Entwicklung und Realisierung des Quadranten-Steuerwerks für die pn-CCD-Kamera an Bord der Satelliten XMM und ABRIXAS

Diplomarbeit

vorgelegt von

Bärbel Kretschmar

Eberhard-Karls-Universität Tübingen
Institut für Astronomie und Astrophysik, Abt. Astronomie

27. Juni 1997

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Seit 1949 Burnright und seine Mitarbeiter vom NRL (Naval Research Laboratory, USA) bei einem Raketenflug entdeckten, daß die Sonne Röntgenstrahlen aussendet, vor allem aber seit 1962 bei dem Versuch Röntgenstrahlung vom Mond zu messen statt dessen die erste andere Röntgenquelle Sco X-1 gefunden wurde, sind die Bereiche der Röntgen- und Gamma-Astronomie zunehmend wichtige Gebiete bei der Erforschung des Weltalls geworden. Da Photonen aus diesem Energiebereich von der Erdatmosphäre absorbiert werden, ist es allerdings unumgänglich für die Hochenergie-Astronomen, ihre Meßinstrumente in sehr große Höhen zu bringen — sei es kurzzeitig mittels ballistischer Raketen oder Ballonflügen oder gleich für mehrere Jahre in Form von Satelliten. Das Astronomische Institut Tübingen (AIT, heute IAAT — Institut für Astronomie und Astrophysik, Abteilung Astronomie) hat in den letzten Jahrzehnten in all diesen Bereichen Erfahrung sammeln können:

In den späten 70ern und frühen 80er Jahren wurden sowohl Ballonkampagnen (Australien, Brasilien, USA), als auch Raketenexperimente (Südspanien, USA) mit Tübinger Beteiligung durchgeführt. Ein vor allem für spätere Experimente wichtiges Beobachtungsobjekt sowohl der Ballon-HEXE[1] als auch des Skylark-Raketenexperiments war der Crab-Nebel, ein Supernova-Überrest (SNR) der gerne als Referenz genommen wird [Staubert *et al.* 1975, bzw. Kendziorra 1976]. Eine weitere, herausragende, Erkenntnis, die aus den Daten der Ballon-HEXE gewonnen wurde, ist die Existenz von Zyklotron-Resonanz-Linien in Röntgen-Doppelstern-Systemen, gemessen an Her X-1 [Trümper *et al.* 1978].

Das AIT war bereits an der Entwicklung des 1983 gestarteten EXOSAT beteiligt. Durch seinen hochexzentrischen Orbit mit einer Umlaufzeit von 90 Minuten, war es mit diesem Satelliten erstmals möglich, lange ununterbrochene Messungen von Röntgenquellen durchzuführen.

Aufbauend auf den Erfahrungen mit der Ballon-HEXE wurde dann die MIR-HEXE gebaut. Sie bildet mit drei weiteren Hochenergie-Experimenten das Kvant-Modul, das an der russischen Raumstation MIR angedockt ist, und wurde im März 1987 in seine Umlaufbahn gebracht, rechtzeitig zur Beobachtung der Supernova 1987a [Englhauser 1996]. Sie liefert auch heute noch sporadisch Daten. Wichtige Forschungsobjekte waren auch hier wieder X-Ray Binaries neben AGNs (Active Galactic Nuclei) und Supernova-Überresten. Anhand des Crab-Nebels erfolgte die Eichung der vier Phoswich-Detektoren [Kretschmar 1991].

Es folgte die Beteiligung an dem 1990 gestarteten ROSAT[2], dessen vordringliche Aufgabe es war, eine Himmelsdurchmusterung im weichen Röntgenbereich von ca. 0,1 keV bis 2 keV durchzuführen. Zu diesem Zweck hat ROSAT als einer der ersten Satelliten ein abbildendes System für Röntgenstrahlen an Bord.

[1] **H**igh **E**nergy **X**-Ray **E**xperiment
[2] **Ro**entgen **Sat**ellit

Die letzten Missionen, bei denen Instrumente mit Tübinger Hardware in die Umlaufbahn gebrachten wurden, waren die Shuttle-Missionen ORFEUS I und II (Orbital and Retrievable Far and Extreme Ultraviolet Spectrometer) im September 1993 und November 1996. Sie bauten auf den ARIES-Höhenforschungs-Raketenexperimenten auf, die photometrische Halo-Beobachtungen gemacht hatten. Als wichtigstes ORFEUS-Ergebnis ist die Messung der Säulendichte intergalaktischer Wolken anhand der Lymen-K_α-Kante zu nennen [Appenzeller *et al.* 1995].

Durch diese langjährige Erfahrung mit Flughardware arbeitet das IAAT heute an vorderster Front der Forschungssatelliten mit; seit einigen Jahren ist es sogar bundesweit das einzige Universitätsinstitut, das überhaupt an der Entwicklung und am Bau von Satelliten für die Astronomie beteiligt ist.

Derzeitige größere Hardware-Projekte sind:

- Als Nachfolge-Projekt von ORFEUS II soll es zum einen einen ORFEUS III geben; zum zweiten soll in Zusammenarbeit mit Russland der Satellit SUV (Spektrum UV) entstehen.
- Im Hochenergie-Bereich wird seit kurzem an der Entwicklung des Gamma-Ray-Imager IBIS mitgearbeitet, der an Bord des Satelliten INTEGRAL (International Gamma-Ray Astrophysics Laboratory) im April 2001 starten soll.
- Bereits seit einigen Jahren ist das IAAT an Entwicklung und Bau einer der Röntgen-Kameras an Bord des Satelliten XMM (X-Ray Multi Mirror Mission) beteiligt. Dies ist ein sogenanntes „Cornerstone“-Projekt aus dem Programm „Horizon 2000“ der ESA (Kapitel 2).
- Auf der Basis dieser pn-CCD-Röntgen-Kamera von XMM beruht der Klein-Satellit ABRIXAS (A Broad band Imaging X-Ray All-sky Survey), ein Projekt der DARA, an dem das Institut demzufolge ebenfalls beteiligt ist.

Die vorliegende Diplomarbeit wurde im Rahmen der beiden letztgenannten Projekte XMM und ABRIXAS erstellt. Es handelt sich um Entwurf, Aufbau und Test einer digitalen Steuereinheit für die CCD-Auslese und die festverdrahtete Signalanalyse der An-Bord-Datenverarbeitung der pn-CCD-Kamera.

Um einen Überblick über diese Satelliten zu bekommen, sollen sie im folgenden kurz vorgestellt werden.

Kapitel 2

Die Missionen XMM und ABRIXAS

Im Jahre 1984 stellte die ESA[1] ein Forschungsprogramm zusammen, das sehr viele Gebiete der Astronomie abdeckt und — da unter anderem Materialproben eines Kometen entnommen werden sollen — auch für andere wissenschaftliche Bereiche von Interesse sein kann. Im Rahmen von „Horizon 2000“ werden seither mehrere Weltraum-Missionen entwickelt, deren letzte kurz nach der Jahrtausendwende gestartet sein soll [Burke 1996].

Es handelt sich dabei um Projekte unterschiedlicher finanzieller Größenordnungen: Kleine Projekte sind z.B. die Kometen-Sonde Giotto, eine Beteiligung am Hubble-Space-Telescope, der Astrometrie-Satellit Hipparcos, mit dessen Daten unter anderem hier am Institut der Tycho-Katalog erstellt wurde, oder auch ISO (Infrared Space Observatory), der dann doch zu einer recht großen Mission wurde. Dann gibt es einige mittelgroße Projekte: M1 ist die Huygens-Sonde an Bord des US-Satelliten CASSINI, die Proben des Saturn-Mondes Titan entnehmen soll; M2 ist der (ebenfalls mit IAAT-Beteiligung gebaute) INTEGRAL (International Gamma-Ray Astrophysics Laboratory) und M3 ist derzeit in der Definitionsphase. Als Wichtigstes sind jedoch die vier „Cornerstones“ zu nennen:

1. Das Solar Terrestrial Science Programm STSP bildet den ersten Eckstein bestehend aus den Missionen SOHO (SOlar and Heliospheric Observatory, seit 2. Dezember 1995 in Orbit) und Cluster, dessen 4 Satelliten im Feuerwerk des ersten (fehlgeschlagenen) Ariane 5-Starts zerstört wurden. Allerdings soll Cluster komplett nachgebaut und bei zwei Soyuz-Starts Mitte 2000 in die Umlaufbahn gebracht werden. Damit kann dann doch das vollständige Programm von Sonnenbeobachtungen und drei-dimensionalen erdnahen Plasma-Messungen durchgeführt werden.
2. Auf den zweiten Eckpfeiler, die X-ray Multi-mirror Mission XMM, soll in Abschnitt 2.1 ausführlicher eingegangen werden.
3. Als dritter Cornerstone soll Rosetta im Januar 2003 mit einer Ariane 5 starten und den Kometen Wirtanen im August 2011 treffen, wo 2013 eine Sonde abgesetzt wird, die Proben entnehmen und analysieren soll.
4. Der vierte (und letzte) Pfeiler von Horizon 2000 ist FIRST (Far-Infrared and Submillimetre Space Telescope), dessen Ariane 5-Start für das Jahr 2007 vorgesehen ist.

Inzwischen wurde das Programm um Horizon 2000 Plus ergänzt: Es sind weitere kleine und mittlere Projekte geplant, die Beteiligung an der Raumstation ist hinzugekommen und drei weitere Cornerstone-Missionen sind vorgesehen: Eine Merkur-Mission, das Interferometer-Observatorium GAIA (Global Astrometric Interferometer for Astrophysics) und das Gravitationswellen-Observatorium LISA (Laser Interferometer Space Antenna). Alle Starts sollen bis zum Jahr 2016 erfolgt sein [Bleeker 1995 und Burke 1996].

[1] **E**uropean **S**pace **A**gency

2.1 XMM

Wie der Name **X**-Ray **M**ulti-Mirror **M**ission schon sagt, ist XMM ein Röntgensatellit, dessen Start mit einer Ariane 5 im August 1999 vorgesehen ist. Eines seiner Hauptinstrumente ist das Kamerasystem EPIC (European Photon Imaging Camera), das in einem Energiebereich von 0,1 bis 15 keV neben spektroskopischen Messungen auch Abbildungen der Röntgenquellen erstellen soll. Zu diesem Zweck sind drei Wolterteleskope des Typs 1 an Bord, in deren Fokalebenen insgesamt 5 Instrumente arbeiten (eine Übersicht findet sich auf S. 5):

1. Die EPIC-pn-CCD[2]-Kamera MAXI (MPE[3]-AIT-X-ray-Imager) wird — bis auf die Data Handling Unit — vom IAAT (Event Analyser) und dem MPE (alles andere) gebaut. Sie befindet sich alleine im Fokus eines der drei Wolterteleskope und hat somit 100% der einfallenden Strahlung für Breitband-Spektroskopie und Abbildungen zur Verfügung.

2. Die beiden RGS' (Reflection Grating Spectrometer), deren Platz hinter den Wolterteleskopen der beiden verbleibenden Spiegelsysteme ist, werden als holländisch-amerikanisches Gemeinschaftsprojekt mit Schweizer Beteiligung entwickelt. Allerdings werden hier nur 40% bis 50% der einfallenden Photonen mittels Reflexionsgittern auf den Detektor abgelenkt, da sie sich die Teleskope jeweils mit einem weiteren Instrument teilen.

3. Die ebenfalls doppelt konzipierten EPIC-MOS-CCD-Kameras werden gemeinsam von französischen und britischen Instituten entwickelt. Sie verwenden den Teil der Strahlung, der nicht auf die RGS' fällt.

Außerdem gibt es noch einen zu den Wolterteleskopen gleichgerichteten sogenannten „optischen“ Monitor, der im optischen und UV-Bereich mißt, und innerhalb der EPIC-Gruppe einen Strahlungsmonitor.

Auf diese Weise können gleichzeitig Messungen der gleichen Quelle in unterschiedlichen Energiebereichen gemacht werden: Sowohl optische Abbildungen als auch Bilder mit mittlerer spektraler Auflösung im Röntgenbereich sind möglich, und die hochauflösenden Gitter-Spektrometer des RGS gestatten erstmals Aufnahmen von Röntgenspektren mit einer Auflösung von ca. 400 (bei 0,5 keV) in diesem Energiebereich.

