treff.S = TestSet \cdot CellSel \cdot SetSel + Set \cdot Data \cdot SetSel \quad \text{(B.1)}$$

$$Treff.R = ResT \cdot MarkOut + Reset \quad \text{(B.2)}$$

$$ExtMark.S = TestSet \cdot CellSel \cdot \overline{SetSel} + Set \cdot Data \cdot ColSel \cdot \overline{SetSel} \quad \text{(B.3)}$$

$$ExtMark.R = ResM \quad \text{(B.4)}$$

Das Layout

Abbildung B.2 zeigt das entsprechende Layout der Speicherlogik. Der aktive Bereich befindet sich im unteren Teil der Abbildung. Da pMOS-Transistoren in einer N-Wanne eingelagert sein müssen, die einen relativ großen Mindestabstand von nMOS-Transistoren erfordert, werden die pMOS-Transistoren zusammengefaßt und in einer gemeinsamen Wanne plaziert. Etwas unterhalb bzw. oberhalb dieser Wanne befinden sich die zu den pMOS-Transistoren komplementären nMOS-Transistoren.
Die eigentlichen Transistoren erkennt man an den grünen Diffusionsbereichen. Unten rechts sieht man das externe Markierungsflipflop, links daneben, oberhalb des Markers „Treff“, das Trefferflipflop. Um die Flipflops herum sind die Logikgatter angeordnet, über die sie angesteuert werden. Deutlich zu erkennen ist auch in der N-Wanne die Wannenkontaktierung und darunter die Substratkontaktierung, die dafür sorgen, dass sich das Halbleitermaterial, in dem sich die FETs befinden, auf Masse-Potential liegt und somit die Drain-Substrat-, bzw. Source-Substrat-Dioden in Sperrrichtung vorgespannt sind. Die Größe des dargestellen Feldes beträgt etwa 100 μm mal 50 μm.

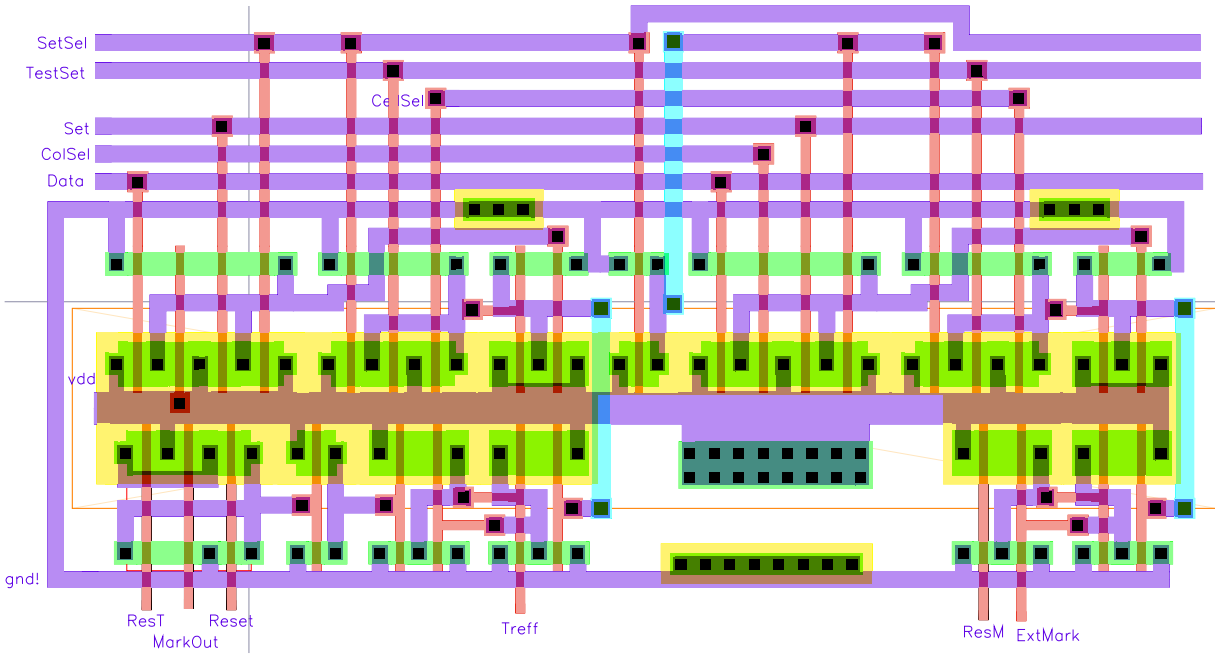


Abbildung B.2: Die Speicherlogik. Die Abbildung zeigt das Halbleiterlayout für die in Abbildung B.1 dargestellte Schaltung

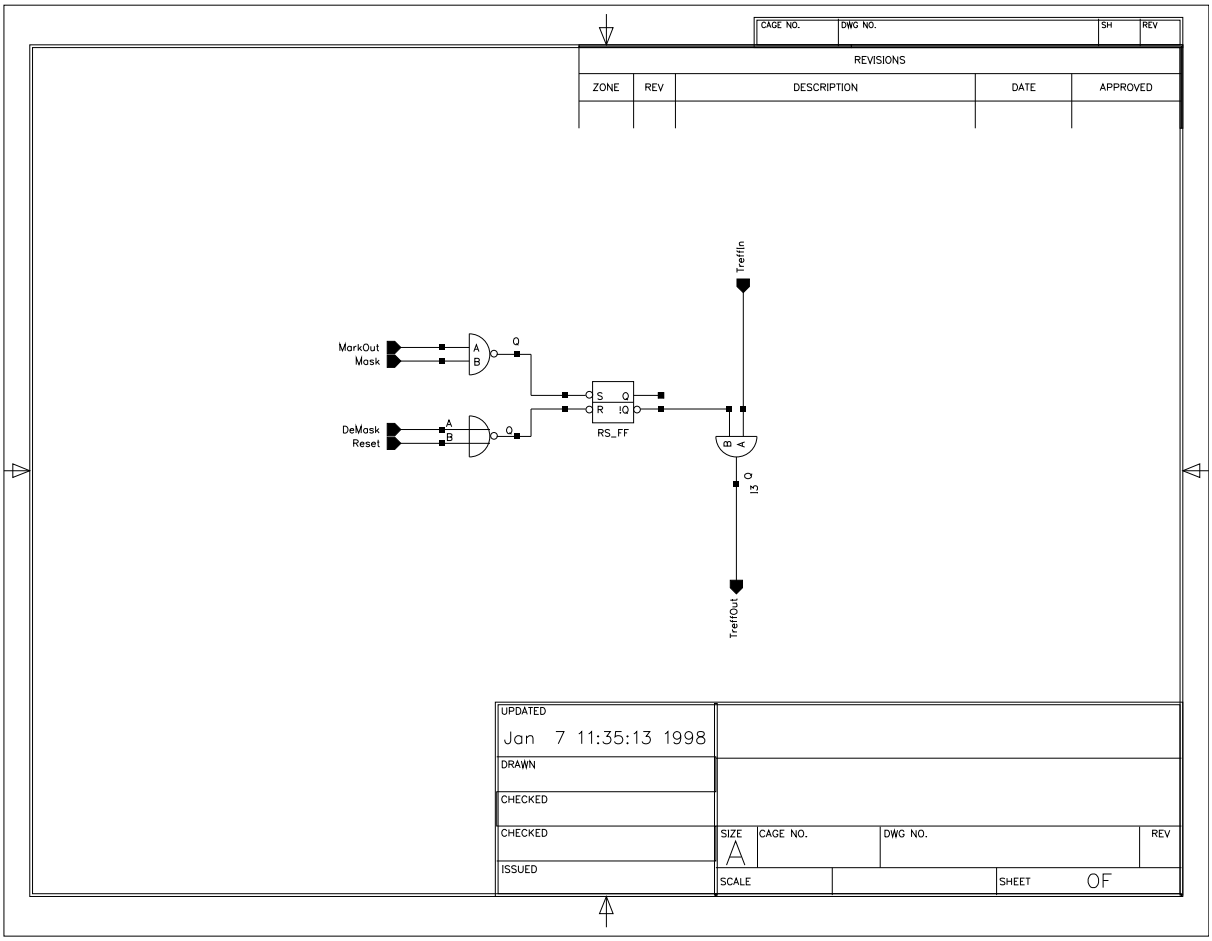


Abbildung B.3: Die Maskierungslogik. Sie dient dazu, Trefferbits temporär auszublenden

B.1.2 Die Maskierungslogik

Die Schaltung

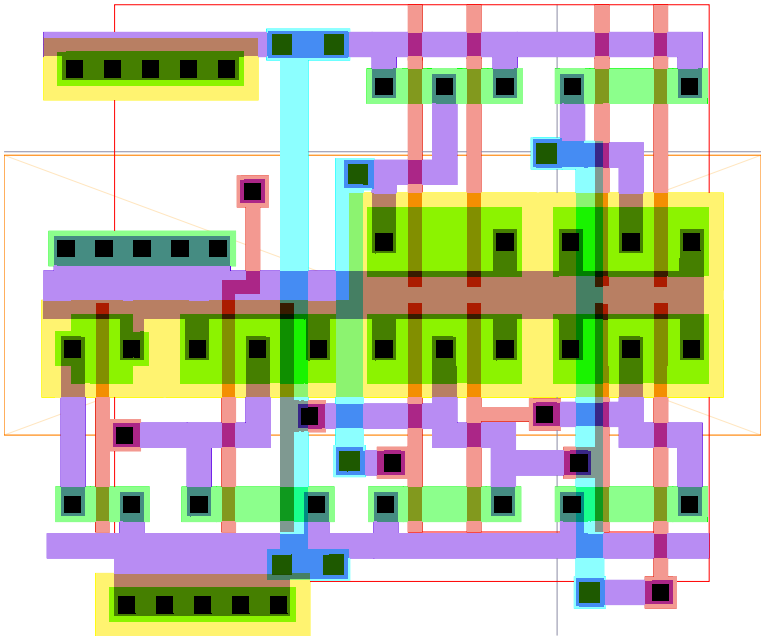
Abbildung B.3 zeigt die Schaltung der Maskierungslogik. Sie dient dazu, das Trefferbit dieser Zelle, das über den Eingang $TreffIn$ in die Maskierungseinheit kommt, am Ausgang $TreffOut$ zeitweise verdecken zu können. Ob das Bit maskiert werden soll, wird in dem RS-Flipflop gespeichert. Gesetzt wird das Flipflop, wenn das $MarkOut$- und das $Mask$-Signal aktiv sind. Beide Signale werden über ein NAND-Gatter verknüpft. Gelöscht wird das Flipflop mit dem $DeMask$-Signal oder dem $Reset$-Signal. Die eigentliche Maskierung geschieht mit dem AND-Gatter I3. Die Logik-Gleichungen, die mit dieser Schaltung realisiert wurden sehen wie folgt aus:

$$Mask.S = MarkOut \cdot Mask \quad \text{(B.5)}$$

$$Mask.R = DeMask + Reset \quad \text{(B.6)}$$

$$TreffOut = TreffIn \cdot \overline{Mask} \quad \text{(B.7)}$$

Abbildung B.4: Das Layout für die Maskierungslogik



Das Layout

In Abbildung B.4 ist das Layout für die Maskierungslogik dargestellt. Unten rechts erkennt man wieder das RS-Flipflop zur Speicherung des Maskierungszustandes. Es besteht aus zwei NAND-Gattern, die Rückgekoppelt sind. Über dem Flipflop befindet sich rechts das NAND- und links das NOR-Gatter, über die die Eingänge angesteuert werden. Links neben dem Flipflop ist das AND-Gatter platziert, das die eigentliche Maskierung ausführt. Das AND-Gatter ist als Kombination eines NAND-Gatters und eines Inverters realisiert. Die Größe des Maskierungsteils beträgt etwa 43 μm mal 37 μm.

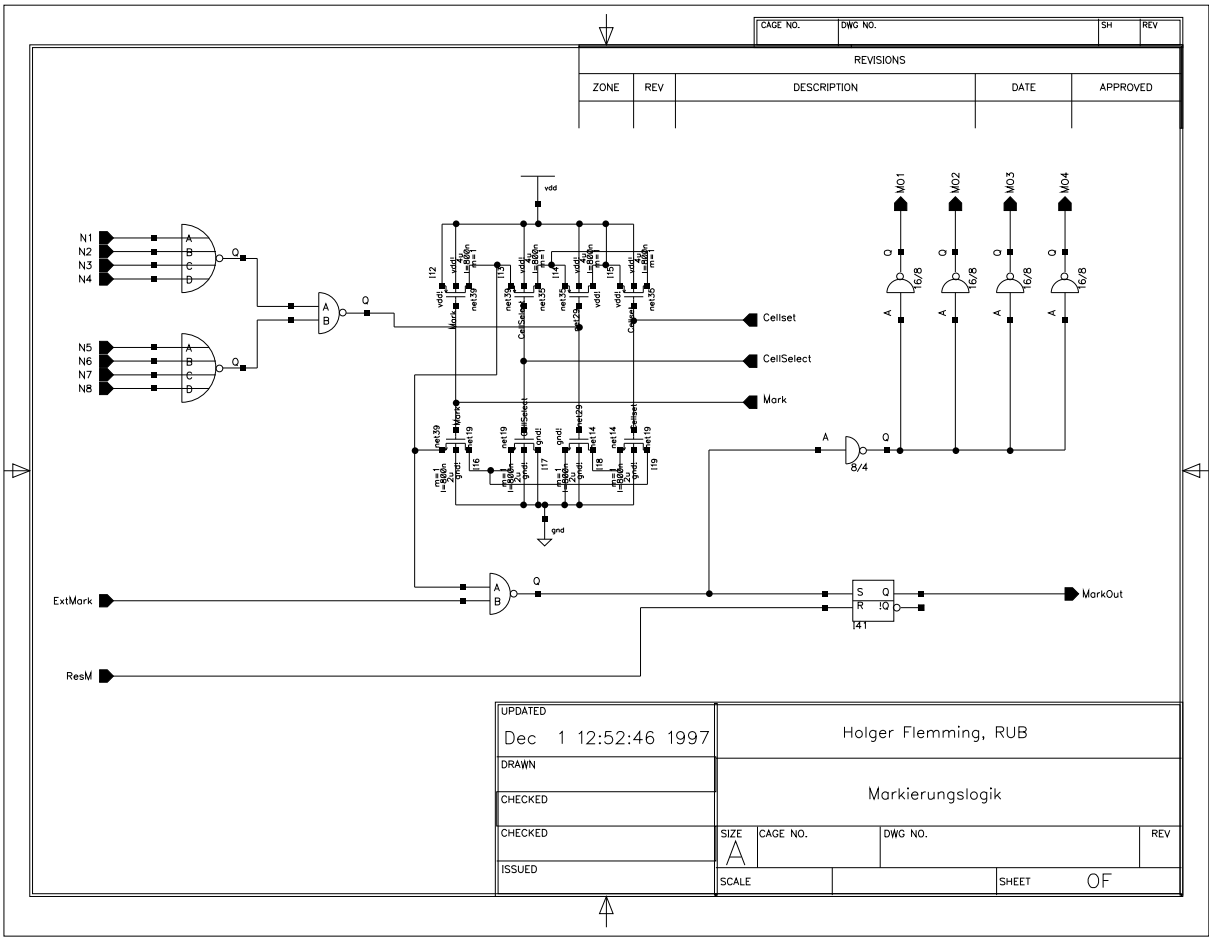


Abbildung B.5: Die Markierungslogik. Sie enthält alle Komponenten, die für die Kommunikation mit den Nachbarzellen und den Markierungsvorgang notwendig sind

B.1.3 Die Markierungslogik

Die Schaltung

Der zentrale Teil der Markierungslogik, die in Abbildung B.5 dargestellt ist, ist eine komplexere Schaltung, die aus vier pMOS- und vier nMOS-Transistoren besteht. Die Gateanschlüsse von jeweils zwei Transistoren sind miteinander verbunden. An den vier Eingängen dieser Schaltung liegen die Signale $Cellset$, $CellSelect$, $Mark$ und eine Oder-Verknüpfung aus den 8 Eingängen, auf die die Markierungsinformation von den acht Nachbarn gelangt, (Im Folgenden einfach N genannt.) an. Diese vier Signale werden mit der Schaltung logisch gemäß Gleichung B.8 verknüpft.

$$Out = \overline{Mark \cdot (CellSelect + N \cdot CellSet)} \tag{B.8}$$

Damit der Ausgang aktiv wird, muß also auf jeden Fall $Mark$ aktiv sein. Ist dies der Fall, gibt es zwei Möglichkeiten, das Ausgangssignal zu aktivieren. Entweder muss der Eingang $CellSelect$ aktiv sein oder es muss $CellSet$ aktiv sein und von mindestens einem der 8 Nachbarn ein Mar-

kierungssignal kommen. Der Ausgang dieses Logikgatters ist Low-aktiv. Somit kann mit dem folgenden NAND-Gatter dieses Signal in negativer Logik mit dem $ExtMark$-Signal NOR-Verknüpft werden, um mit diesem gemeinsamen Signal das Markierungsflipflop zu setzen. Zudem wird dieses Signal auch über Treiber an die 8 Nachbarn weitergegeben, wobei zwei Nachbarn von einem Treiber[2] bedient werden. Das Ausgangssignal des Markierungsfliplops wird als $MarkOut$-Signal nach außen zur Verfügung gestellt. Die Funktion der Markierungslogik lässt sich in den folgenden Logikgleichungen ausdrücken:

$$MarkOut.S = \overline{ExtMark} + Mark \cdot \left(CellSelect + CellSet \cdot \sum_{i=1}^{8} N_i \right) \quad \text{(B.9)}$$

$$MarkOut.R = ResM \quad \text{(B.10)}$$

$$MO_i = MarkOut.S \quad \text{(B.11)}$$

Das Layout

Das Layout der Markierungslogik ist in Abbildung B.6 dargestellt. Unten links nehmen die beiden vierfach-NOR-Gatter und das NAND-Gatter, die für die Oder-Verknüpfung der acht Markierungseingänge benötigt werden, einen Großteil des Platzes ein. Rechts daneben befindet sich das Gatter, das die logische Verknüpfung nach Gleichung B.8 realisiert, und ein NAND-Gatter. Darüber ist das Markierungsflipflop und links daneben die Inverter untergebracht, die das Markierungssignal so weit treiben, dass es zu den Nachbarzellen weitergeleitet werden kann.

[2] die W/L-Verhältnisse der Transistoren sind: pMos: 16μ / 0,8μ, nMOS: 8μ / 0,8μ

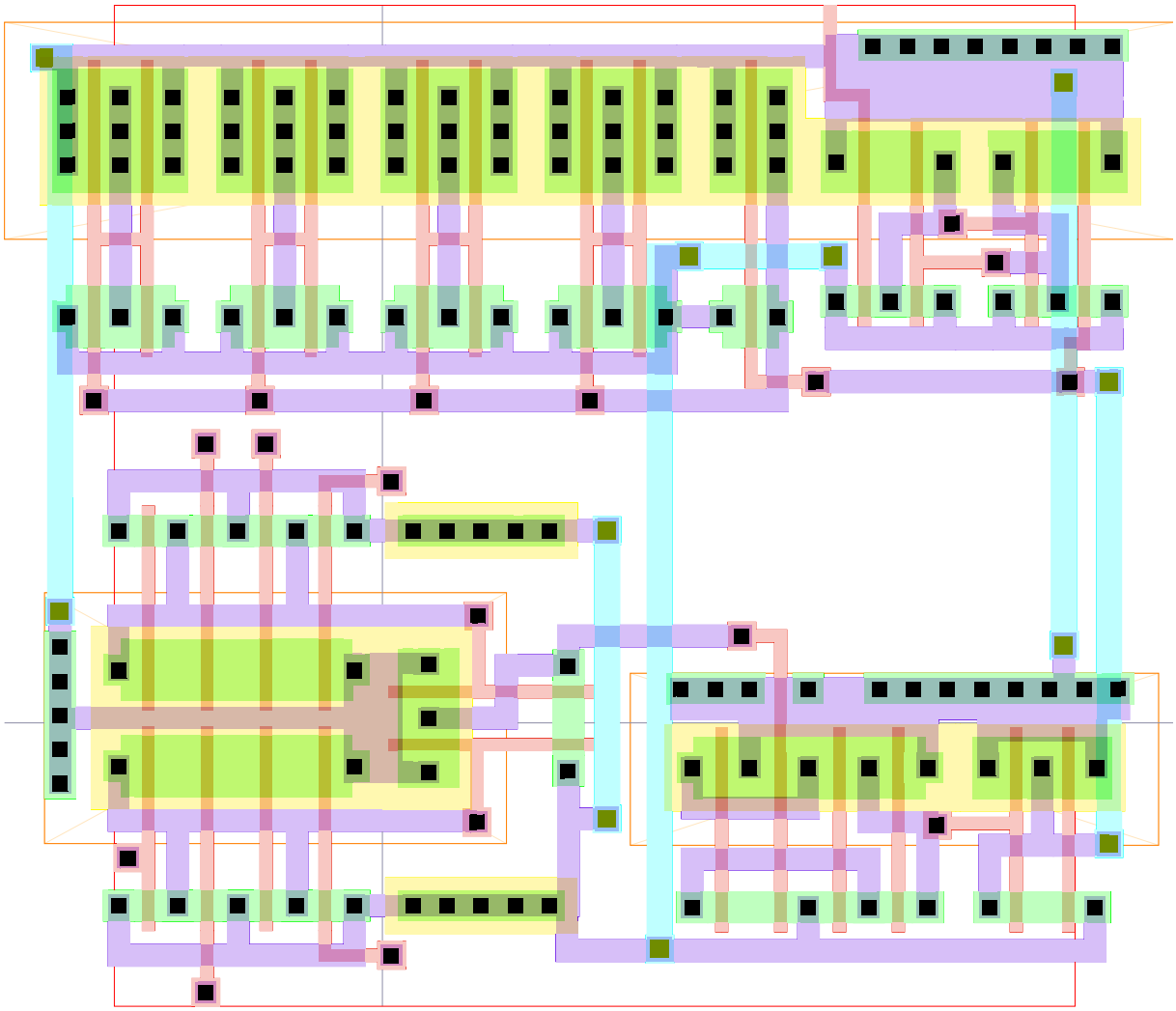


Abbildung B.6: Das Layout der Markierungslogik

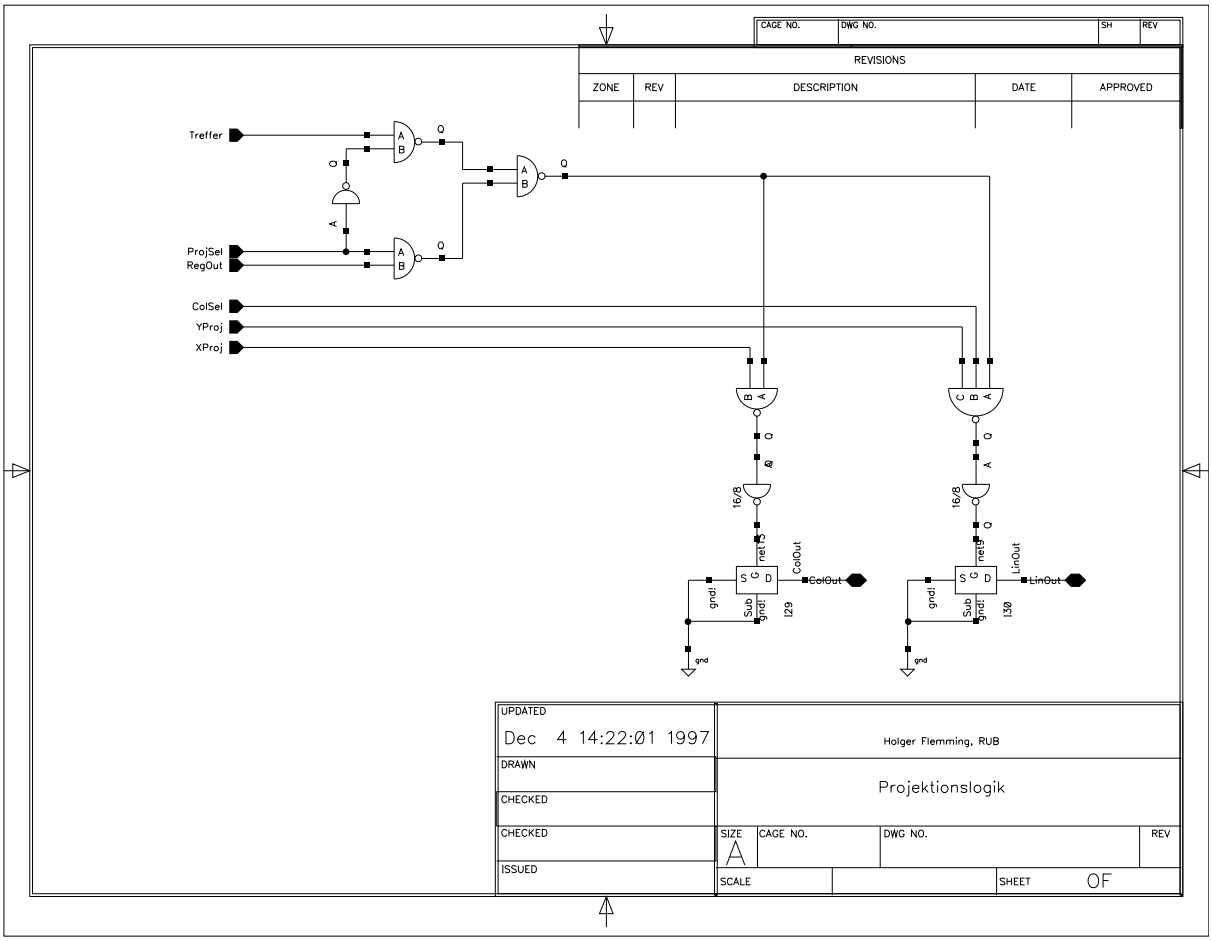


Abbildung B.7: Die Projektionslogik. Dargestellt sind die Projektionsauswahl im oberen Teil und im unteren Teil die Treiber und die Projektionstransistoren

B.1.4 Die Projektionslogik

Die Schaltung

Über die Projektionslogik, die in Abbildung B.7 dargestellt ist, gelangen die Informationen aus der Zelle nach außen. Zwei verschiedene binäre Informationen, die an den Eingängen $Treffer$ und $RegOut$ anliegen, können projiziert werden. Die drei NAND-Gatter und der Inverter bilden einen Multiplexer, mit dem zwischen diesen beiden Eingangssignalen ausgewählt wird. Ist $ProjSel = 0$ wird das $Treffer$-Signal projiziert, ist dagegen $ProjSel = 1$ wird das $RegOut$-Signal projiziert. Die beiden folgenden NAND-Gatter bestimmen, wann projiziert wird. Das zweifach-NAND-Gatter verknüpft das ausgewählte Siganal mit dem $XProj$-Signal. Wenn $XProj$ aktiv ist, wird somit die gesamte Matrix Spaltenweise projiziert. Über einen Treiber wird das Signal auf das Gate eines großen Projektionstransistors (W=50μ,L=0,8μ) gegeben. Der Drain-Anschluß wird direkt mit dem entsprechenden Spaltenausgang verbunden.
Für die zeilenweise Projektion ist ein Dreifach-NAND-Gatter verantwortlich, mit dem $YProj$, das ausgewählte Signal und zusätzlich noch $ColSel$ verknüpft wird. Damit wird erreicht, daß nur

selektiv eine Spalte projiziert wird. Auch hier wird der Ausgang wieder über einen Treiber auf das Gate eines Projektionstransistors gegeben. Die Funktion der Projektionslogik läßt sich durch die folgenden Gleichungen beschreiben:

$$\overline{ColOut} = XProj \cdot (Regout \cdot ProjSel + Treffer \cdot \overline{ProjSel}) \quad \text{(B.12)}$$

$$\overline{LineOut} = YProj \cdot ColSel \cdot (Regout \cdot ProjSel + Treffer \cdot \overline{ProjSel}) \quad \text{(B.13)}$$

Das Layout

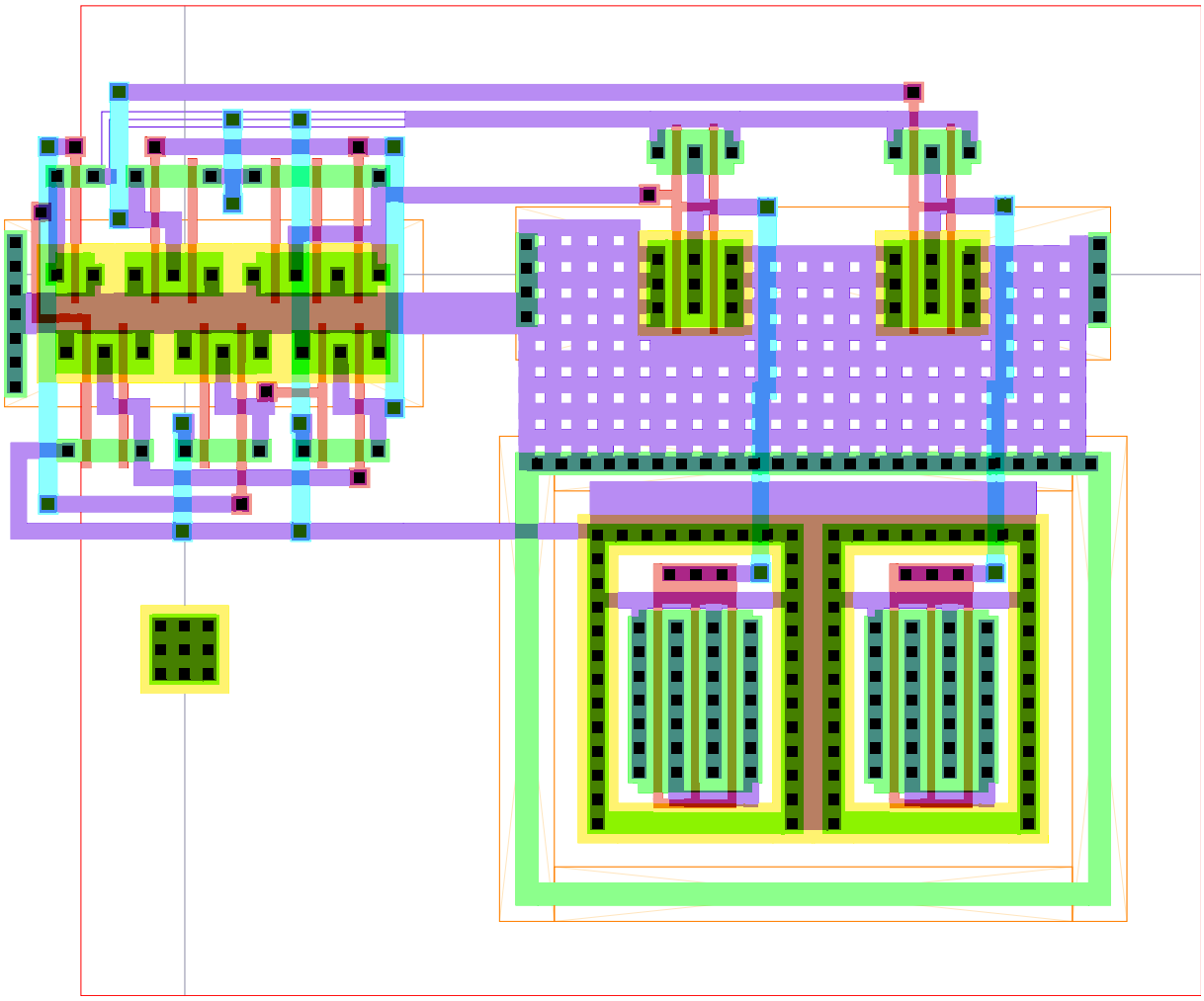


Abbildung B.8: Das Layout der Projektionslogik

Das Layout der Projektionslogik ist in Abbildung B.8 zu sehen. Den meisten Platz nehmen die beiden unten rechts zu erkennenden Projektionstransistoren ein. Es handelt sich dabei um

nMOS-Transistoren mit einer Gate-Breite von 50 μm, die in drei Teiltransistoren mit je 16,6 μm-Breite aufgeteilt sind. Da diese Transistoren direkt mit Bondingpads verbunden werden, wurde um die Transistoren herum ein p- und dann ein n-Gardring gelegt, mit denen Latch-up-Effekte vermieden werden sollen, die zur Zerstörung des Chips führen können. Oberhalb dieser Transistoren befinden sich die Treiberstufen, links daneben die Ansteuerlogik für die Projektion.