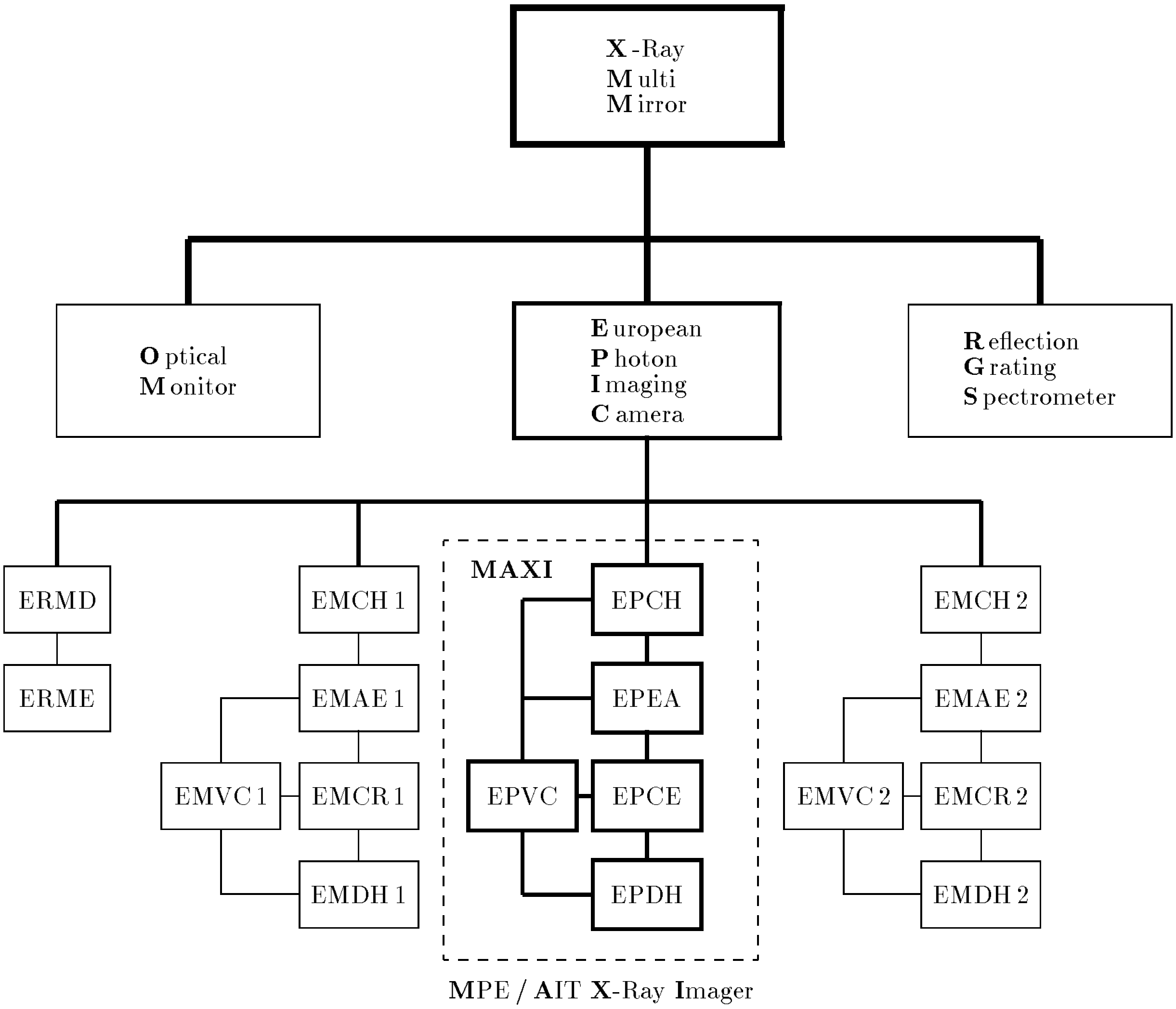
Für weitreichendere Informationen zu XMM soll auf Siegmund & Gummin (1996) verwiesen werden, aus dem auch Tabelle 2.1 mit einigen wichtigen Missionsdaten entnommen ist.

2.1.1 Die Wolterteleskope

Große Aufmerksamkeit wurde auf die Entwicklung und Fertigung dieser Hochpräzisions-Spiegelsysteme (mit einer Fokallänge von 7,5 m) gelegt, die die Hauptmasse des Satelliten ausmachen. Es handelt sich um jeweils 58 Spiegelschalen einer Dicke von ca 1 mm und Spiegelradien von 15,9 bis zu 35 cm, die mit einem Abstand von $\geq$ 3 mm geschachtelt

[2] **C**harge **C**oupled **D**evice; ein abbildender Halbleiter-Detektor, der seine Daten als Elektronen in Pixeln speichert. Eine kurze Erklärung findet sich in Abschnitt 3.1 und auf Seite 26.

[3] **M**ax **P**lanck Institut für **E**xtraterrestrische Physik in Garching bei München



E PIC **R** adiation **M** onitor	**E** PIC **M** OS # 1	**E** PIC **P** N	**E** PIC **M** OS # 2
D etector	**C** amera **H** ead	**C** amera **H** ead	**C** amera **H** ead
E lectronics	**A** nalog **E** lectronics	**E** vent **A** nalyser	**A** nalog **E** lectronics
	C ontrol & **R** ecognition	**C** ontrol **E** lectronics	**C** ontrol & **R** ecognition
	D ata **H** andling	**D** ata **H** andling	**D** ata **H** andling
	V oltage **C** onverter	**V** oltage **C** onverter	**V** oltage **C** onverter

Abbildung 2.1: *Die Instrumente auf XMM in ihrer logisch-hierarchischen Anordnung*

Telescope		**Pointing**	specified	goal
Focal length	7.5 m	Solar aspect angle pitch/roll	20°	20°
Field of view (half cone)	15 arcmin	Absolute pointing error	60 "	
Number of mirror modules	3	Absolute pointing drift (16 h)	45 "	2 "
Mirror shells per module	58	Absolute pointing drift (1 h)	5 "	
Carrier/effective surface	Nickel/Gold	Relative pointing error (2 min)	5 "	2 "
Reflective area at 1.5/8 keV	4425/1740 cm^2	Relative pointing error (10 sec)		0.25 "
Mirror shell thickness	0.47 – 1.07 mm	Absolute measurement accuracy	10 "	1 "
Mass		**Power**		
Total spacecraft	3900 kg	Solar array BOL	1945 W	
Experiments	600 kg	EOL	1600 W	
Mirror modules	1420 kg	Batteries	18 Ah (each)	
Hydrazine	485 kg	Experiments	360 W	
Thermal control		**Data management**		
Mirror module heating	100 W (2x)	Telemetry rate (S-band)	70 kbps	
Mirror support platform	72 W	Experiment allocation	64 kbps	
Passive thermal control		Telecommand rate (S-band)	2 kbps	
with safeguard heaters		Transmitter power	5 W	
Orbit		**Operations**		
Period	47.8 hr	Launch date	August 1999	
Apogee	11400 km	Minimum observation altitude	40000 km	
Perigee	7000 km	Coverage	> 40 hr/orbit	
Inclination	70°	Launcher	Ariane 5	
Argument of perigee	300°	Nominal mission duration	2 years	
Right ascension of ascending node	43°	Extendet mission duration	10 years	
Maximum eclipse duration	1.3 hr			

Tabelle 2.1: *Wichtige Daten der XMM-Mission, wie sie in Lumb* et al. *1996 angegeben sind.*

werden. Sie werden von der Firma Media Lario (Italien) aus einer 2000 Å dicken Goldschicht auf monolithischem Nickel als Trägermaterial gefertigt und frei hängend montiert [Casteren 1996].

Sie liefern eine Sammelfläche von mehreren 1000 cm^2 (4500 cm^2 bei 2 keV und 1750 bei 8 keV), die größte Sammelfläche, die zum Jahrtausendwechsel in diesem Energiebereich fliegen wird [Lumb *et al.* 1996].

2.1.2 Der optische Monitor OM

Das Herzstück des in England entwickelten optischen Monitors ist ein 30 cm Ritchey-Cretien-Teleskop und ein CCD-Detektor-System. Es stehen mehrere Filter zur Verfügung, so daß es möglich ist, gleichzeitig mit den Röntgenmessungen verschiedene Abbildungen desselben Objekts in unterschiedlichen Wellenlängenbereichen aufzunehmen — vom Optischen bis ins Ultraviolette. Dies ist einer der großen Vorteile von XMM: Für die eindeutige Interpretation von Röntgendaten ist es manchmal unerläßlich, auch Daten aus niedrigeren Energiebereichen zu haben. Diese gleichzeitige Beobachtung ist bisher aufgrund von Wetter- oder Terminplan-Problemen mit erdgebundenen Teleskopen selten möglich gewesen.

2.1.3 Das Gitterspektrometer RGS

Das RGS ermöglicht die Messungen mit der höchsten spektralen Auflösung auf XMM. Um gleichzeitig orts- wie energie-auflösend zu messen, wurde eine Rowland-Kreis-Anordnung (Durchmesser 6,7 m) gewählt: Als Reflektionsgitter wurden 202 sogenannte „Blaze-Gitter“ aus SiC genommen, während der Detektor aus einem CCD-Array besteht. Damit die Rowland-Kreis-Bedingungen erfüllt werden, sind die einzelnen CCDs (Pixelgröße 27 μm Kantenlänge) auf unterschiedlichen Ebenen angeordnet. Die damit erreichte Auflösung in erster Ordnung beträgt 400 bei 0,5 keV [Brinkman *et al.* 1996].

2.1.4 Das EPIC-System

Wie die Grafik auf S. 5 bereits veranschaulicht, besteht die „European Photon Imaging Camera“ aus der pn-CCD-Kamera, den beiden (gleichen) MOS[4]-CCD-Kameras und dem Strahlungsmonitor.

Vor jeder Detektoröffnung befindet sich ein Filterrad mit 6 Positionen. Eine davon ist völlig offen, 4 sind mit Filtern unterschiedlicher Dicke bestückt, die es erlauben niederenergetische Strahlung auszufiltern, und in einer Position werden die Detektoröffnungen mittels einer recht dicken Aluminiumschicht geschlossen.

Der **Radiation Monitor** schlägt Alarm, wenn die Teilchenstrahlung zu hoch für die empfindlichen CCDs wird, und kann damit das eben beschriebene Schließen der Detektoröffnungen zum Schutz bewirken.

Die **MOS-CCD-Kameras** beruhen auf herkömmlichen CCDs, wie sie auch für Video-Kameras verwendet werden. Ihr Vorteil ist eine große Anzahl kleiner Pixel und damit eine hohe Ortsauflösung, die allerdings auch Nachteile mit sich bringt (vergl. Tabelle 2.2): Zum einen wird dadurch eine hohe Auslesezeit benötigt, und zum anderen muß jeder Detektor in 7 Einzel-CCDs aufgeteilt werden, die sich in unterschiedlichen Ebenen befinden (siehe Abbildung 2.2 (b)).

2.1.4.1 Das EPIC-pn-CCD-Kamera-System EPCS

Wie bereits aus der Grafik auf S. 5 ersichtlich ist, besteht MAXI ähnlich wie die MOS-Kameras aus 5 Untersystemen, die nun im Einzelnen kurz vorgestellt werden sollen.

Der Kamerakopf EPCH

hat als Herzstück den CCD-Detektor, der im Gegensatz zum MOS-Detektor eine völlige Neuentwicklung des MPE-Halbleiterlabors bei München ist. Im Vergleich mit den MOS-CCDs haben die pn-CCDs zwar eine geringfügig schlechtere Energieauflösung; der große Vorteil der pn-CCDs liegt jedoch in der eindeutig kürzeren Auslesezeit und der deutlich besseren Nachweis-Wahrscheinlichkeit der Photonen („Quantum-Efficiency“) im gesamten relevanten Energiebereich (vergleiche Tabelle 2.2). Die Ursache für diese bessere Quantum-Efficiency liegt in der vollständigen Depletierbarkeit des pn-Übergangs [Kuster 1997]; diejenige für die kürzere Auslesezeit in der Pixelgröße: Die Pixel der pn-CCD-Kamera sind mit einer Kantenlänge von 150 μm größer als die der MOS-Kamera. Die Ortsauflösung wird dadurch allerdings kaum beeinträchtigt, da dies einer Winkelauflösung von 4,1”entspricht,

[4] **M**etall-**O**xid-**S**chicht

Parameter	MOS-CDD	pn-CCD
CCD Format	7 CCDs	12 CCDs
Pixel (each CCD)	600 x 600	64 x 200
Pixelsize	40 x 40 μm	150 x 150 μm
(arcsec)	1.1 x 1.1	4.1 x 4.1
Field of view	33.5 x 33.5	27.5 x 27.5
(arc min)	(max)	
Full field time	3	0.0558 (without
resolution (sec)		integration)
Timing Resolution	1 ms	40 μs
Energy	135 eV @ 6.4 keV	140 eV @ 6.4 keV
Resolution	55 eV @ 1 keV	60 eV @ 1 keV
Quantum	87 % @ 6.4 keV	99 % @ 6.4 keV
Efficiency	65 % @ 8 keV	97 % @ 8 keV
	45 % @ 0.5 keV	88 % @ 0.5 keV
Operating Temperature	-80° C	-128° C

Tabelle 2.2: *Einige Charakteristika im Vergleich von MOS- und pn-CCD-Kamera wie sie tatsächlich gemessen wurden [Kuster 1997].*

die Spiegelmodule selber aber eine Winkelauflösung von ca 20” (Half Energy Width HEW) haben.

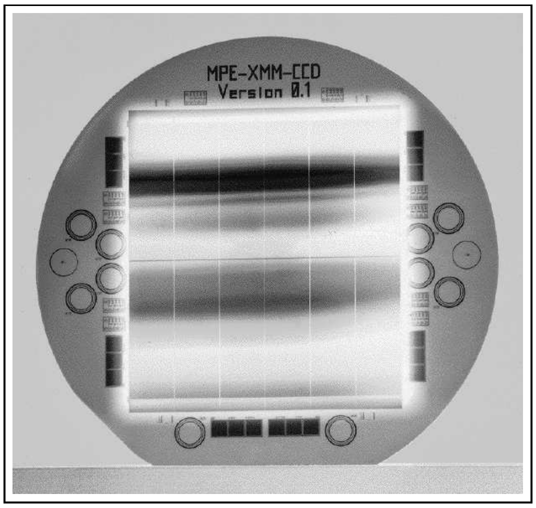
In der Fokalebene befindet sich im Wesentlichen eine einzige Silizium-Scheibe (Wafer), in die die 12 pn-CCDs à 200 x 64 Pixel eingearbeitet wurden (siehe Abbildung 2.2 (a)). Aus Betriebssicherheitsgründen sind die CCDs auf 4 Quadranten verteilt (vergleiche Abbildung 2.3) und einzeln ansteuerbar; auf diese Weise muß im Falle eines Ausfalls, sowohl einzelner CCDs als auch ganzer Quadranten, nicht das ganze System abgeschaltet werden.

Der Kamerakopf und vor allem die Funktionsweise der pn-CCDs ist in der Diplomarbeit von Kuster (1997) genauer erklärt; die CMOS[5]-Verstärker-Einheit (jeweils ein CAMEX[6] und sein Steuer-Chip TIMEX) wird in Zusammenhang mit den Voraussetzungen für die Entwicklung des Quadrantensequenzer in Abschnitt 3.2.1 näher erläutert.

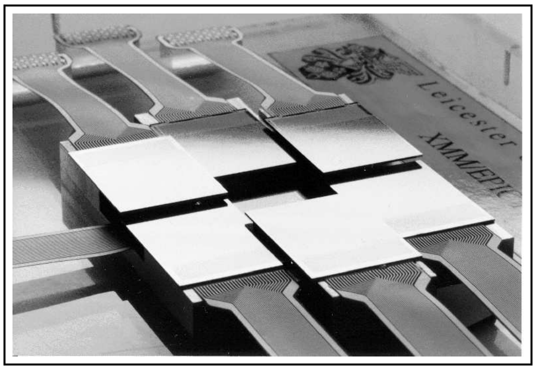
Der „Event Analyser“ EPEA

übernimmt einerseits die Digitalisierung, die Filterung und das Übertragen der gemessenen Daten an die auf Seite 10 vorgestellten „Data Handling Unit„ EPDH. Andererseits liefert er die vollständige Ablaufsteuerung des gesamten pn-CCD-Kamera-Systems. Auch dieser Teil soll in Abschnitt 3.3 genauer erklärt werden.

[5] **C**omplementary **MOS**, bestehend aus komplementär aufgebauten p- und n-MOS Schaltungsteilen
[6] **CMOS** **A**mplifier and **M**ultiplex**ex**er Chip, ursprünglich **CMOS** **A**nalog **M**ultiplex**ex**ing readout chip



(a) *Die Vorderseite der 12 pn-CCDs. Für die Messungen wird allerdings die Rückseite beleuchtet.*



(b) *Der MOS-CCD-Array aus 7 Einzel-CCDs.*

Abbildung 2.2: *Die verschiedenen Detektor-Felder der EPIC-Kameras. Deutlich zu erkennen ist die unterschiedliche Struktur: Die MOS-Kamera (b) hat 7 CCDs auf verschiedenen Ebenen, während die pn-Kamera (a) 12 CCDs hat, die monolithisch auf einem einzigen Silizium-Wafer gefertigt werden.*

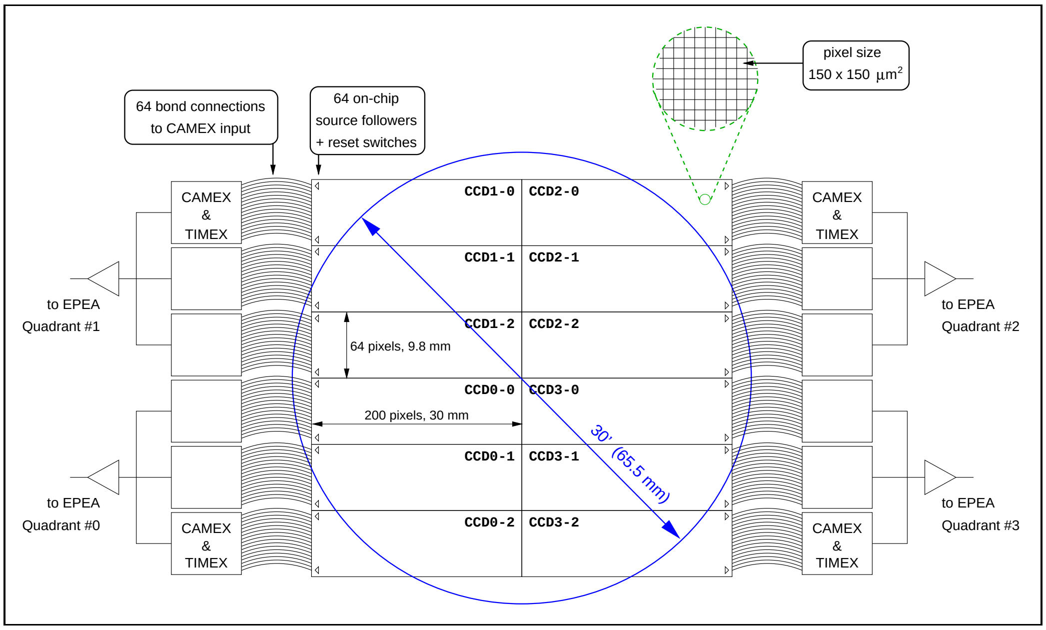


Abbildung 2.3: *Die Fokalebene der XMM-pn-CCD-Kamera mit ihren auf 4 Quadranten aufgeteilten 12 CCDs. Neben jedem CCD ist die zugehörige, über Bonddrähte verbundene, Verstärker-Einheit zu erkennen.*

Die „Control-Electronics" EPCE

hat die Kontrolle über alle weiteren Einheiten: Sie schaltet und kontrolliert die verschiedenen Spannungen, die von der **„Low Voltage Control Unit" EPVC** außer für die

EPDH für das gesamte EPCS zur Verfügung gestellt werden, steuert das Filterrad und überwacht die Temperatur-Stabilisierung des Kamerakopfes. Zusammen mit dem „Low Bitrate Interface“ LBR bildet sie den „Mittelsmann“ für Kommandodaten in der einen Richtung und Housekeeping-Daten in der anderen Richtung zwischen EPEA und EPDH, die als nächstes beschrieben werden soll.

Die „Data-Handling“-Einheit EPDH

bildet die Schnittstelle zwischen dem „On Board Data Handling“-System OBDH des Satelliten und dem Kamera-System. Sie verwaltet die Kommandos vom OBDH an das EPCS, koordiniert die Daten und die Housekeeping-Informationen und reiht beides in den Telemetriestrom des Satelliten zur Erde ein.

2.2 ABRIXAS

Wie in der Einleitung bereits erwähnt, handelt es sich bei ABRIXAS um ein Kleinsatelliten-Projekt der DARA[7], dessen Start für Mitte 1999 mit einer russischen COSMOS-Rakete vorgesehen ist.

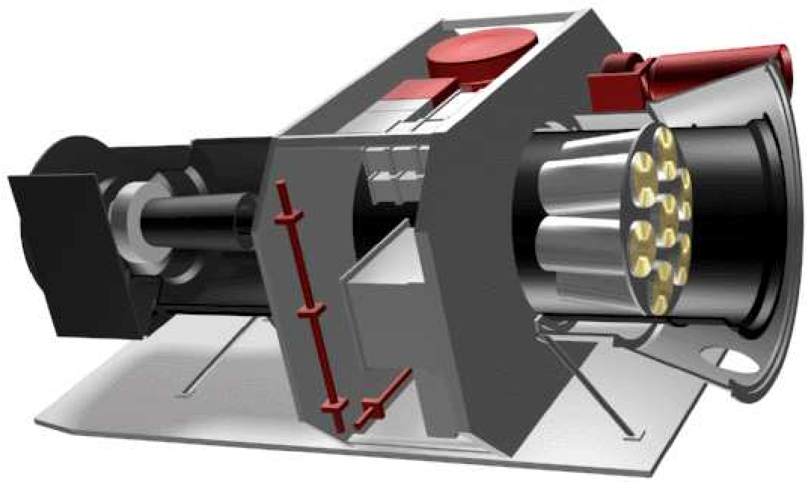


Abbildung 2.4: *Der ABRIXAS-Satellit. Rechts sind die 7 Wolterteleskope zu erkennen, links befindet sich der Detektor.*

Sein Name „**A** **Br**oad Band **I**maging **X**-Ray **A**ll-Sky **S**urvey“ ist auch seine wissenschaftliche Zielsetzung: Die Erweiterung der erfolgreichen ROSAT-Himmelsdurchmusterung zu höheren Energien hin.

Aus finanziellen, aber auch aus (entwicklungs-) zeitlichen Gründen enthält ABRIXAS als Detektorsystem dasjenige der XMM-pn-CCD-Kamera MAXI (siehe vorhergehenden Abschnitt 2.1.4). Die beteiligten wissenschaftlichen Institute sind das Astronophysikalische Institut Potsdam (AIP), das MPE und das IAAT. Der Satellit selber wird von OHB in Bremen gebaut, und die Spiegelsysteme — wie bei XMM mehrere Woltertelskope des Typs I — kommen von der Firma Zeiss, Oberkochen.