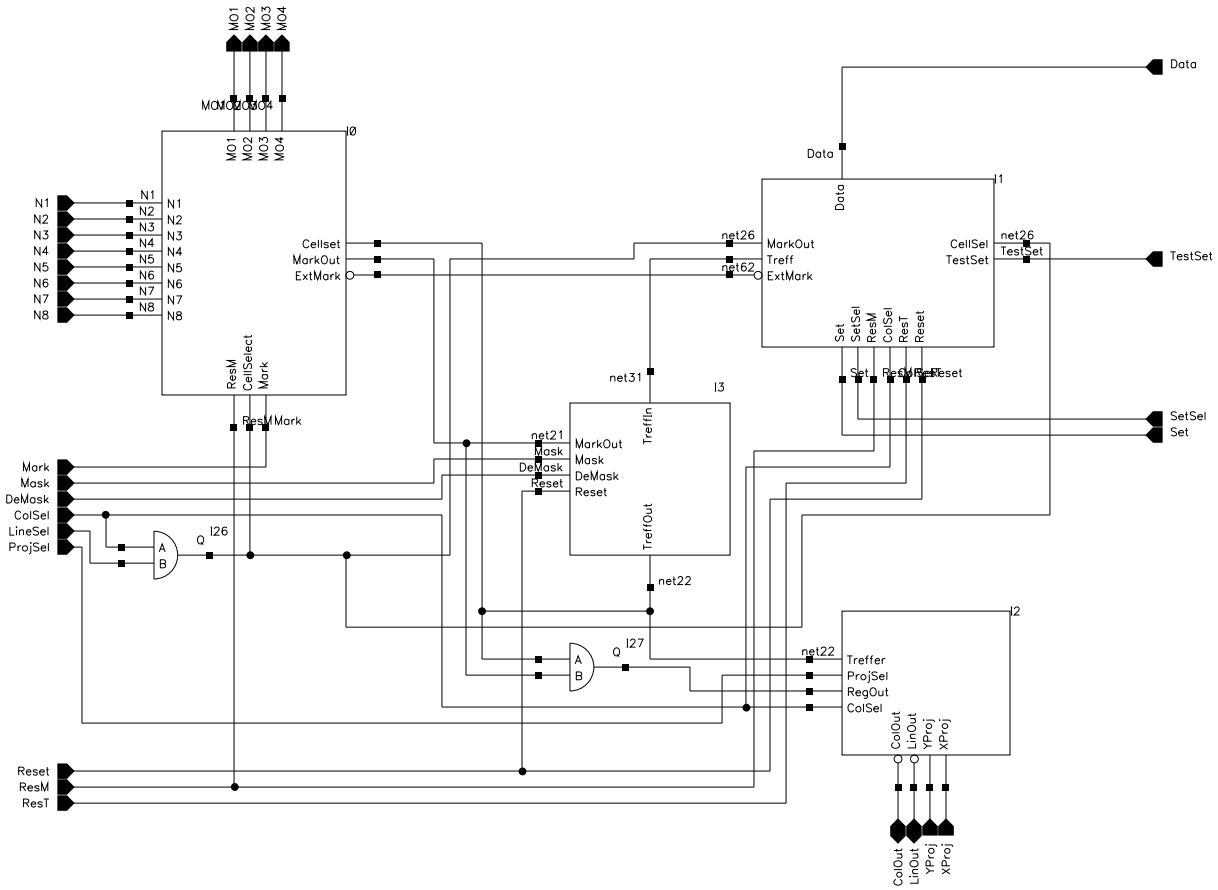


Abbildung B.9: Schaltbild einer Zelle. Die Zelle besteht im Wesentlichen aus den vier bereits beschriebenen Funktionsblöcken und einigen Verknüpfungsgattern

B.1.5 Die Gesamtzelle

Die Schaltung

Die Zelle, die in Abbildung B.9 dargestellt ist, setzt sich aus den vier beschriebenen Funktionsblöcken sowie zwei AND-Gattern zusammen, die für die Zusammenschaltung dieser Blöcke benötigt werden. Oben rechts ist die Speicherlogik I1 zu sehen. Die Signale $Data$, $TestSet$, $SetSel$, Set, $Reset$, $ResT$, $ResM$ und $ColSel$ sind direkt mit den entsprechenden Eingängen der Zelle verbunden. Das $CellSel$-Signal wird vom AND-Gatter I26 erzeugt, mit dem die Eingänge $ColSel$ und $LineSel$ verknüpft werden. Der $MarkOut$-Eingang der Speicherlogik ist nicht mit dem $MarkOut$-Ausgang der Markierungslogik verbunden, sondern mit dem $CellSel$-Signal. Durch die Einführung der Maskierungslogik war ein selektives Löschen von markierten Zellen nicht mehr notwendig. Statt dessen wurde damit die Möglichkeit geschaffen, Zellen löschen zu können, die von außen selektiert werden.
Das Treffersignal wird in der Maskierungslogik I3 weiterverarbeitet, wofür der $Treff$-Ausgang der Speicherlogik mit dem $TreffIn$-Eingang der Maskierungslogik verbunden ist. Weiterhin erhält die Maskierungslogik das $MarkOut$-Signal von der Markierungslogik, sowie die Steuer-

signale $Mask$, $DeMask$ und $Reset$.
Der Ausgang $TreffOut$ wird sowohl dem $CellSet$ Eingang der Markierungslogik als auch dem $Treffer$-Eingang der Projektionslogik zugeführt.
Die Markierungslogik ist direkt mit den Zelleingängen $ResM$ und $Mark$ verbunden. Von den Nachbarn erhält sie die Informationen über die Eingänge $N_1, N_2, \ldots, N_8$. Die Ausgänge MO_1, $MO2$, ..., $MO4$ gehen wiederum zu den Nachbarn hin. Der Ausgang $MarkOut$ der Markierungslogik wird in dem AND-Gatter I27 mit dem $TreffOut$-Signal der Maskierungslogik verknüpft und das so entstandene Verknüpfungssignal dem $RegOut$-Eingang der Projektionslogik zugeführt. Die weiteren Eingänge der Projektionslogik I2, $ProjSel$, $colSel$, $XProj$ und $YProj$ sind direkt mit den entsprechenden Eingängen der Zelle verbunden. Die Ausgänge $ColOut$ und $LineOut$ führen nach außen. Damit erhält man für die Funktion einer kompletten Zelle die folgenden Logikgleichungen:

$$Treff.S = TestSet \cdot ColSel \cdot LineSel \cdot SetSel + Set \cdot Data \cdot SetSel \quad \text{(B.14)}$$

$$Treff.R = ResT \cdot ColSel \cdot LineSel + Reset \quad \text{(B.15)}$$

$$ExtMark.S = TestSet \cdot ColSel \cdot LineSel \cdot \overline{SetSel} + Set \cdot Data \cdot ColSel \cdot \overline{SetSel} \quad \text{(B.16)}$$

$$Extmark.R = ResM \quad \text{(B.17)}$$

$$Mask.S = MarkOut \cdot Mask \quad \text{(B.18)}$$

$$Mask.R = DeMask + Reset \quad \text{(B.19)}$$

$$Mark.S = ExtMark + Mark \cdot \left(ColSel \cdot LineSel + Treffer \cdot \sum_{i=1}^{8} Mark.S_{P_i} \right) \quad \text{(B.20)}$$

$$Mark.R = ResM \quad \text{(B.21)}$$

$$Treffer = Treff \cdot \overline{mask} \quad \text{(B.22)}$$

$$ColOut = XProj \cdot (RegOutProjSel + Treffer\overline{ProjSel}) \quad \text{(B.23)}$$

$$LineOut = YProj \cdot ColSel \cdot (RegOutProjSel + Treffer\overline{ProjSel}) \quad \text{(B.24)}$$

Das Layout

Abbildung B.10 zeigt das Layout der gesamten Zelle, die im Wesentlichen aus den vier beschriebenen Blöcken, Speicherlogik, Maskierungslogik, Markierungslogik und Projektionslogik besteht. Die Zelle wurde nicht auf minimalen Platzbedarf optimiert, sondern so aufgebaut, dass sich problemlos mehrere Zellen nebeneinander anordnen lassen. Alle Steuerleitungen sind entweder horizontal oder vertikal durchgeführt. Lediglich die Kommunikationsleitungen zwischen den Zellen und die Zuführung der Datensignale erfordern noch ein spezielles Leitungsrouting. Um die Zelle herum befindet sich wieder ein p-Gardring, um die Zellen gegeneinander abzublocken. Die Größe der Zelle beträgt etwa 220 μm mal 195 μm.

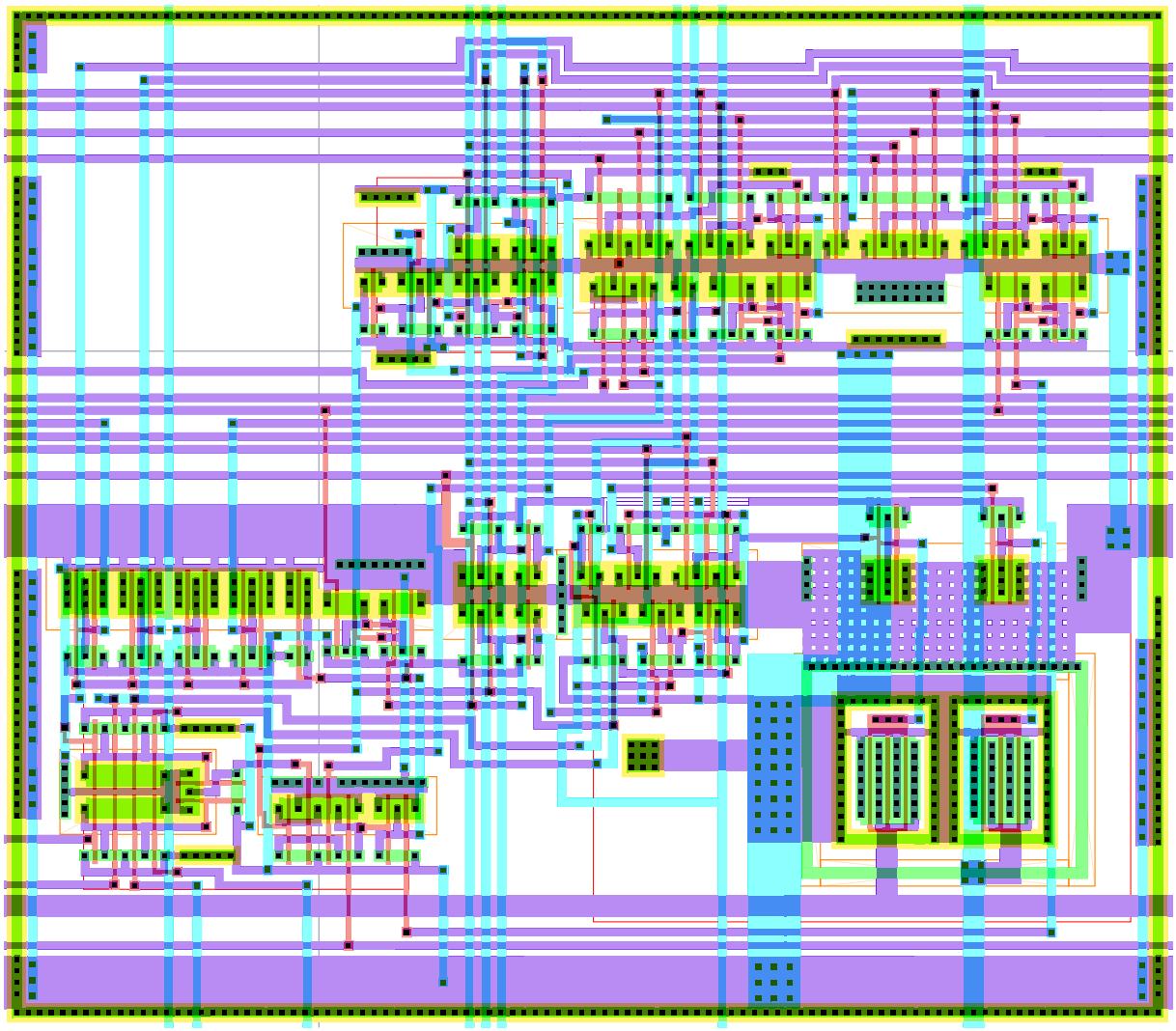


Abbildung B.10: Das Layout der Logikzelle

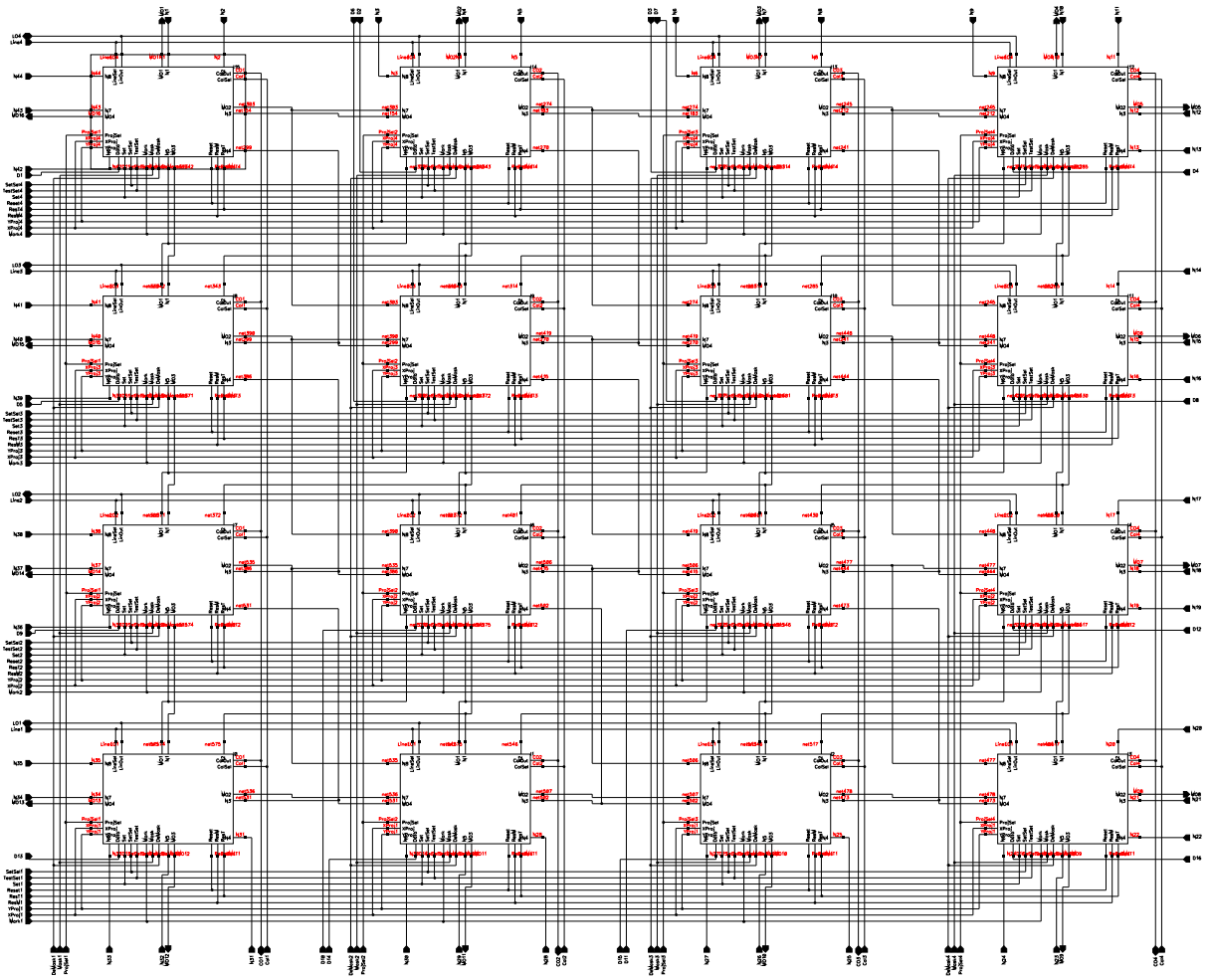


Abbildung B.11: Das Schaltbild der vier mal vier Matrix

B.2 Die Zellmatrix

B.2.1 Die vier mal vier Matrix und die internen Treiber

Die Schaltung

Auf einem Chip sind insgesamt 16 Zellen in einer vier mal vier Matrix angeordnet. Diese 16 Zellen müssen in geeigneter Weise miteinander verbunden werden, um die Kommunikation zwischen Nachbarn zu gewährleisten. Die Verbindungen erfolgen über die MO_i Ausgänge und die N_i Eingänge. Die korrekten Verbindungen sind so durchzuführen, dass jeweils die folgenden Verbindungen hergestellt werden:

- MO_1 wird mit N_5 des Nachbarn oben und N_6 des Nachbarn rechts oben verbunden.
- MO_2 wird mit N_7 des Nachbarn rechts und N_8 des Nachbarn rechts unten verbunden.
- MO_3 wird mit N_1 des Nachbarn unten und N_2 des Nachbarn links unten verbunden.

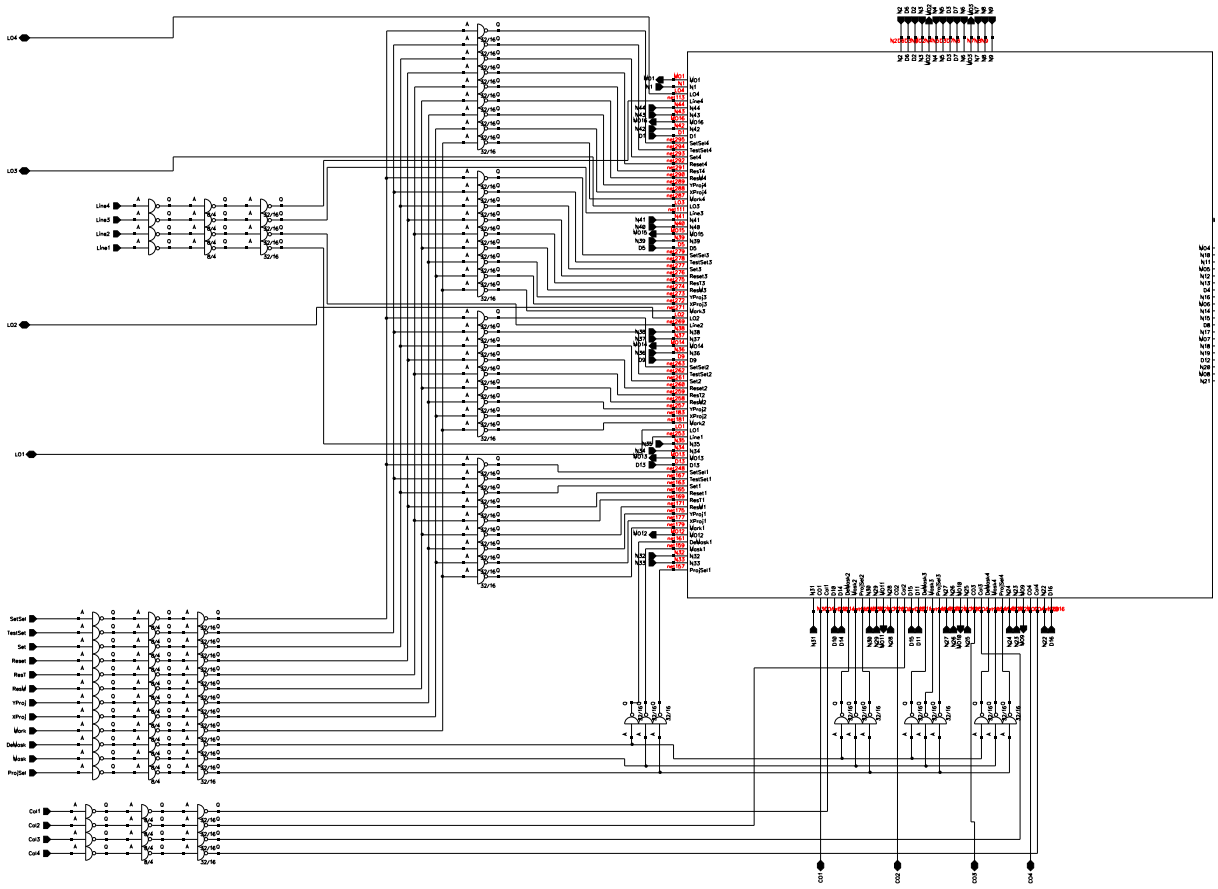


Abbildung B.12: Das Schaltbild der Treiberketten für die Ansteuerung der Logikmatrix

- MO_4 wird mit N_3 des Nachbarn links und N_4 des Nachbarn links oben verbunden.

Dies ist in Abbildung B.11 zu sehen. Die Ausgänge $ColOut$, sowie die Eingänge $ColSel$ und $ProjSel$ werden spaltenweise miteinander verbunden. Die Ausgänge $LinOut$, sowie die Eingänge $LineSel$, $XProj$, $YProj$, Set, $SetSel$, $TestSet$, $Mark$, $Mask$, $DeMask$, $ResM$, $Reset$ und $ResT$ jeder Zelle werden dagegen zeilenweise miteinander verbunden. Dies berücksichtigt die Layoutstruktur der Zelle, sodass diese Verbindungen allein durch das Aneinanderreihen der Zellen realisiert werden können. Die $Data$-Eingänge aller Zellen werden nach außen geführt.
Die Steuereingänge aller Zellen einer Zeile oder Spalte der Matrix stellen bereits eine relativ große kapazitive Last dar. Um diese Last treiben zu können, sind für jede Steuerleitung auf dem Chip Treiberketten vorgesehen, die stufenweise die Steuersignale verstärken.
In den Endtreibern betragen die Verhältnisse W/L der Transistoren für den pMOS-Transistor $32\mu/0,8\mu$ und für den nMOS-Transitor $16\mu/0,8\mu$. Abbildung B.12 zeigt das Schaltbild der Treiberketten.

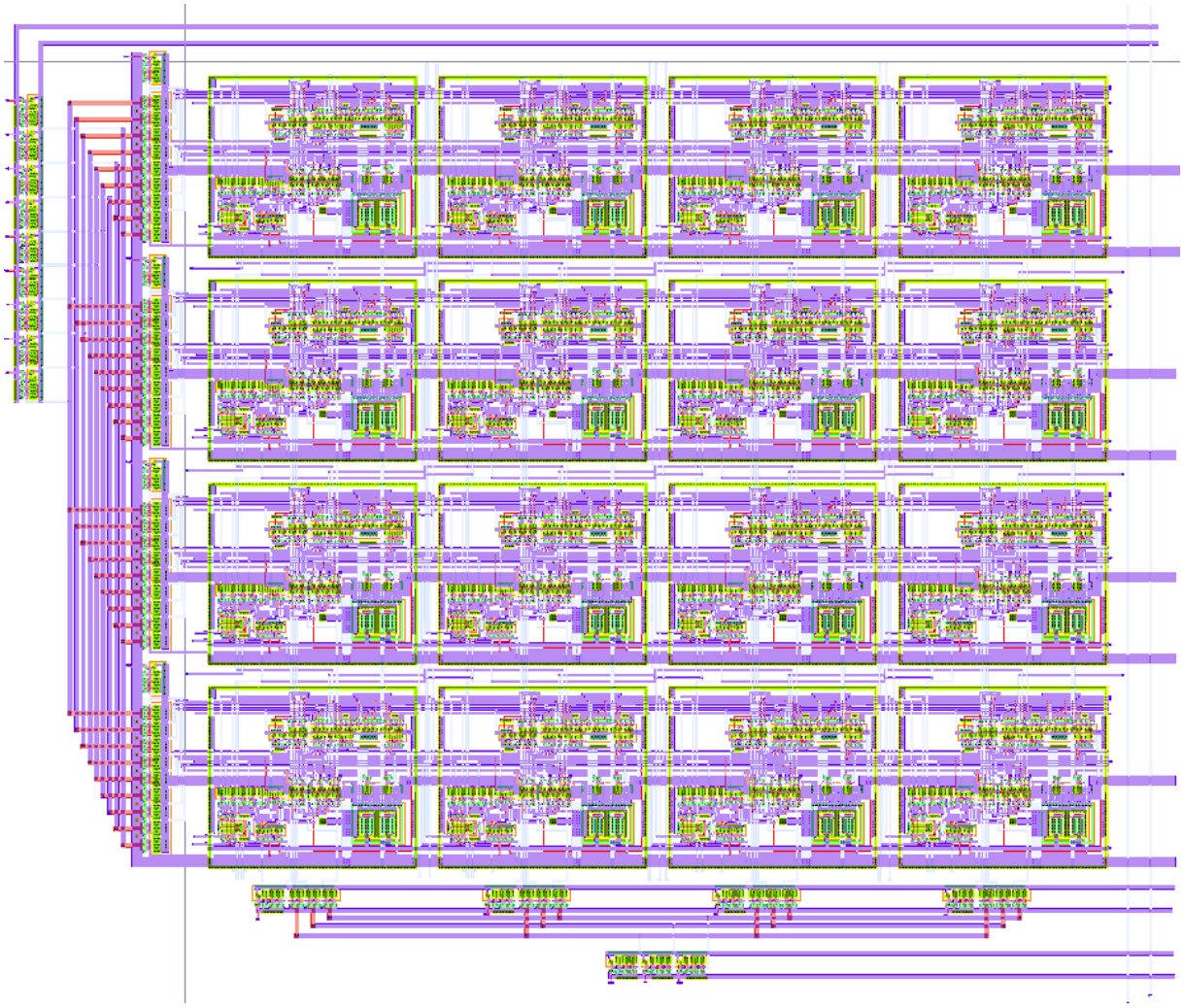


Abbildung B.13: Die Matrix mit allen Treibern zur Zuführung der Steuersignale

Das Layout

In Abbildung B.13 ist die Anordnung der 16 Zellen nebeneinander zu sehen. Hier sieht man deutlich den Vorteil, den das verwendete Layout der Zellen für die Zusammenschaltung von Zellen bietet. Die Steuer- und Projektionsleitungen brauchten zwischen den Zellen nur verlängert zu werden, sodass sie über die gesamte Horizontale, bzw. Vertikale durchgehen. Dadurch wurde das Signalrouting zwischen den Zellen wesentlich vereinfacht. Zwischen den Zellen wird nur ein Zwischenraum von etwa 20 μm als Routing-Kanal benötigt. Innerhalb dieser Routing-Kanäle sind die Kommunikationsleitungen zwischen den Zellen, sowie die Zuführungen der Datensignale untergebracht. Die gesamte Matrix ist etwa 950 μm mal 840 μm groß.
Links neben der Matrix und unterhalb sind die Treiber untergebracht. Jede Spalte bzw. Zeile hat zunächst eigene Treiber, die wiederum durch gemeinsame Vortreiber angesteuert werden.

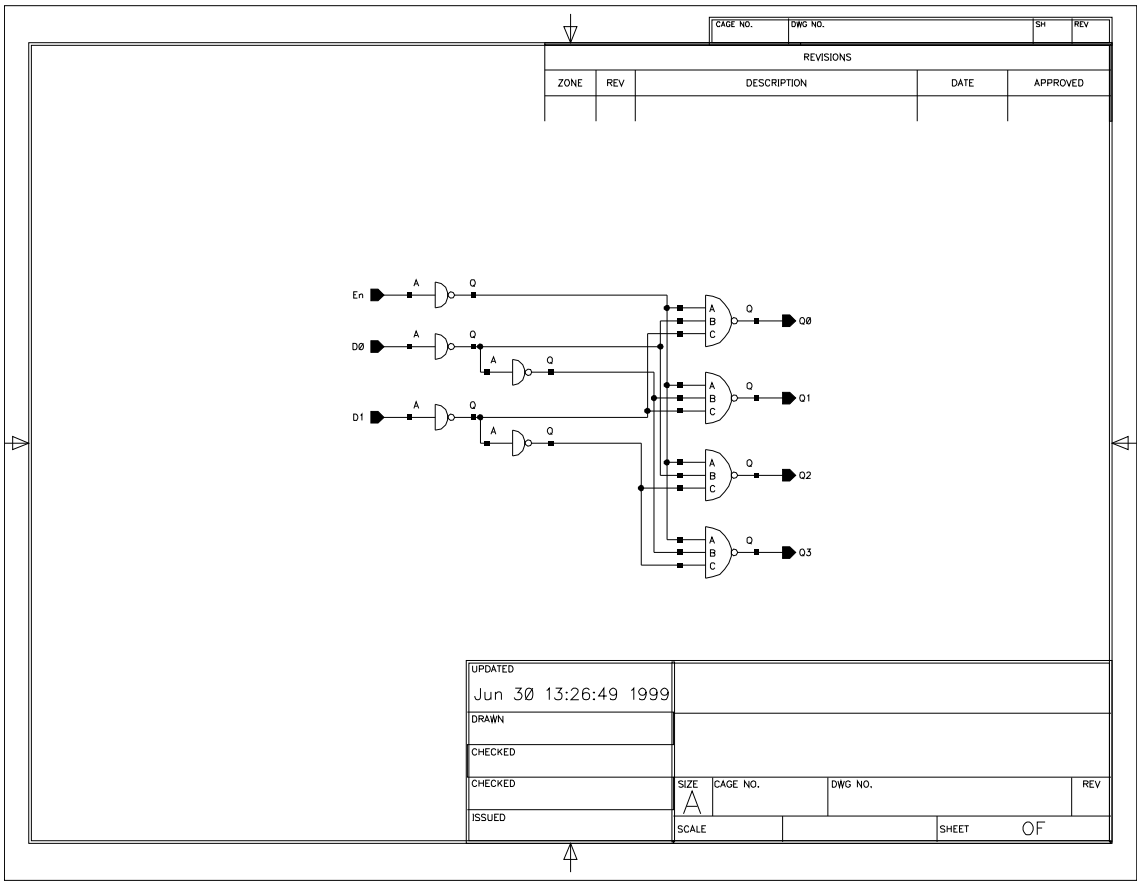


Abbildung B.14: Das Schaltbild des vier aus zwei Dekoders

B.2.2 Die Spalten- und Zeilendekoder

Die Schaltung

Sowohl für die Auswahl der Spalten, als auch für die Auswahl der Zeilen sind auf dem Chip Dekoder vorgesehen, die je zwei Spalten-, bzw. Zeilenadressbits dekodieren und die entsprechenden Selektionsleitungen aktivieren. Das Schaltbild dieser beiden identischen Dekoder ist in Abbildung B.14 dargestellt. Der Dekoder verfügt über drei Eingänge, $D0$, $D1$ und EN. $D0$ und $D1$ sind die beiden Adresseingänge. EN ist das Enable Signal, das am Eingang in negativer Logik vorliegt. Um das Signal in positiver Logik zu erhalten, wird es zunächst ebenfalls invertiert. Über die entsprechenden NAND-Verknüpfungen werden die vier Ausgangssignale $Q0$ bis $Q3$ erzeugt. Diese Signale werden von der Zellularlogikmatrix in negativer Logik benötigt, sodass nach den NAND-Gattern keine Invertierung mehr notwendig ist.

Das Layout

In Abbildung B.15 ist das Layout des Adressdekoders zu sehen. Unten befinden sich die Eingangsinverter. Zunächst ganz links der Inverter für das En-Signal, dann die Doppelinverter für

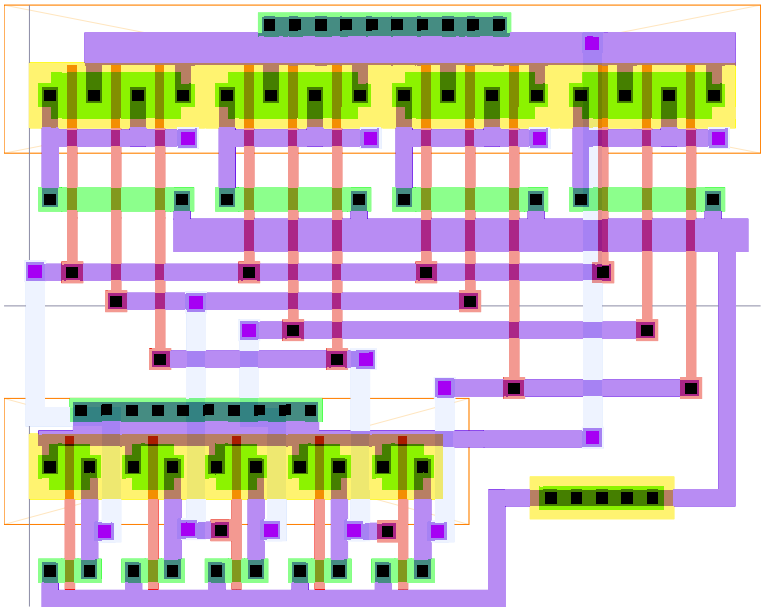


Abbildung B.15: Das Layout der Adressdekoder für die Zeilen- und Spaltenadressierung

die Adresssignale. Über den Routingchanel im mitleren Bereich werden die Ausgangssignale dieser Eingangsstufe auf die vier NAND-Gatter, die in dem Bild oben zu sehen sind, verteilt. Diese vier NAND-Gatter liefern die erforderlichen Ausgangssignale.

B.3 IO-Strukturen

Für die besonderen Funktionen des Zellularlogikchips, z.B. der Projektion und der Kaskadierung mehrerer Chips, mußten spezielle Ein-/Ausgabestrukturen entworfen werden. Da sich in vorangegangenen Tests gezeigt hat, daß sich auch die Geschwindigkeit der Standard Ein-/Ausgabestrukturen, die in der Herstellerbibliothek vorhanden sind, noch deutlich steigern läßt, wurde auch für diese Strukturen nicht auf die Bibliotheksfunktionen zurück gegriffen, sondern Eigenentwicklungen durchgeführt.

B.3.1 Der TTL-Eingabeblock

Die Schaltung

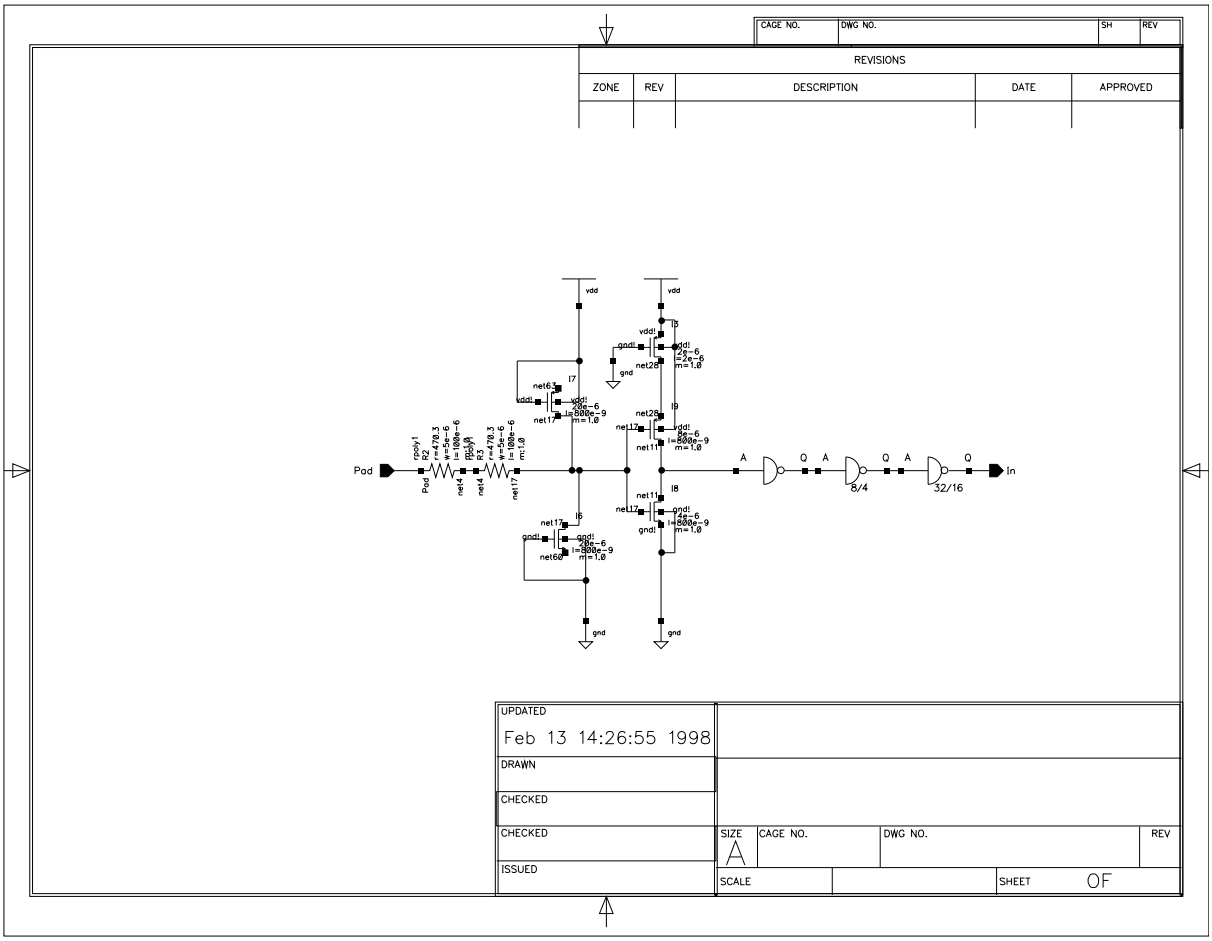


Abbildung B.16: Das Schaltbild der TTL-Eingangsstruktur

Der TTL-Eingabeblock ist eine der Schaltungen, die auch als Standardbaustein in den Herstellerbibliotheken vorhanden sind. Dennoch wurde auch hier eine eigene Schaltung entworfen,

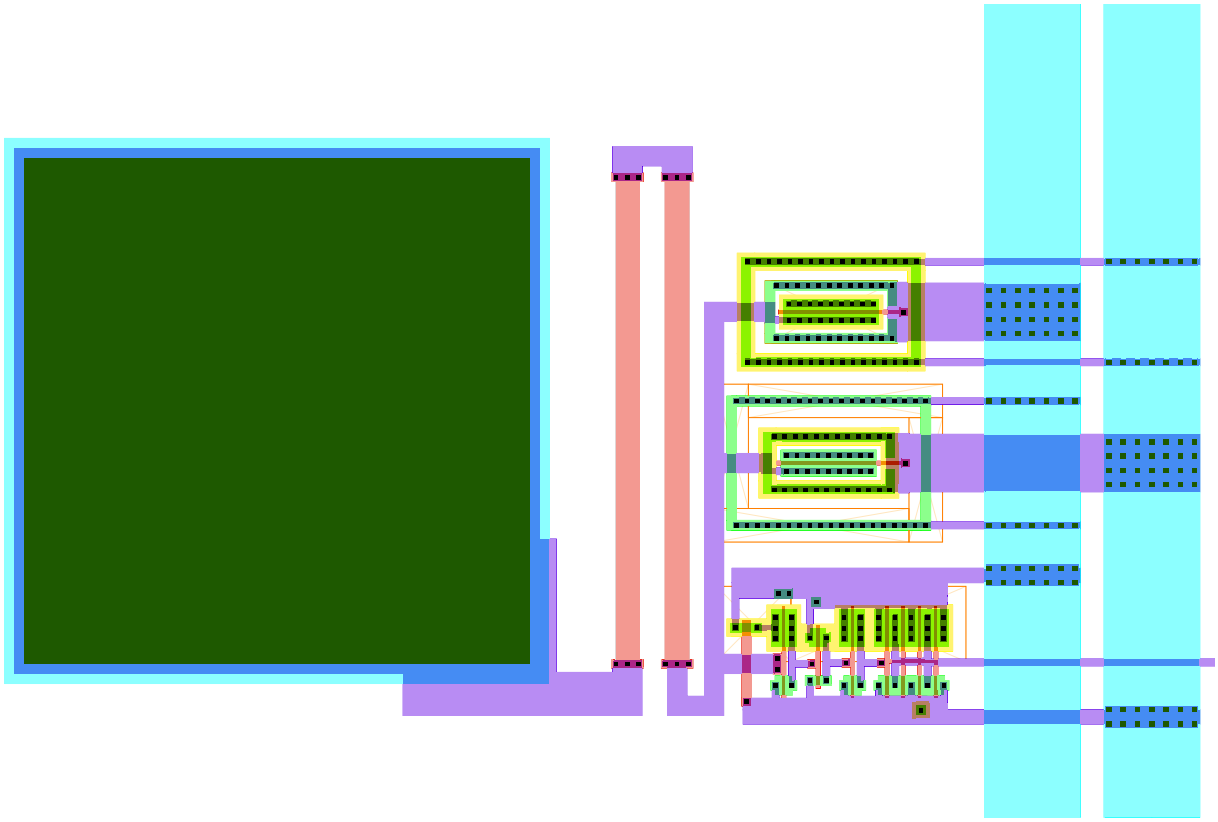


Abbildung B.17: Das Layout eines Eingabepads mit TTl-Kompatibler Schaltschwelle

die in Abbildung B.16 dargestellt ist. Vom eigentlichen PAD geht das Signal über zwei Poly1-Widerstände von je 470 Ω zur Eingangsstufe. Parallel zur Eingangsstufe befinden sich zwei Transistoren, die als Schutzdioden dienen, um Überspannungen nach Masse oder +5 V abzuführen. Die Eingangsstufe besteht aus einem Inverter, der aus den Transistoren I8 und I9 gebildet wird. Um die Schaltschwelle auf TTL-Pegel anzupassen, wird der pMOS-Transistor nicht direkt mit der Versorgungsspannung verbunden, sondern mit einer, über Transistor I3, der als Widerstand geschaltet ist, herabgesetzten Versorgungsspannung. Zu diesem Zweck liegt das Gate von I3 fest auf Massepotential. Hinter dieser Eingangsstufe befinden sich noch drei weitere Inverter mit zunehmender Transistorgröße, die eine Treiberkette bilden, um längere Leitungen auf dem Chip treiben zu können.