[7]**D**eutsche **A**gentur für **R**aumfahrt-**A**angelegenheiten

Fokal-Länge	1,6 m
Spiegel-Länge	30 cm
größter Spiegel-Durchmesser	16,3 cm
kleinster Spiegel-Durchmesser	7,6 cm
Spiegel-Anzahl	27 Stück
Spiegel-Abstand	$\geq$ 1 mm
Gesichtsfeld	7 mal 40'
Energieband	0,5 bis 10 keV
ΔE bei 6 keV	160 eV
Hersteller	Zeiss
Winkelauflösung der Spiegel	$\leq$ 30"
Winkelauflösung der Pixel	ca 4"

Tabelle 2.3: *Einige ABRIXAS-Parameter*

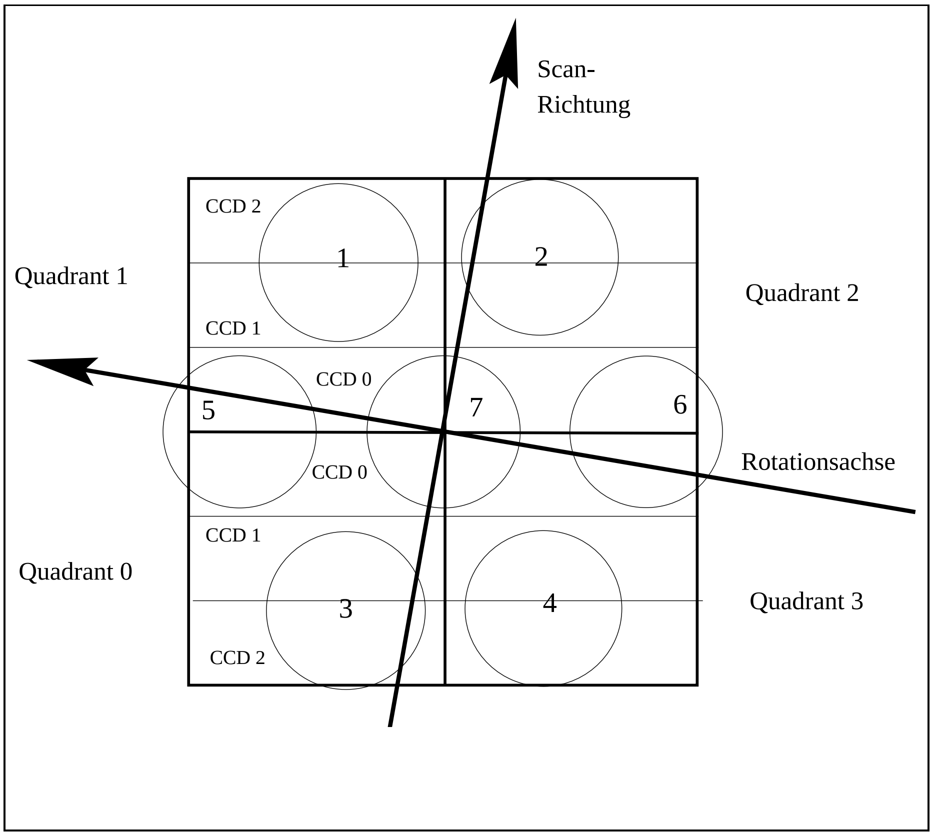


Abbildung 2.5: *Die Fokalebene von ABRIXAS. Zu erkennen ist sowohl die Verteilung der Gesichtsfelder der 7 Spiegelsysteme auf den 12 CCDs als auch die Himmelsbewegung darüber.*

Um den Anforderungen an einen Kleinsatelliten genügen zu können, wurden das XMM-Spiegelsystem von einer Fokallänge von 7,5 m auf 1,6 m verkleinert. Dabei wurde darauf geachtet, daß das Verhältnis von Fokallänge zum Durchmesser konstant gehalten wurde, da nur so die Glanzwinkel und damit der Meßbereich im Energieband erhalten bleiben. Außerdem ist es vorteilhaft, einen möglichst großen Himmelsbereich gleichzeitig beobachten zu können, weshalb 7 Spiegelsysteme verwendet werden, deren Bildfelder sich wie in Abbildung 2.5 auf die 12 CCDs verteilen. Sonstige interessante ABRIXAS-Parameter sind in Tabelle 2.3 zusammengefaßt.

Kapitel 3

Anforderungen an den Quadrantensequenzer

Der Begriff „Sequenzer“ ist der elektrotechnische Ausdruck für „Steuerwerk“. Der „Quadrantensequenzer“, kurz „QSEQ“ genannt, ist demzufolge die elektronische Schaltung, die für die Steuerung eines Quadranten zuständig ist. Dabei ist mit „Quadrant“ ein Viertel der Detektorfläche gemeint, also 3 CCDs.

Der QSEQ ist ein Teil der in Abschnitt 2.1.4.1 erwähnten EPEA. Seine Aufgaben und das Zusammenspiel mit dem Rest des Systems sollen in diesem Kapitel erläutert werden. Dazu gehört die allgemeine Beschreibung der Betriebsmodi der EPIC-pn-CCD-Kamera ebenso, wie die konkreten Anforderungen des sogenannten Operationswerkes, also des Kamerakopfes mit seinen analogen Verstärkern (siehe Abschnitt 3.2), des ADC[1], der die analogen Daten digitalisiert, und des Common-Mode-Filters (ab Seite 34), der neben dem Hinzufügen der Ortsinformation die Rohdaten einer ersten Bearbeitung und Auswahl unterzieht — letzteres vor allem aus Gründen der begrenzten Telemetrie-Kapazität von $< 40\,\mathrm{kbit/sec}$.

3.1 Die verschiedenen Betriebsmodi

Um sehr unterschiedliche Quellen untersuchen zu können, kann die pn-CCD-Kamera in verschiedenen Modi betrieben werden: Sowohl die Auslesegeschwindigkeit als auch Detektorbereiche („Fenster“) können gewählt werden. Beides ist eng mit dem Aufbau der Detektorfläche und dem Auslesevorgang der Daten verbunden:

Wie bereits mehrfach erwähnt ist die Fokalebene in 4 Quadranten eingeteilt, die von 0 bis 3 durchnummeriert sind. Jeder Quadrant enthält die CCDs 0 bis 2, die jeweils aus 200 Zeilen (0 bis 199) à 64 Pixel (Spalten 0 bis 63) aufgebaut sind. Die einzelnen CCDs werden zeilenweise ausgelesen, indem die Ladungen in Spaltenrichtung nach außen geschoben werden. Die Nummerierung sowohl der Zeilen als auch der Pixel ergibt sich aus der Auslesereihenfolge, immer bei 0 beginnend (die innerste Zeile hat also die Nummer 199).

Die nächsten Seiten werden verdeutlichen, daß die Wahl der Auslesegeschwindigkeit sich aus der Wahl des Schiebemodus zusammen mit der Aktivierung der Quadranten ergibt; in Bezug auf die Wahl der Detektorfläche ist es leicht verständlich, daß die Fenster immer nur so gesetzt werden können, daß keine Zeile eines CCDs abgeschnitten wird. Dies liegt an der parallelen Übernahme aller 64 Pixel einer Zeile in den CAMEX (Abschnitt 3.2.1.1).

Außerdem soll darauf hingewiesen werden, daß

1. immer nur ein CCD pro Quadrant aktiv sein kann,

[1]**A**nalog to **D**igital **C**onverter

und

2. der variable Teil der Integrationszeit nur zwischen der Abarbeitung der einzelnen Quadranten eingestellt werden kann, da der Quadrantensequenzer in seiner ursprünglichen Version zwischen „seinen“ CCDs keine Pausen macht.

3.1.1 Der Fullframe-Mode

Dieser Modus ist der Standard-Betrieb für XMM und der alleinige Betriebsmodus für ABRIXAS. In ihm werden alle CCDs nacheinander vollständig ausgelesen und verarbeitet. Durch die relativ lange Auslesezeit kann dies allerdings zu Problemen mit sogenannten „Out-of-Time-Events“ führen: Dies sind Photonen, die während eines Auslesezyklus detektiert werden, durch den Schiebevorgang allerdings in Zeilen mit einer „falschen“ Nummer fallen und die die Bildstruktur dadurch massiv verwischen können. Dieser Effekt kann vermindert werden, indem das Verhältnis der Anzahl Photonen, die während der „inaktiven“ Zeit erkannt werden, zu der Anzahl Photonen, die während der Auslesezeit einfallen, verbessert wird. Dies entspricht einer Verbesserung von Daten-Aufnahme- zu Daten-Auslesezeit, weshalb die Integrationszeit bis dasselbe CCD wieder ausgelesen wird bei zyklischer Abarbeitung der Quadranten variiert werden kann. Dies erlaubt außerdem, auch die Helligkeit der Quelle bei der Beobachtung zu berücksichtigen: Je heller die Quelle ist, desto kürzer sollte die Integrationszeit gewählt werden. Abbildung 3.1 soll diesen Modus etwas verdeutlichen.

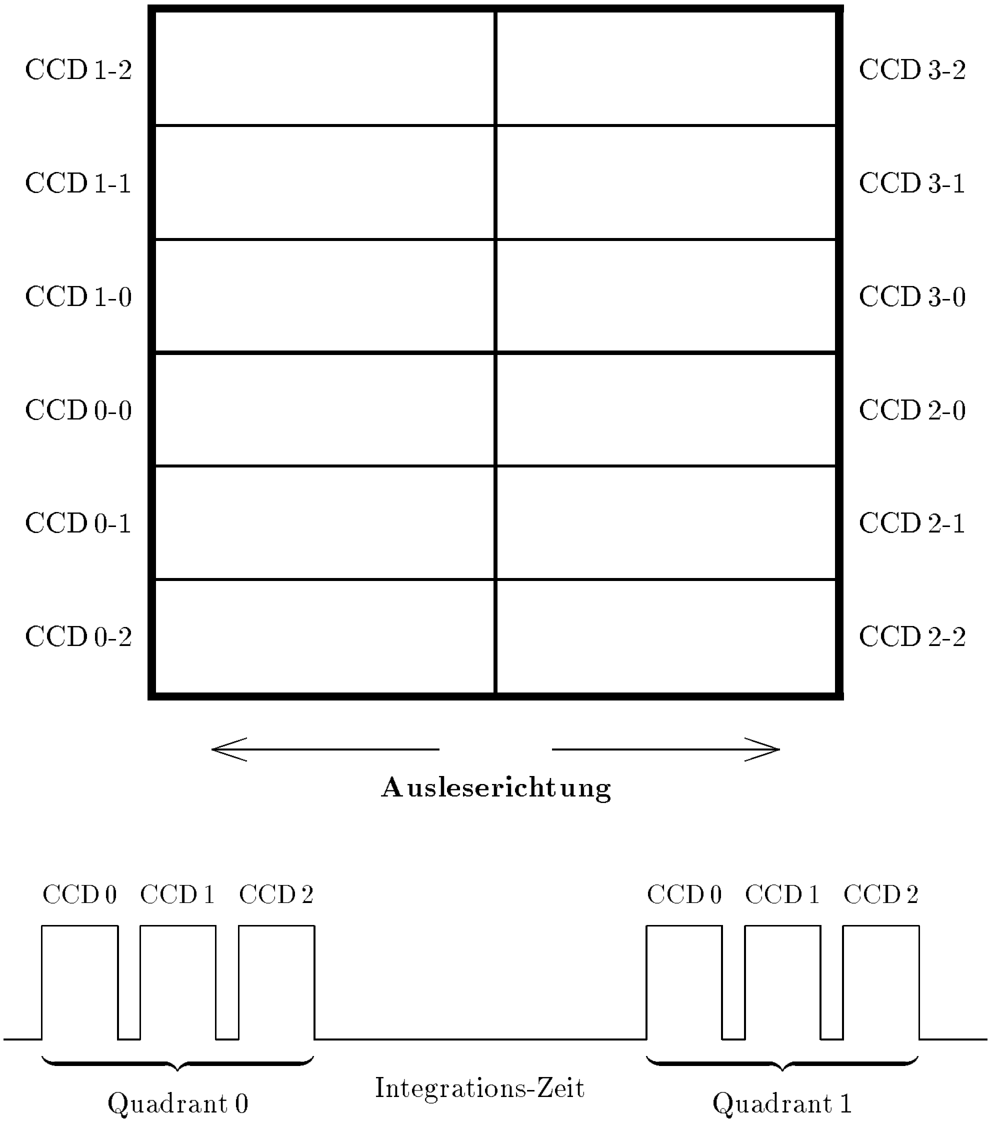


Abbildung 3.1: *Im Fullframe-Mode genutzte Detektorfläche und CCD-Reihenfolge. In der CCD-Abfolge sind hier nur 2 Quadranten angedeutet; natürlich kommen danach auch Quadrant 2 und 3.*

3.1.2 Der Large-Window-Mode

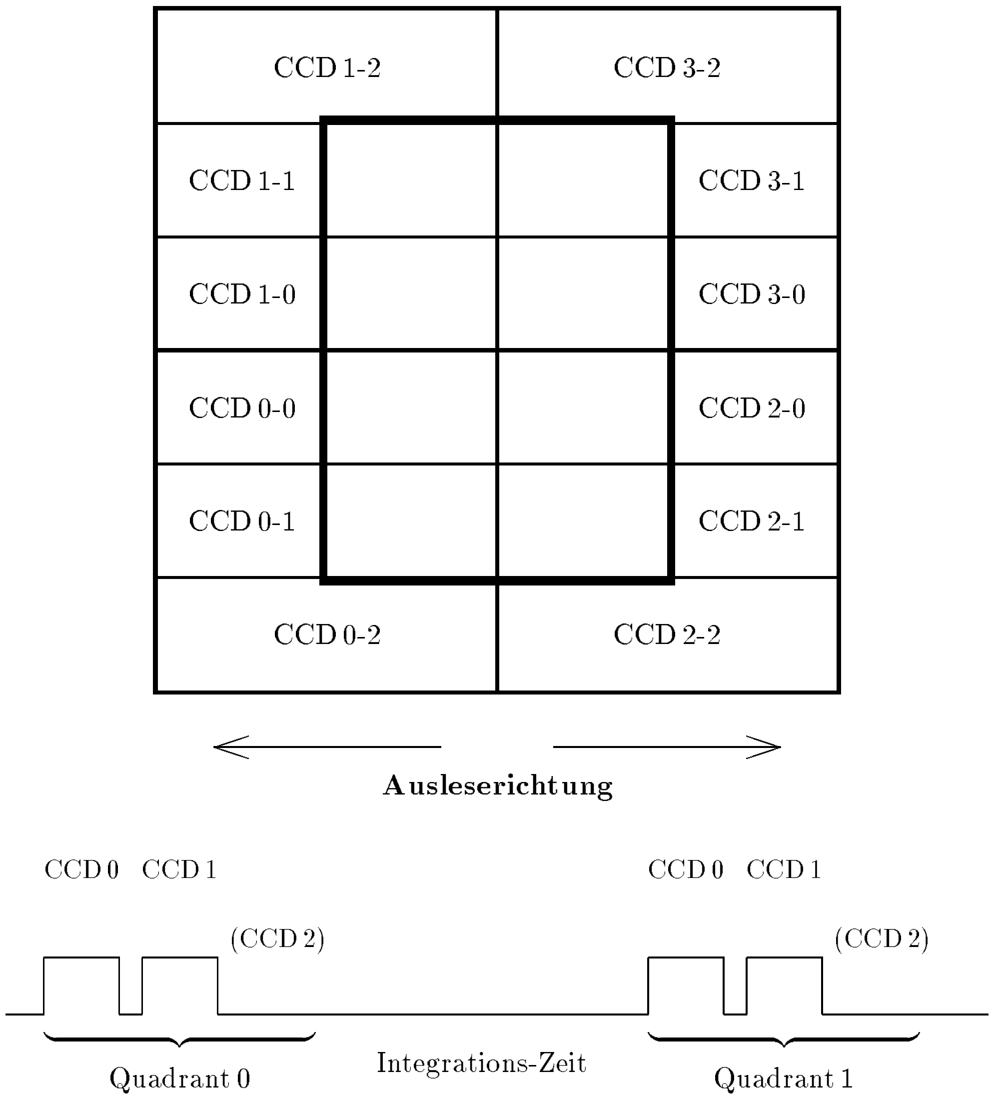


Abbildung 3.2: *Im Large-Window-Mode genutzte Detektorfläche und CCD-Reihenfolge. Man beachte, daß CCD 2 bei dieser Fensterwahl im Ausleseturnus „übergangen" wird, da es nicht benötigt wird. Bei einer anderen Fenstergröße kann es notwendig werden, CCD 2 mit zu berücksichtigen. Auch hier werden die Quadranten 2 und 3 selbstverständlich im Anschluß an Quadrant 1 ausgelesen.*

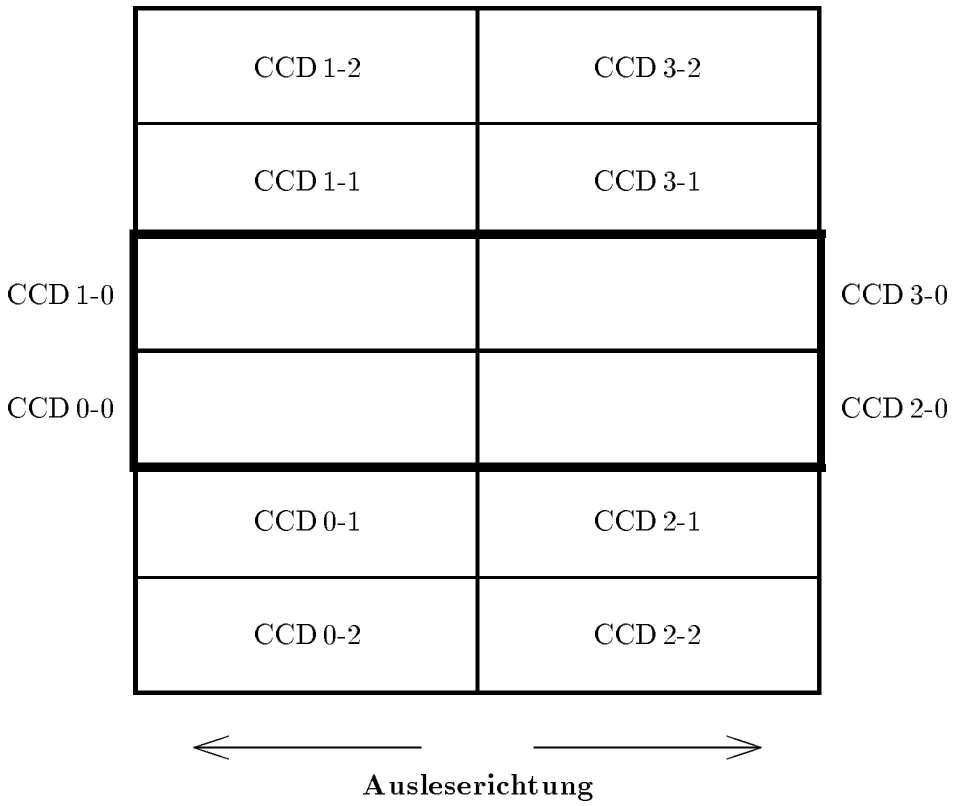
Soll eine Quelle möglichst ohne diese „Out-of-Time-Events" beobachtet werden, z.B. um die Struktur einer ausgedehnten Quelle zu untersuchen, so kann der Large-Window-Mode eingeschaltet werden. Voraussetzung dafür ist, daß sie sich im Zentrum des Detektors befindet, innerhalb eines CCDs nicht mehr als die Hälfte der Fläche einnimmt, und daß sich in der verbleibenden Detektorfläche nur deutlich schwächere Quellen befinden. Sind diese Voraussetzungen erfüllt, so kann die äußere Hälfte jedes CCDs als Zwischenspeicher genutzt werden: Mindestens 100 Zeilen werden schnell geschoben (und ihr Inhalt weggeworfen), wodurch die inneren 100 Zeilen nach außen „in Sicherheit“ gebracht werden; erst danach werden diese 100 Zeilen normal ausgelesen und verarbeitet.

Als Fenster kommt somit maximal die innere Hälfte jedes CCDs in Frage. Allerdings kann es auch kleiner sein, indem mehr als 100 Zeilen weggeworfen werden, oder nicht alle CCDs mit einbezogen werden. Eine sinnvolle Fenstergröße ist die Wahl der inneren 100 Zeilen der inneren 2 CCDs, also die halben CCDs 0 und 1. Dies ist in Abbildung 3.2 dargestellt.

3.1.3 Der Timing-Mode

Um sehr helle Quellen kontinuierlich beobachten zu können, muß ein CCD kontinuierlich mit einer schnelleren Auslesezeit abgearbeitet werden. Andernfalls würden sogenannte „Pile-ups“ die Energieauflösung deutlich verschlechtern: Fallen mehrere Photonen in dasselbe Pixel, so kann nur noch ihre gemeinsame Energie, nicht jedoch die Energie der einzelnen Photonen ermittelt werden. Daher sollen im Timing-Mode immer 10 Zeilen integriert und gemeinsam verarbeitet werden, was der ungefähren Ausdehnung der Punktbild-Übertragungs-Funktion („Point Spread Function“) der Teleskope entspricht. Trotzdem ist ein CCD-Rahmen im Timing-Mode mit einer Größe von 200 dieser Makrozeilen definiert, was dazu führt, daß die Abbildung gestaucht wird; 2000 physikalische CCD-Zeilen werden auf 200 Daten-Zeilen abgebildet. Effektiv ist ein Pixel damit 1x10 „Pixel-Einheiten“ groß.

Ein Nebeneffekt dieses schnelleren Auslesens ist eine bessere Zeitauflösung. Nachdem in diesem Betriebsmodus außerdem ein einziges CCD pro Quadrant kontinuierlich arbeitet, eignen sich die in diesem Betriebsmodus gemessenen Daten zur Zeitanalyse sich schnell ändernder Quellen. Eine Integrationszeit macht hier keinen Sinn; es wird ja nur ein CCD pro Quadrant angeschaltet, dafür werden allerdings alle Quadranten parallel abgearbeitet. Das aktive CCD kann zwar innerhalb eines Quadranten beliebig ausgewählt werden, in der Regel werden jedoch die 4 zentralen CCDs ein gemeinsames Fenster bilden (siehe Abbildung 3.3).



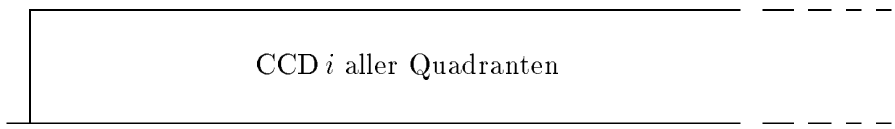


Abbildung 3.3: *Im Timing-Mode genutzte Detektorfläche und CCD-Reihenfolge. Man beachte, daß alle 4 inneren CCDs gleichzeitig ausgelesen werden (je eines pro Quadrant).*

3.1.4 Der Burst-Mode

Werden Quellen mit einer Helligkeit von $\geq 100\,$mCrab beobachtet, so ist eine kontinuierliche Datenaufnahme nicht mehr möglich und es muß in den „Burst-Mode“ geschaltet werden. Dieser Betriebs-Modus erlaubt die maximal mögliche Zeitauflösung: Die Schiebedauer einer CCD-Zeile. Allerdings geht dabei jegliche Ortsinformation in Schieberichtung verloren. Wie im Timing-Mode können auch hier bis zu vier CCDs (eines pro Quadrant) parallel ausgelesen werden.