Das Layout

Zentraler Teil aller IO-Strukturen ist das Bondingpad, das die eigentliche Kontaktstelle zur Außenwelt darstellt. Dieses Pad besteht aus einer großflächigen Schicht aus Metall1 und Metall2 unter einem Fenster in der Passivierungsschicht, um die Kontaktierung mit dem Bondingdraht zu ermöglichen. In den Spzifikationen von Europractice ist für das Bonden der Chips ein Fenster von mindestens 100 μm mal 100 μm Größe erforderlich. Zwischen den Pads muß ein Abstand von mindestens 55 μm gelassen werden. Abbildung B.17 zeigt das Layout des Eingangspads.

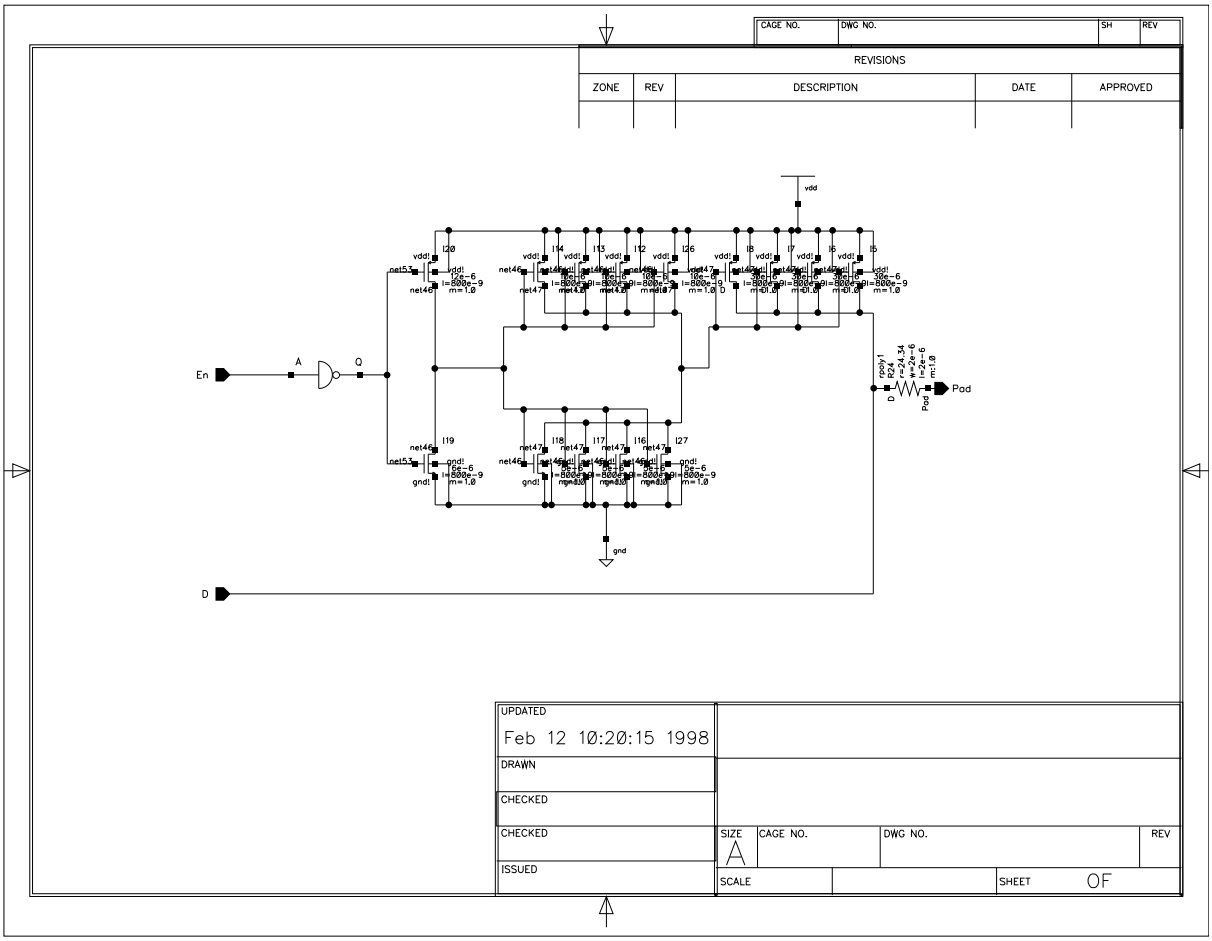


Abbildung B.18: Das Schaltbild der Projektionsausgänge. Der Projektionsausgang, der von den Zellen kommt, ist direkt mit dem Pad verbunden. Oben sieht man Treiber und Endtransistoren für die Bypasslogik

Links ist das eigentliche Bondingpad zu sehen, das in der rechten unteren Ecke kontaktiert wird. Das Eingangssignal geht dann über einen großflächigen Widerstand, der als Leiterbahn aus Polysilizium auf die Pad-Elektronik. Das Signal geht dort zunächst auf den Inverter mit reduzierter Schaltschwelle und schließlich auf herkömmliche Inverter, mit denen genügend Treiberleistung generiert wird, um längere Leiterbahnen treiben zu können.
Um die Schaltung, insbesondere die Gates der beiden Transistoren im ersten Inverter, vor Zerstörung zu schützen, enthält sie zwei Schutzdioden, die die Signalspannung hinter dem Schutzwiderstand auf die Betriebsspannung begrenzen. Diese Dioden befinden sich über dem Inverterblock. Zum Schutz vor Latch-up-Effekten sind die Dioden wieder mit zwei Guardringen umgeben.

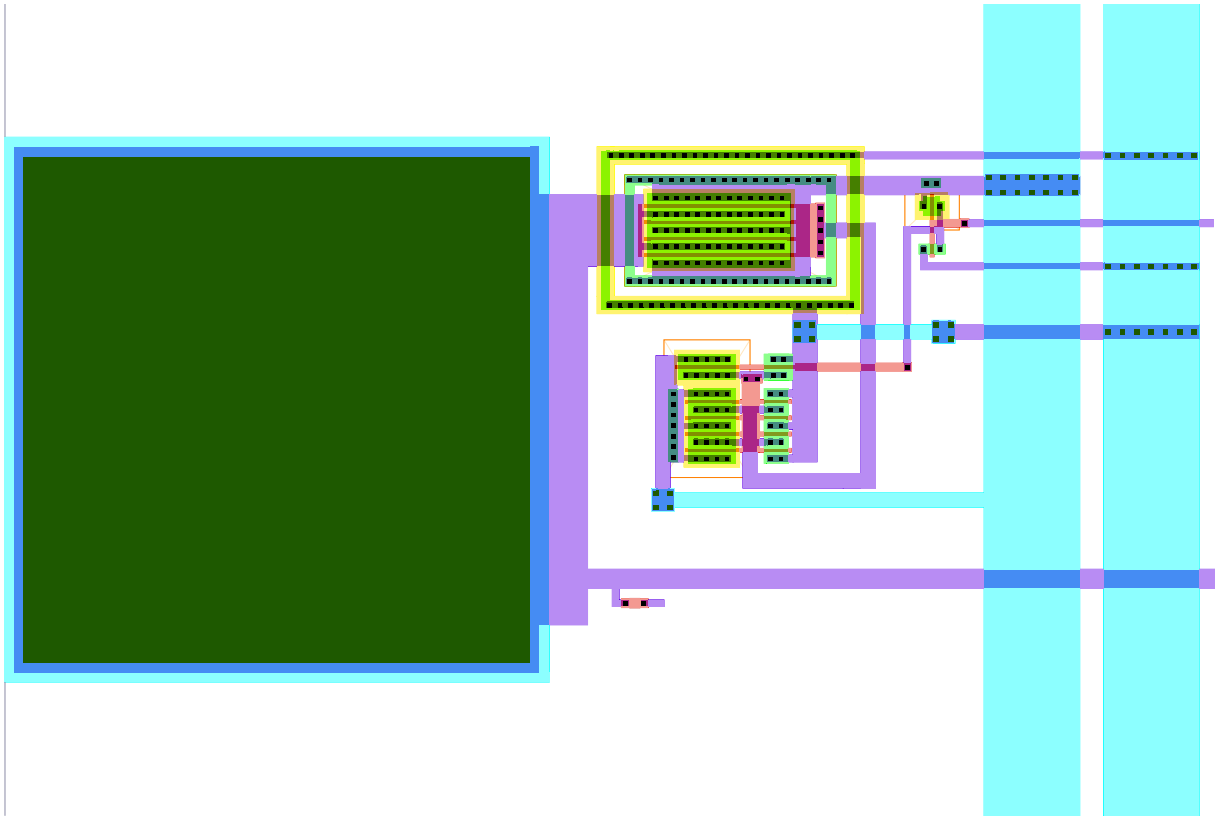


Abbildung B.19: Das Layout eines Projektionspads

B.3.2 Der Projektionsausgang

Die Schaltung

Für die Projektion sind bereits Transistoren in den Zellen untergebracht, die hinreichend groß sind, um direkt die Ausgänge treiben zu können. Deshalb wird in der Projektionszelle die Datenleitung direkt mit dem Ausgang verbunden. Normalerweise wird bei einer derartigen Wired-Or-Schaltung die Relaxation in den Ruhezustand, d.h. der Übergang vom aktiven Low-Pegel zum Ruhezustand (High-Pegel), durch den externen Pull-Up-Widerstand hervorgerufen. Dieser Übergang wäre jedoch mit Schaltflanken von einigen 10 ns oder mehr verbunden.
Um dies abzukürzen, ist in der Ausgangsschaltung ein Bypass-Transistor eingebaut, der den Ausgang mit der Versorgungsspannung verbindet, sobald der Projektionsvorgang beendet ist. Damit lassen sich die Schaltflanken auf wenige Nanosekunden verkürzen. Dieser Bypass-Transistor ist in Abbildung B.18 oben zu erkennen (Einzeltransistoren I5, I6, I7 und I8) und links davon die Treiber (I12-I14, I16-I18 sowie I26 und I27).

Das Layout

Das Layout des Projektionspads ist in Abbildung B.19 zu sehen. Rechts neben dem Bondingpad ist der große Bypasstransistor zu erkennen, darunter die Treiberstufen für diesen Transistor. Die

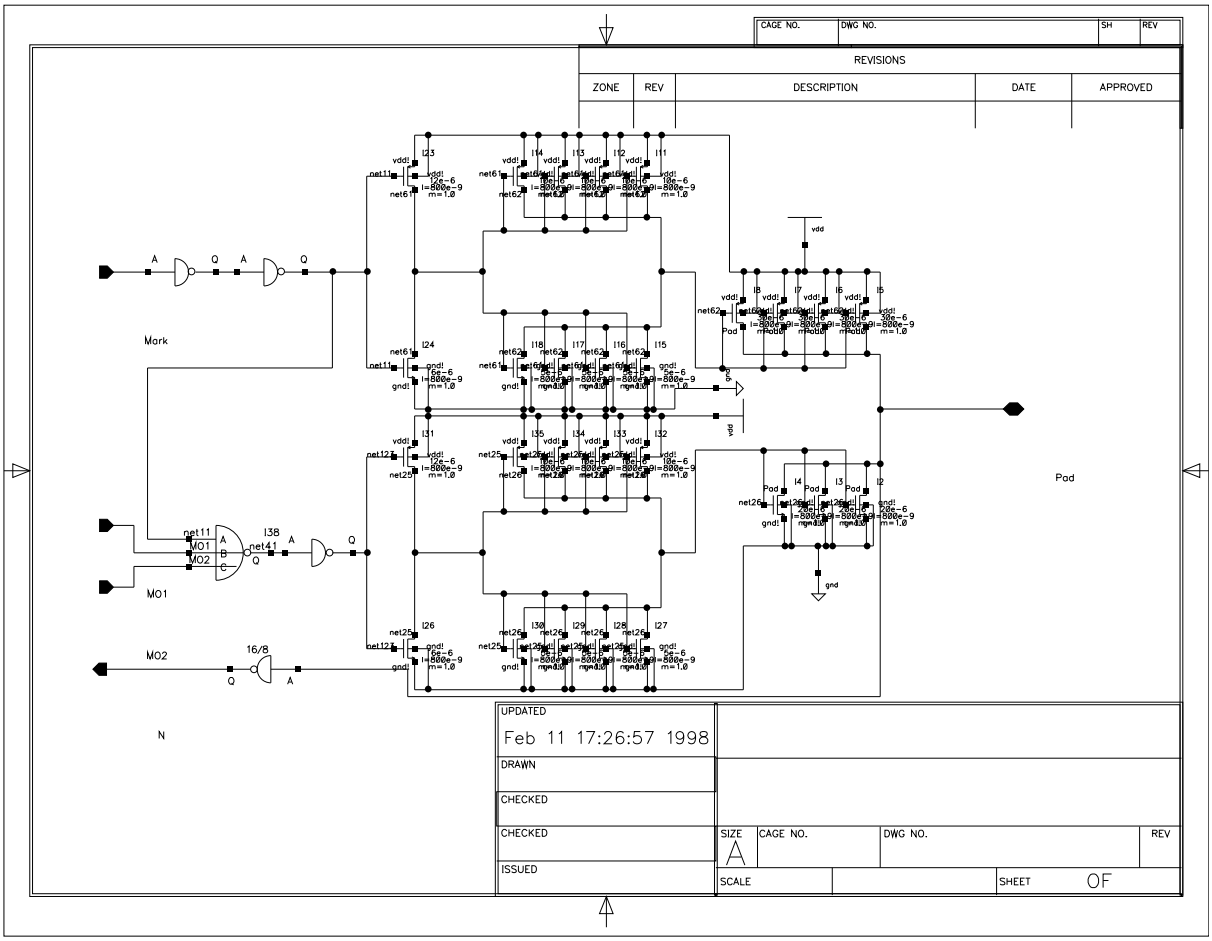


Abbildung B.20: Das Schaltbild des bidirektionalen Kaskadierungspads

Verbindung zu den Projektionstransistoren auf dem Chip geschieht über eine direkte Leiterbahnanbindung, die unterhalb der Treiber zu sehen ist.

B.3.3 Die Kaskadierungspads

Die Schaltung

Die komplexeste Ein- / Ausgabestruktur stellen die Kaskadierungspads dar. Mit ihnen ist es möglich, die Markierungsinformationen bidirektional mit den Nachbarzellen eines anderen Chips auszutauschen. Die Funktionsweise ist bereits ausführlich in Abschnitt 6.5.3 beschrieben. Abbildung B.20 zeigt das Schaltbild der Kaskadierungslogik. Von links kommen das Steuersignal $Mark$, sowie die Kaskadierungssignale zweier Zellen $MO1$ und $MO2$. Das $Mark$-Signal wird mit zwei Invertern aufbereitet, um damit über die Treiber $I23/I24$, sowie $I11 \ldots I14$ und $I15 \ldots I18$ den pMOS-Ausgangstransistor $I5 \ldots I8$ direkt anzusteuern.
Die Kaskadierungssignale werden zusammen mit dem aufbereiteten $Mark$-Signal im Logikgat-

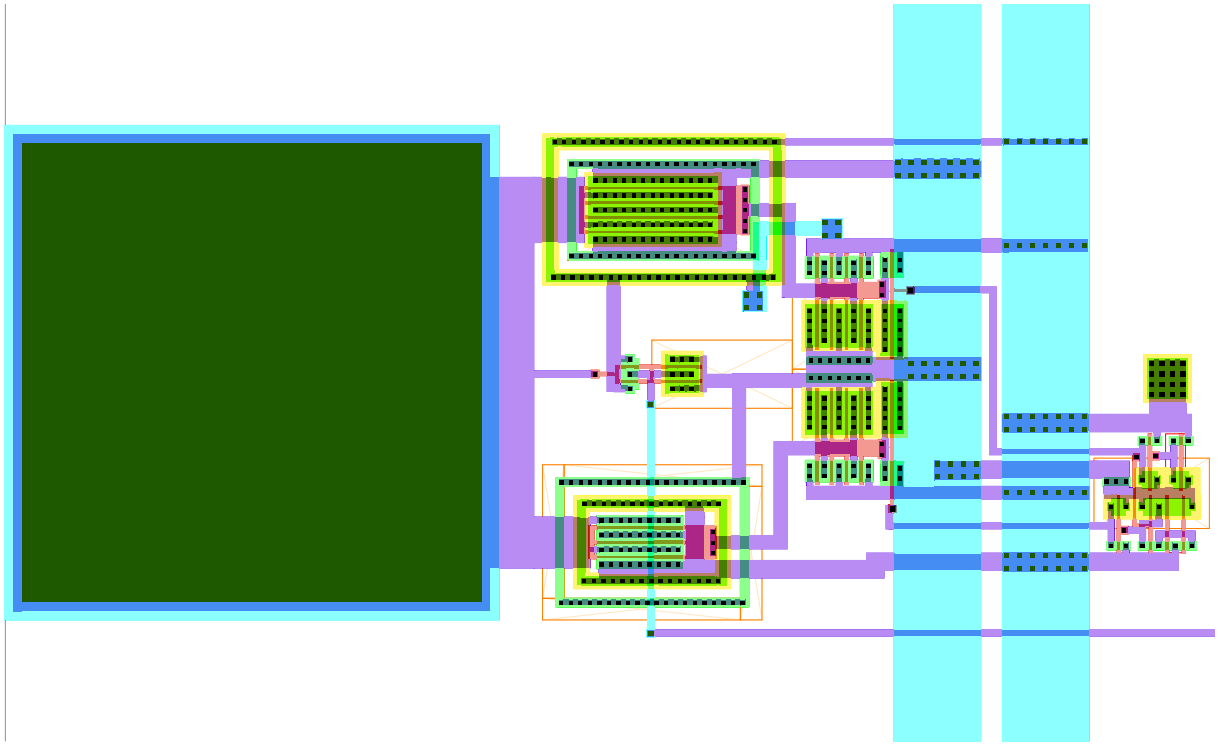


Abbildung B.21: Das Layout eines Kaskadierungspads

ter $I38$ entsprechend der Gleichung

$$Out = \overline{Mark \cdot (MO1 + MO2)} \tag{B.25}$$

verknüpft. Dieses Signal wird invertiert, um dann damit wieder über Treiberstufen den nMOS-Ausgangstransistor anzusteuern. Somit wird der Ausgang auf Low-Pegel gezogen, wenn $Mark$ aktiv ist und mindestens einer der beiden Kaskadierungssignale $MO1$ oder $MO2$ aktiv ist. Wenn $Mark$ nicht aktiv ist, hält der pMOS-Transistor den Ausgang auf High-Pegel.
Der Zustand des Pads wird über einen Inverter, der die vierfache Gatebreite[3] besitzt, um längere Leitungen zu treiben, dem Chip zurückgemeldet.

Das Layout

Abbildung B.21 zeigt das Layout des Kaskadierungspads. Rechts neben dem Bondingpad sind die beiden Ausgangstransistoren deutlich zu erkennen. Dazwischen befindet sich der Inverter zur Rückmeldung des Pad-Zustands, rechts daneben die Treiber für die Schalttransistoren. Die Ansteuerlogik ist rechts neben den Versorgungsspannungsleiterbahnen zu sehen.

[3] pMOS : $16\mu/0,8\mu$, nMOS : $8\mu/0,8\mu$.

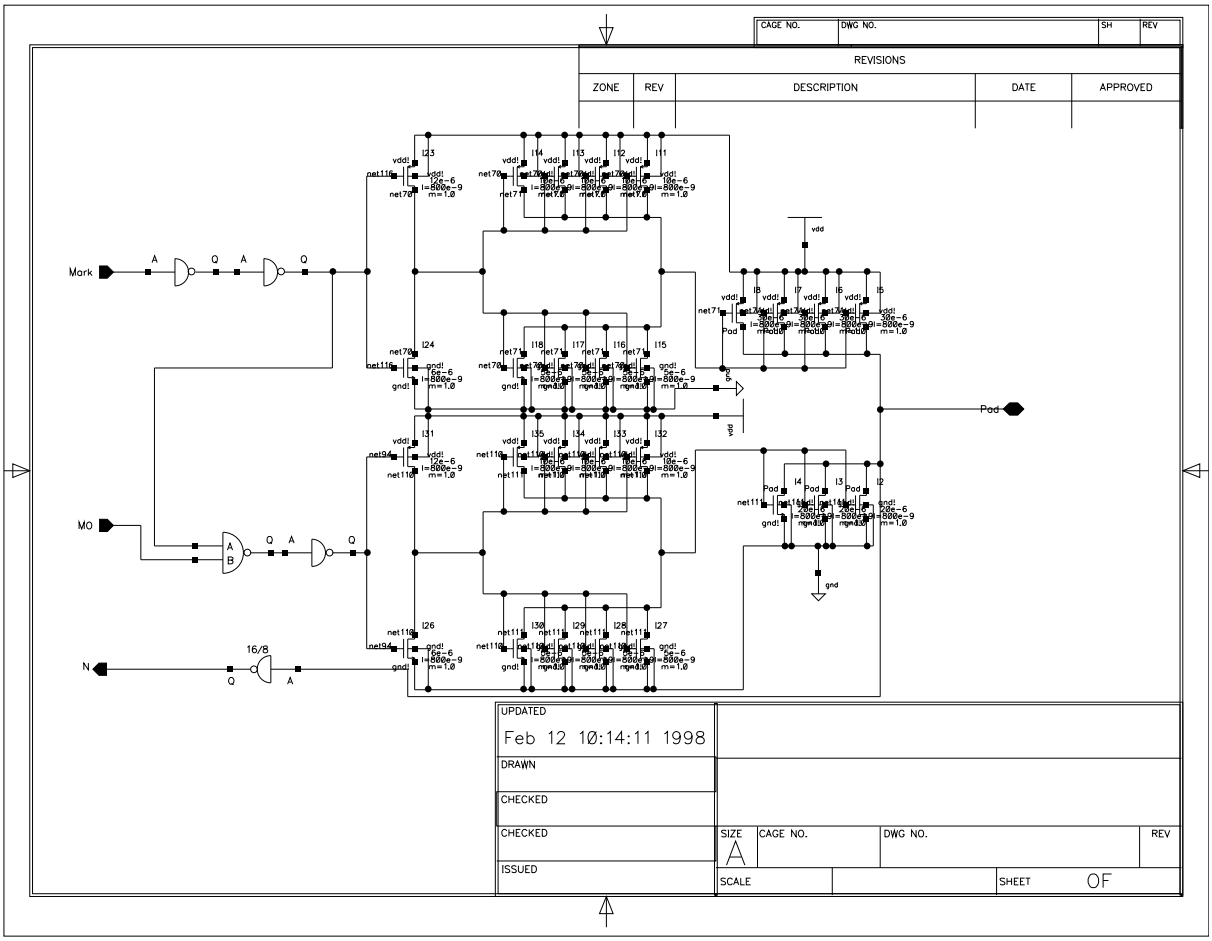


Abbildung B.22: Das Schaltbild eines Eckpads

B.3.4 Das Eckpad

Die Schaltung

Das Eckpad ist ebenfalls ein Kaskadierungspad und daher mit diesem im Aufbau nahezu identisch. Der einzige Unterschied besteht darin, dass das Eckpad nur einen MO-Eingang besitzt. Dadurch kann die Eingangsverknüpfung, für die im Kaskadierungspad noch ein spezielles Gatter entworfen werden mußte, jetzt mit einem einfachen NAND-Gatter realisiert werden.

Das Layout

Dieses Pad, dessen Layout in Abbildung B.23 zu sehen ist, ist im Unterschied zum normalen Kaskadierungspad von der Geometrie her so aufgebaut, dass es sich in den vier Ecken des Chips platzieren läßt. Die einzelnen Komponenten des Layouts sind jedoch identisch mit denen des Kaskadierungspads.

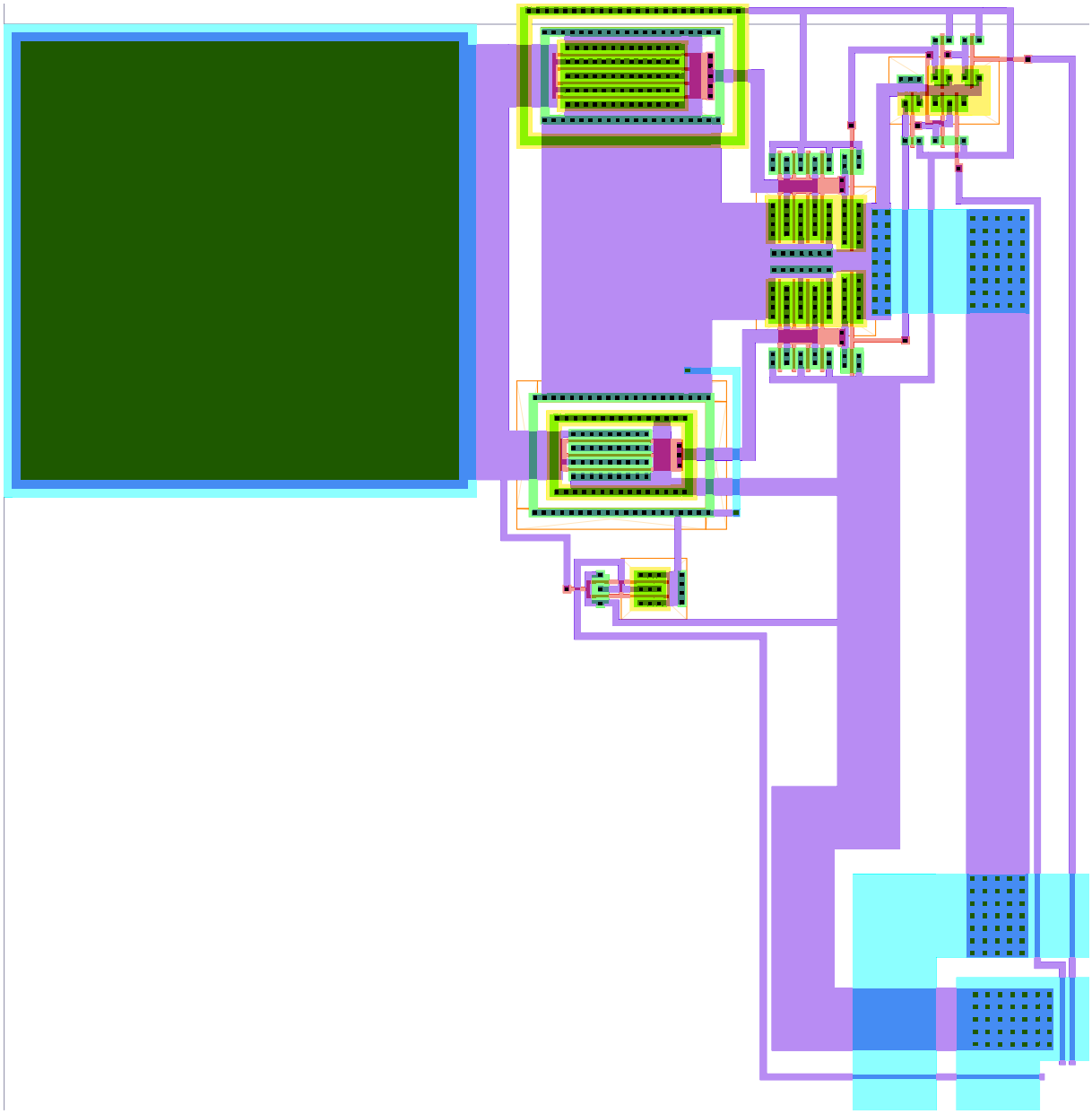


Abbildung B.23: Das Layout eines Eckpads

B.3.5 Die Versorgungsspannung

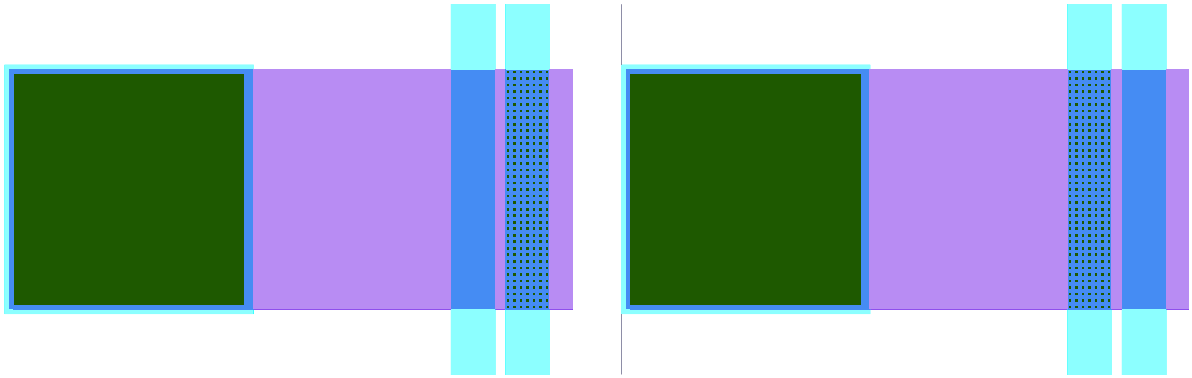


Abbildung B.24: Das Layout der Versorgungspads, Links das für den Masseanschluß, rechts das für die +5V-Versorgung

Für die Zuführung der Versorgungsspannung sind keine besonderen Schaltungsmaßnahmen notwendig. Es handelt sich dabei lediglich um Bondingpads mit großflächigen Zuleitungen nach außen. Zudem ist eine Verbindung zu den Ringleitungen realisert, über die die anderen Pads mit Spannung versorgt werden. Hierin unterscheiden sich das Ground- und das Vdd-Pad.

B.4 Der Gesamtchip

B.4.1 Die Gesamtschaltung

In Abbildung B.25 ist das Schaltbild des gesamten Chips zu sehen. Der zentrale Block enthält die Zellularlogik-Matrix mit allen Treibern, wie in Abbildung B.12 dargestellt. Die Spalten- und Zeilenselektionseingänge der Logikmatrix werden von den beiden Adressdekodern angesteuert. Alle anderen Steuerleitungen und Ausgänge der Logikmatrix gehen direkt auf entsprechende Pads.
Das Schaltungsschema der Kaskadierungspads ist dabei jedoch etwas komplexer. Ein Kaskadierungspad wird immer zwischen zwei Zeilen, bzw. Spalten angeordnet. Diese Anordnung führt dazu, dass über diese eine Kaskadierungsleitung sowohl die waagerechte Verbindung zwischen zwei Zellen, als auch die Eckverbindung zwischen zwei Zellen realisiert werden kann.
Um eine derartige Anordnung zu realisieren, muss jedes Kaskadierungspad mit den zwei Logikzellen verbunden werden, zwischen denen es sich befindet. Entsprechend muß auch jede Zelle mit zwei Kaskadierungspads verbunden werden. Dies ist auch in der Schaltung in Abbildung B.25 realisiert. Da die Kaskadierungspads nur aktiv werden, während die Markierungsphase läuft, benötigen alle Pads am entsprechenden Eingang das Markierungsignal. Dieses Signal wird direkt hinter dem Eingangspad für das Markierungssignal abgegriffen, mit einer Inverterkette aus vier Invertern verstärkt und dann allen Kaskadierungspads zugeführt.
Eine ähnliche Schaltungsanordnung findet man bei den Projektionsausgangspads. Solange der Projektionsmechanismus nicht aktiviert wird, muss in den Projektionspads der Bypass-Transistor geschaltet sein. Dazu benötigen die Projektionspads das $YPROJ$-, bzw. $XPROJ$-Signal. Diese Signale werden ebenfalls direkt an den Eingangspads abgegriffen und über Inverterketten aus drei Invertern verstärkt. Das $\overline{XPROJ}$-Signal wird allen Projektionspads für die CO_n-Signale und das $\overline{YPROJ}$-Signal denen für die LO_n-Signale zugeführt.

B.4.2 Das Layout des Gesamtchips

Aus den zuvor beschriebenen Komponenten wird schließlich der gesamte Chip zusammengesetzt. Das vollständige Layout ist in der Abbildung B.26 dargestellt. Links ist eine Beschriftung aufgebracht, die den Namen des Chips, das Entwurfsdatum und das Institut angibt. Die Größe des gesamten Chips beträgt etwa 3,26 mal 2,91 mm^2.
Der Chip wird in PLCC-68-Gehäuse gebondet. Die Tabelle B.1 enthält alle Pinbezeichnungen mit den entsprechenden Pinnummern. Dabei sind $D1$ bis $D16$ die Dateneingänge der 16 Zellen, $\overline{\mathrm{RAS}}$, $RA0$ und $RA1$ die Zeilenselektions und -adressleitungen, $\overline{\mathrm{CAS}}$, $CA0$ und $CA1$ die Spaltenselektions und -adressleitungen, $RO0$ bis $RO3$ die Zeilenprojektions- und $CO0$ bis $CO1$ die Spaltenprojektionsausgänge. Die Versorgungsspannung beträgt 5 V und wird über die Pins GND und Vdd zugeführt. Die Bezeichnung Substrat bezieht sich auf direkte Substratkontaktierungen. Die Pins mit der Bezeichnung Kas1 bis Kas16 dienen zur Kaskadierung der Chips. Die anderen Pinbezeichnungen entsprechen den Steuersignalen auf dem Chip.

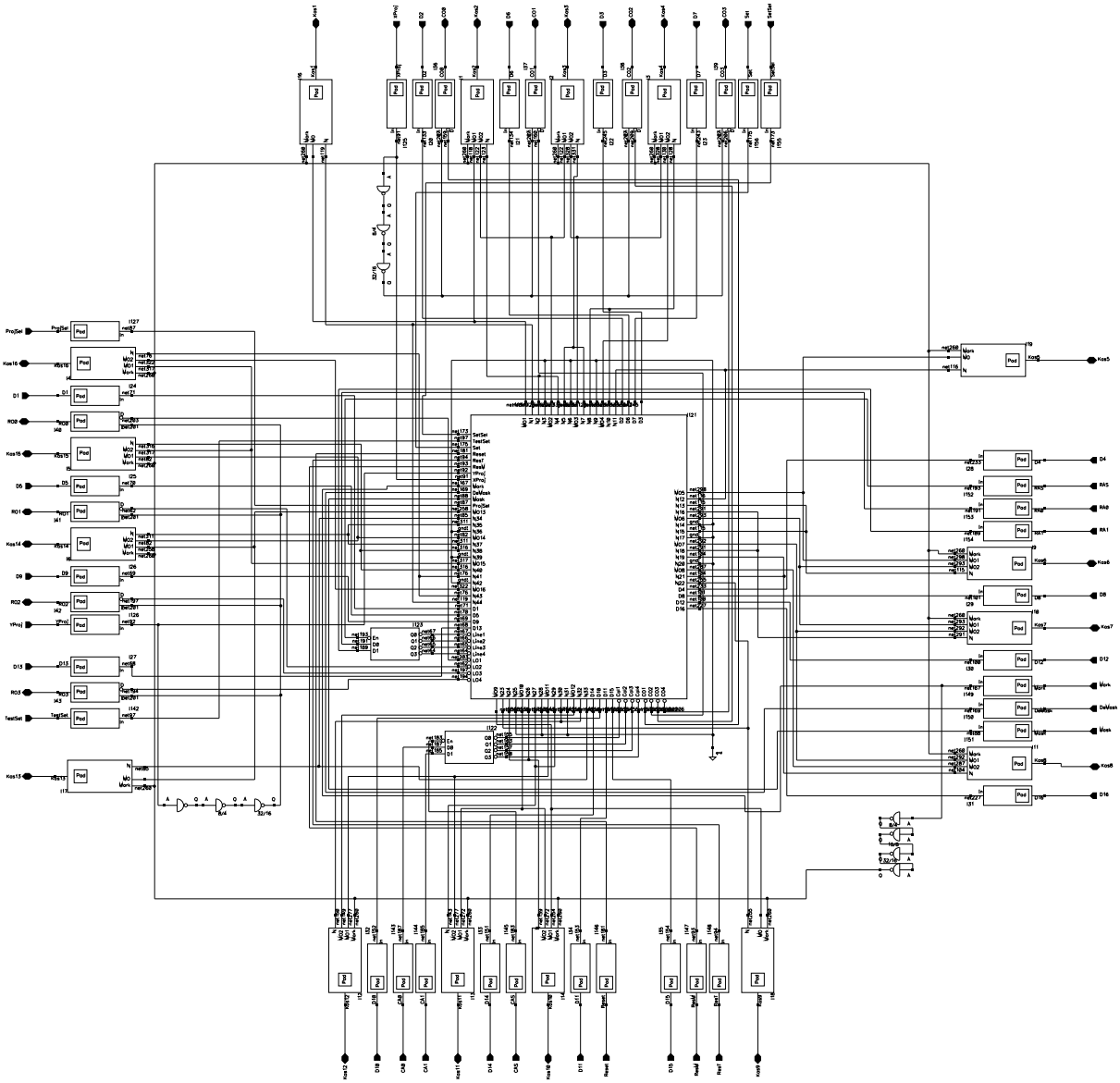


Abbildung B.25: Das Schaltbild des gesamten Chips

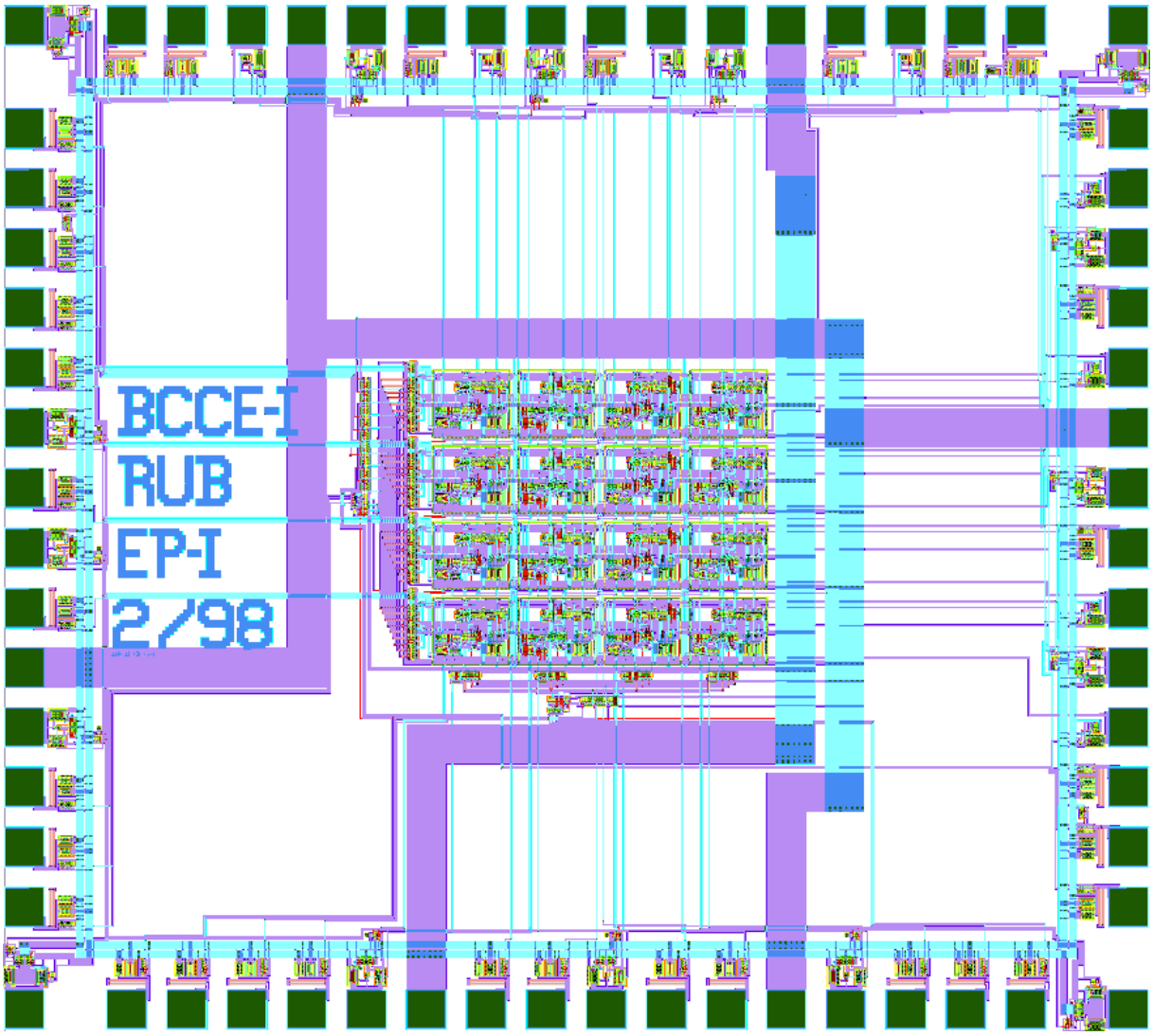


Abbildung B.26: Das Layout des gesamten Chips

Pinnummer	Pinbezeichnung
1	Kas3
2	CO1
3	D6
4	Kas2
5	GND
6	CO0
7	Reset
8	Set
9	Kas1
10	ResT
11	ResM
12	Mark
13	D1
14	Substrat
15	D5
16	Kas16
17	D9
18	Kas15
19	D13
20	GND
21	Kas14
22	Substrat
23	$\overline{\mathrm{RAS}}$
24	RA0
25	RA1
26	Kas13
27	TestSet
28	DeMask
29	Mask
30	ProjSel
31	Kas12
32	Vdd
33	D10
34	D14

Pinnummer	Pinbezeichnung
35	Kas11
36	D15
37	D11
38	GND
39	Kas10
40	$\overline{\mathrm{CAS}}$
41	CA0
42	CA1
43	Kas9
44	SetSel
45	YProj
46	D16
47	RO0
48	Substrat
49	Kas8
50	RO1
51	D12
52	Kas7
53	GND
54	RO2
55	D8
56	Substrat
57	Kas8
58	RO4
59	D4
60	Kas5
61	XProj
62	D7
63	CO3
64	D3
65	Vdd
66	Kas4
67	CO2
68	D2

Tabelle B.1: Pinbelegung des Zellularlogik-ASICs

Anhang C

Der Netzlistengenerator

Die Schaltung der Haupttriggerplatinen besteht im Wesentlichen aus einer regelmäßigen Anordnung identischer Schaltkreise, den Zellularlogik ASICs. Die Verbindungslisten solcher Schaltungen lassen sich recht einfach mit Computerprogrammen erzeugen, ohne dass eine große Gefahr besteht, dass einzelne Fehler auftreten, wie dies bei einer manuellen Erstellung über einen Schaltplan der Fall wäre. Die für den Entwurf der Platinen verwendete CAD-Software EAGLE der Firma CADSoft ermöglicht es, solche von einem Rechner erzeugten Bauteillisten und Netzlisten in Form von Script-Dateien einzulesen. Aus diesem Grund wurde für diese Arbeit ein Programm entwickelt, das die entsprechenden Netzlisten erzeugt. Dieses Programm wurde dabei weitestgehend flexibel gehalten. In einer Eingabedateil erhält das Programm eine Beschreibung der Schaltung. Dazu wurde eine einfach strukturierte Programmiersprache entwickelt, die vom Netzlistengenerator mit einem entsprechenden Interpreter abgearbeitet wird.
Bei der Abarbeitung werden entsprechend den in der Schaltungsbeschreibung vorkommenden Befehlen Bauteile und Verbindungen zwischen diesen Bauteilen angelegt. Um die Schaltungsbeschreibungen kompakt zu halten, wurden einige Schaltungsblöcke bereits vordefiniert. Dazu gehören einfache Strukturen wie Treiber, Inverter oder ODER-Verknüpfungen zweier Bussysteme, aber auch komplexere Strukturen, wie Prioritätsencoder und beliebige Matrizen der Zellularlogik.
Die Ausgabe des Netzlistengenerators erfolgt in einer Scriptdatei, die direkt von EAGLE eingelesen werden kann.

C.1 Elemente der Schaltungsbeschreibungssprache

C.1.1 Datentypen

Die für den Netzlistengenerator entwickelte Sprache enthält vier Datentypen: Integer-Zahlen, Bauteile, Netze und Busse, wobei die letzten drei Typen speziell auf diesen Anwendungsfall abgestimmt sind. Der Datentyp *Bauteil* beschreibt ein elektronisches Bauelement mit allen Eigenschaften, die für seine Erzeugung in einer EAGLE-Skriptdatei notwendig sind. Dazu gehören die EAGLE-Bauteilbibliothek, in der es zu finden ist, und sein Name innerhalb dieser Bibliothek, sowie der Name des Bauteils in der Schaltung und ein Bauteilwert. Ggf. kann auch noch der Ort, an dem dieses Bauteil auf der Platine platziert werden soll, und die Ausrichtung in diesem Datentyp gespeichert werden.

Der Datentyp Netz enthält neben einem Netznamen alle Bauteile und deren Anschlüsse, die durch dieses Netz verbunden werden. Mit einem Bus können mehrere logisch zusammengehörende Netze zusammengefasst werden. Im Netzlistengenerator wird dies so realisiert, dass die Namen aller Netze eines Busses einen gemeinsamen Präfix haben und fortlaufend durchnummeriert sind. In der Datenstruktur Bus wird dieser gemeinsame Präfix, sowie Anfang und Ende der Nummerierung gespeichert.

C.1.2 Konstanten und Variablen

Konstanten können in der Schaltungsbeschreibungssprache lediglich vom Typ Integer sein. Variablen dagegen können von allen vier in der Sprache bereit gestellten Typen sein. Sowohl Konstanten als auch Variablen müssen zu Beginn des Programmcodes deklariert werden und werden durch einen eindeutigen Konstanten- oder Variablennamen gekennzeichnet.

C.1.3 Anweisungen

Die für den Netzlistengenerator entwickelte Sprache kennt zwei Arten von Anweisungen: Zuweisungsanweisungen und Blockaufrufe. In Zuweisungsanweisungen wird einer Variablen ein neuer Wert zugewiesen. Dabei kann es sich um Integer-Ausdrücke, Netznamen und Bauteile handeln. Innerhalb dieser Zuweisungen können aber auch neue Verbindungen erzeugt oder Bauteile hinzugefügt werden, da diese Operationen stehts einen Rückgabewert des entsprechenden Datentyps erzeugen. Drei Funktionen stehen dazu zur Verfügung.

Verbinde_einen fügt einen Anschluss eines Bauteils zu einem Netz hinzu, dass durch den Netznamen oder eine Variable vom Typ Netz gegeben ist. Wird kein gültiger Netzname, sondern statt dessen ein leerer String als Netzname übergeben, generiert die Funktion einen neuen, bisher nicht benutzten Netznamen.

Verbinde_zwei verbindet zwei Anschlüsse eines oder zweier Bauteile. Dazu kann der Funktion wieder ein Netzname übergeben werden. Existiert dieser Netzname bereits, werden diese beiden neuen Knotenpunkte des Netzes zu allen bereits vorhandenen Knotenpunkten zusammengefügt. Wird dagegen wieder ein Leerstring als Netzname übergeben, wird ein neues Netz mit diesen beiden Knotenpunkten erzeugt.

Bauteil Ein Bauteil erfordert vier String-Konstanten als Parameter, die der Reihe nach für den Bibliotheksnamen, den Bauteilnamen in der Bibliothek, den Präfix des Bauteilnamens in der Schaltung und dem Bauteilwert stehen. Der Bauteilname, mit dem das somit erzeugte Bauteil in der Schaltung gekennzeichnet wird, setzt sich aus dem Präfix, der dieser Funktion als Parameter übergeben wird, und einer fortlaufenden Nummerierung zusammen, die von dieser Funktion selbstständig durchgeführt wird und dafür sorgt, dass die Bauteilnamen in der Schaltung eindeutig sind.
Ein besonderes Bauteil wurde mit dem Befehl *Board9U* in die Sprache implementiert. Dabei handelt es sich um die Umrandung einer VME-9U-Platine mit allen Steckern. Zudem können automatisch alle notwendigen Netze für die Kontaktierung des Zellularlogikbusses erzeugt werden.

Dazu ist als erster Parameter eine 1 und als zweiter Parameter ein Bus, der mit dem Zelladressbus verbunden werden soll, zu übergeben.

Die andere Art von Anweisungen, die die Programmstruktur zur Verfügung stellt, sind Blockaufrufe, mit denen bereits größere Schaltungsblöcke erzeugt werden können, die fest im Netzlistengenerator implementiert sind. Diese Blöcke werden im folgenden kurz beschrieben.

Inverter ist ein Block, der eine größere Anzahl von Invertern erzeugt. Als Parameter werden zwei Busse übergeben, die mit den Eingängen und Ausgängen der Inverter verbunden werden. Die Anzahl der Inverter wird durch den Start und den Stoppwert des Eingangsbusses bestimmt, vom Ausgangsbus wird nur der Startwert betrachtet und der Stoppwert entsprechend der Anzahl der Inverter neu gesetzt.

Treiber entspricht dem Block *Inverter*, jedoch werden hier nicht invertierende Treiber erzeugt.

NOR erzeugt eine Reihe von NOR-Gattern, deren Eingänge mit zwei Bussen und deren Ausgänge mit einem weiteren Bus verbunden werden, die als Parameter diesem Blockaufruf übergeben werden. Bestimmend für die Anzahl der Gatter ist wieder der erste Bus, bei den anderen Bussen wird der Stoppwert entsprechend angepasst.

PRIOENC . Dieser Block erzeugt einen Prioritätsencoder entsprechend der Schaltung, die in [Go98] beschrieben ist. Als Parameter werden zwei Busse für die Eingangs- und Ausgangssignale und ein Netzname für das GS-Signal der Encoder übergeben.

PRIOSEL erzeugt einen Prioritätsselektor, wie er für den Aufbau der Zellularlgik benötigt wird. Dieser Block hat zwei Eingangsbusse und zwei separate Eingangsnetze. Abhängig vom Zustand dieser beiden Eingangsnetze wird einer der beiden Eingangsbusse über einen in diesem Block befindlichen Multiplexer mit dem Ausgangsbus verbunden, der entsprechend der Funktion des Selektors über ein Netz mehr als die beiden Eingangsbusse verfügt. Der Zustand des Selektors wird über einen Ausgang signalisiert, der mit einem dritten Netz verbunden ist.

LATCH erzeugt einen Speicherbaustein aus D-Flipflops. Die Eingänge dieser Flipflops sind mit dem ersten Bus und die Ausgänge mit dem zweiten als Parameter übergebenen Bus verbunden. Die Steuereingänge *Latchen* und *Output-Enable* werden mit zwei weiteren Netzen verbunden, die ebenfalls als Parameter übergeben werden.

ZELLUL erzeugt eine komplette Zellularlogik-Matrix aus $i \cdot j$ ASICs. Als Parameter werden zunächst die beiden Werte i und j übergeben, anschließend die Busse für die Zeilenprojektionsausgänge, die Spaltenprojektionsausgänge, die Kaskadierungsanschlüsse für den oberen und den unteren Rand, sowie der Zelladressbus. Anschließend wird noch ein Netzname für das Zelladressenable-Signal und Flag übergeben, das durch die Werte 0 oder 1 angibt, ob sich die ASICs auf der Vorder- oder Rückseite der Platine befinden sollen. Diese Blockoperation erzeugt alle für die Zellularlogik notwendigen Bauteile, einschließlich der Adressdekoder, aller benötigten Pull-up-Widerstände und der Steckverbinder für die Datenzufuhr.

Adapter erzeugt die kompletten Netzlisten für die beiden Adapterplatinen. Welche der beiden Adapterplatinen benötigt wird, wird dem Netzlistengenerator mit zwei Flags mitgeteilt.

C.2 Die Syntax der Schaltungsbeschreibungssprache

Die Syntax der Sprache, mit der in der Eingabedatei des Netzlistengenerators die Schaltung beschrieben werden kann, wird hier in einer erweiterten Backus-Naur-Form (EBNF) [Wi86] wiedergegeben. Bei dieser Syntaxbeschreibung wird die Sprache als Abfolge von Symbolen betrachtet, wobei zwischen *terminalen* und *nicht terminalen* Symbolen unterschieden wird. Terminale Symbole, die im Folgenden in großen Buchstaben gesetzt sind, werden so als Bestandteile der Sprache benutzt. Einzelne Zeichen, die in der folgenden Syntaxbeschreibung in Anführungszeichen gesetzt werden, sind ebenfalls terminale Symbole. Nicht terminale Symbole, die in der Beschreibung kursiv gesetzt werden, sind dagegen als Platzhalter für eine komplexere Abfolge von anderen Symbolen zu verstehen.
Um die Abfolge der Symbole zu beschreiben, werden in der EBNF selbst einige Metasymbole benutzt, wie z.B. das |-Zeichen. Die Symbole, die durch dieses Metasymbol getrennt werden, können an dieser Stelle alternativ eingesetzt werden. Das |-Zeichen betrifft alle Symbole, die vor und nach dem Zeichen auftreten. Eine Folge ab|cd heißt also, dass alternativ ab oder cd in der Sprache auftreten können. Soll sich das |-Zeichen nur auf einen Teil der Symbolfolge beziehen, kann dies durch Klammern ausgedrückt werden. Die Folge a(b|c)d stellt also alternativ die beiden Folgen abd oder acd dar. Symbolfolgen, die in geschweiften Klammern { } auftreten, können beliebig oft wiederholt werden, wobei auch der Fall möglich ist, dass sie gar nicht auftreten. Die eckigen Klammern [ ] werden dazu benutzt, Symbolfolgen zu kennzeichnen, die gar nicht oder einmal auftreten. Die Syntaxregeln, die die Beschreibungssprache für den Netzlistengenerator definieren, sind in Tabelle C.1 aufgeführt. Die oberste Ebene ist dabei das nicht terminale Symbol *schaltung*, das die gesamten Sprachregeln beinhaltet. Die nicht terminalen Symbole *buchstabe*, *ziffer* und *zeichen* sind trivial, sodass sie nicht weiter aufgeschlüsselt werden. Beim Symbol *buchstabe* wird nicht zwischen Groß- und Kleinbuchstaben unterschieden. Für *zeichen* können alle Zeichen des ASCII-Zeichensatzes eingesetzt werden.

C.3 Die Programmstruktur

Die Programmstruktur des Netzlistengenerators ist im Blockdiagramm in Abbildung C.1 dargestellt. Zentraler Teil des Programmes sind zwei unabhängige Module, die Listen der verwendeten Bauteile und aller Verbindungen zwischen diesen Bauteilen verwalten. Diese Module stellen sowohl Funktionen zur Erzeugung von Bauteilen und Verbindungen zur Verügung, als auch die Möglichkeit, aus diesen Listen entsprechende Scriptdateien zu erzeugen. Letzteres geschieht in Verbindung mit der Script-Ausgabe. Das Anlegen der Bauteil- und Netzlisten geschieht gesteuert vom Programm in der Eingabedatei durch den Interpreter, der dabei auf die Hilfe eines Programmmoduls zurückgreifen kann, das die Schaltungsblöcke erzeugen kann. Der gesamte Netzlistengenerator ist in der Programmiersprache Pascal implementiert.

schaltung	=	SCHALTUNG ’=’ *bezeichner* ’;’ *block* ’.’
block	=	[KONST {*bezeichner* ’=’ *zahl* ’;’ }] [VAR { *bezeichner* { ’,’ *bezeichner* } ’:’ (NETZ | BUS | INTEGER | BAUTEIL) ’;’ }] BEGINN *anweisung* { ’;’ *anweisung* } ENDE
anweisung	=	(*bezeichner* ’:=’ *ausdruck* | *netzname* | *bauteil*) |(*bezeichner* ’.’ START | STOPP ’:=’ *ausdruck*) | *blockaufruf*
blockaufruf	=	(INVERTER ’(’ *busname* ’,’ *busname* ’)’ | (NOR ’(’ *busname* ’,’ *busname* ’,’ *busname* ’)’ | (PRIOENC ’(’ *busname* ’,’ *busname* ’,’ *netzname* ’)’ | (PRIOSEL ’(’ *busname* ’,’ *busname* ’,’ *netzname* ’,’ *netzname* ’,’ *busname* ’,’ *netzname* ’)’ | (ZELLUL ’(’ *ausdruck* ’,’ *ausdruck* ’,’ *busname* ’,’ *busname* ’,’ *busname* ’,’ *busname* ’,’ *busname* ’,’ *netzname* ’,’ *ausdruck* ’)’ | (TREIBER ’(’ *busname* ’,’ *busname* ’)’ | (LATCH ’(’ *busname* ’,’ *busname* ’,’ *netzname* ’,’ *netzname* ’)’ | (ADAPTER ’(’ *ausdruck* ’,’ *ausdruck* ’)’
netzname	=	*string* | *bezeichner* [’[’ *ausdruck* ’]’] | *verbinde_einen* | *verbinde_zwei*
verbinde_einen	=	VERBINDE_EINEN ’(’ *netzname* ’,’ *bauteil* ’,’ *ausdruck* ’)’
verbinde_zwei	=	VERBINDE_ZWEI ’(’ *netzname* ’,’ *bauteil* ’,’ *ausdruck* ’,’ *bauteil* ’,’ *ausdruck* ’)’
bauteil	=	(BAUTEIL ’(’ *string* ’,’ *string* ’,’ *string* ’,’ *string* ’)’) | *bezeichner* | *board9u*
board9u	=	BOARD9U ’(’ *ausdruck* ’,’ *busname* ’)’
ausdruck	=	*summand* { (+|-) *summand* }
summand	=	*faktor* { (*|/) *faktor* }
faktor	=	(*bezeichner* [’.’ (START | STOPP)]) | *zahl* | (’(’ *ausdruck* ’)’)
busname	=	*bezeichner*
bezeichner	=	*buchstabe* { *buchstabe* | *ziffer* | ’_’ }
zahl	=	*ziffer* { *ziffer* }
string	=	’“ { *zeichen* } ’“

Tabelle C.1: Sprachregeln der Schaltungsbeschreibungssprache des Netzlistengenerators

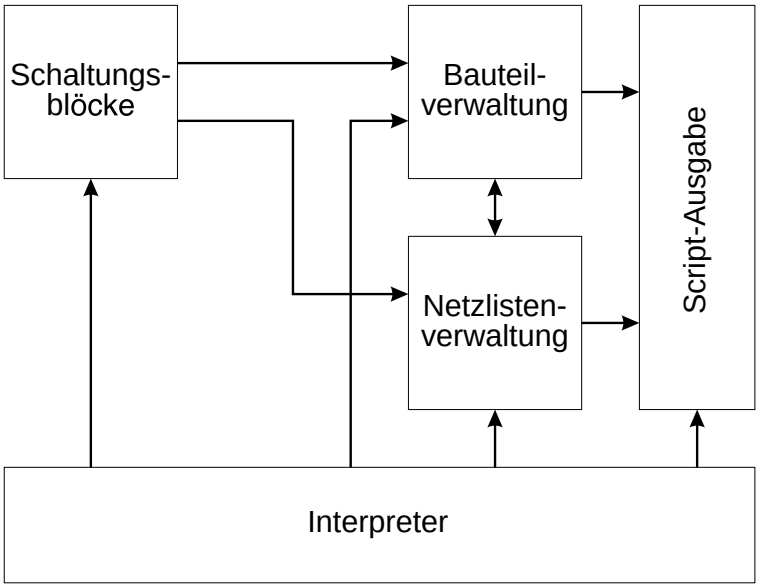


Abbildung C.1: Die Programmstruktur des Netzlistengenerators

C.4 Programm des Netzlistengenerators für die erste Triggerplatine

```
schaltung pcb1;

  var  adr                 : bus;
       rout,cout,kas1,kas2 : bus;
       select              : netz;
       cbuf                : bus;
       radr,rbuf           : bus;
       status              : netz;
       dname               : netz;
       bord                : bauteil;

  beginn
    adr := "AD";
    adr.start := 0;
    bord := board9u(1,adr);

    rout := "RO";
    rout.start := 0;

    cout := "CO";
    cout.start := 0;
    cout.stopp := 31;

    kas1 := "KTOP";
    kas1.start := 0;
```

```
kas2 := "KBOT";
kas2.start := 0;

select := "ADREN";

zellul(8,8,rout,cout,kas1,kas2,adr,"ADREN",0);

cbuf := "COB";
cbuf.start := 0;
cbuf.stopp := 31;

inverter(cout,cbuf);

radr := "RADR";
radr.start := 0;
radr.stopp := 5;

status := radr[5];


prioenc(rout,radr,status);

dname := verbinde_einen(cbuf[0],bord,201);
dname := verbinde_einen(cbuf[1],bord,202);
dname := verbinde_einen(cbuf[2],bord,203);
dname := verbinde_einen(cbuf[3],bord,204);
dname := verbinde_einen(cbuf[4],bord,205);
dname := verbinde_einen(cbuf[5],bord,206);
dname := verbinde_einen(cbuf[6],bord,207);
dname := verbinde_einen(cbuf[7],bord,208);
dname := verbinde_einen(cbuf[8],bord,209);
dname := verbinde_einen(cbuf[9],bord,210);
dname := verbinde_einen(cbuf[10],bord,211);
dname := verbinde_einen(cbuf[11],bord,212);
dname := verbinde_einen(cbuf[12],bord,213);
dname := verbinde_einen(cbuf[13],bord,214);
dname := verbinde_einen(cbuf[14],bord,215);
dname := verbinde_einen(cbuf[15],bord,216);
dname := verbinde_einen(cbuf[16],bord,217);
dname := verbinde_einen(cbuf[17],bord,218);
dname := verbinde_einen(cbuf[18],bord,219);
dname := verbinde_einen(cbuf[19],bord,220);
dname := verbinde_einen(cbuf[20],bord,221);
dname := verbinde_einen(cbuf[21],bord,222);
```

```
    dname := verbinde_einen(cbuf[22],bord,223);
    dname := verbinde_einen(cbuf[23],bord,224);
    dname := verbinde_einen(cbuf[24],bord,225);
    dname := verbinde_einen(cbuf[25],bord,226);
    dname := verbinde_einen(cbuf[26],bord,227);
    dname := verbinde_einen(cbuf[27],bord,228);
    dname := verbinde_einen(cbuf[28],bord,229);
    dname := verbinde_einen(cbuf[29],bord,230);
    dname := verbinde_einen(cbuf[30],bord,231);
    dname := verbinde_einen(cbuf[31],bord,232);

    dname := verbinde_einen(radr[0],bord,265);
    dname := verbinde_einen(radr[1],bord,266);
    dname := verbinde_einen(radr[2],bord,267);
    dname := verbinde_einen(radr[3],bord,268);
    dname := verbinde_einen(radr[4],bord,269);
    dname := verbinde_einen(radr[5],bord,270)


  ende.
```

C.5 Programm des Netzlistengenerators für die zweite Triggerplatine

```
schaltung pcb2;

  var adr,rout,cout,kas1,kas2              : bus;
      cbuf1,cbuf2,cbuf3                    : bus;
      radr1,radr2,unladr                   : bus;
      select,dname                         : netz;
      x_status,y_status1,y_status2,y_status : netz;
      latchen,encen                        : netz;
      bord                                 : bauteil;

  beginn
    adr := "AD";
    adr.start := 0;
    adr.stopp := 10;
    bord := board9u(1,adr);

    rout := "RO";
    rout.start := 0;

    cout := "CO";
```

```
cout.start := 0;
cout.stopp := 31;

kas1 := "KTOP";
kas1.start := 0;

kas2 := "KBOT";
kas2.start := 0;

select := "ADREN";

zellul(8,7,rout,cout,kas1,kas2,adr,"ADREN",1);

cbuf1 := "COB1";
cbuf1.start := 0;
cbuf1.stopp := 31;

cbuf2 := "COB2";
cbuf2.start := 0;
cbuf2.stopp := 31;

cbuf3 := "COB3";
cbuf3.start := 0;
cbuf3.stopp := 31;

inverter(cout,cbuf1);


dname := verbinde_einen(cbuf2[0],bord,201);
dname := verbinde_einen(cbuf2[1],bord,202);
dname := verbinde_einen(cbuf2[2],bord,203);
dname := verbinde_einen(cbuf2[3],bord,204);
dname := verbinde_einen(cbuf2[4],bord,205);
dname := verbinde_einen(cbuf2[5],bord,206);
dname := verbinde_einen(cbuf2[6],bord,207);
dname := verbinde_einen(cbuf2[7],bord,208);
dname := verbinde_einen(cbuf2[8],bord,209);
dname := verbinde_einen(cbuf2[9],bord,210);
dname := verbinde_einen(cbuf2[10],bord,211);
dname := verbinde_einen(cbuf2[11],bord,212);
dname := verbinde_einen(cbuf2[12],bord,213);
dname := verbinde_einen(cbuf2[13],bord,214);
dname := verbinde_einen(cbuf2[14],bord,215);
dname := verbinde_einen(cbuf2[15],bord,216);
dname := verbinde_einen(cbuf2[16],bord,217);
```

```
dname := verbinde_einen(cbuf2[17],bord,218);
dname := verbinde_einen(cbuf2[18],bord,219);
dname := verbinde_einen(cbuf2[19],bord,220);
dname := verbinde_einen(cbuf2[20],bord,221);
dname := verbinde_einen(cbuf2[21],bord,222);
dname := verbinde_einen(cbuf2[22],bord,223);
dname := verbinde_einen(cbuf2[23],bord,224);
dname := verbinde_einen(cbuf2[24],bord,225);
dname := verbinde_einen(cbuf2[25],bord,226);
dname := verbinde_einen(cbuf2[26],bord,227);
dname := verbinde_einen(cbuf2[27],bord,228);
dname := verbinde_einen(cbuf2[28],bord,229);
dname := verbinde_einen(cbuf2[29],bord,230);
dname := verbinde_einen(cbuf2[30],bord,231);
dname := verbinde_einen(cbuf2[31],bord,232);

nor(cbuf1,cbuf2,cbuf3);

unladr := "UNLADR";
unladr.start := 0;
unladr.stopp := 4;

x_status := "XEMPTY";

prioenc(cbuf3,unladr,x_status);

radr1 := "RADR1";
radr1.start := 0;
radr1.stopp := 4;

y_status1 := "YSTAT1";

prioenc(rout,radr1,y_status1);

radr2 := "RADR2";
radr2.start := 0;
radr2.stopp := 4;

y_status2 := "YSTAT2";

dname := verbinde_einen(radr2[0],bord,265);
dname := verbinde_einen(radr2[1],bord,266);
dname := verbinde_einen(radr2[2],bord,267);
dname := verbinde_einen(radr2[3],bord,268);
dname := verbinde_einen(radr2[4],bord,269);
```

```
    dname := verbinde_einen(y_status2,bord,270);

    unladr.start := 5;
    unladr.stopp := 10;

    y_status := "YEMPTY";

    priosel(radr2,radr1,y_status2,y_status1,unladr,y_status);

    unladr.start := 0;


    latchen := "LATCHEN";
    encen := "ENCEN";

    latch(unladr,adr,latchen,encen)

  ende.
```

C.6 Programm des Netzlistengenerators für die erste Adapterplatine

```
schaltung adapter1;

  var brd : bauteil;
      adr : bus;

  beginn
    brd := board9u(0,adr);
    adapter(0,1)
  ende.
```

C.7 Programm des Netzlistengenerators für die zweite Adapterplatine

```
schaltung adapter2;

  var brd : bauteil;
      adr : bus;

  beginn
    brd := board9u(0,adr);
    adapter(1,1)
```

```
ende.
```

Anhang D

Das Controller-Board

Zur Steuerung der gesamten Zellularlogik dient ein Controller-Board, das als VMEbus Modul aufgebaut wurde. Dieses Modul gliedert sich in eine Reihe von Baugruppen auf, wobei die wesentlichen der Sequencer, der von A. GOLDSCHMIDT in seiner Diplomarbeit [Go98] entwickelt wurde, und das VME-Interface sind. In Abbildung D.1 ist derjenige Teil des Schaltplans des Controllers zu sehen, das das VMEbus-Interface darstellt.