Es wird davon ausgegangen, daß im Detektorzentrum eine helle bzw. stark variable Quelle pointiert ist; die anderen Quellen im Gesichtsfeld sind damit verglichen schwach. Von

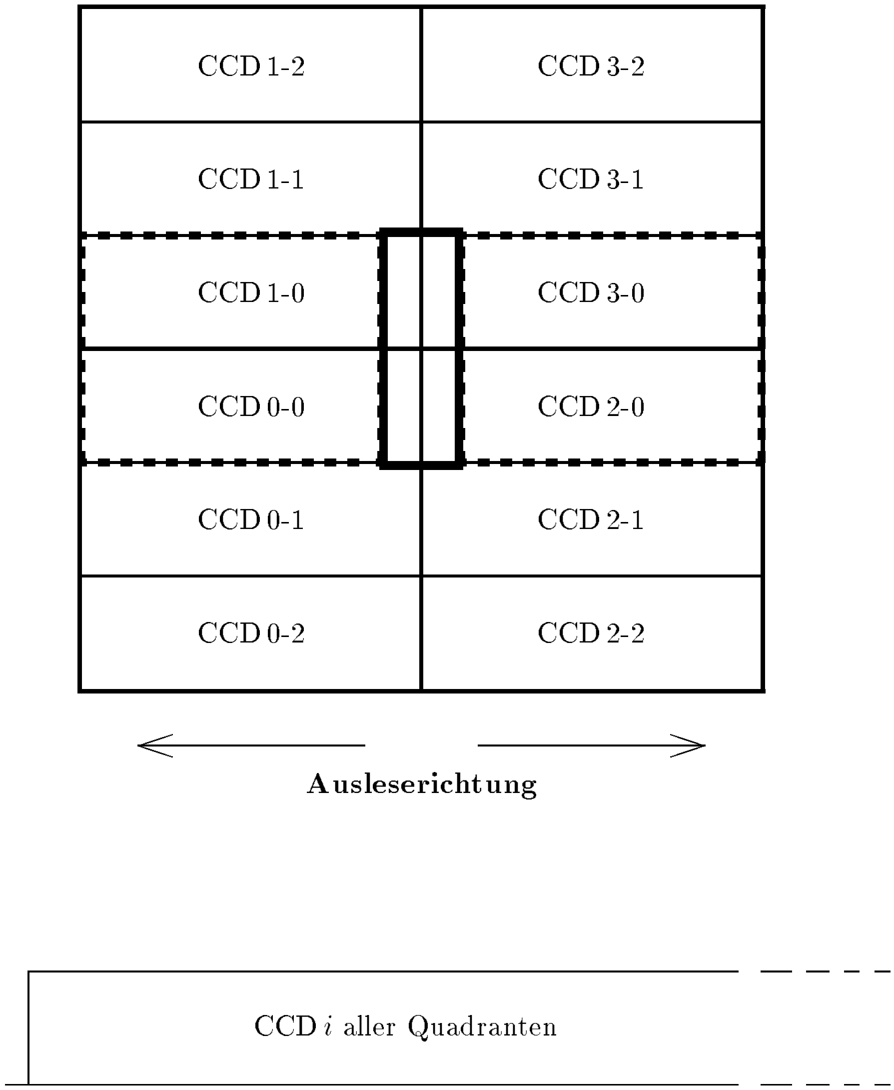


Abbildung 3.4: *Im Burst-Mode genutzte Detektorfläche und CCD-Reihenfolge. Man beachte, daß wie im Timing-Mode alle 4 inneren CCDs gleichzeitig ausgelesen werden (je eines pro Quadrant), die eigentlich aktive Fläche jedoch nur die innersten Zeilen sind.*

einem aktiven CAMEX aus betrachtet liegt diese Quelle in den letzten Zeilen des CCDs; die anderen Zeilen sind also „leer“. Dies kann dazu verwendet werden, Daten zeilenweise in schneller Abfolge aufzunehmen, in diesen „freien“ Zeilen zu speichern und dann langsam auszulesen und zu verarbeiten. Während dieses langsamen — genauer gesagt normalen — Auslesens werden die inneren Zeilen jedoch überbelichtet, so daß keine vollen 200 Zeilen sinnvoll ausgelesen werden können (in Abbildung 3.4 sind exemplarisch 180 ausgelesene Zeilen dargestellt). Diese überbelichteten Zeilen werden jedoch beim nächsten schnellen Schieben automatisch weggeworfen.

3.1.5 Der Small-Window-Mode

Was die Zeitanalyse und die Bilderfassung betrifft, ist der Small-Window-Mode ein Zwischenmodus. Hier werden wie bei den auf Zeitanalyse optimierten Burst- und Timing-Modi alle Quadranten parallel ausgelesen. Innerhalb der Quadranten werden jedoch alle CCDs abgearbeitet. Dabei ist im inneren CCD 0 ein Fenster von $n \cdot 20$ Zeilen gesetzt, das im Wechsel mit CCD 1 und CCD 2 ausgelesen wird (vergleiche Abbildung 3.5). Dadurch hat man sowohl die Bilder fast der gesamten Detektorfläche, als auch die schnellen Auslesezyklen für die Zeit- (oder Struktur-) Analyse der zentralen Quelle.

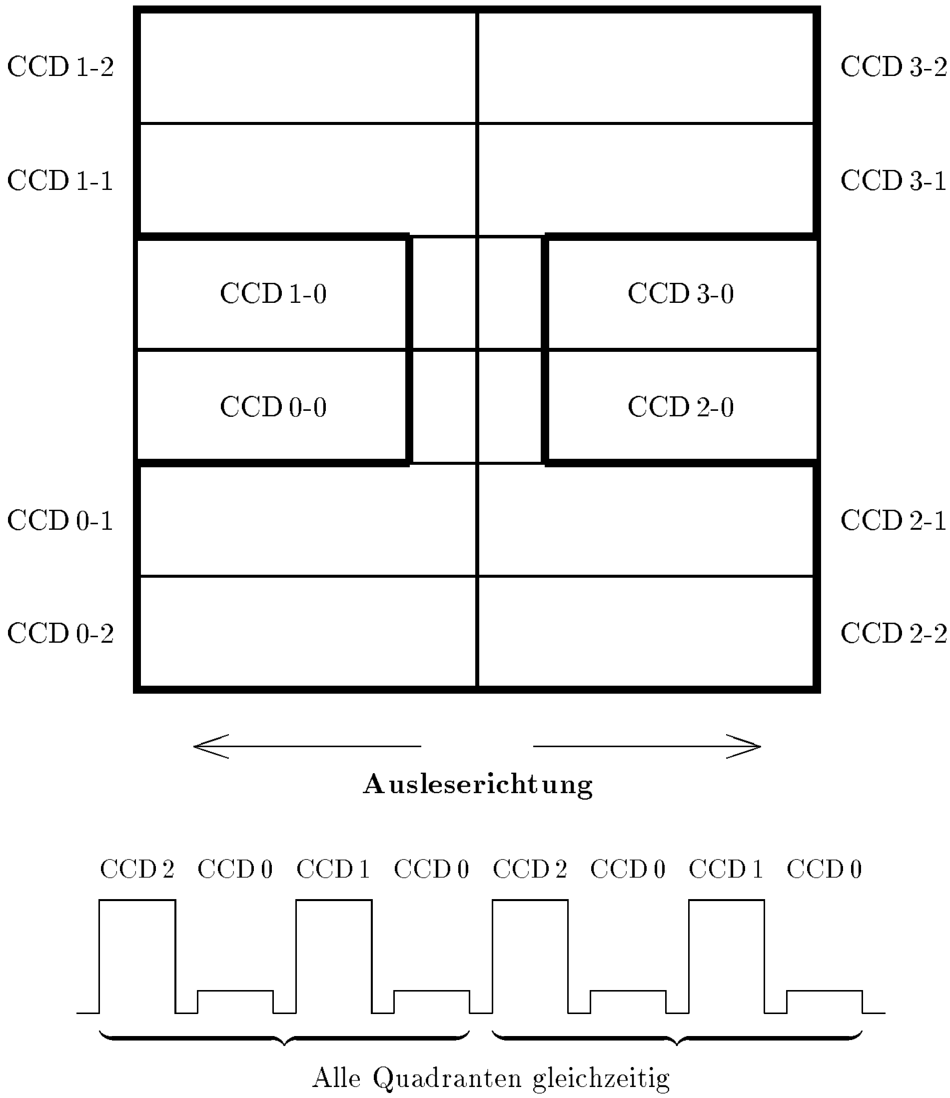


Abbildung 3.5: *Im Small-Window-Mode genutzte Detektorfläche und CCD-Reihenfolge. Man beachte, daß die angegebene Reihenfolge für jeweils einen Quadranten gilt, jedoch alle Quadranten parallel arbeiten.*

3.2 Die vom Kamerakopf verlangten Signale

Um die in Abschnitt 3.1 beschriebenen Betriebsmodi zu realisieren, würde es im Prinzip ausreichen, die im Detektor enthaltenen Daten Pixel für Pixel abzuholen und dabei darauf zu achten, daß sie aus dem gewünschten CCD stammen. Dies ist leider so einfach nicht realisierbar. Wenn ein Photon im Detektor absorbiert wurde, so sammeln sich die dadurch entstandenen Elektronen in den einzelnen Pixeln, genauer in einer von jeweils 3 Potentialmulden, wie es in den Diplomarbeiten von Kuster (1997) und Schuster (1994) erklärt ist. Allerdings liefern diese Elektronen beim Auslesen nur ein sehr schwaches Signal, das im Rauschen unterginge, würde es nicht sofort verstärkt werden: Zuerst noch auf dem Detektor-Wafer mittels eines sogenannte JFETs[2], danach in mehreren Stufen in einer extra Schaltung, dem CAMEX, der seinerseits vom TIMEX gesteuert wird (vergleiche Abschnitt 3.2.1). Auf der Fotographie der CCD-Platine (Abbildung 3.6) ist sowohl der CAMEX als auch der TIMEX neben dem CCD im Hintergrund erkennbar.

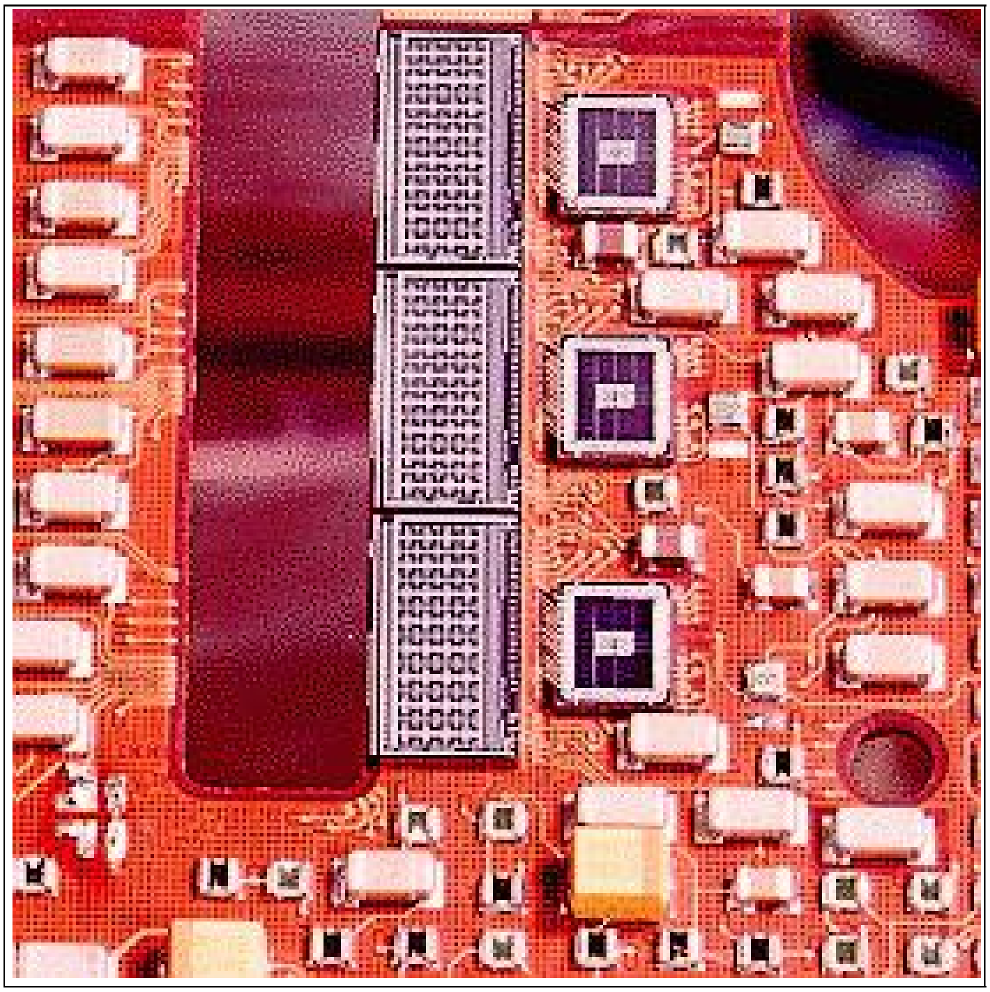


Abbildung 3.6: *Die unbeleuchtete Seite der CCD-Platine. Durch die linke Öffnung ist das CCD zu sehen; direkt daneben sind die 3 CAMEX-Bausteine und deren jeweiliger TIMEX angebracht.*

[2]FET = **F**eld-**E**ffekt-**T**ransistor, ein auf der MOS-Technologie aufbauender Transistor im Gegensatz zu den bipolaren (normalen) Transistoren, die wie die Dioden auf pn-Übergängen basieren.