D.1 Das VMEbus-Interface

Auf dem Schaltbild ist auf der linken Seite der Steckverbinder zum VMEbus-System zu erkennen. Das Controllerboard benutzt lediglich den Stecker J1 des VMEbus. Damit ist der Adressraum auf 24 Bits und der Datentransfer auf 16 Bits begrenzt, was für den Controller jedoch keine Beschränkung darstellt. Die Spannungsversorgung des Boards erfolgt aus dem VMEbus–System. Es werden $+5\,\mathrm{V}$ und $-12\,\mathrm{V}$ benötigt. Die Spannungen werden über ein pi-Filter, bestehend aus zwei Elektrolytkondensatoren mit 22 $\mu\mathrm{F}$ und einer UKW-Breitbanddrossel gefiltert und dann auf dem Board verteilt. Oben ist rechts neben den Steckverbindern der Adressdekoder, der die oberen 16 Bits der VMEbus-Adresse mit einer auf zwei DIP-Switches S1 und S2 eingestellten Basisadresse vergleicht. Der Dekoder besteht aus zwei schnellen 8-Bit-Komparatoren vom Typ 74F521 (IC1 und IC2). Das Vergleichsergebnis liegt bei beiden Bausteinen in invertierter Logik an Pin 19 an und wird mit dem Oder-Gatter IC3A zusammengefasst. Das Ergebnis wird in einen programmierbaren Logikbaustein der Firma Lattice vom Typ ispLSI 1016E übertragen. In diesem Baustein ist die gesamte VMEbus-Logik für Datenzugriffe implementiert.
Die Logik wird in einer Hardware-Beschreibungssprache definiert, die vom Lattice Entwicklungssystem für die ispLSI-Bausteine in eine sogenannte Fuse-map umgesetzt wird. Diese Fusemap enthält alle Informationen über Verbindungen, die zwischen den Gatefunktionen des Chips geschaffen werden müssen. Mit einem speziellen Programmieradapter kann diese Fuse-map dann in den Chip geladen werden.
Der Quellcode für die Logikbeschreibung beginnt mit der Definition aller Ein- und Ausgänge. Dabei werden auch interne Knotenpunkte definiert, die nicht nach außen geführt werden. Diese Punkte erhalten in der Deklaration die Bezeichnung *node*.

```
MODULE VME_DTB

TITLE 'VME-Interface, DTB-Slave'
```

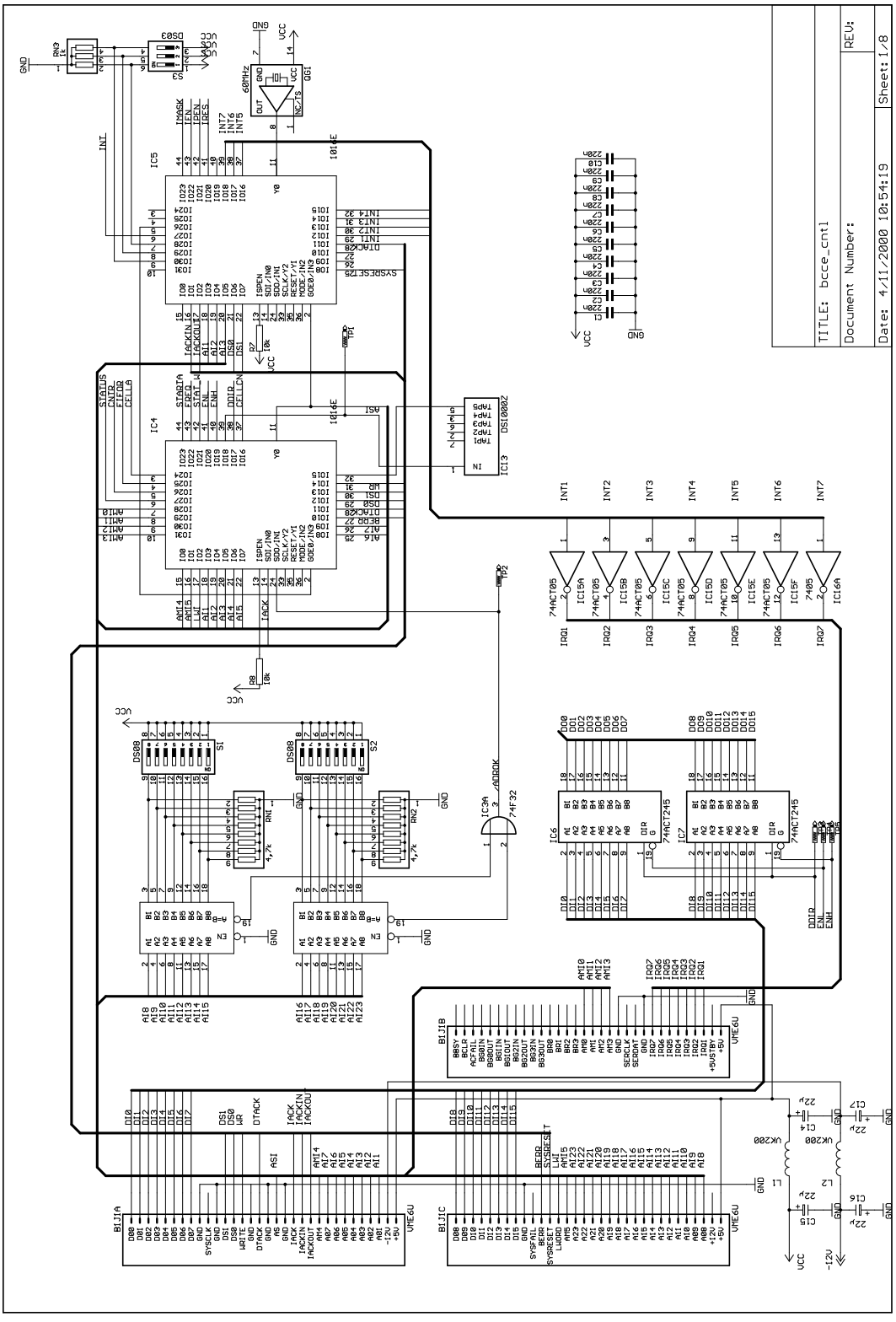


Abbildung D.1: Blatt 1 des Schaltbildes des Zellularlogik-Controller-Boards. Dargestellt ist hier im Wesentlichen das VMEbus-Interface

```
!AS             PIN     11;

A1,A2,A3,A4     PIN     18,19,20,21;
A5,A6,A7        PIN     22,25,26;

AM0,AM1,AM2,AM3 PIN     7,8,9,10;
AM4,AM5         PIN     15,16;

!LWORD          PIN     17;


!IACK           PIN     14;

!AVALID         PIN     24;

!DS0,!DS1       PIN     29,30;

!WRITE          PIN     31;

delay           PIN     32;

!DTACK          PIN     28      istype 'com';

!BERR           PIN     27      istype 'com';

select          node    istype 'reg';

sa0,sa1,sa2     node    istype 'reg';

wr              node    istype 'reg';

dtran           node    istype 'collapse';

dt_fin          node    istype 'collapse';

dtrans          node    istype 'reg';

d_start         PIN     39      istype 'reg';

STATUS_R        PIN     6       istype 'com';

CNTR            PIN     5       istype 'com';

!FIFOR          PIN     4       istype 'com';

CELLA           PIN     3       istype 'com';

STARTA          PIN     44      istype 'com';

FREQ            PIN     43      istype 'com';

STATUS_W        PIN     42      istype 'com';

!CELLCN         PIN     37      istype 'com';

!ENL,!ENH       PIN     41,40   istype 'com';

DIR             PIN     38      istype 'com';

IACYC           PIN     2;
```

Anschließend werden einige Signale zu Gruppen zusammengefasst. Mit diesen Signalgruppen können gemeinsame Operationen durchgeführt werden.

```
AM = [AM5,AM4,AM3,AM2,AM1,AM0];
```

```
AL = [A3,A2,A1];

AH = [A7,A6,A5,A4];

sa = [sa2,sa1,sa0];
```

Mit dem Schlüsselwort *equations* werden danach die Logikgleichungen eingeleitet, die mit der Definition des internen Knotenpunktes *select* beginnen.

```
select.d = AVALID & (AM == ^h3d) & !IACK & (AH == 0)
         # AVALID & (AM == ^h39) & !IACK & (AH == 0);

select.clk = AS;

select.ar = delay;
```

Bei diesem Knotenpunkt handelt es sich nach der Deklaration um ein D-Flipflop. Der mit *select.d* bezeichnete Dateneingang des Flipflops wird aktiv, wenn eine gültige Basisadresse erkannt wurde (*AVALID*), kein Interrupt-Acknowlege-Zyklus abläuft, die Adressbits A4 bis A7 low sind (AH) und ein Adressmodifier-Code von 39h oder 3dh anliegt. Dabei handelt es sich um die Adressmodifiercodes für Standard Datenzugriffe im User, bzw. Supervisormodus. Als Clock-Signal für dieses Flipflop dient das Adress-Strobe-Signal AS. Das $delay$-Signal, das aktiv wird, wenn der Buszyklus eingeleitet wurde, setzt das Flipflop wieder zurück.

```
dtran = select & DS0 & !DTACK.PIN & !BERR.PIN
      # select & DS1 & !DTACK.PIN & !BERR.PIN;

dt_fin = dtrans & !DS0 & !DS1;
```

Zwei weitere interne Knotenpunkte $dtran$ und dt_fin signalisieren, dass der Buszyklus nun startet, bzw. beendet wird. $dtran$ wird aktiv, wenn die Karte selektiert wurde, mindestens eines der beiden Data-Strobe-Signale $DS0$ und $DS1$ aktiv ist und weder $DTACK$ noch $BERR$ aktiv sind. Der Test auf den Zustand der beiden Leitungen $DTACK$ und $BERR$ ist unbedingt notwendig, um auch mit Busmaster-Karten arbeiten zu können, die Address-Pipelining verwenden. Sobald beide Data-Strobe-Signale wärend eines Datentransfers wieder inaktiv werden, wird der Datentransfer mit dem *dt_fin*-Signal wieder beendet.

```
sa.d = AL;

sa.clk = dtran;

wr.d = WRITE;

wr.clk = dtran;
```

Sobald der Buszyklus eingeleitet wird, müssen die Zustände der unteren drei Adressleitungen (AL), sowie das Write-Signal zwischengespeichert werden, da ihr Zustand auf dem VMEbus nur zu diesem Zeitpunkt definiert ist. Dies geschieht in den Registern sa und wr, deren Clock-Eingang mit dem $dtran$-Signal gespeist wird.

```
dtrans.d = 1;

dtrans.clk = dtran;

dtrans.ar = dt_fin;
```

Im $dtrans$-Flipflop (nicht zu verwechseln mit dem $dtran$-Signal) wird festgehalten, ob momentan ein Buszyklus läuft. Gesetzt wird es vom $dtran$-Signal, wogegen es vom dt_fin-Signal zurückgesetzt wird.
Zwischen dem Start eines Bustransfers und dem Aktivieren der Acknowlege-Leitungen durch den Bus-Slave muss eine Mindestverzögerungszeit liegen. Diese Verzögerung wird durch die aktive Verzögerungsleitung DS1000 (IC13) der Firma Dallas Semiconductor kontrolliert. Aktiviert wird diese Verzögerungsleitung mit dem Ausgang d_start, der gesetzt wird, sobald $dtrans$ aktiv wird.

```
d_start.d = 1;

d_start.clk = dtrans;

d_start.ar = dt_fin;
```

Die Trennung von $dtrans$ und d_start, verbunden mit der Verwendung eines weiteren Flipflops, das vom Signal durchlaufen werden muss, bringt einige Nanosekunden weitere Verzögerungszeit. Der Ausgang der Verzögerungsleitung ist mit dem Eingang $delay$ verbunden. Der nächste Abschnitt in den Logikgleichungen beschäftigt sich mit dem Data Acknowlegement-Signal $DTACK$ und dem Buss-Error-Signal $BERR$.

```
BERR.OE = delay & dtrans & LWORD
        # delay & dtrans & wr & (sa == 4)
        # delay & dtrans & wr & (sa == 5)
        # delay & dtrans & wr & (sa == 6)
        # delay & dtrans & !wr & (sa == 1)
        # delay & dtrans & !wr & (sa == 2)
        # delay & dtrans & !wr & (sa == 3);

DTACK.OE = delay & dtrans & !LWORD & wr & (sa == 1)
         # delay & dtrans & !LWORD & wr & (sa == 2)
         # delay & dtrans & !LWORD & wr & (sa == 3)
         # delay & dtrans & !LWORD & !wr & (sa == 4)
         # delay & dtrans & !LWORD & !wr & (sa == 5)
         # delay & dtrans & !LWORD & !wr & (sa == 6)
         # delay & dtrans & !LWORD & (sa == 0);

DTACK = 1;
BERR = 1;
```

Bei beiden Ausgängen handelt es sich um Open-Drain-Ausgänge, die durch Tristate-Ausgänge realisiert werden, deren Ausgangspegel fest auf low gelegt sind. Die eigentliche Ausgangsschaltfunktion nimmt dann das Output-Enable-Signal dieses Ausgangs wahr. Zu beachten ist dabei, dass in den Logikgleichungen eine 1 als Ausgangspegel angegeben wird, da beide Ausgänge in der Pin-Deklaration in negativer Logik deklariert sind. $DTACK$ wird immer dann aktiviert, wenn ein gültiger Datentransfer stattgefunden hat. Dazu ist es erforderlich, dass ein Datentransfer aktiv ist, die Verzögerungszeit abgelaufen ist und kein Langwort-Transfer aktiviert wurde. Des Weiteren muss auf eines der 7 Register in richtiger Datentransferrichtung zugegriffen worden sein.
Wurde dagegen ein Langwort-Transfer eingeleitet oder in falscher Datentransferrichtung auf ein Register zugegriffen, dann wird das $BERR$-Signal aktiviert, dass dem Bus-Master anzeigt, dass ein Busfehler aufgetreten ist.
Während ein Datentransfer läuft, muss eine Verbindung zwischen dem Datenbus des VMEbus und dem internen Datenbus auf der Controller-Karte geschaffen werden. Dazu gibt es die beiden Bustreiber IC6 und IC7 vom Typ 74ACT245. Diese beiden Treiber erhalten die beiden Steuersignale ENL und ENH, sowie das gemeinsame Steuersignal DIR, die von dem folgenden Abschnitt der Logikgleichungen erzeugt werden

```
ENL = dtrans & DS0 # IACYC & DS0;

ENH = dtrans & DS1 # IACYC & DS1;

// Die Richtung des Datentransfers

DIR = wr & !IACYC;
```

ENL wird aktiviert, wenn ein Datentransfer läuft und dabei $DS0$ aktiv ist, ENH dagegen wird aktiv, wenn dabei $DS1$ aktiv ist. Die beiden Bustreiber werden aber auch mit den Enable-Signalen aktiviert, wenn ein Interrupt-Bestätigungszyklus läuft. Dazu erhält IC4 das $IACYC$-Signal von IC5, das die Interrupt-Verarbeitung übernimmt.

```
STATUS_R = dtrans & !wr & (sa == 0)
         # IACYC;

CNTR = dtrans & !wr & (sa == 5);

FIFOR = dtrans & !wr & (sa == 4);

CELLA = dtrans & wr & delay & (sa == 3);

STARTA = dtrans & wr & delay & (sa == 2);

FREQ = dtrans & wr & delay & (sa == 1);

STATUS_W = dtrans & wr & delay & (sa == 0);

CELLCN = dtrans & !wr & (sa == 6 );
```

Zum Schluss wird die Ansteuerung der Ausgänge, mit denen die Register auf der Karte aktiviert werden, aufgeführt. Bei Lesezugriffen auf Register werden die entsprechenden Steuersignale sofort nach dem Einleiten des Datentransfers aktiviert, wogegen bei Schreibzugriffen zusätzlich noch die durch die Verzögerungsleitung bestimmte Verzögerungszeit abgewartet wird, damit zum Zeitpunkt, an dem das Register die Daten übernimmt, diese bereits stabil auf dem internen Datenbus der Karte liegen. Der lesende Zugriff auf das Statusregister erfolgt zudem auch dann, wenn ein Interruptbestätigungszyklus läuft. Der Quelltext wird abgeschlossen mit dem Schlüsselwort *END*.
Direkt neben IC4 befindet sich auf dem Schaltbild IC5, das für die Bearbeitung von Interruptzyklen zuständig ist. Auch dabei handelt es sich um einen ispLSI1016 von Lattice. Die Logikbeschreibung dieses ICs beginnt wieder mit der Deklaration aller Pins.

```
MODULE VME_INT

TITLE 'VME Interrupt Sysbsystem'

IRQA0,IRQA1     PIN     7,8;
IRQA2           PIN     9;

CLK             PIN     11;

!IACKIN         PIN     16;
!IACKOUT        PIN     17      istype 'reg';

A1,A2,A3        PIN     18,19,20;

!DS0,!DS1       PIN     21,22;

!AS             PIN     2;

DTACK           PIN     28      istype 'reg';

IRQ1,IRQ2,IRQ3  PIN     29,30,31 istype 'com';
IRQ4,IRQ5,IRQ6  PIN     32,37,38 istype 'com';
IRQ7            PIN     39       istype 'com';

IACYC           PIN     5       istype 'reg';

INT             PIN     6;

int_pending     PIN     42      istype 'reg';

int_mask        PIN     44;
int_en          PIN     43;
int_reset       PIN     41;

sys_reset       PIN     25;

freig           node    istype 'collapse';

dc0,dc1,dc2     node    istype 'reg';

del1,del2       node    istype 'collapse';

IP,ak           node    istype  'reg';
```

Zudem werden wieder zwei Gruppen definiert.

```
irqa = [IRQA2,IRQA1,IRQA0];

ai = [A3,A2,A1];
```

Danach beginnen die Logikgleichungen, wobei zunächst das Interrupt-Pending-Flipflop definiert wird, das eine Interruptaufforderung der Controller-Karte speichert. Damit es gesetzt werden kann, muss allerdings der Interrupt-Enable-Eingang int_en aktiv sein. Das Flipflop wird zurückgesetzt, wenn ein System-Reset erfolgt, speziell der Interrupt mit dem int_reset-Signal zurückgesetzt wird oder ein Interrupt-Bestätigungszyklus erfolgt.

```
equations

int_pending.d = 1;

int_pending.clk = INT & int_en;

int_pending.ar = sys_reset # int_reset # ak;
```

Sobald das Interrupt-Pending-Flipflop gestzt ist und der Interrupt-Maskierungseingang int_mask aktiv ist, wird, abhängig von dem Wert, der an den drei Interrupt-Adresseingängen $IRQA0$ bis $IRQA2$ anliegt, einer der 7 Interrupt-Ausgänge aktiviert.

```
IRQ1 = int_pending & int_mask & (irqa == 1);

IRQ2 = int_pending & int_mask & (irqa == 2);

IRQ3 = int_pending & int_mask & (irqa == 3);

IRQ4 = int_pending & int_mask & (irqa == 4);

IRQ5 = int_pending & int_mask & (irqa == 5);

IRQ6 = int_pending & int_mask & (irqa == 6);

IRQ7 = int_pending & int_mask & (irqa == 7);
```

An den Eingängen $IRQA0$ bis $IRQA2$ ist der DIP-Switch S3 angeschlossen, mit dem von Hand auf der Karte die VME-Interrupt-Ebene eingestellt werden kann. Die Interruptausgänge gehen auf Open-Drain-Treiber IC15 und IC16 vom Typ 74ACT05, deren Ausgänge mit den Interruptleitungen des VMEbus verbunden sind.

```
freig = IACKIN & DS0 # IACKIN & DS1;

dc0.d = freig & !dc0;

dc1.d = freig & !dc1 & dc0
      # freig & dc1 & !dc0;

dc2.d = freig & !dc2 & dc1 & dc0
      # freig & dc2 & !dc1
      # freig & dc2 & !dc0;
```

```
dc0.clk = CLK;
dc1.clk = CLK;
dc2.clk = CLK;
```

$dc0$ bis $dc2$ definieren einen 3-bit-Zähler, der losläuft, sobald der Knoten $freig$ aktiv wird. Dies ist der Fall, wenn der $IACKIN$ Eingang und eines der beiden Data-Strobe-Signale aktiv werden und somit ein Interrupt-Bestätigungszyklus erkannt wird. Die Taktfrequenz für den Zähler erzeugt der 60-MHz-Quarzoszillator QG1. Eine Taktperiode dauert somit 16,6 ns. Abgeleitet werden davon zwei Verzögerungssignale $del1$, 66,4 ns, und $del2$, 83 ns nachdem ***freig*** aktiv wurde.

```
del1 = dc2 & !dc1 & !dc0;
del2 = dc2 & !dc1 & dc0;
```

Um Interrupt-Bestätigungszyklen bearbeiten zu können, wird zunächst ein weiteres internes Flipflop IP definiert, das anzeigt, dass eine Interruptleitung aktiviert wurde. Dieses Flipflop wird stets gesetzt, wenn eines der beiden Data-Strobe-Signale aktiviert wird.

```
IP.d = int_pending & int_mask;
IP.clk = DS0 # DS1;

IACKOUT.d = !IP # (ai != irqa);
IACKOUT.clk = del2;
IACKOUT.ar = !AS;
```

Das $IACKOUT$-Signal gehört zu den Daisy-Chain-Signalen $IACKIN$ und $IACKOUT$, mit denen der Busmaster das Interrupt-Bestätigungssignal $IACK$ entlang einer Prioritätsfolge von einem Slave auf dem VMEbus zum nächsten transferiert. $IACKOUT$ muss vom Slave aktiviert werden, wenn am $IACKIN$-Eingang ein Signal anliegt und die Karte selber nicht angesprochen wird.
Dies ist der Fall, wenn die Karte selbst keinen Interrupt angefordert hat oder wenn der Interrupt-Level, der vom Master auf den untersten drei Adressleitungen des VMEbus angezeigt wird, nicht mit dem Interrupt-Level der Karte übereinstimmt. $IACKOUT$ ist als D-Flipflop realisiert, das gesetzt wird, wenn die Verzögerungszeit $del2$ abgelaufen ist und erst dann zurückgesetzt wird, wenn der Zyklus mit der Deaktivierung des Adress-Strobe-Signals *AS* beendet wird.

```
IACYC.d = IP & (ai == irqa);
IACYC.clk = dc2 & !dc1 & !dc0;
IACYC.ar = !DS0 & !DS1;
```

Wird die Karte dagegen vom aktuellen Interruptzyklus angesprochen, d.h. die Karte hat selbst einen Interrupt angefordert und der vom Master angezeigte Interruptlevel stimmt mit dem der Karte überein, wird mit dem $IACYC$-Signal, das zur VMEbus-Datentransferlogik in IC4 weitergeleitet wird, der Transfer des Statusregisters eingeleitet. Das Clocksignal für dieses Signal erfolgt 66,4 ns nach der Freigabe des Zählers. Zurückgesetzt wird es, sobald beide Data-Strobe-Signale inaktiv werden.

```
ak.d = IP & (ai == irqa);
ak.clk = del2;
ak.ar = !DS0 & !DS1;
```

Im ak-Flipflop wird intern gespeichert, wenn ein korrekter Interrupt-Bestätigungszyklus erkannt wurde. Am Dateneingang dieses Flipflops liegt der gleiche Zustand an, wie am Dateneingang des $IACYC$-Flipflops. Als Taktsignal wird jedoch das Verzögerungssignal $del2$ benutzt. Ebenso wie $IACYC$ wird ak zurückgesetzt, wenn beide Data-Strobe-Signale wieder inaktiv werden. Um dem Bus-Master zu signalisieren, dass der Interrupt-Acknowlege-Zyklus erkannt wurde, dient wie beim Datentransfer das $DTACK$-Signal. Als Ausgangstreiber des $DTACK$-Signals muss genau wie in IC4 ein Open-Drain-Ausgang mit Hilfe eines Tristate-Ausgangs nachgebildet werden.

```
DTACK.D = 1;
DTACK.OE = ak;
DTACK.clk = CLK
```

Der Quellcode wird wieder mit dem Schlüsselwort *END* abgeschlossen.

D.2 Das Statusregister

Mit dem 8-fach-D-Flipflop IC8 und den beiden Bustreibern IC9 und IC10, die in Abbildung D.2 zu sehen sind, wird das Steuerregister realisiert. Mit den Eingängen von IC8 sind die oberen 8 Bits des internen Datenbusses verbunden. Die Ausgänge werden je nach Bedeutung des entsprechenden Steuerbits teilweise über den Treiber IC14 auf der Platine verteilt. Sie liegen zudem an den Eingängen des Tristate-Treibers IC9 an. Diese Bustreiber werden aktiviert, wenn ein Lesezugriff auf das Statusregister erfolgt. Somit können die gesetzten Steuerbits wieder ausgelesen werden. Die Bustreiber in IC10 sind mit den Bits 0 bis 7 des internen Datenbusses verbunden. An ihren Eingängen liegen die Signale an, die den Statusbits 0 bis 7 des Statusregisters entsprechen. IC11 dient dazu, die von der VMEbus-Logik kommenden Steuersignale zu treiben und auf der Platine zu verteilen. An den Eingängen der Bustreiber IC17 und IC18 liegen die Ausgänge der beiden Clusterzähler $CNTA$ und $CNTB$ an. Die beiden Bustreiber-ICs werden von der VMEbus-Logik aktiviert, wenn ein Lesezugriff auf diese beiden Zähler erfolgt. Die Ausgänge der Bustreiber sind wieder mit dem internen Datenbus verbunden.

D.3 Die FIFOs und die Zellzähler

Blatt 3 des Schaltbildes (siehe Abbildung D.3) zeigt ganz links die beiden FIFO-Speicher IC19 und IC20 vom Typ IDT7203. Diese Bausteine haben jeweils 9 Eingänge, auf die die 12 Bits des Zelladressbusses sowie zwei Flag-Signale aufgeteilt werden, die den Beginn eines neuen Events und den Beginn eines neuen Clusters in den Auslesedaten anzeigen. Die Bits 0 bis 7 des Zelladressbusses liegen an D0 bis D7 von IC19 an. Der Eingang D8 dieses ICs bleibt unbenutzt und wird fest auf Messe gelegt. Die restlichen vier Bits liegen zusammen mit dem New-Cluster-Flag und dem New-Event-Flag, die vom Sequenzer gesetzt und gelöscht werden können, an D0 bis D5 von IC20. Die übrigen Eingänge von IC20 sind fest mit Masse verdrahtet.

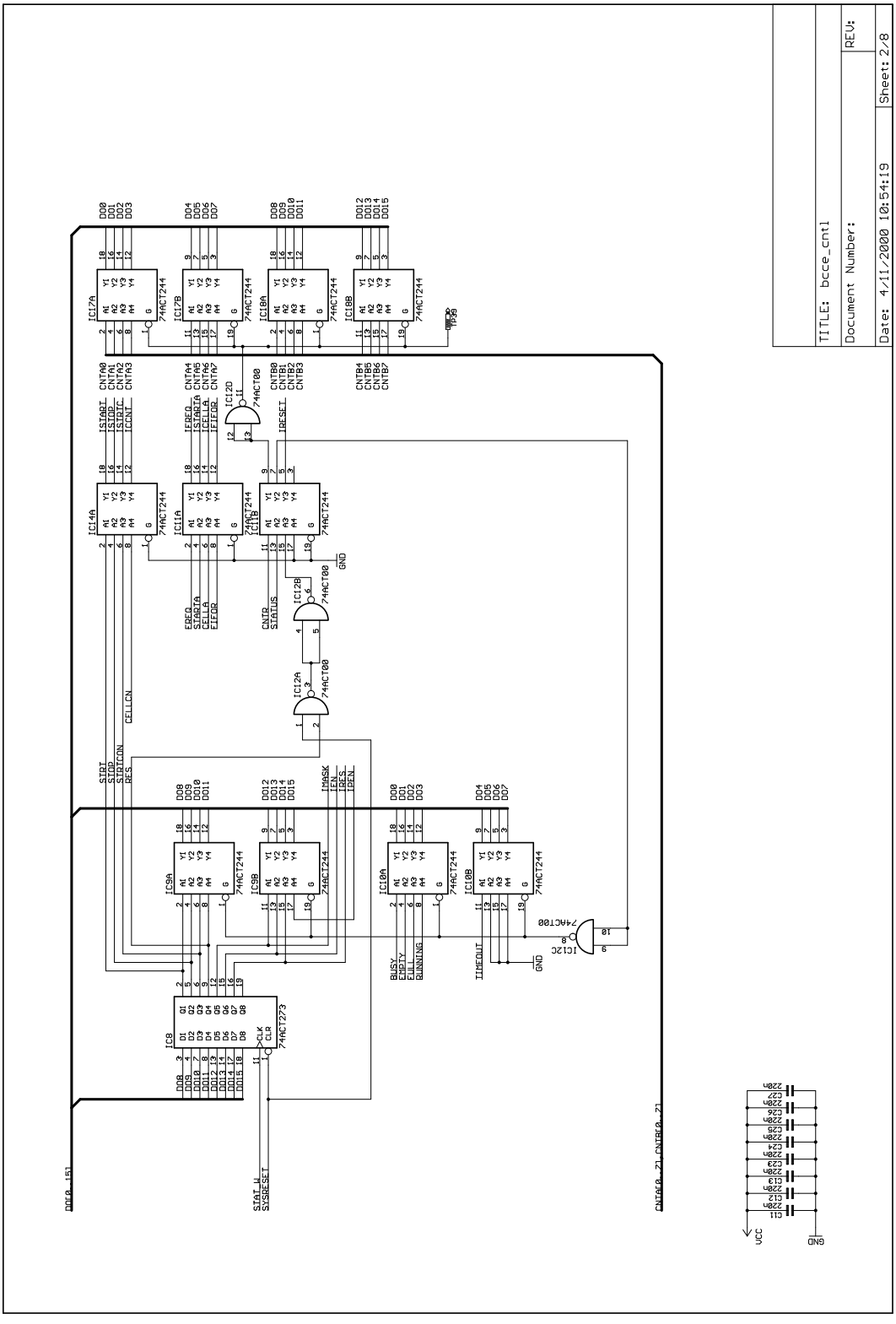


Abbildung D.2: Blatt 2 des Schaltbildes der Controller-Platine. Zu sehen ist im Wesentlichen das Steuerregister

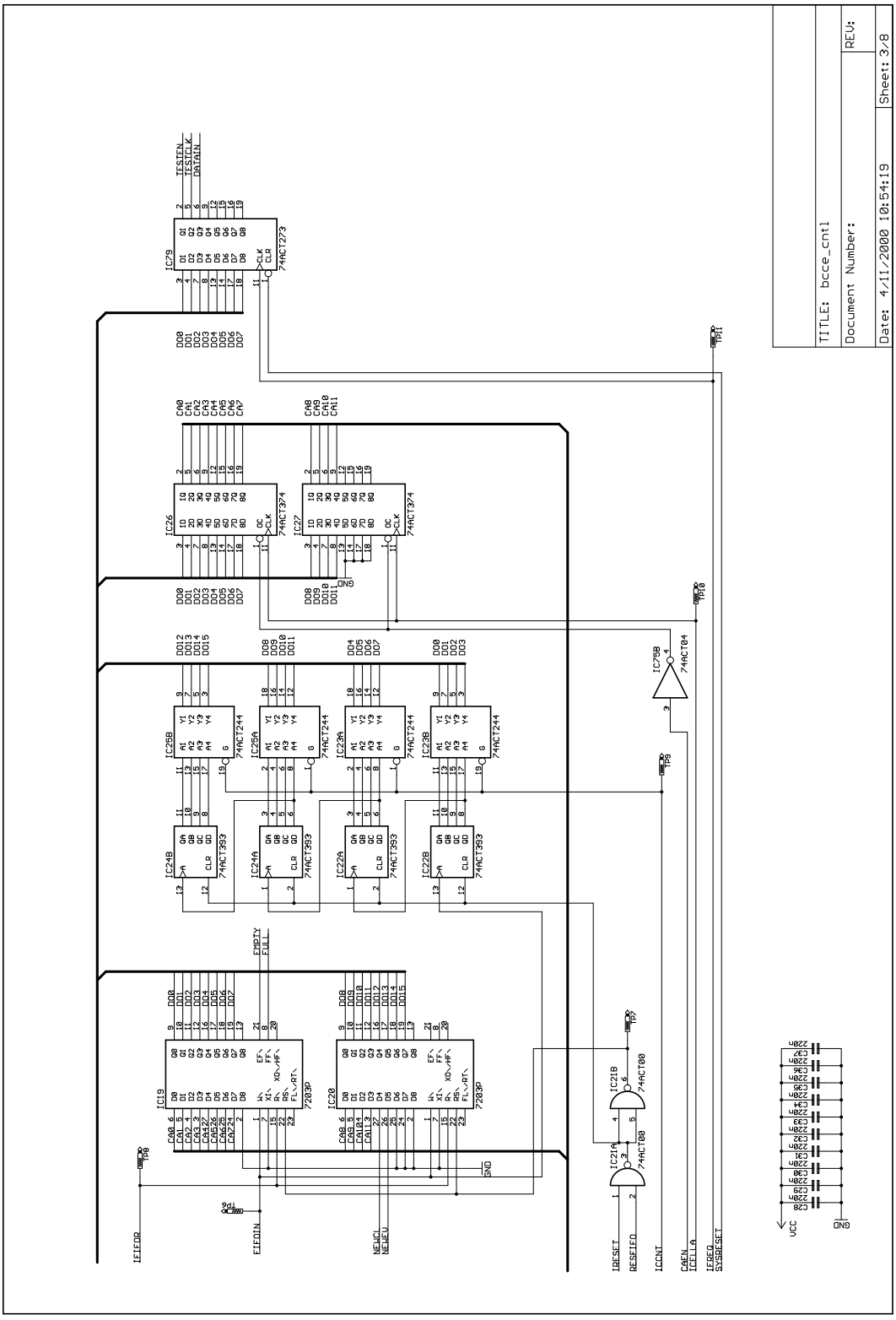


Abbildung D.3: Blatt 3 des Schaltbildes der Controller-Platine

Die Ausgänge der FIFOs sind mit dem internen Datenbus verbunden. Zudem erzeugen die FIFOs zwei Statusbits, $Empty$ und $Full$, die über das Statusregister der Controller-Karte ausgelesen werden können. Zur Ansteuerung benötigen die FIFOs ein Write- und ein Read-Signal. Während das Write-Signal, hier als $FIFOIN$ bezeichnet, vom Sequencer erzeugt wird, kommt das Read-Signal vom VMEbus-Controller und wird hier als $FIFOR$ bezeichnet. Um die FIFO-Speicher zu initialisieren, benötigen sie vor der ersten Benutzung ein Reset-Signal, das aus einer Oder-Verknüpfung der beiden Resetsignale $IReset$ und $ResFIFO$ erzeugt wird. $IReset$ ist das globale Reset-Signal und selbst wieder eine Oder-Verknüpfung aus dem VME-Systemreset und dem Steuerbit Res des Statusregister, $ResFIFO$ dagegen wird vom Sequencer erzeugt, sodass dieser in der Lage ist, vor dem Auslesen eines Ereignisses den FIFO-Speicher zu initialisieren. Da beide Signale in negativer Logik vorliegen, wird zur Oder-Verknüpfung das NAND-Gatter IC21A benutzt. Das Reset-Signal des FIFOs und das $FIFOIN$-Signal gehen auch auf einen 16-Bit-Zähler, der aus den ICs IC22 und IC24 aufgebaut ist. Dieser Zähler zählt somit die Einträge in den FIFO-Speicher mit. Über die Bustreiber IC23 und IC25 kann der Zählerstand bei einem entsprechenden VMEbus-Zugriff von der CPU abgefragt werden.
Um rechnergesteuert selektiv Zellen setzen zu können, kann in die beiden 8-Bit-D-Flipflops IC26 und IC27 eine Zelladresse geschrieben werden. Dazu erzeugt die VMEbus-Logik bei einem entsprechenden Zugriff einen Clk-Impuls auf der $ICELLA$-Leitung. Der Sequencer kann durch die Aktivierung des Output-Enable-Signals $CAEN$ die in diesem Register gespeicherten Daten auf den Zelladressbus geben.
IC79 ist ein weiteres 8-Bit-Register, das von der VMEbus-Logik über die $IFREQ$-Leitung gesetzt wird. Dieses Register dient derzeit nur dazu, die Steuersignale für den Testmustergenerator in den Latches zu speichern. Dazu werden die Ausgänge Q1 bis Q3 benutzt, die die Signale $TestEn$, $TEstClk$ und $DataIn$ darstellen.

D.4 Der Sequencer

Die Funktion des Sequencers ist bereits ausführlich in [Go98] beschrieben. Hier sind nur noch einmal der Vollständigkeit halber die Schaltbilder in Abbildungen D.5 und D.6 wiedergegeben.

D.5 Die Sequenceransteuerung

Es ist vorgesehen, bei Bedarf den internen Sequencer deaktivieren zu können und einen externen Sequencer zu benutzen. Dieser externe Sequencer wird dann als Piggy-Pack auf die Controller-Platine aufgesteckt. Dazu gibt es den Stecker $PP1$, der auf dem Blatt 6 des Schaltbildes in Abbildung D.6 zu sehen ist. Dieser Stecker ist in drei Teilstecker unterteilt, wovon ein achtpoliger Stecker die Zustandsbits für bedingte Sprunganweisungen enthält und ein zweiter Teilstecker mit 18 Pins mit der Startadresse, der Versorgungsspannung und den Steuersignalen $IStart$, $IStop$ und $IReset$ belegt ist. Weiterhin gibt der Sequencer über diesen Teilstecker das $RunE$-Signal zurück, mit dem er anzeigt, dass er sich im Betriebszustand befindet. Die meisten Signale befinden sich auf dem dritten Teilstecker, auf dem sämtliche Ausgangssignale des Sequencers zusammengefasst sind. Diese Ausgangssignale werden ebenso wie das $RunE$-Signal über TriState-Treiber geleitet, die nur aktiviert werden, wenn der interne Sequencer deaktiviert ist. Dazu gibt es ein Steuersignal $Intern$, das mit einem Kurzschlussstecker auf zwei Pfostensteckern von $JP1$

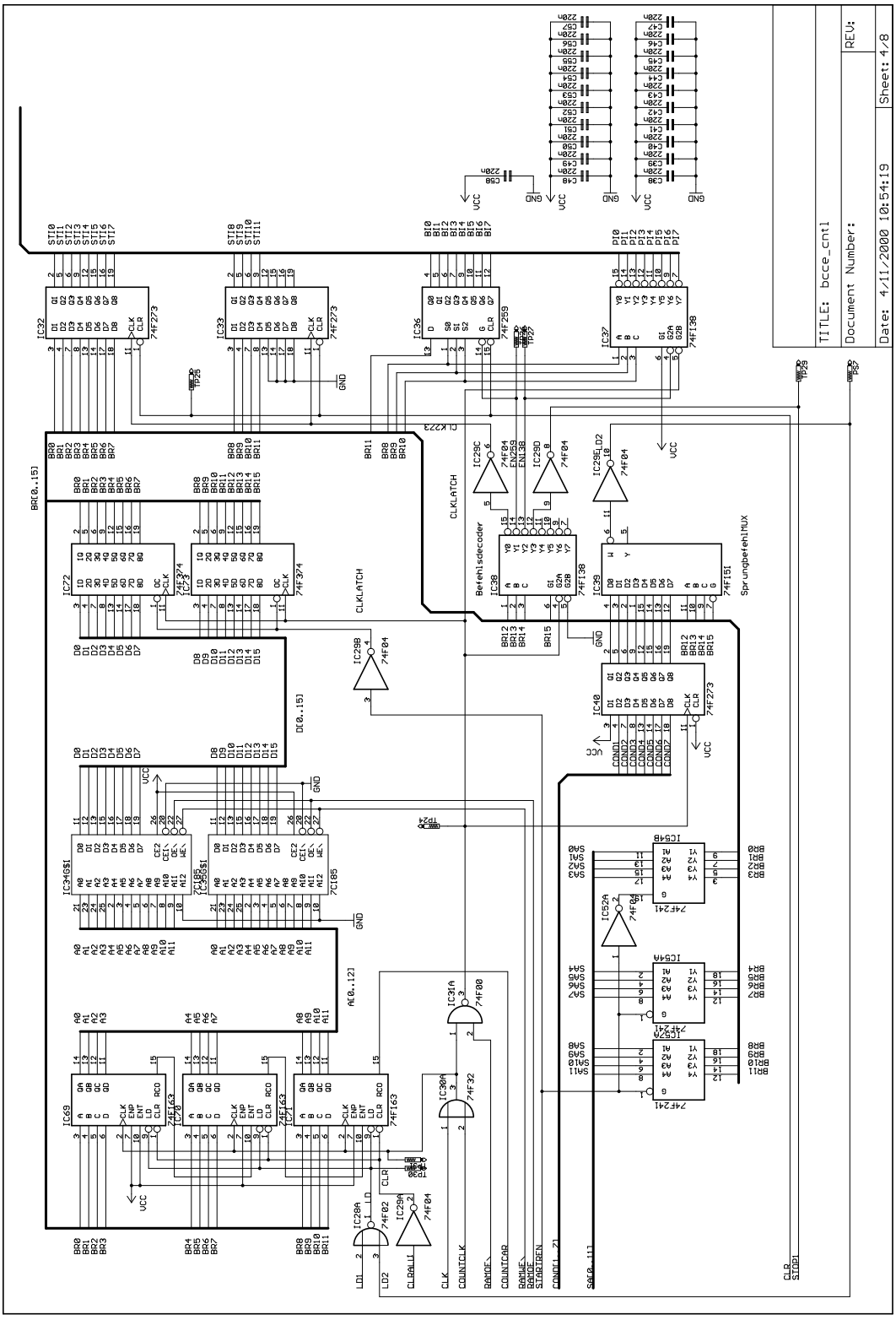


Abbildung D.4: Blatt 4 des Schaltbildes der Controller-Platine

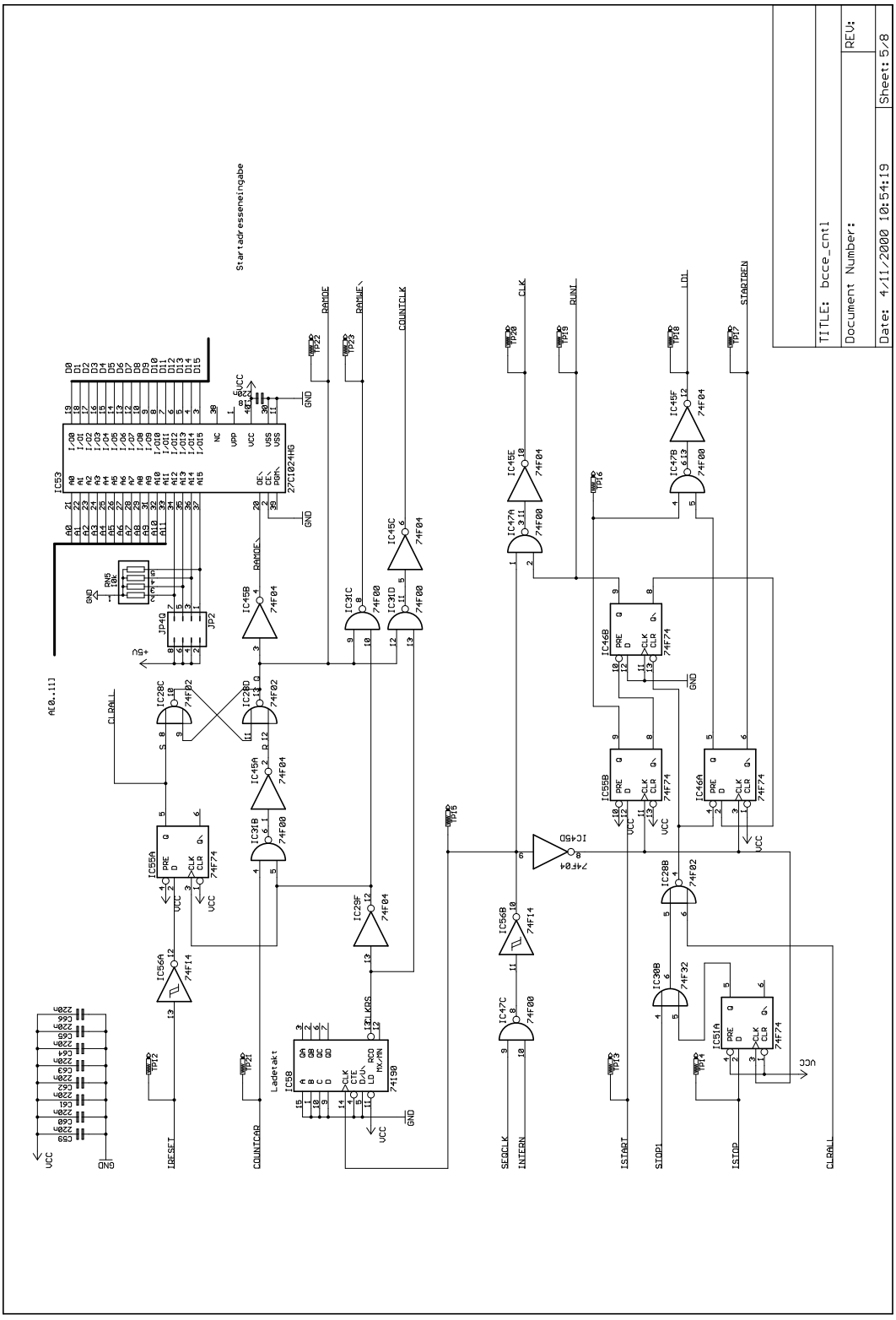


Abbildung D.5: Blatt 5 des Schaltbildes der Controller-Platine

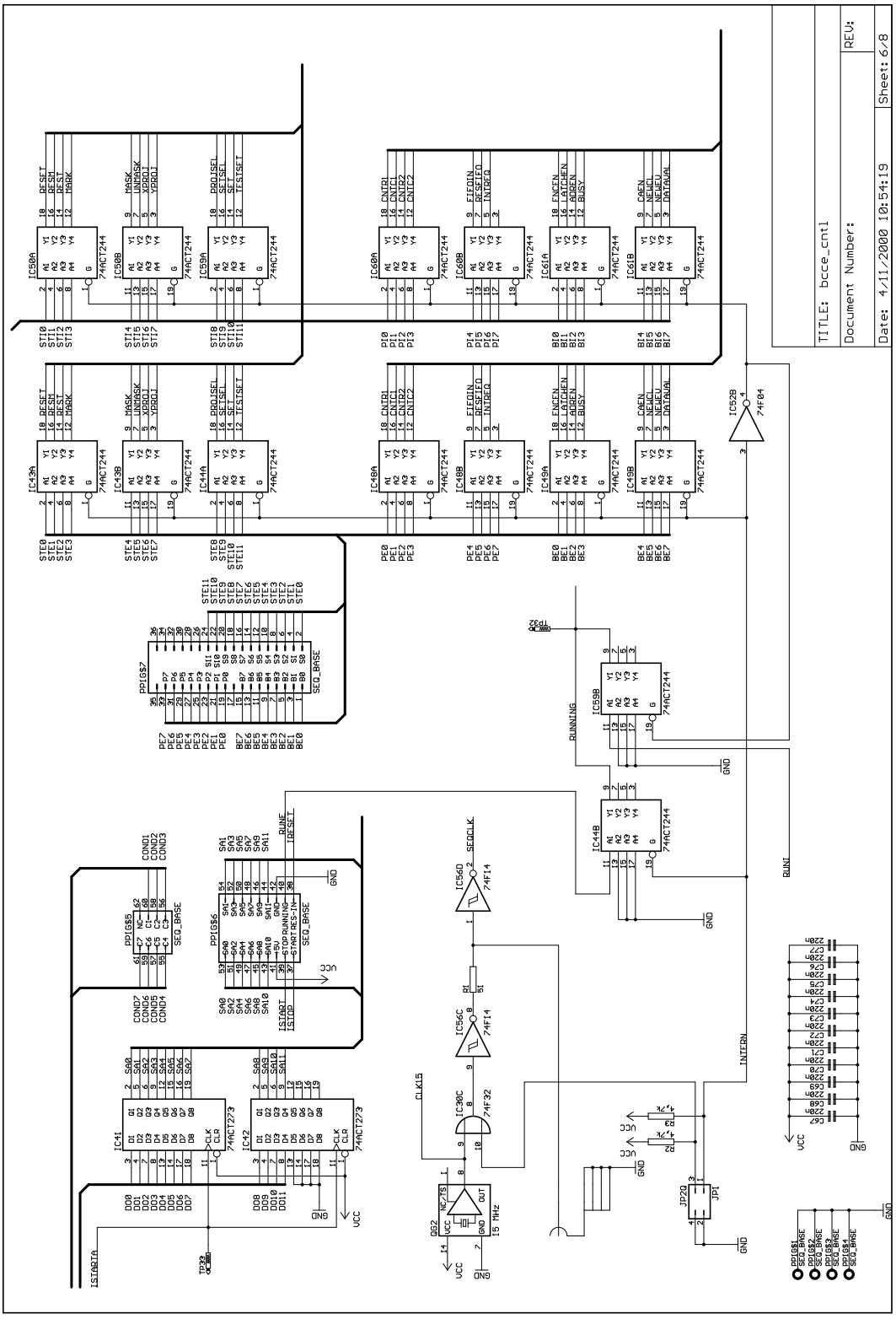


Abbildung D.6: Blatt 6 des Schaltbildes der Controller-Platine

erzeugt werden kann. Die Ausgänge des internen Sequencers werden ebenfalls über TriState-Treiber geleitet, die jedoch nur dann aktiviert werden, wenn der interne Sequencer aktiviert ist.
Als Taktquelle für den internen Sequencer dient der 15-MHz-Quarzoszillator QG2, dessen Ausgangssignal über das Oder-Gatter IC30C, den Schmitt-Trigger IC56C und den 51 Ω-Widerstand R1 auch auf eine Lemo-Buchse nach außen geführt wird. Dort kann das Taktsignal z.B. auch für einen externen Sequencer benutzt werden. Mit diesem Punkt ist auch der Eingang des Schmitt-Triggers IC56D verbunden, der das Taktsignal regeneriert und dem internen Sequencer zur Verfügung stellt.
Damit diese Taktquelle aktiviert ist, müssen die beiden oberen Pfostenstecker im Steckerfeld JP1 miteinander verbunden sind. Lässt man diese Verbindung offen, wechselt der Eingang an Pin 10 am Oder-Gatter IC30C auf High, womit der Ausgang des Schmitt-Triggers IC56C fest auf Low-Pegel liegt. Die Lemo-Buchse dient dann als Eingang und der Widerstand R1 als Abschlusswiderstand, sodass der interne Sequencer auch mit einer externen Taktquelle betrieben werden kann.

D.6 Clusterzähler und Timeout-Zähler

Auf dem Blatt 7 des Schaltbildes in Abbildung D.7 sind oben die beiden Clusterzähler IC62 und IC63 zu sehen. Diese Zähler werden mit den Steuersignalen $CntC1$ und $CntR1$ sowie $CntC2$ und $CntR2$ angesprochen, die vom Sequencer mit dessen Pulsgenerator erzeugt werden.
Die Ausgänge der Zähler sind zum einen über die beiden Bussysteme $CntA$ und $CntB$ mit dem entsprechenden VMEbus-Leseregister (siehe Abbildung D.7) und zum anderen mit TTL-ECL-Konvertern vom Typ MC10124 (IC64 is IC67) verbunden. Die symmetrischen ECL-Ausgänge der Zähler werden über die beiden Pfostenstecker SV1 und SV2 nach außen geführt, sodass die Haupttriggerelektronik diese Zählerstände als Triggerinformation benutzen kann. Unter den TTL-ECL-Konvertern sieht man in Abbildung D.7 die Timeout-Zähler IC68 und IC74, die mit dem Taktsignal des internen 15-MHz-Quarzgenerators angesteuert werden. Mit den Reset-Eingängen werden diese Zähler im Normalzustand beim Zählerstand 0 festgehalten. Sobald das $Busy$-Signal aktiv wird, werden die Zähler freigegeben. Der Zähler IC74 zählt so lange, bis $Busy$ wieder inaktiv wird, der Zähler IC68 lässt sich dagegen auch wieder mit dem Aktivieren des $DataVal$-Signals anhalten. Dies wird mit den Gattern IC21C, IC52E und IC52F erreicht. Sobald der Ausgang Q8 von Zähler IC68 aktiv wird, was nach 128 Taktzyklen, also nach 8,5 μs eintritt, oder der Ausgang Q9 von Zähler IC74 aktiv wird, was nach 256 Taktzyklen oder 17 μs eintritt, werden die entsprechenden Timeout-Flipflops IC76A oder IC77A gesetzt.
Ein weiteres Flipflop IC78A wird bei einem aktiven $Busy$-Signal gesetzt, wenn nicht gleichzeitig das $Running$-Signal aktiv ist. Die invertierten Ausgänge der Timeout-Flipflops werden mit dem NAND-Gatter IC80A verknüpft, und als Timeout-Bit zum Statusregister geleitet. Mit dem weiteren NAND-Gatter IC80B werden die Ausgänge aller drei Flipflops, sowie ein Interrupt-Anforderungs-Signal, das vom Sequencer erzeugt werden kann, verknüpft. Dieses Verknüpfungssignal dient als Interrupt-Aufforderung und wird direkt zur Interruptlogik auf dem Schaltplan in Abbildung D.1 weitergeleitet. Die Flipflops können mit dem $IReset$-Signal wieder zurückgesetzt werden.
Mit den Ausgängen der drei Flipflops IC76A bis IC78A sind zudem die Setz-Eingänge der drei Flipflops IC76B bis IC78B verbunden. Die Zustände dieser Flipflops werden mit Leuchtdioden

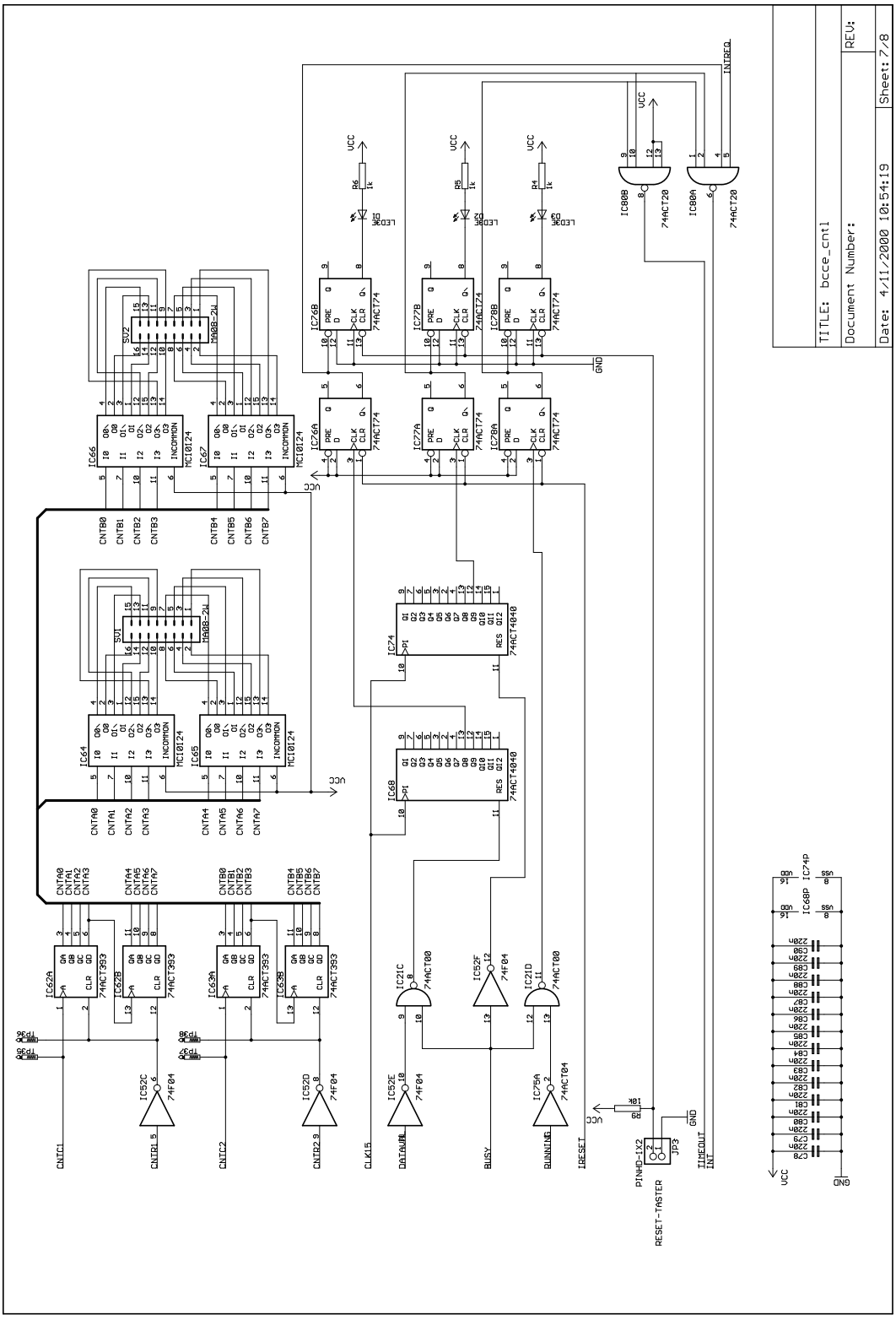


Abbildung D.7: Blatt 7 des Schaltbildes der Controller-Platine

angezeigt, sodass von außen eine optische Kontrolle von Fehlerzuständen auf der Controllerplatine gegeben ist. Diese Flipflops können durch Betätigen eines Reset-Tasters, der an JP3 angeschlossen wird, zurückgesetzt werden.

D.7 BCCE-Busanschluss und Spannungsversorgung

Auf dem Blatt 8 des Schaltbildes in Abbildung D.8 sind die restlichen Schaltungsteile zu sehen, die für die Verbindung nach außen notwendig sind. IC81 ist ein TTL-Leitungstreiber, der die Status- und Steuersignale $Busy$, $Dataval$, $TestEn$, $TestClk$ und $DataIn$ treibt. Die Ausgänge des Treibers sind mit dem Stecker $SV3$ verbunden. Über diesen Stecker gelangen auch die Steuersignale $Gate$ und $FastClear$ auf die Karte, die ebenfalls mit dem Leitungstreiber IC81 regeneriert werden. Das $Gate$-Signal wird zunächt mit dem Oder-Gatter IC3B mit dem $IStrtc$-Signal verknüpft, das durch Setzen des entsprechenden Bits im Statusregister erzeugt werden kann. Der Ausgang des Oder-Gatters ist mit dem Konditionsbit $Cond3$ des Sequencers verbunden, während das regenerierte $FastClr$ direkt mit dem $Cond4$-Bit des Sequencers verbunden ist.
Darunter befindet sich auf Blatt 8 der Spannungsregler IC82, der aus der 12-V-Spannungsversorgung, die der VMEbus zur Verfügung stellt, eine Spannung von -5 V erzeugt, die für den Betrieb der TTL-ECL-Konverter benötigt wird.
Rechts in Abbildung D.8 ist die Belegung von J2 des VMEbus-Interfaces zu sehen. Hier werden die Steuersignale des BCCE-Busses zur Zellularlogik geführt.

D.8 Das Layout der Controllerplatine

Der Controller wurde nahezu vollständig in SMD-Technik auf einer Vierlagen-Multilayer-Platine aufgebaut. Das Layout, das wieder mit der Leiterplatten-CAD-Software EAGLE entworfen wurde, ist in Abbildung D.9 zu sehen. Die Leiterbahnen konnten nahezu vollstängig mit dem Autoroutermodul des CAD-Programms entflochten werden. Abbildung D.10 zeigt den Bestückungsplan der Controllerplatine auf der Bestückungsseite. Den unteren Teil der VMEbus-Platine nimmt der interne Seqeuncer ein. Dort befindet sich auch das EPROM IC53 vom Typ 27C1024, in dem die Sequencerprogramme abgelegt werden können. Dieses EPROM wird gesockelt, sodass es jederzeit durch ein anderes mit einem verändertem Sequencerprogramm ersetzt werden kann. Der restliche Platz auf der Karte wird im Wesentlichen durch das VMEbus-Interface, das sich oben rechts befindet, und der Elektronik, die für die Realisierung der verschiedenen Register und der FIFOs notwendig ist, eingenommen. Einige Bauteile, die auf der Bestückungsseite keinen Platz mehr fanden, wurden auf der Lötseite der Platine untergebracht. Dies ist in Abbildung D.11 zu sehen. Abbildung D.12 zeigt eine Fotografie des Controllers.

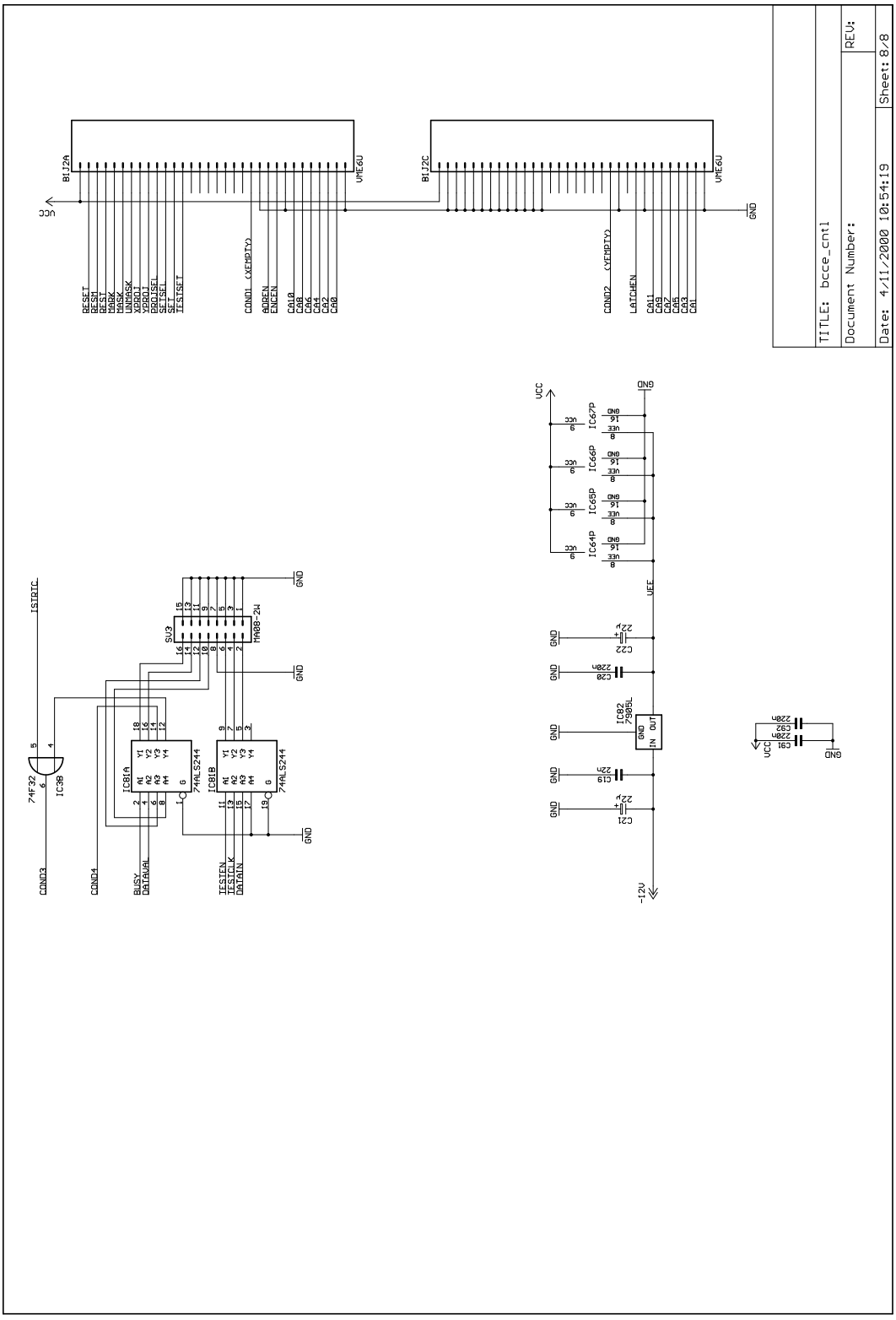


Abbildung D.8: Blatt 8 des Schaltbildes der Controller-Platine

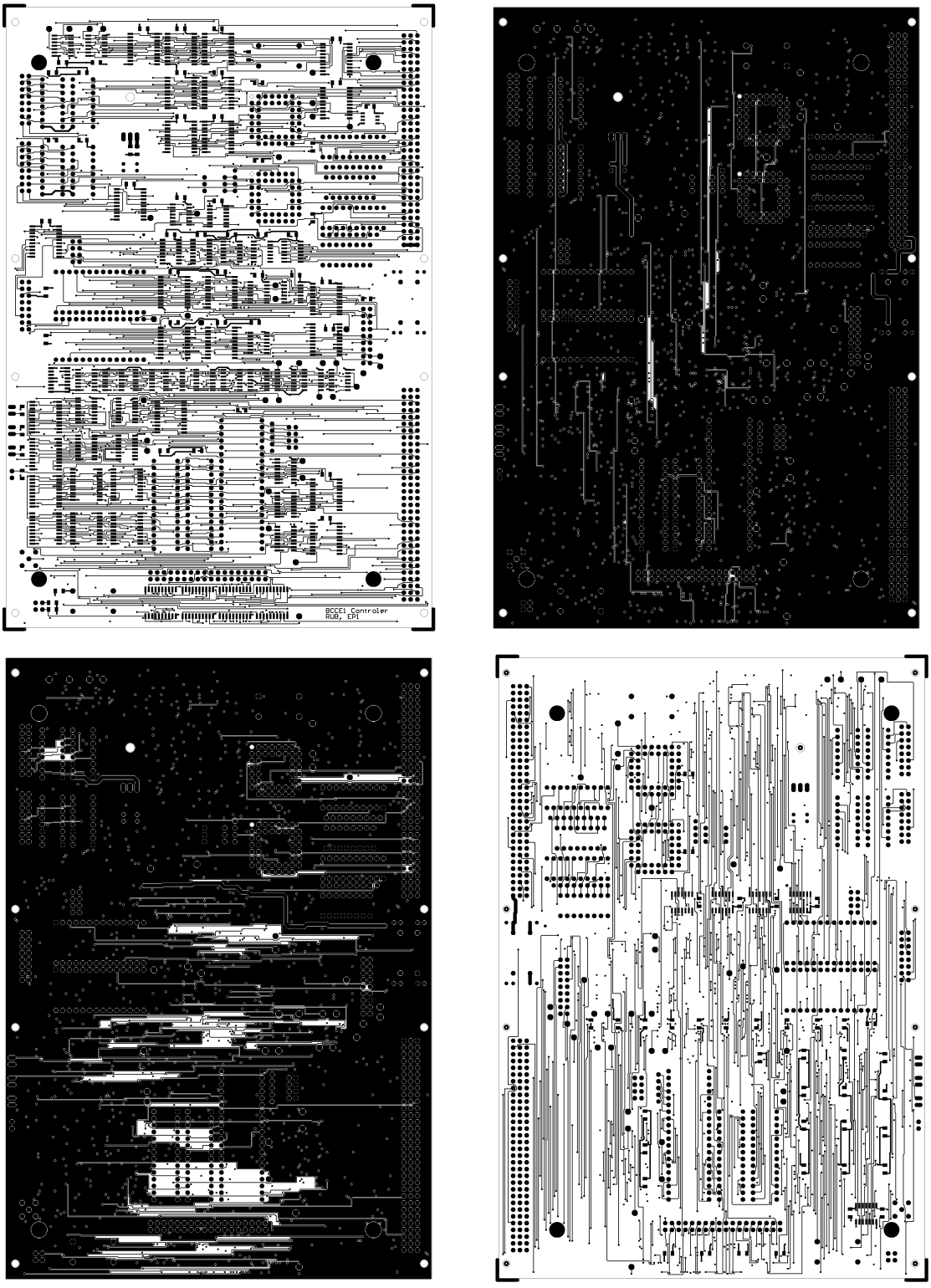


Abbildung D.9: Das Layout der Controllerplatine. Oben Links ist der Top-Layer, oben rechts der erste Innenlayer, unten links der zweite Innenlayer und unten rechts der Bottom-Layer zu sehen