Abgesehen von den Signalen, die bestimmen, aus welchem CCD die nächsten Daten kommen sollen (siehe Abschnitt 3.2.3), und denjenigen, die die Ladungen zu den Auslese-Anoden schieben (die Phi-Pulse aus Abschnitt 3.2.2), müssen also auch Steuerpulse für den TIMEX erzeugt werden. Außerdem werden noch ein Testsignal und ein Signal zur Verbesserung der Datenqualität benötigt (AMOS und FLUSH, Abschnitt 3.2.2).

3.2.1 Der CAMEX und sein TIMEX

Eine sehr ausführliche (und auch verständliche) Beschreibung sowohl des CAMEX als auch des TIMEX findet sich in der Diplomarbeit von Martini (1993). Trotzdem sollen beide Bausteine im folgenden kurz erläutert werden:

3.2.1.1 Der CAMEX

Der CAMEX übernimmt und verarbeitet alle 64 Pixel einer CCD-Zeile zur selben Zeit, weshalb die Daten im CCD immer so transportiert werden, daß ganze Zeilen parallel verschoben werden. Durch seinen Aufbau unternimmt er gleichzeitig eine Offset-Korrektur und eine 4-fache Abtastung des einzelnen Signals und stellt es danach zum seriellen Auslesen in den 64 Sample-and-Hold-Registern zur Verfügung. Die Dauer dieses Ablaufs bestimmt die Verarbeitungs-Dauer einer Zeile, wie im folgenden erläutert werden soll:

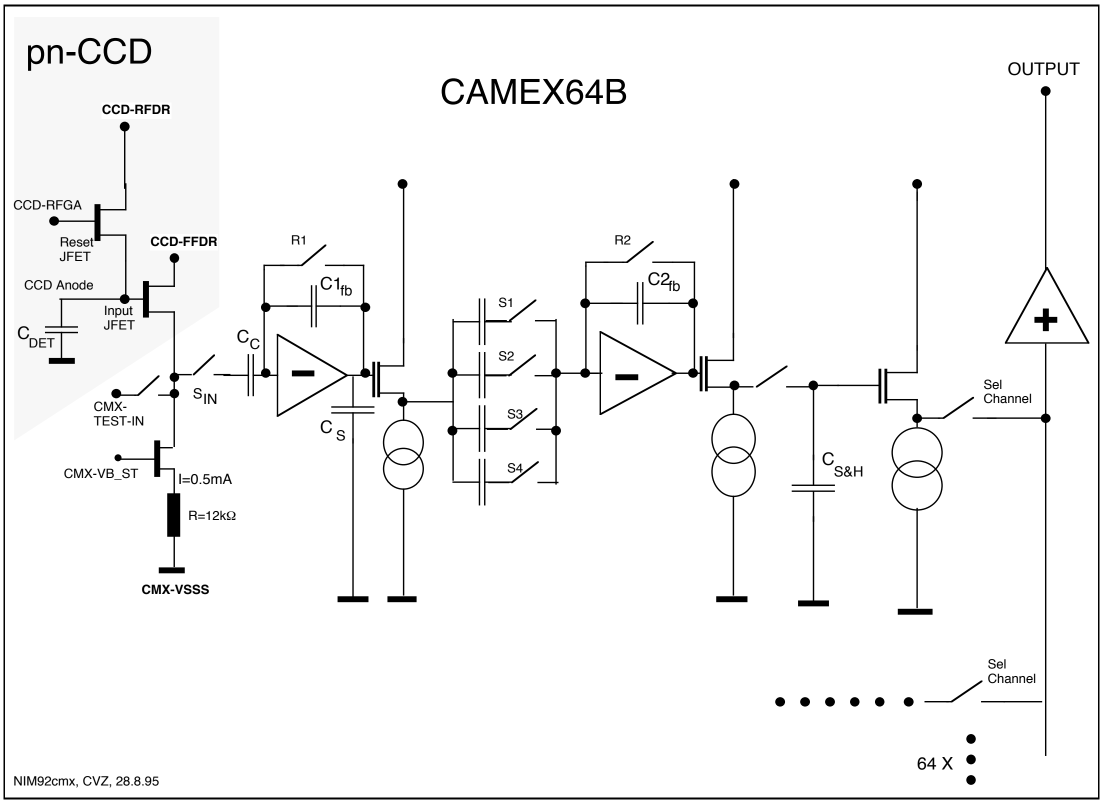


Abbildung 3.7: *Die schematische Darstellung eines CAMEX-Kanals. Der grau unterlegte Bereich oben links befindet sich noch auf dem CCD und ist mittels Bond-Drähten mit dem nicht unterlegten CAMEX-Bereich verbunden. (Das Bild wurde freundlicherweise vom Halbleiter-Labor in München zur Verfügung gestellt.)*

Zum besseren Verständnis sei zuerst auf Abbildung 3.7 verwiesen, wo alle Verarbeitungs-Stufen eines Pixels vom CCD (grau unterlegter Bereich) bis zur Übergabe an den ADC der EPEA (siehe Abschnitt 3.3) schematisch dargestellt sind. Beim zeilenweisen Schieben gelangen die Daten auf die CCD-Anode, die einen Kondensator gegen Masse darstellt. Dieser Kondensator ist als C_{DET} in der Zeichnung angedeutet. Die an ihm anliegende Spannung steuert den JFET, wodurch die Ladung noch auf dem CCD verstärkt wird, bevor sie als Eingangs-Spannung am CAMEX anliegt. Dieses Prinzip der ständig steigenden Spannung an der CCD-Anode — es werden ja mit jeder Zeile zusätzliche Ladungen auf den Kondensator geschoben — macht es erforderlich, daß für jedes neue Pixel zunächst die Basis-Spannung (Offset) bestimmt wird, bevor das neue Signal verarbeitet werden kann.

Zu diesem Zweck werden zunächst alle Verstärkerstufen des CAMEX zurückgesetzt, indem die Schalter R_1 und R_2 geschlossen werden (außer S_{in} sind alle anderen Schalter offen). Sodann wird der Schalter R_1 geöffnet und die Vorverstärkerstufe somit aktiviert.

Nun soll zuerst der Offset und danach das Signal der in der Zwischenzeit auf die CCD-Anode geschobenen neuen Zeile 4-fach abgetastet werden. Damit sich CAMEX und CCD nicht wechselseitig stören, müssen sie während des Schiebens entkoppelt sein (S_{in} offen). Da Offset und Signal unter den gleichen Bedingungen verarbeitet werden sollten, wird S_{in} als nächstes geöffnet. Nach der Zeit, die das Schieben benötigen würde, wird der Schalter wieder geschlossen, womit der Offset in den CAMEX übernommen wird. Werden sodann die 4 S_i nacheinander geschlossen und wieder geöffnet („Baseline-Sampling“), so stellt sich an den 4 zugehörigen Kondensatoren der Mittelwert der abgetasteten Spannung als Basiswert für die nächste Verstärkerstufe ein.

Damit sind die Voraussetzungen gegeben, daß S_{in} ein zweites Mal geöffnet und die nächste Zeile auf die CCD-Anode geschoben werden kann. Aus den eben erwähnten Gründen, darf S_{in} erst wieder geschlossen werden, nachdem die Aktivität an der Anode beendet ist. Hieraus ergibt sich die wichtige Anforderung an den Quadrantensequenzer, daß die schiebenden Phi-Pulse vollständig unter dem S_{in}-Puls liegen müssen.

Nach dem Schließen von S_{in} wird die zweite Verstärkerstufe durch Öffnen von R_2 aktiviert. Für die nun anstehende Signal-Verstärkung werden die 4 S_i-Schalter nun im möglichst gleichen zeitlichen Muster (relativ zu S_{in}) wie beim Baseline-Sampling betätigt. Dadurch lädt sich der Kondensator $C2_{fb}$ sukzessive auf die 4-fache Differenz zwischen Signal- und Baseline-Spannung auf, womit die CAMEX-Rauschfilterung — durch das jeweils 4-fache Abtasten — mit gleichzeitigem Verstärken beendet wäre.

Zuletzt wird das so erhaltene Signal in das Sample-and-Hold-Register (der Kondensator $C_{S\&H}$) übernommen, indem der zugehörige (nicht beschriftete) Schalter geschlossen wird, und die 64 Pixel der Zeile n stehen zum Abholen bereit.

Damit dieses Abholen auch bei Pixel 0 beginnt, wird als letztes sichergestellt, daß die Sel-Channel-Schalter aller anderen Pixel geöffnet sind (RD-Puls im nächsten Abschnitt 3.2.1.2).

Während nun Zeile n Pixel für Pixel in den ADC übernommen wird, kann für Zeile $(n+1)$ bereits wieder das Baseline- und Signal-Sampling samt Verstärkung durchgeführt werden; es muß nur darauf geachtet werden, daß Zeile $(n+1)$ erst dann in das S&H-Register übernommen wird, nachdem alle 64 Pixel der Zeile n ausgelesen sind. Auf diese Art entsteht der Anfang einer Pipeline (vergleiche Abschnitt 3.4), die eine zusätzliche Anforderung an den Quadrantensequenzer darstellt.

3.2.1.2 Der TIMEX

Dieses ist der Steuerchip für den CAMEX: Er bewirkt das richtige Schalten der eben beschriebenen Schalter (vergleiche auch Abbildung 3.7), indem er aus einer definierten Pulsfolge eines einzelnen Signals (des „Timing"-Signals) für jeden Schalter (bis auf die Channel-Select-Schalter) ein eigenes Ansteuerungs-Signal generiert. Dieser Teil des TIMEX ist im Wesentlichen ein Schieberegister, das mit den negativen Flanken des Timing-Signals getaktet wird. Durch geschicktes Verknüpfen der Register-Ausgänge Q_i miteinander sowie mit Timing selber, entstehen die erforderlichen Ansteuerungs-Signale (im oberen Teil der Abbildung 3.8 dargestellt). Eine Übersicht über alle Timing-Pulse und ihre Wirkung gibt Tabelle 3.1.