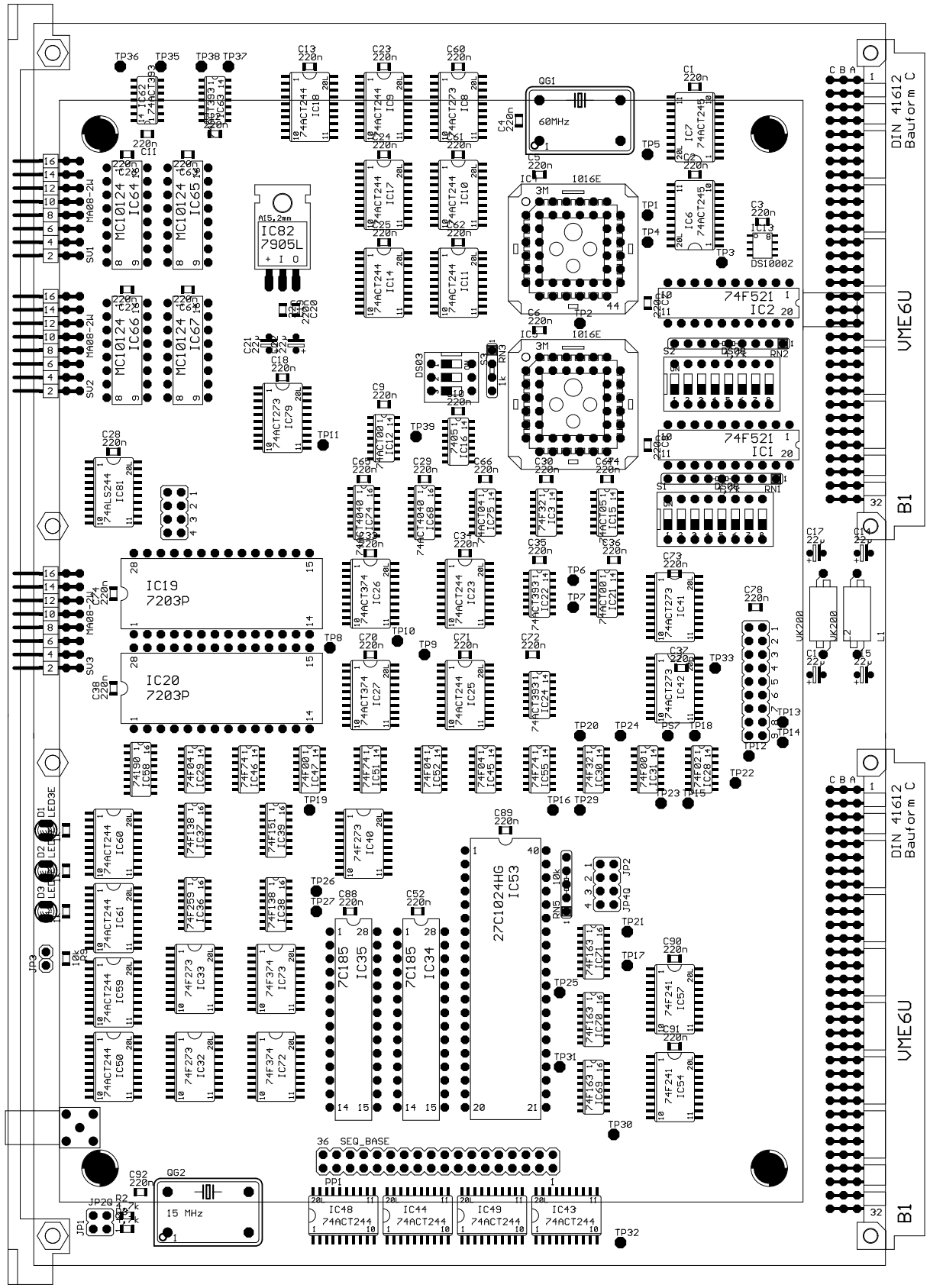


Abbildung D.10: Der Bestückungsplan für die Vorderseite der Controllerplatine

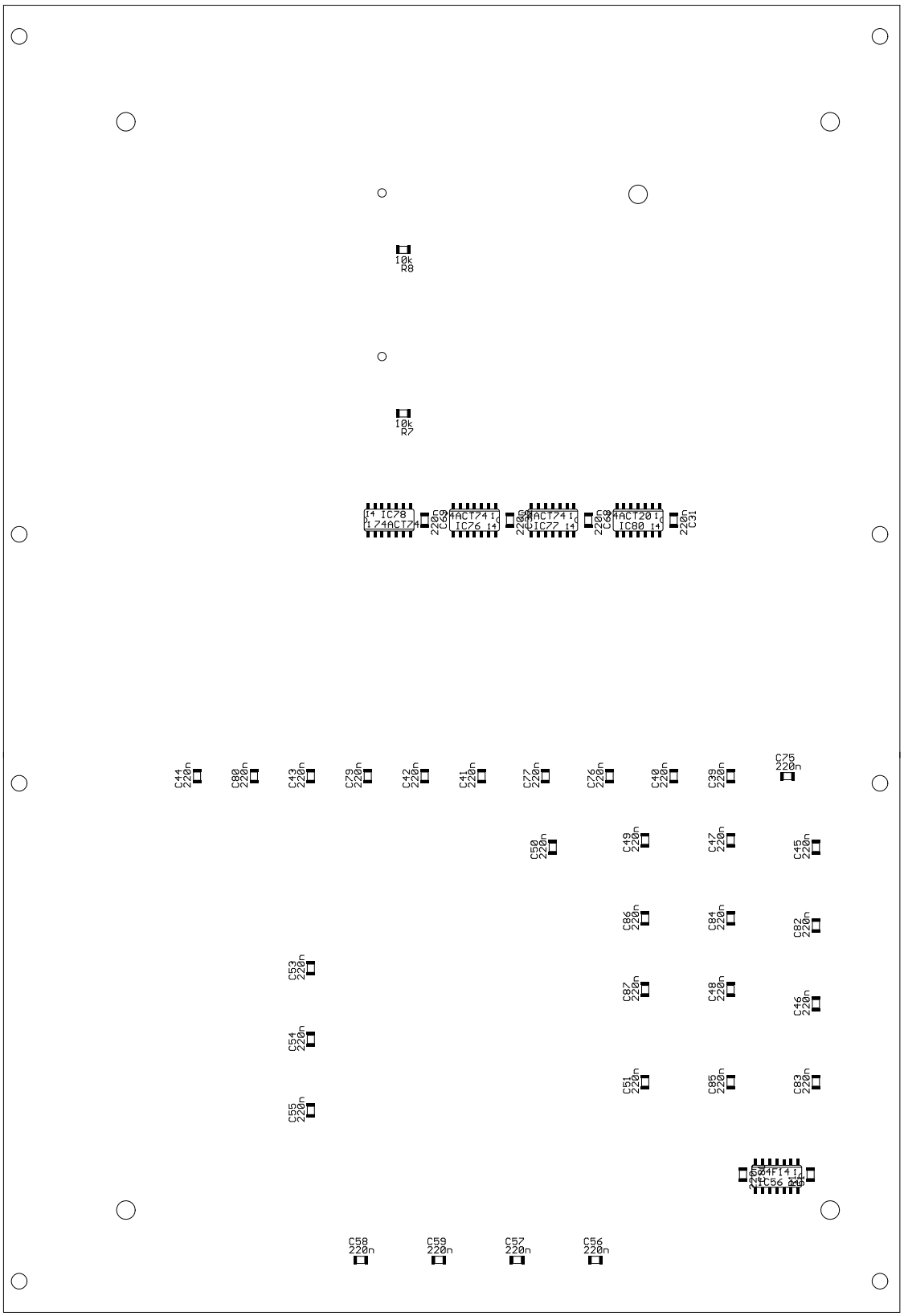


Abbildung D.11: Der Bestückungsplan für die Lötseite der Controllerplatine

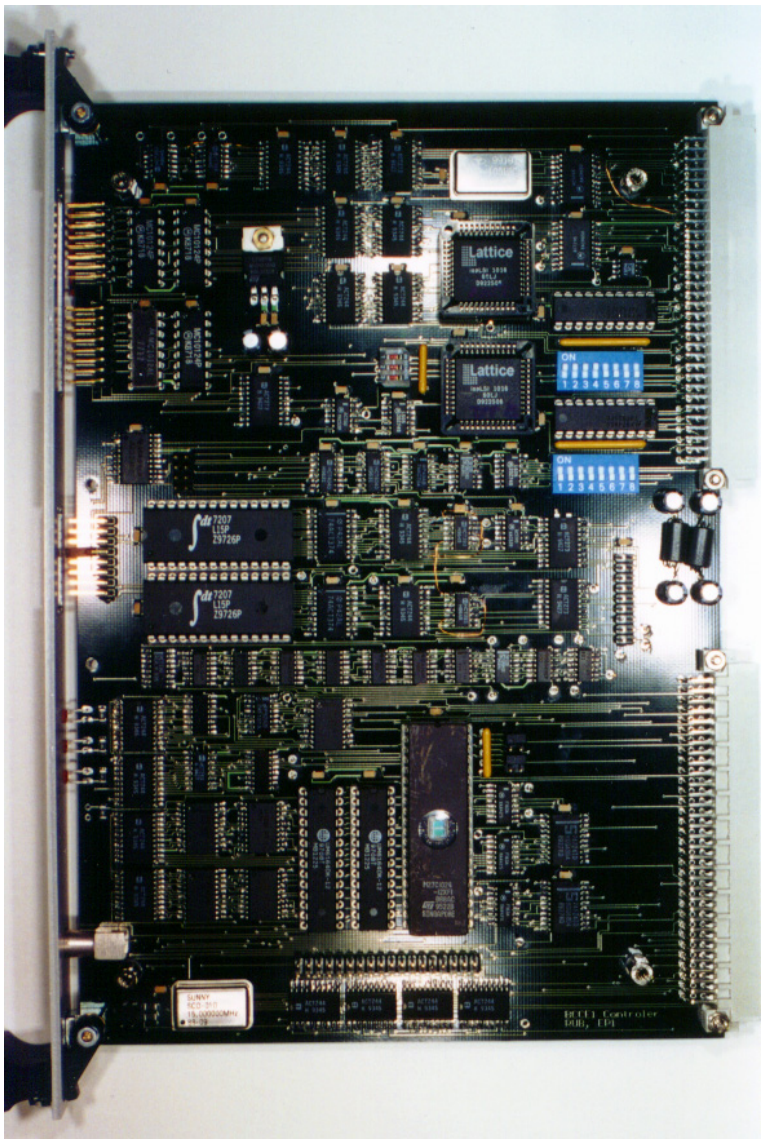


Abbildung D.12: Der Aufbau der Controller-Platine. Unten befindet sich der Sequencer, wovon deutlich das EPROM und daneben das Befehls-RAM zu erkennen sind. In der Mitte links sind die beiden FIFO-Speicher-Bausteine zu sehen, oben rechts ist das VME-Interface angeordnet. Deutlich sind die DIP-Schalter zur Einstellung der Basis-Adresse zu erkennen

D.9 Anschlüsse auf dem Controllerboard

Das Controllerboard verfügt über eine ganze Reihe von Anschlüssen, die unterschiedlichen Zwecken dienen. Standardmäßig verfügt eine 6-Höheneinheiten-VMEbus-Platine über zwei VG-Messerleisten auf der Rückseite. Davon wird auf diesem Board die obere für den VMEbus und die untere zur Übertragung der vom Sequencer erzeugten Steuersignale an die Zellularlogik benutzt.

D.9.1 Der VMEbus-Anschluss

Die Belegung des VMEbus-Steckers ist in [Vi87] spezifiziert, wobei die Controllerkarte jedoch nicht alle VMEbus-Signale benötigt. Die Belegung mit den benutzten Signalen zeigt Tabelle D.1. Die Belegung der unteren VG-Messerleiste, die für den BCCE-Bus benutzt wird, ist aus Tabelle D.2 zu ersehen.

D.9.2 Die Anschlussmöglichkeiten auf der Frontplatte

Auf der Frontseite befinden sich ebenfalls eine Reihe von Anschlussmöglicheiten, die zur Kommunikation des Controllers mit der Triggerelektronik des Crystal-Barrel-Experimentes dienen. Ganz oben sind zwei 16-polige Pfostensteckerleisten angebracht, an denen die Zählerstände der beiden Clusterzähler binär codiert als Differential-ECL-Signale anliegen. Die obere der beiden Steckerleisten ist dabei für den Zähler A vorgesehen, die untere für den Zähler B. Bit 0 der Zähler ist an beiden Steckerleisten ganz oben; auf der linken Seite befinden sich die nicht invertierten, auf der rechten Seite die invertierten ECL-Signale.
Ein Stück darunter befindet sich eine dritte 16-polige Pfostensteckerleiste, über die eine Reihe von Steuersignalen übertragen wird. Im Gegensatz zu den Zählerausgängen werden hier TTL-Pegel verwendet. Die Steuersignale können entweder über das VMEbus-Interface oder vom Sequencer gesetzt werden. Die eingehenden Steuersignale sind mit Konditionsbits des Sequenzers verbunden und können somit dessen Programmablauf beeinflussen. Damit ist die Bedeutung der einzelnen Anschlüsse programmabhängig. Für den vorgesehenen Betrieb als Triggercontroller im Bochumer Zellularlogik-Clustertrigger ist jedoch eine Belegung laut Tabelle D.3 vorgesehen. Dabei haben die einzelnen Pins die folgende Bedeutung:

Busy Mit dem Busy-Signal zeigt der Clusterencoder der Haupttriggerelektronik an, dass er zur Zeit ein Treffermuster des Barrels bearbeitet und nicht in der Lage ist, neue Daten anzunehmen. Solange diese Leitung auf Low-Pegel liegt, kann die Clustererkennung mit einem Gate-Signal gestartet werden.

Data Valid Nachdem die Clusterzählung abgeschlossen ist und der engültige Zählerstand an den Zählerausgängen anliegt, erzeugt der Controller ein High-Signal am Data-Valid-Ausgang. Die Haupttriggerelektronik darf erst nachdem dieses Signal anliegt die Zählerdaten auswerten.

Gate Mit einem High-Pegel an diesem Pin wird die Clustererkennung gestartet. Dazu muss das Gate-Signal mindestens 150 ns anliegen, da der Status vom Sequencer zyklisch abgefragt

A	B	C
D0		D8
D1		D9
D2		D10
D3		D11
D4		D12
D5		D13
D6		D14
D7		D15
GND		GND
GND		BERR
DS1		SYSRESET
DS0		LWORD
WRITE		AM5
GND		A23
DTACK	AM0	A22
GND	AM1	A21
AS	AM2	A20
GND	AM3	A19
IACK	GND	A18
IACKIN		A17
IACKOUT		A16
AM4	GND	A15
A7	IRQ7	A14
A6	IRQ6	A13
A5	IRQ5	A12
A4	IRQ4	A11
A3	IRQ3	A10
A2	IRQ2	A9
A1	IRQ1	A8
-12 V		-12 V
$+5$ V	$+5$ V	$+5$ V

Tabelle D.1: Die Belegung des VMEbus-Steckers J1 mit den von der Controllerkarte benötigten Signalen

A	B	C
+5V		+5V
RESET		GND
ResM		GND
ResT		GND
Mark		GND
Mask		GND
UnMask		GND
XProj		GND
YProj		GND
ProjSel		GND
SetSel		GND
Set		GND
TestSet		GND
xempty		yempty
GND		GND
AdrEn		
EncEn		LatchEn
GND		GND
CA10		CA11
CA8		CA9
CA6		CA7
CA4		CA5
CA2		CA3
CA0		CA1
GND		GND

Tabelle D.2: Pinbelegung des Zellularlogik-busses auf J2 des VMEbus-Systems

Pin	Steuersignal	Richtung
1	Busy	Ausgang
2	DataValid	Ausgang
3	Gate	Eingang
4	FastClear	Eingang
5	Gnd	
6	TestEn	Ausgang
7	TestClk	Ausgang
8	DataIn	Ausgang

Tabelle D.3: Die Pinbelegung des Pfostensteckers für die Steuersignale

wird. Wenn das Gate-Signal vom Sequencer erkannt wurde, wird von diesem das Busy-Signal zur Bestätigung aktiviert.

Fast Clear Nachdem der Clusterencoder die Clusterzählung abgeschlossen hat und dies mit dem Data-Valid-Signal anzeigt, kann die Auslese der Zellularlogik jederzeit mit dem Fast-Clear-Signal abgebrochen werden. Da auch dieses Signal zyklisch vom Sequencer abgefragt wird, ist die Zeit, die dieses Signal anliegt, nicht festgelegt. Wenn das Signal erkannt wurde, zeigt der Sequencer dies jedoch dadurch an, dass das Busy-Signal inaktiv wird. Das Fast-Clear-Signal muss also so lange aktiv bleiben, bis Busy inaktiv wird.

Test Enable Mit dem Test-Enable-Signal schaltet der Controller die Latchmodule in den Testmodus. Dadurch werden statt der Barreldaten von den Latches die Daten des Testmustergenerators auf den Eingang des Clustertriggers gegeben.

Test Clock dient als Taktsignal für den Testmustergenerator in den Latchmodulen.

Data In ist das Datensignal für den Testmustergenerator der Latchmodule.

Ganz unten auf der Frontplatte der Controllerkarte befindet sich eine Koaxialbuchse in QLA-Norm, die je nach Konfiguration auf der Karte als Takt Ein- oder Ausgang für den Sequencertakt benutzt werden kann.

D.9.3 Der Steckplatz für den externen Sequencer

Auf der Controllerplatine ist, wie in der Schaltungsbeschreibung erwähnt, die Möglichkeit vorgesehen, einen externen Sequencer zu benutzen. Dazu ist ein Steckplatz vorhanden, auf den dieser externe Sequencer als Piggy-Pack-Platine aufgesteckt werden kann.
Dieser Steckplatz besteht aus drei Steckleisten, auf die jeweils zweireihige Pfostensteckerleisten aufgesteckt werden können.

D.10 Die Konfiguration des Controllerboards

Es gibt eine Reihe von Einstellungen, die auf dem Controllerboard vorgenommen werden können. Dabei ist grundsätzlich zwischen solchen Einstellungen zu unterscheiden, die z.B. über Schalter oder Kurzschlussstecker auf dem Board selber vorgenommen werden, und solchen, die per Software durch Zugriff über das VMEbus-Interface auf die internen Register vorgenommen werden. Die Softwarekonfigurationsmöglichkeiten wurden bereits in Abschnitt 7.4.1 beschrieben.

D.10.1 Die Hardware-Konfiguration

Auf dem Bestückungsplan in Abbildung D.10 sind eine Reihe von Dip-Schaltern und Pfostensteckerleisten zu erkennen, auf denen mit Kurzschlusssteckern Einstellungen vorgenommen werden müssen.

Takt- und Sequencerauswahl In der unteren linken Ecke der Karte befinden sich zwei Pfostensteckerpaare JP1, über die die Taktquelle für den Sequencer und der Sequencer selbst ausgewählt werden können. Mit dem oberen der beiden Paare wird die Taktquelle des internen Seqeuncers ausgewählt. Ist hier ein Kurzschlussstecker gesetzt, wird der interne 15-MHz-Oszillator benutzt, wobei die Koaxialbuchse auf der Frontseite als Taktausgang dient. Sind die beiden Pfostenstecker dagegen offen, ist der interne Quarzoszillator deaktiviert. Es ist jetzt möglich, ein externes Taktsignal über die Koaxialbuchse in die Controllerkarte einzuspeisen.
Mit den beiden darunter liegenden Pfostensteckern kann man den Sequencer auswählen. Bei einem aufgesteckten Kurzschlussstecker wird der externe Sequencer, ohne Kurzschlussstecker dagegen der interne Sequencer aktiviert.

Die Sequencer-Programmauswahl Rechts neben dem EPROM, IC53, befindet sich eine weitere Pfostensteckerleiste mit zwei mal vier Pinnen, mit der eine Auswahl zwischen 16 verschiedenen Programmpaketen getroffen werden kann, die beim Einschalten des Sequencers aus dem EPROM in das RAM geladen werden können. Die Auswahl ist wieder mit Kurzschlusssteckern zu treffen, wobei ein gesetzter Stecker eine Eins an dem entsprechendem Bit erzeugt, ein offenes Pfostenpaar dagegen eine Null. Das niederwertigste Bit (LSB) befindet sich auf der Unterseite und ist mit 4 bezeichnet.

Die VMEbus Basisadresse Oben rechts, direkt neben der oberen Bus-Steckerleiste erkennt man zwei 8-fach-DIL-Schalter. Mit Hilfe dieser Schalter werden die oberen 16 Bits, d.h. A_8 bis A_{23} der VMEbus-Basisadresse eingestellt, an der das Controllerboard von der VMEbus-CPU angesprochen werden kann. Der untere Schalter $S1$ deckt die Bits A_8 bis A_{15} ab und der obere Schalter die Bits A_{16} bis A_{23}. Das niederwertigste Bit befindet sich auf beiden Schaltern jeweils links. Ein offener Schalter entspricht einer Null, ein geschlossener Schalter einer Eins.

Die Interrupt-Ebene Etwa in gleicher Höhe wie $S2$ befindet sich in der Mitte der Platine ein dreifach-DIL-Schalter, mit dem die Interrupt-Ebene der Controllerkarte eingestellt werden kann. Auch hier gilt wieder, dass ein offener Schalter einer Null und ein geschlossener Schalter einer Eins entspricht. Wenn alle drei Schalter offen sind, ist damit der Interrupt-Controller von

der Hardware gesperrt und kann von der Software nicht aktiviert werden. Bei einer anderen Einstellung wird bei einem auftretenden Interrupt ein VME-Interrupt der entsprechenden Ebene erzeugt.

Anhang E

Die Kristallzuordnung

E.1 Grundsätzliches

Abbildung E.1 zeigt die auf eine Ebene abgerollte Kristallmatrix des Barrels. Die Strahlrichtung des Photonenstrahls zeigt von links nach rechts. Die eingezeichneten Richtungen für den θ- und ϕ-Winkel entsprechen den Konventionen des Crystal-Barrel-Experimentes. Die geschlossene Torusstruktur der Kristallmatrix ist in der Horizontalebene aufgetrennt.
Die Grundnachbarschaft, die für die Clusteridentifikation benutzt werden muss, ist eine Moore-Nachbarschaft der Größe 1. Zur Einhaltung der korrekten Nachbarschaftsbeziehungen im Randbereich des Barrels, wo die Kristalle (Kristalltyp 11, 12 und 13) einen ϕ-Winkel von 12° statt 6° abdecken, müssen diese Kristalle auf zwei Logikzellen abgebildet werden. In Abbildung E.2, in der die 6 unterschiedlichen im Randbereich vorkommenden Nachbarschaftsbeziehungen dargestellt sind, ist zu sehen, dass damit die Topologie des Barrels wieder korrekt abgebildet wird.

E.2 Zuordnung der Kristalle auf die Shapermodule

Jeweils drei Receiver/Shaper-Module werden für einen Doppelsektor einer Barrelhälfte benutzt. Abbildung E.3 zeigt die Zuordnung der Kristalle eines Doppelsektors auf die Kanäle dieser drei Module. Der Index n nummeriert die Doppelsektoren. Er läuft von 1 bis 30, wobei der Sektor 61 ($n = 30$, $2 \cdot 30 + 1$) dem Sektor 1 entspricht. Der erste Doppelsektor ($n01$) umfasst also die Sektoren 2 und 3, der letzte ($n = 30$) die Sektoren 60 und 1.

E.3 Zuordnung auf die Diskriminatormodule

Die Verbindung zwischen Shaper und Diskriminatoren erfolgt über einzelne Koaxialkabel, sodass an dieser Stelle die Freiheit besteht, die Kanäle neu den Diskriminatoren zuzuordnen. Dabei erwies es sich für die folgende Weiterführung der Signale als vorteilhaft, wenn abwechselnd je ein Kristall des ersten und des folgenden Sektors auf die Diskriminatorkanäle gelegt wird, wie dies in Abbildung E.4 dargestellt ist. Die Aufteilung der gesamten Zellularlogik in ϕ-Richtung auf zwei Platinen, auf die die Barreldaten von verschiedenen Seiten zugeführt werden, macht

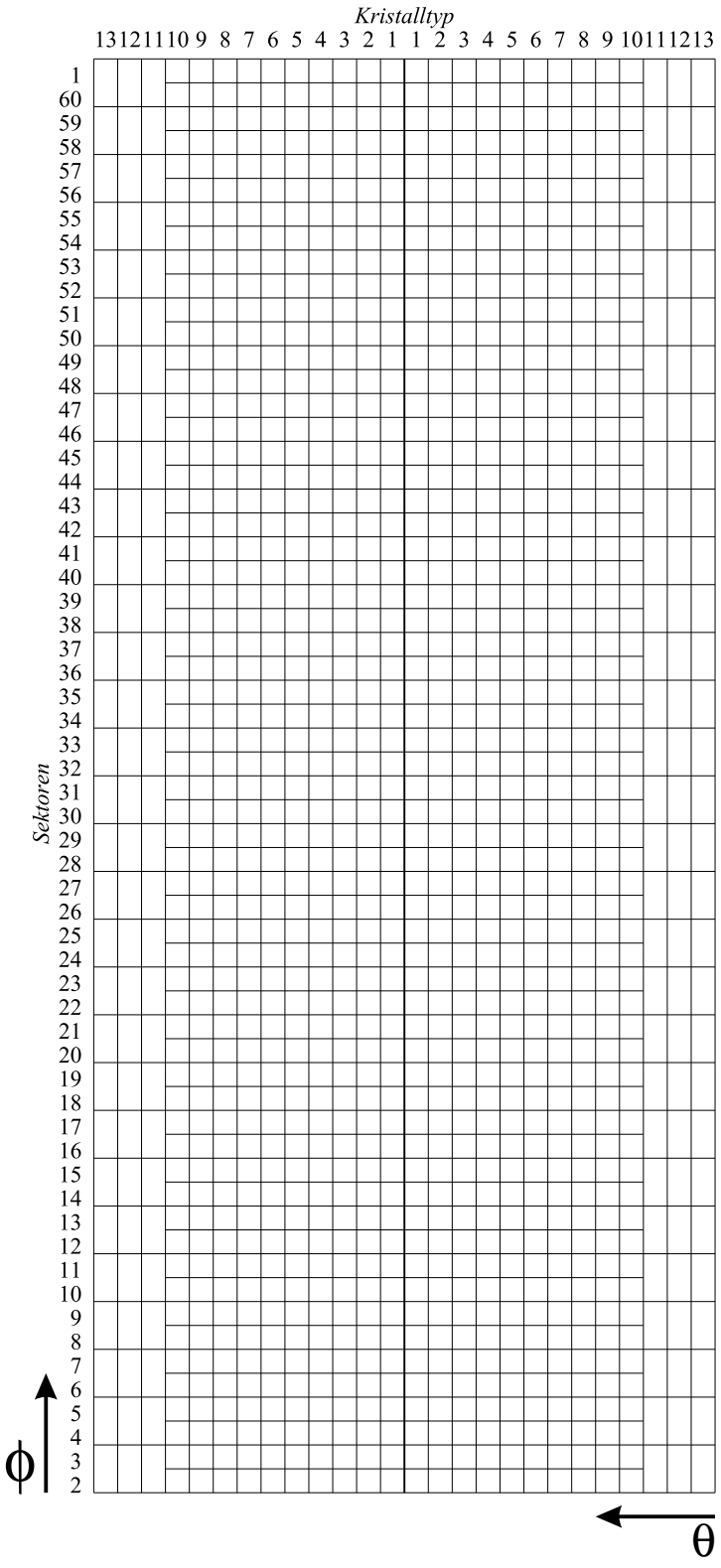


Abbildung E.1: Projektion der Kristallmatrix des Crystal Barrels auf eine Ebene

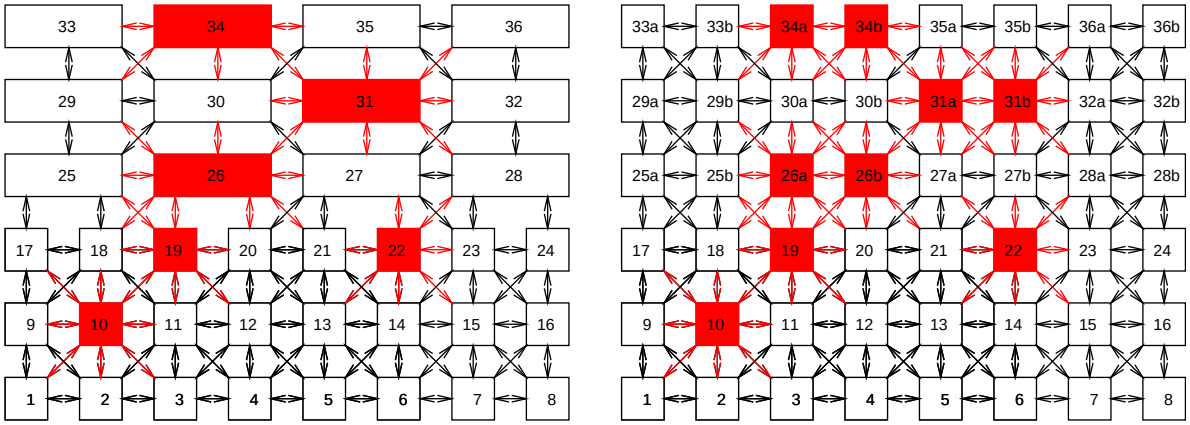


Abbildung E.2: Die unterschiedlichen Nachbarschaften im Randbereich der Barrelmatrix. Im rechten Bild wird gezeigt, dass diese Nachbarschaftsbeziehungen korrekt eingehalten werden, wenn jedem Randkristall zwei Zellen zugeordnet werden

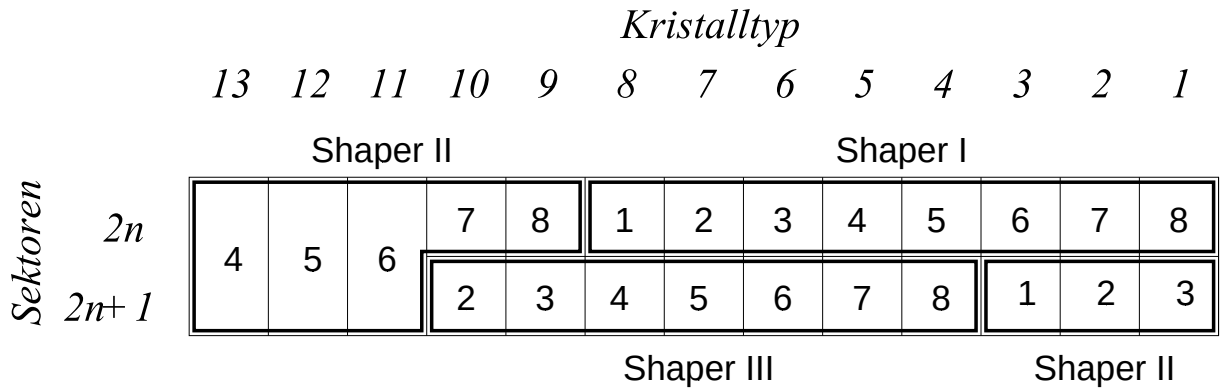


Abbildung E.3: Die Zuordnung der Kristalle eines Doppelsektors zu den Shapermodulen

zwei unterschiedliche Zuordnungsschemata erforderlich. Die Trennlinie liegt zwischen den Sektoren 33 und 34. Die dargestellten Zuordnungen gelten für die Up-Stream-Barrelhälfte. Für die Down-Stream-Barrelhälfte müssen die beiden Zuordnungsschemata vertauscht werden.

E.4 Zuordnung auf die Latchmodule

Von den Diskriminatoren werden die Signale über Flachbandkabel auf die Latchmodule geführt. Die Signale von jeweils drei Diskriminatormodulen, also 24 Kanäle, werden auf ein Latchmodul gegeben, das über 32 Eingangskanäle verfügt. Entsprechend der beiden verschiedenen Zuordnungsschemata bei den Diskriminatoren gibt es auch bei den Latchmodulen zwei verschiedene Zuordnungen. Diese sind in Abbildung E.5 wiedergegeben.
In den Latchmodulen werden die Signale aus dem Randbereich des Barrels auf jeweils zwei Ausgänge aufgeteilt. Die daraus resultierende Zuordnung der Ausgangskanäle der Latchmodule

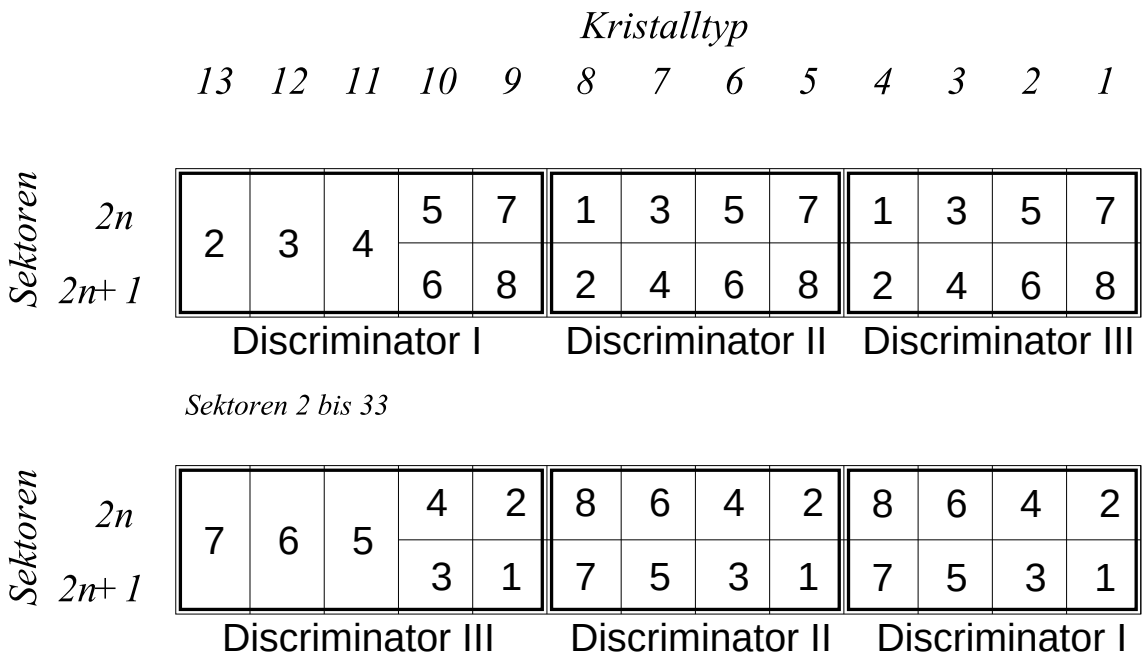


Abbildung E.4: Die Zuordnung der Diskriminatorkanäle

Abbildung E.5: Die Zuordnung der Latchkanäle

Kristalltyp

Sektoren	13	12	11	10	9	8	7	6	5	4	3	2	1
2n	6	8	10	12	14	16	18	20	22	24	26	28	30
2n+ 1	7	9	11	13	15	17	19	21	23	25	27	29	31

Sektoren 2 bis 33

Sektoren													
2n	25	23	20	19	17	15	13	11	9	7	5	3	1
2n+ 1	24	22	21	18	16	14	12	10	8	6	4	2	0

Sektoren 34 bis 1

Abbildung E.6: Die Zuordnung der Latchausgänge

zu den Logikzellen ist in Abbildung E.6 zu sehen. Die Signale werden von diesen Ausgängen über Flachbandkabel direkt auf die Eingänge des Clustertriggers gegeben.

Anhang F

Verwendete Software

F.1 Die Sequencer-Firmware

```
 >>>>> PASS 1 <<<<<
 >>>>> PASS 2 <<<<<


Assemblerausdruck von SEQ2.S

0000        ; Steuerleitungen des Sequencers
0000
0000        Reset   equ     000000000001b
0000        ResM    equ     000000000010b
0000        ResT    equ     000000000100b
0000        Mark    equ     000000001000b
0000        Mask    equ     000000010000b
0000        UnMask  equ     000000100000b
0000        XProj   equ     000001000000b
0000        YProj   equ     000010000000b
0000        ProjSel equ     000100000000b
0000        SetSel  equ     001000000000b
0000        Set     equ     010000000000b
0000        TestSet equ     100000000000b
0000
0000        ; Output-Bits fuer Switch-Register
0000
0000        EncEn   equ     0       ; Enable-Signal fuer Prioritaetsenc.
0000        LatchEn equ     1       ; Enebale-Signal fuer Encoderlatch
0000        DecEn   equ     2       ; Enabel-Signal fuer Adressdecoder
0000        Busy    equ     3       ; Busy-Signal fuer 1st-Level-Trigger
0000        CAEn    equ     4       ; Legt VME-Register auf Zelladressbus
0000        NewCl   equ     5       ; Flag fuer Clusteranfang
0000        NewEv   equ     6       ; Flag fuer Eventanfang
0000        DataVal equ     7       ; Data-Valid-Signal fuer 1st-Level-Trigger
0000
0000        ; Output-Bits des Pulsers
0000
0000        CntR1   equ     0       ; Reset fuer Zaehler 1
0000        CntC1   equ     1       ; Clock fuer Zaehler 1
0000        CntR2   equ     2       ; Reset fuer Zaehler 2
0000        CntC2   equ     3       ; Clock fuer Zaehler 2
0000        FiFoIn  equ     4       ; Clock-Signal fuer FIFO-Input
0000        FiFoR   equ     5       ; Reset-Signal fuer FIFO
0000        IntReq  equ     6       ; Interrupt-Request
0000
0000        ; Condition Bits fuer Sprungbefehle
0000
0000        uncon   equ     0
```

```
0000        XEmpty  equ     1
0000        YEmpty  equ     2
0000        StrtSig equ     3
0000        FastClr equ     4
0000
0000        ;
0000        ; Zunaechst ein Programm, mit dem die Trefferflipflops einzelner
0000        ; Zellen gesetzt werden koennen.
0000        ;
0000        ; Nachdem das Sequencerprogramm gestartet wurde, muss der Busmaster
0000        ; die Zelladressen der anzusprechenden Zelle in das Controllregister
0000        ; schreiben und einen Startcon-Puls ausloesen.
0000        ; Das Sequencerprogramm erzeugt dann die TestSet-Sequenz und wartet
0000        ; danach auf das naechste Ereignis.
0000        ;
0000        ; Das Programm muss extern beendet werden.
0000        ;
0000
0000  F000  testset nop             ;
0002  9A00          stb     DecEn   ; Adressdekoder deaktivieren
0004  9800          stb     EncEn   ; Prioritaetsencoder deaktivieren
0006  9100          clb     LatchEn ; Latches deaktivieren
0008  8007          stv     Reset+ResT+ResM ; Reset des Chips
000A  8000          stv     0       ;
000C  9C00          stb     CAEn    ; VME-Adressdaten auf ZA-Bus
000E  9200          clb     DecEn   ; Adressdekoder aktivieren
0010  8200          stv     SetSel  ; SetSel-Leitung auf Trefferflipflop
0012
0012  300B  wt_bit  jp      StrtSig,setbit  ; Startsignal gegeben?
0014  0009          jp      uncon,wt_bit    ; nein, weiter warten
0016
0016  8A00  setbit  stv     SetSel+TestSet  ; TestSet-Puls geben
0018  8200          stv     SetSel  ;
001A  0009          jp      uncon,wt_bit    ; Auf's naechste Bit warten
001C
001C        ;
001C        ; Moeglicherweise ist es notwendig, die Trefferflipflops einzeln zu setzen
001C        ; ohne dass dabei auf ein Start-Signal reagiert werden darf.
001C        ; Dazu gibt es zwei kurze Programme, um die Zellularlogik zu initialisieren
001C        ; und um eine einzelne Zelle zu setzen.
001C        ;
001C        ; Zunaechst das Programm zum Initialisieren
001C        ;
001C  9B00  init    stb     Busy    ; 1st-Level-Trigger mittteilen, dass wir
001E        ;                                   beshaeftigt sind
001E  8000          stv     0       ; Steuersignale Initialisieren
0020  8007          stv     Reset+ResT+ResM ; Reset des Chips
0022  8000          stv     0       ; und Steuersignale wieder im Grundzustand
0024  9300          clb     Busy    ;
0026  B000          hlt             ; Fertig
0028
0028        ;
0028        ; Und jetzt das Programm, um eine einzelne Zelle zu setzen
0028        ;
0028  9B00  set     stb     Busy    ; 1st-Level-Trigger mittteilen, dass wir
002A        ;                                   beshaeftigt sind
002A  9A00          stb     DecEn   ; Adressdecoder deaktivieren
002C  9800          stb     EncEn   ; prioritaetsencoder deaktivieren
002E  9C00          stb     CAEn    ; VME-Adressdaten auf ZA-Bus
0030  9200          clb     DecEn   ; Adressdecoder aktivieren
0032  8200          stv     SetSel  ; SetSel-Leitung auf Trefferflipflop
0034  8A00          stv     SetSel+TestSet  ; TestSet-Puls
0036  8200          stv     SetSel
0038  8000          stv     0       ; Steuersignale auf 0 setzen
003A  9A00          stb     DecEn   ; Adressdecoder wieder deaktivieren
003C  9400          clb     CAEn    ; VME-Adressen wieder vom Bus nehmen
003E  9300          clb     Busy    ; Busy deaktivieren
0040  B000          hlt             ; Fertig
```

```
0042
0042        ; Jetzt kommt ein Programm, das die Cluster zunaechst Zaehlt und dann
0042        ; eine Liste aller gesetzten Zellen nach Clustern sortiert
0042        ; ins FIFO schreibt.
0042        ;
0042        ; Waehrend das programm arbeitet ist Busy gesetzt. Nachdem Busy wieder
0042        ; geloescht wurde, steht im Zaehler die Anzahl der Cluster und im FIFO
0042        ; die Adressen aller gesetzten Zellen.
0042        ;
0042        ; Der Sequencer wird nach dem Programmablauf nicht gestoppt. Damit kann
0042        ; der korrekte Ablauf ueberprueft werden: nach dem korrekten Ablauf muss
0042        ; der Sequencer noch laufen, Busy dagegen geloescht sein.
0042        ;
0042        ; Dieses Programm dient lediglich dazu fuer Testzwecke die Auslese vorzunehmen,
0042        ; da es nciht auf externe Signale wie Start oder FastClear reagiert.
0042        ;
0042  9400  test_ro clb     CAEn    ; VME-Daten vom ZA nehmen
0044  9200          clb     DecEn   ; Adressdecoder aktivieren
0046  9000          clb     EncEn   ; prioritaetsencoder aktivieren
0048  9900          stb     LatchEn ; Latches auf transparent schalten
004A  A000          pulse   CntR1   ; Zaehler 1 Reseten
004C  A200          pulse   CntR2   ; Zaehler 2 Reseten
004E  A500          pulse   FiFoR   ; FIFO reseten
0050  9E00          stb     NewEv   ; New-Event-Flag setzen
0052  9700          clb     DataVal ; Daten nicht gueltig
0054  9B00          stb     Busy    ; Busy aktivieren
0056
0056  8042  cl_loop stv     XProj+ResM      ; XProjektion starten, Marker-FF loeschen
0058  8040          stv     XProj   ; ResM deaktivieren
005A  1036          jp      XEmpty,fettich  ; Leer, alle Cluster gezaehlt
005C  80C0          stv     YProj+XProj     ; YProjektion starten
005E  F000          nop                     ; Warten
0060  80C8          stv     YProj+XProj+Mark        ; markierung starten
0062  F000          nop                     ; Einen Zyklus warten
0064  8010          stv     Mask            ; markierten Cluster maskieren
0066  A100          pulse   CntC1           ; Cluster Zaehlen
0068  8000          stv     0               ; Alle Steuersignale deaktivieren
006A  002B          jp      uncon,cl_loop   ; naechster Cluster
006C
006C  9F00  fettich stb     DataVal         ; zaehler Daten nun gueltig
006E  8020          stv     UnMask          ; Maskierung aufheben
0070  8042  next_cl stv     XProj+ResM      ; Xprojektion starten, Marker-FF loeschen
0072  8040          stv     XProj           ; ResM deaktivieren
0074  1051          jp      XEmpty,ende     ; jetzt wirklich fertig
0076
0076  80C0          stv     YProj+XProj     ; YProjektion starten
0078  F000          nop                     ; einen Zyklus warten
007A  80C8          stv     YProj+XProj+Mark        ; Markierung starten
007C  9100          clb     LatchEn         ; latch deaktivieren
007E  9D00          stb     NewCl           ; New-Cluster-Flag aktivieren
0080  A400          pulse   FiFoIn          ; Adresse in FiFo schreiben
0082  8004          stv     ResT            ; zelle loeschen
0084  8100          stv     ProjSel         ; Nun markierte Zelle projizieren
0086  9900          stb     LatchEn         ; Latch wieder transparent schalten
0088  9500          clb     NewCl           ; Kein neuer Cluster mehr
008A  9600          clb     NewEv           ; Auch kein neues Event mehr
008C  8140  loop1   stv     ProjSel+XProj   ; XProjektion starten
008E  F000          nop                     ; Warten
0090  1038          jp      XEmpty,next_cl  ; Cluster abgearbeite, naechster Cluster
0092  81C0          stv     ProjSel+XProj+YProj     ; YProjektion starten
0094  F000          nop                     ; Warten
0096  9100          clb     LatchEn         ; Latch deaktivieren
0098  8104          stv     ProjSel+ResT    ; Zelle loeschen
009A  8100          stv     ProjSel         ;
009C  A400          pulse   FiFoIn          ; Adresse in FiFo schreiben
009E  9900          stb     LatchEn         ; Latch wieder transparanet schalten
00A0  0046          jp      uncon,loop1     ; Naechste Zelle
00A2
```

```
00A2  9300  ende    clb     Busy            ; Busy deaktivieren
00A4  9A00          stb     DecEn           ; Adressdecoder disablen
00A6  9800          stb     EncEn           ; Prioritaetsencoder deaktivieren
00A8  9100          clb     LatchEn         ; Latches sperren
00AA  0055  endls   jp      uncon,endls     ; Endlosschleife
00AC
00AC        ;
00AC        ; Das letzte programm schliessliche dient als Haupt Triggerloop. Es wartet
00AC        ; auf ein Startsignal, laed dann die Treffermuster in die Zellularlogik und
00AC        ; zahelt dann die Cluster. nach dem Zaehlen wird ueber die DataVal-Leitung dem
00AC        ; 1st-Level-Trigger ein Signal gegeben und mit der Auslese der Zelllisten
00AC        ; begonnen.
00AC        ; Diese Auslese kann vom 1st-Level-Trigger jederzeit mit einem FastClear-Signal
00AC        ; unterbrochen werden.
00AC        ; Wahrend der gsamten Aktion wird das Busy-Signal gesetzt
00AC        ;
00AC
00AC  9400  trigger clb     CAEn            ; VME-Daten vom ZA nehmen
00AE  9200          clb     DecEn           ; Adressdecoder aktivieren
00B0  9000          clb     EncEn           ; prioritaetsencoder aktivieren
00B2  9900          stb     LatchEn         ; Latches auf transparent schalten
00B4  8000          stv     0               ; Alle Steuersignale loeschen
00B6  A200          pulse   CntR2           ; zaehler 2 Reseten (wird z.Zt. nicht benoetigt)
00B8
00B8  8007  init_tr stv     Reset+ResM+ResT ; Zellularlogik initialisieren
00BA  8200          stv     SetSel          ; SetSel auf Treffer-FF setzen
00BC  9700          clb     DataVal         ; Data Valid-Signal zurueck nehmen
00BE
00BE  3061  evnt_lp jp      StrtSig,event   ; Liegt ein Event vor?
00C0  005F          jp      uncon,evnt_lp   ; nein, weiter warten
00C2
00C2  9700  event   clb     DataVal         ; Daten nicht gueltig
00C4  9B00          stb     Busy            ; Busy aktivieren
00C6  A000          pulse   CntR1           ; Zaehler 1 Reseten
00C8  8600          stv     Set+SetSel      ; Trefferdaten setzen
00CA  8200          stv     SetSel          ; Daten sind gesetzt
00CC
00CC  8042  cnt_lp  stv     XProj+ResM      ; XProjektion starten, Marker-FF loeschen
00CE  8040          stv     XProj           ; ResM deaktivieren
00D0  1071          jp      XEmpty,cnt_en   ; Leer, alle Cluster gezaehlt
00D2  80C0          stv     YProj+XProj     ; YProjektion starten
00D4  F000          nop                     ; Warten
00D6  80C8          stv     YProj+XProj+Mark        ; markierung starten
00D8  F000          nop                     ; Einen Zyklus warten
00DA  8010          stv     Mask            ; markierten Cluster maskieren
00DC  A100          pulse   CntC1           ; Cluster Zaehlen
00DE  8000          stv     0               ; Alle Steuersignale deaktivieren
00E0  0066          jp      uncon,cnt_lp    ; naechster Cluster
00E2
00E2  9F00  cnt_en  stb     DataVal         ; zaehler Daten nun gueltig
00E4  A500          pulse   FiFoR           ; FIFO mit Zaehler initialisieren
00E6  9E00          stb     NewEv           ; NewEvent-Falg aktivieren
00E8  8020          stv     UnMask          ; Maskierung aufheben
00EA
00EA  4093  nxt_cl  jp      FastClr,fstcl   ; Fast-Clear
00EC  8042          stv     XProj+ResM      ; Xprojektion starten, Marker-FF loeschen
00EE  8040          stv     XProj           ; ResM deaktivieren
00F0  1090          jp      XEmpty,ro_end   ; jetzt wirklich fertig
00F2
00F2  80C0          stv     YProj+XProj     ; YProjektion starten
00F4  F000          nop                     ; einen Zyklus warten
00F6  80C8          stv     YProj+XProj+Mark        ; Markierung starten
00F8  9100          clb     LatchEn         ; latch deaktivieren
00FA  9D00          stb     NewCl           ; New-Cluster-Flag aktivieren
00FC  A400          pulse   FiFoIn          ; Adresse in FiFo schreiben
00FE  8004          stv     ResT            ; zelle loeschen
0100  8100          stv     ProjSel         ; Nun markierte Zelle projizieren
0102  9900          stb     LatchEn         ; Latch wieder transparent schalten
```

```
0104  9500          clb     NewCl           ; Kein neuer Cluster mehr
0106  9600          clb     NewEv           ; Auch kein neues Event mehr
0108  4093  loop2   jp      FastClr,fstcl   ; Fast-Clear
010A  8140          stv     ProjSel+XProj   ; XProjektion starten
010C  F000          nop                     ; Warten
010E  1075          jp      XEmpty,nxt_cl   ; Cluster abgearbeite, naechster Cluster
0110  81C0          stv     ProjSel+XProj+YProj     ; YProjektion starten
0112  F000          nop                     ; Warten
0114  9100          clb     LatchEn         ; Latch deaktivieren
0116  8104          stv     ProjSel+ResT    ; Zelle loeschen
0118  8100          stv     ProjSel         ;
011A  A400          pulse   FiFoIn          ; Adresse in FiFo schreiben
011C  9900          stb     LatchEn         ; Latch wieder transparanet schalten
011E  0084          jp      uncon,loop2     ; Naechste Zelle
0120
0120  9300  ro_end  clb     Busy            ; Busy deaktivieren
0122  A600          pulse   IntReq          ; An CPU Melden, dass Daten vorliegen
0124
0124  005C          jp      uncon,init_tr   ; Trigger neu initialisieren fuer's naechste
0126        ;                                 Event
0126  A500  fstcl   pulse   FiFoR           ; FiFo gleich wieder loeschen
0128  9300          clb     Busy            ; Busy-Leitung deaktivieren
012A  005C          jp      uncon,init_tr   ; und auf auf's naechste Event warten
012C
```

F.2 Die Datenacquisistion für den Clustertrigger

F.2.1 Das Header-File

```
#ifndef __FACE_DAQ_
#define __FACE_DAQ_

#include "VIC/vicmap_elba.h"
#include "vmv.h"
#include "absolutes.h"
#include "structs.h"
```

// Status-bits in the BCCE-Controller-status-register

```
#define BSTAT_BUSY 1
#define BSTAT_EMPTY 2
#define BSTAT_FULL 4
#define BSTAT_RUNNING 8
#define BSTAT_TIMEOUT 16
#define BSTAT_START 256
#define BSTAT_STOP 512
#define BSTAT_STARTCON 1024
#define BSTAT_RESET 2048
#define BSTAT_IMASK 4096
#define BSTAT_IEN 8192
#define BSTAT_IRES 16384
#define BSTAT_IPEN 32768
```

// Control-bits in the latch-control-register

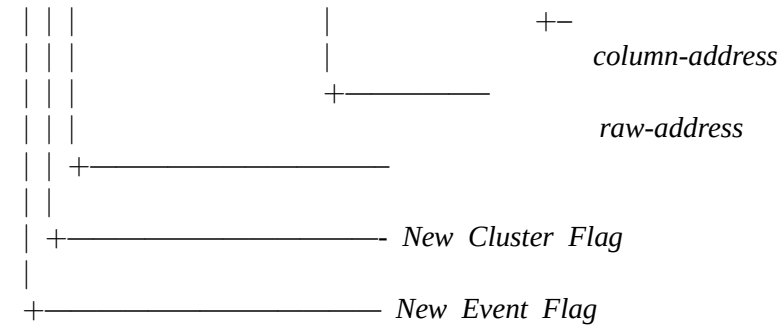
```
#define BLC_TESTEN 1
#define BLC_TESTDATA 2
#define BLC_TESTCLK 4

//start-addresses of some Sequencer-routines
#define SEQUENCER_TRIGGER_LOOP 86
#define SEQUENCER_CELL_SET 0
#define SEQUENCER_ANALYZE 14

// base-address of the bcce-controller-board
#define BCCE_CONTROLLER_BASIS 0xfcef0100

// the Controller registerset
struct bcreg
{
   short status;
   short latch;
   short start_adr;
   short cell_adr;
   short fifo;
   short counter;
   short cell_counter;
};

typedef struct bcreg BCCE_REGISTERS;


extern "C" int tsleep(int);

int initFace(void);
int resetFace(void);
int readFace(FACE_BUFFER *genBuf);
int stopFace( void );
short* latchControl( void );
void startCon( void );


#endif
```

F.2.2 Der Programmcode

```
#define EVB_SRC

#include <stdlib.h>
#include <stdio.h>
#include <time.h>
```

```
#include "face.h"                                   // Including some header-files

static VMV_REGISTERS *vreg;                  // Pointer to the VME-Interface-registerset
static BCCE_REGISTERS *breg;                 // Pointer to the BCCE-Controller-registerset

// initFace initialize the VMV_interface _and_ the BCCE_Controller

unsigned short get_status( void )
{
  return breg->status;
}


int initFace(void)
{
  vreg = (VMV_REGISTERS *) (VMV_REGISTER_BASE);          // Pointer to the beginning
                                                         // of registerset
  vreg->csre = 0x0100;                                                          //
  vreg->inth = 0x0000;                                                          //
  vreg->mapam = 0x39;                                    // Adressmodifier-Code 0x39
  vreg->mapad = 0x0000;                                  // MSB Adressbits 0x000
  vreg->mapcr = 2;                                       // VMV-Crate 2
  vreg->timout = 0x0030;                                 // timeout

// next, we set the breg-pointer to the begining of the controller-registerset

  breg = (BCCE_REGISTERS* ) (BCCE_CONTROLLER_BASIS);

  /* first we generate a reset-signal to initialize the Controller. This
     means toggle the RESET-Bit to low-state.
     As well we generate a high-state on the IRES-bit, to initialize the
     interrupt-unit of the card.

     then interrupts will be enabled (IEN to high-state), but all
     interrupts stay masked
  */

  breg->status = BSTAT_IRES;                                                    //
  breg->status = BSTAT_RESET | BSTAT_IEN;                                       //

  /* At this point a delay of at least 2.7 ms has to be inserted, because
     this time is needed by the sequencer to load the program-code from
     EPROM into RAM
  */

  int ticks = tsleep(0x80000100);
```

```
  // switch latches into normal mode
  breg→latch = 0;

  unsigned short stat = get_status();                          // get status-register

  // startaddress of the sequencer-trigger-loop
  breg→start_adr = SEQUENCER_TRIGGER_LOOP;

  // starting the sequencer
  breg→status = stat | BSTAT_START;
  breg→status = stat;


  if (!(get_status() & BSTAT_RUNNING) )                  // Is the sequencer running ?
    {
      fprintf(stderr,"Sequencer laeuft nicht Statuswort :  %x\n",get_status());

      return -1;                                         // No, This is a fatal Error !!
    }
  else
{
fprintf(stderr,"Sequencer laeuft.\n");
    return 0;                                            // Yes, trigger is clear for first
}  // event
}

// resetFace just resets the interrupt-unit of the trigger-controller
int resetFace(void)
{

  unsigned short stat = get_status() & ~BSTAT_IRES;          // get the status-register
  breg→status = stat | BSTAT_IRES;                         // toggle IRES-bit to high
  breg→status = stat;                                        // an clear IRES-bit

  return 0;
}


/* readFace is the readout-function for the cluster-trigger. The function
   needs a buffer of type FACE_BUFFER in which the cluster-data will be
   written.

   First short-word consists of the two cluster-counter, the lower byte is
   counter a and the higher byte ist cluster b.
   all other words are cell addresses, builded by the following scheme:

   Bit 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
                   | | |                 |              |
```




```
*/

int readFace(FACE_BUFFER *genBuf)
{
  unsigned short *rawBuf;                                   // buffer-pointer

  if(genBuf)                                  // no readout, if there is no buffer-space
  {
    volatile unsigned int stat = get_status() & BSTAT_BUSY;
    int cnt = 0;
    while (stat && cnt++ < 100)
      stat = get_status() & BSTAT_BUSY;
    fprintf(stderr,"Busy Delaycounter %d \n",cnt);

    if (!stat)
    {
      // first we set the rawBuf-pointer to the first word of the data-buffer
      rawBuf = (unsigned short *) &(genBuf->data[0]);

      // writing the cluster-counter in the first word of the buffer
      *rawBuf++ = breg->counter;

      // getting the number of cell-addresses an writing them into the buffer
      int anzahl = (int) breg->cell_counter;
      for (int i = 0;i<anzahl;i++)
        *rawBuf++ = breg->fifo;

      // the size of the needed buffer-space is:
      //          2 words : size-entry
      //          1 word  : cluster-counter
      // <anzahl> words : cell-addresses

      genBuf->size= 3+anzahl;

      return 0;
    }
    else
      return -1;                                          // Error: Busy-timeout
  }
```

```
   else
      return -2;                                          // Error: No Buffer-space
}

// stopFace stops the sequencer

int stopFace( void )
{
   // getting the status-register
   unsigned short stat = get_status() & ~BSTAT_STOP & ~BSTAT_START & ~BSTAT_IEN;


   breg->status = stat | BSTAT_STOP;                      // toggle the STOP bit to high-state
   breg->status = stat;                                   // and clear the STOP bit

   return 0;
}

// latchControl returns a pointer to the latch-control-register in the
// bcce-controller. this is useful, while using the testpattern of the
// Latch-modules

short* latchControl( void )
{
   return &(breg->latch);                  // returns the pointer to the latch-control-register
}

// this function is just for test-capabilities

void startCon( void )
{
   unsigned short stat = get_status() & ~BSTAT_STARTCON;

   breg->status = stat | BSTAT_STARTCON;
   breg->status = stat;
}
```

Literaturverzeichnis

[Af98] A. AFANASEV, P.R. PAGE: Photoproduction and electroproduction of $J^{PC} = 1^{-+}$ exotics
Phys. Rev. D57 (1998) 6771

[Am00] Austria Mikrosysteme AG: CMOS Technology Selection Guide
Produktinformation der Firma AMS im Internet unter
www.ams.co.at/products/technology/sel_cmos.html, Stand 24.3.2000

[Ar98] Arbeitsbericht über den Bau eines Kalorimeter-Triggers für Experimente mit dem Crystal Barrel Detektor an der Elektronenstretcher Anlage ELSA
Ruhr-Universität Bochum, 1998

[As90] M. ASNER: A cluster-finding trigger processor
NIM A291 (1990) 577-586

[Bh88] R.K. BHADURI: Models of the Nucleon
Addison Wesley, 1988

[Br86] K. BRAUNE: The Fast Cluster Encoder
CB-Note-41, 12.6.1986

[Bu91] V. D. BURKERT: Light Quark Baryons
in Nuclear Physics B (Proc. Suppl.) 21 (1991) 232-242

[Ca86] S. CAPSTICK, N. ISGUR: Baryons in a relativized quark model with chromodynamics
Physical Review D 34 (1986) 2809-2835

[Ca98] C. CASO ET. AL.: The Particle Data Book
The European Physical Journal C3 (1998) 1

[CB00] DIE CRYSTAL-BARREL-ELSA-KOLLABORATION: Anlage zum Antrag an die Deutsche Forschungsgemeinschaft auf Gewährung von Sachbeihilfen innerhalb des Schwerpunktprogramms ,,Untersuchung der hadronischen Struktur von Nukleonen und Kernen mit elektromagnetischen Sonden”
Universität Bonn, 2000

[CB92] DIE CRYSTAL-BARREL-KOLLABORATION: The Crystal Barrel spectrometer at LEAR
NIM A321 (1992) 69-108

[Cl79] F. CLOSE: An Introduction to Quarks and Partons
Academic Press, London, New York, San Francisco 1979

[Eh00] A. EHMANNS: Entwicklung, Aufbau und Test eines neuen Auslesesystems für den Crystal-Barrel-Detektor zur Messung photoinduzierter Reaktionen an ELSA
Dissertation, Bonn 2000

[Fa68] D. FAIMAN, A.W. HENDRY: Harmonic-Oscillator Model for Baryons
Phys. Rev. 173, 1720 (1968)

[Fö99] A. FÖSEL und die CB-ELSA-Kollaboration: Photoproduction of η and η' Mesons Using the Crystal Barrel Detector at ELSA
Proposal to the PAC, Bonn 1999

[Gl96] L.Y. GLOZMAN, D.O. RISKA: Quark model explanation of the N* $\rightarrow$Nη branching ratios
Phys. Lett. B366 (1996) 305

[Go85] S. GODFRAY, N. ISGUR: Mesons in a relativized quark model with chromodynamics
Phys. Rev. D32 (1985) 189

[Go98] A. GOLDSCHMIDT: Entwicklung und Test einer Triggerelektronik für CB-ELSA
Diplomarbeit Bochum 1998

[Go99] R. W. GOTHE und die CB-ELSA-Collaboration: Inelastic Photon Scattering in the Exclusive Channels p(γ,$\pi^0\gamma$p) und p(γ,$\eta\gamma$p)
Proposal to the PAC, Bonn 1999

[Hö98] G. HÖHLER: Status of the Nucleon Resonances $S_{11}(1535)$ and $S_{11}(1650)$
πN Newslett. 14 (1998) 168

[IE76] CAMAC Instrumentation and Interface Standards
The Institute of Electrical and Electronics Engineers, 1976

[Is77] N. ISGUR, G. KARL: Hyperfine interactions in negative parity Baryons
Phys. Lett. 72B (1977) 109

[Ka95] N. KAISER, P.B. SIEGEL, W. WEISE: Chiral dynamics and the $S_{11}(1535)$ nucleon resonance
Phys. Lett. B362 (1995) 23

[Ke95] TH. KERSCHNER: Test einer zellulären Logik zur schnellen Clusteridentifizierung bei zweidimensionalen Detektorfeldern
Diplomarbeit Bochum 1995

[Ko01] B. KOPF: Dissertation in Vorbereitung

[Le94] W. R. LEO: Techniques for Nuclear and Partiticle Physics Experiments
Springer-Verlag, 1994

[Li69] D. B. LICHTENBERG: Baryon Supermultiplets of $SU(6) \times O(3)$ in a Quark-Diquark Model
Phys. Rev. 178 (1969) 2197

[Ma88] H. MATTHÄY: A cellular logic array for twodimensional cluster identification
Beiträge zur Theorie der Polyautomaten - Vierte Folge -, Hrsg.: R. Vollmar, U. Golze, TU Braunschweig, 1988

[Me98] U.G. MEISSNER: Chiral Dynamics — Status and Perspectives
Proceedings of BARYONS 98, World Scientific S. 135

[Me98a] B.C. METSCH: A relativistic quark model for mesons and baryons
Proceedings of BARYONS 98, World Scientific S. 61

[Mu88] G. MUSIOL, J. RANFT, R. REIF, D. SEELIGER: Kern- und Elementarteilchenphysik
VCH Verlagsgesellschaft 1988

[Ne66] J. VON NEUMANN: A system of 29 states with a General Transistion Rule
in: Theory of Self Reproducing Automata, University of Illinois Press, 1966

[No91] R. NOVOTNY: The BaF_2 Photon Spectrometer TAPS
IEEE Trans. on Nucl. Sc. 38 (1991) 378

[Po93] POVH, RITH, SCHOLZ, ZETSCHE: Teilchen und Kerne
Springer, 1993

[Sc89] W. SCHOTT Das CsI(Tl) Kalorimeter des Crystal Barrel Detektors
Dissertation, Karlsruhe, 1989

[Sc94] W. J. SCHWILLE ET. AL.: Design and construction of the SAPHIR detector
NIM A 344 (1994) 470-486

[Sm99] J. SMYRSKI und die CB-ELSA-Collaboration: Study of Δ^* resonances decaying into $\Delta(1232)\eta$ and search for the exotic meson $\hat{\rho}(1380)$ in the reaction $\gamma p \to p\pi^0\eta$ using the CB-ELSA detector at ELSA
Proposal to the PAC, Bonn 1999

[Th99] U. THOMA und die CB-ELSA Collaboration: Study of Baryon resonances decaying into $\Delta(1232)\pi^0$ in the reaction $\gamma p \to p\pi^0\pi^0$ with the Crystal Barrel detector at ELSA
Proposal to the PAC, Bonn 1999

[Vi87] VMEbus International Trade Association (VITA): The VMEbus Specification
VITA-Publication, 1987

[Vo79] R. VOLLMAR: Algorithmen in Zellularautomaten
Teubner 1979

[We94] N. E. WESTE, K. ESHRAGHIAN: Principles of CMOS VLSI Design
Addison Wesley 1994

[Wi86] N. WIRTH: Compilerbau
Teubner 1986

Abbildungsverzeichnis

Tabellenverzeichnis

Danksagung

Diese Arbeit bliebe letztlich unvollständig, würde ich zum Schluss nicht noch all jenen ein Wort des Dankes widmen, die auf ihre Weise zum Entstehen dieser Arbeit beigetragen haben.

An erster Stelle sind hier sicherlich mein Themensteller, Herr PROF. DR. HELMUT KOCH und mein Betreuer Herr DR. HELMUT MATTHÄY zu nennen. Beide hatten stets ein offenes Ohr für meine Probleme und Wünsche und haben dafür gesorgt, dass mir jederzeit alle für die Bewerkstelligung der Arbeit notwendigen Hilfsmittel zur Verfügung standen.

Ein Dank gilt aber auch allen anderen Mitarbeitern des Lehrstuhls von Prof. Koch, für die freundschaftliche Aufnahme und die angenehme Arbeitsatmosphäre. Ein besonderer Dank gilt dabei Herrn HARALD A.S. DEPPE und Herrn DR. BERND LEWANDOWSKI für die Einführung in die Benutzung des Halbleiter-CAD-System von Cadance. Letzterer stand mir zudem selbst aus dem fernen Kalifornien als Systemadministrator zur Seite, damit meine Arbeit fortschreiten konnte, während H. Deppe mir auch nach seinem Wechsel zum ASIC-Labor der Uni Heidelberg neben zahlreichen Unverschämtheiten auch den ein oder anderen nützlichen Tipp übermittelte und mir zeigte, dass man manchmal ungewöhnliche Wege im Halbleiterdesign beschreiten muss.

Nicht unerwähnt bleiben sollten auch die Mitarbeiter der Elektronik- und der Feinmechanikwerkstatt. Sie waren beim Aufbau und Test der vielen Elektronikkomponenten eine wertvolle Hilfe

Ein Dank gilt allen Kollegen in der CB-ELSA-Kollaboration für die gute Zusammenarbeit auch über Institutsgrenzen hinweg. Stellvertretend für alle anderen möchte ich Herrn DIETER WALTHER, der eine unschätzbare Hilfe beim Einbau des Triggers in Bonn war, und Herrn DR. JÖRN LANGHEINRICH erwähnen, der eine nahezu unerschöpfliche Ideenquelle für neue Histogramme und Darstellungsformen zum Test des Gesamtsystems war.
Bei der Auswertung und Darstellung der ersten Messergebnisse des CB-ELSA-Experimentes war Herr BERTRAM KOPF eine wichtige Hilfe. An dieser Stelle vielen Dank dafür.

Für Korrekturen zu dieser Arbeit vielen Dank an Herrn DR. VOLKER CREDE, Herrn DR. MATTHIAS STEINKE, Herrn DR. HELMUT MATTHÄY und Frau MICHAELA PASTORS.

Ein besonderer Dank gilt all meinen Freunden im Deutschen Amateur-Radio-Club e.V. und im Wittener Luftsport-Club e.V. (damit seien auch die ‘Kamener’ eingeschlossen), wo ich gerade auch dann, wenn diese Arbeit mal nicht so gut lief, den nötigen Ausgleich fand. Besonders erwähnen möchte ich dabei Herrn THOMAS PFEIFFER, der diese Arbeit ganz nebenbei auch noch

mit hilfreichen Hinweisen zu Produktionsmöglichkeiten von Platinen und Elektronikbaugruppen erleichterte.

Ein letzter mir aber deshalb nicht minder wichtiger Dank gilt meinen Eltern, die mich über alle die Jahre in jeder Beziehung unterstüzten und ohne die ich diese Arbeit sicher nicht hätte bewerkstelligen können.

Lebenslauf

Name:	Holger Flemming
Geburtstag:	13. Oktober 1968
Geburtsort:	Wattenscheid
Eltern:	Heinz Flemming Margarete Flemming, geb. Lewandowski
Staatsangehörigkeit:	deutsch
Schulausbildung:	**1975-1979** Besuch der Grundschule „Ruhrstraße“ in Bochum-Wattenscheid **1979-1980** Besuch der Hauptschule „Ruhrstraße“ in Bochum-Wattenscheid **1980-1989** Besuch des Gymnasiums „Märkische Schule“ in Bochum-Wattenscheid **5.5.1989** Abitur Vom 5.6.1989 bis zum 31.8.1990 Grundwehrdienst, zunächst in Neumünster, dann in Rendsburg
Universitäts- ausbildung	**Oktober 1990** Beginn des Physikstudiums **Ab Juni 1995** Anfertigung der Diplomarbeit mit dem Thema „Aufbau einer Ausleseelektronik für eine Miniaturdrahtkammer“ **30.7.1996** Diplom **Ab dem 1.9.1996** Anstellung als Wissenschaftlicher Mitarbeiter am Institut für Experimentalphysik der Ruhr-Universität Bochum