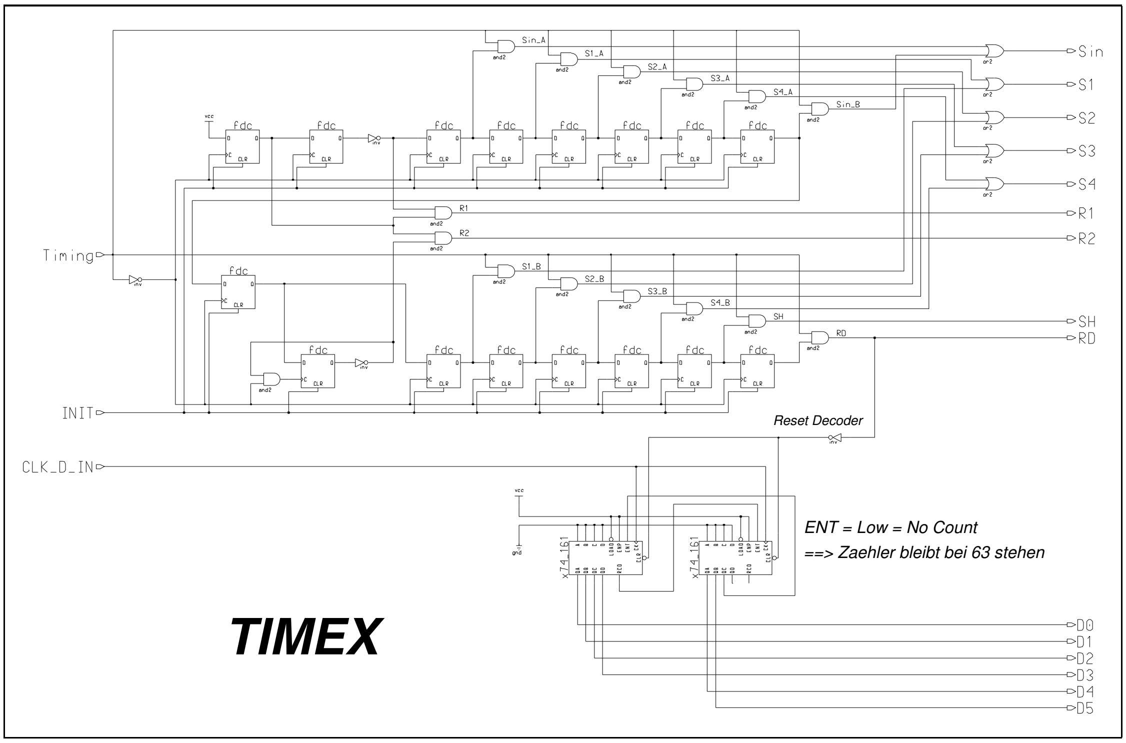


Abbildung 3.8: *Der schematische Aufbau des TIMEX. Zu erkennen ist im oberen Teil das Schieberegister und die zugehörige Logik, die aus dem Timing-Signal die CAMEX-Schalter-Ansteuerungs-Signale generiert; im unteren Teil ist der vom CLK_D_IN-Signal getaktete Zähler, der das serielle Auslesen der Pixel steuert und mit dem letzten Timing-Puls rückgesetzt wird. Dieser Schaltplan soll nur den bei XMM/ABRIXAS benötigten Teil des TIMEX verdeutlichen; er entspricht der tatsächlich verwendeten Schaltung nicht vollständig.*

Was die Channel-Select-Schalter, also die Ausgänge der einzelnen CAMEX-Kanäle betrifft, so hängen sie alle an einem Multiplexer (oft auch „Decoder" genannt). Dies ist eine Schaltung, die (abgesehen von eventuellen „enable"-Eingängen) n Steuer- und $(2^n - 1)$ durchnummerierte Daten-Eingänge hat. Abhängig von der Nummer, die von den Steuer-Bits gebildet wird, wird genau eine der Daten-Leitungen auf den Ausgang durchgeschaltet. Läßt man den Multiplexer von einem n-Bit-Zähler steuern, so können von Eingang Nr. 0 bis Eingang Nr. $(2^n - 1)$ alle Eingänge nacheinander am Ausgang abgegriffen werden. Für die $64 = 2^6$ Pixel ist daher im TIMEX ein 6-Bit-Zähler integriert, der immer mit der Vorderflanke des CLK_D_IN-Signals hochzählt und von dem letzten Timing-Puls rückgesetzt

Puls-Name	**Erklärung**
S_{r1} (R_1)	Schließt die beiden Verstärkerschalter R1 und R2 und setzt somit den CAMEX zurück; R1 wird wieder geöffnet, R2 bleibt geschlossen
S_{in}	Entkopplung von CAMEX und CCD vor Übernahme der Baseline-Spannung
S_1 bis S_4	Baseline-Sampling
S_{in}'	Entkopplung von CAMEX und CCD während des Schiebens der Elektronen auf die Anode, also vor der Übernahme der Signal-Spannung in den CAMEX
S_{r2} (R_2)	Öffnet auch R2 und aktiviert somit den Hauptverstärker
S_1' bis S_4'	Signal-Sampling und -Verstärkung
SH	Sample and Hold: Übernahme der verstärkten Spannung in das Ausleseregister
RD	Reset Decoder: Fange mit dem seriellen Auslesen wieder bei Pixel 0 an

Tabelle 3.1: *Die Pulse des Timing-Signals und ihre Wirkung.*

wird (siehe Abbildung 3.8). Da Pixel 0 bereits mit dem eben genannten Reset-Decoder-Puls anliegt, werden für die restlichen Pixel 63 CLK_D_IN-Pulse benötigt, die alle vor dem SH-Puls des Timing-Signals liegen müssen. Bei Messungen fällt oftmals auf, wie nach dem letzten CLK_D_IN-Puls zunächst Pixel 63 der Zeile n am Ausgang anliegt, mit dem SH-Puls ist es dann Pixel 63 der Zeile $(n+1)$, und nach dem RD-Puls ist Pixel 0 der eben genannten Zeile $(n+1)$ zu sehen.

Als letztes Signal für den TIMEX ist das initialisierende INIT zu nennen, das vor allen anderen Pulsen kommen muß und die CAMEX-Ansteuerung erst ermöglicht.

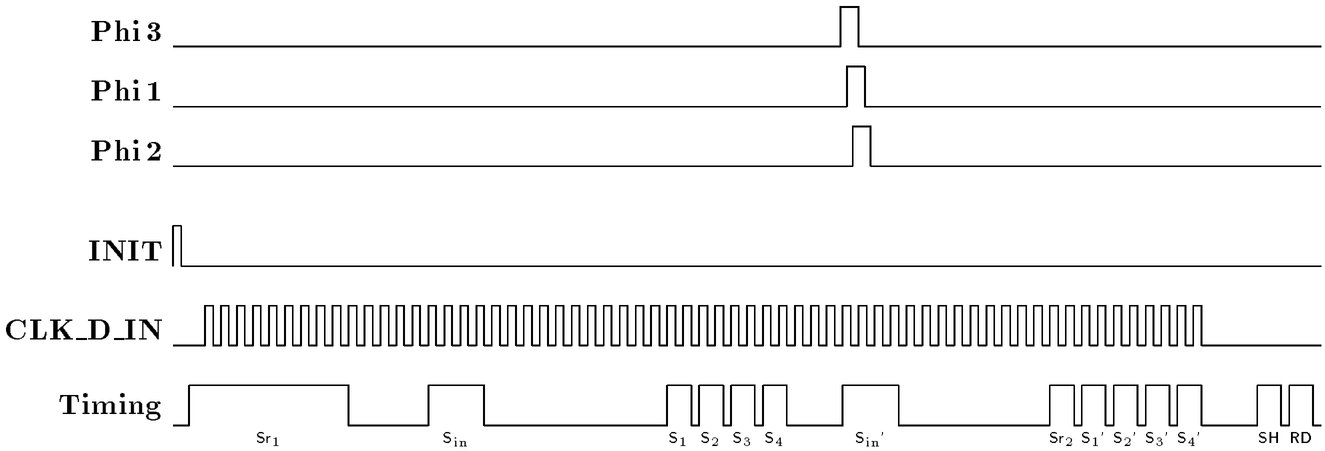


Abbildung 3.9: *Die TIMEX-Signale und die Phi-Pulse im zeitlichen Überblick. Die Lage der Phi-Pulse und das Puls-Muster des Timing-Signals berücksichtigt bereits die Messungen aus Kapitel 4.*

Der relative zeitliche Verlauf aller TIMEX-Signale kann Abbildung 3.9 entnommen werden, in der die Messungen aus Kapitel 4 bereits berücksichtigt wurden. Außerdem sind in dieser Graphik auch die Schiebe-Pulse eingezeichnet, auf die genauer in Abschnitt 3.2.2 eingegangen wird. Deutlich zu sehen ist, wie dieses Pulsmuster die (Verarbeitungs-) Dauer einer Zeile bestimmt: Mit der Verarbeitung einzelner Pixel kann erst nach dem INIT-Puls begonnen werden, und die beiden letzten Timing-Pulse dürfen erst gegeben werden, nachdem auch das 64. Pixel abgearbeitet wurde.

3.2.2 Der Testpuls AMOS und die Phi-Pulse

Es sind zwei Möglichkeiten vorgesehen, wie das Verstärker- und Datenverarbeitungs-System des Kamerakopfes auch im Flug getestet werden kann: Das eine ist eine reine Bestimmung des System-Rauschens der Elektronik, bei dem keinerlei Signal-Ladung in die Verstärkerkette gelangen darf, das andere ist ein Test der Verstärkung als solche, für den eine definierte Eingangs-Spannung für die 64 CAMEX-Eingänge notwendig ist, die genau zu dem Zeitpunkt angelegt wird, zu dem sonst die Signal-Ladungen in den CAMEX geschoben würden.

Damit im ersten Fall sogar die thermischen Elektronen von der Anode abgesogen werden, muß das CCD rückwärts schieben. Dies wird durch ein Vertauschen der Phi-Puls-Reihenfolge erreicht, was in der folgenden Erklärung des Schiebvorgangs offensichtlich wird.

Die Phi-Pulse steuern die auf S. 21 erwähnten 3 Mulden der Potentiale Φ_1, Φ_2 und Φ_3. Es handelt sich um 3 Potentiale pro Zeile, damit sich die Daten beim „Umschütten" der Elektronen in eine Nachbarzeile (der Schiebevorgang) nicht vermischen können, was die Orts- aber auch die Energie-Information ja verfälschen würde („Split Events" und „Pile-Ups", siehe Bihler 1996).

Man kann sich dieses Sammeln und Schieben der Daten in einem mechanischen Analogon als Eimerkette verdeutlichen; die Eimer stellen die Potentialmulden dar, das Wasser darin die Elektronen. Jeder Eimer ist dicht über seinem Boden mit seinen beiden Nachbarn verbunden, und immer jeder dritte Eimer steht auf der selben Höhe. Wird in einen Eimer Wasser gefüllt, so fließt es durch die Schläuche in den tiefsten Eimer, der dem Potential Φ_1 entspricht. Soll das Wasser (die Elektronen) nun in eine Richtung (z.B. nach rechts) transportiert werden, so muß der rechte Nachbareimer (Potential Φ_3) tiefer stehen als Eimer 1. Dies geschieht indem Eimer 3 abgesenkt und Eimer 1 angehoben wird, allerdings nur soweit, daß das Wasser nicht in den linken Nachbareimer 2 fließen kann. Diese Prozedur wird noch 2 mal mit den Eimern 3 und 2 wiederholt, und das Wasser ist (nach rechts) im nächsten Eimer der Nummer 1, also der nächsten Zeile, angekommen.

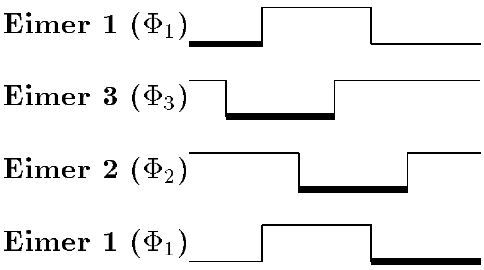


Abbildung 3.10: *Der zeitliche Verlauf des Schiebevorgangs im Eimermodell.*

Die digitalen Steuerpulse Phi 1, Phi 2 und Phi 3, die vom Quadrantensequenzer kommen müssen, werden im Kamerakopf dazu verwendet, analoge Schalter zu schalten. Daher sehen alle 3 Pulse gleich aus, wie Abbildung 3.11 verdeutlicht. Großer Wert wurde zum einen auf die in der Zeichnung angedeutete Drittelung der einzelnen Pulse gelegt, zum anderen müssen alle Pulse vollständig unter dem Timing-S_{in}-Puls liegen, wie in Abschnitt 3.2.1.1 bereits erklärt.

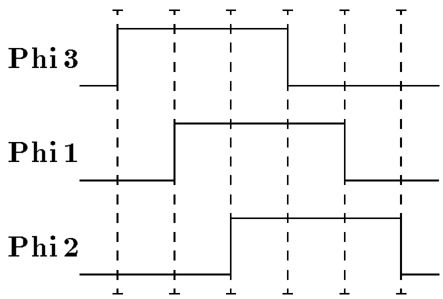


Abbildung 3.11: *Das geforderte Zeitverhalten der Schiebe-Pulse.*

Die Spannung für den zweiten Test wird mit Hilfe des Signals AMOS (Anoden-MOS) generiert. Der Name „AMOS“ wird für zwei leicht unterschiedliche Dinge gebraucht: Zum einen ist damit ein Potential gemeint, das an einer MOS-Struktur oberhalb der Anode auf dem CCD anliegt, zum anderen ist es, analog zu den Phi-Pulsen, der Name des vom Quadrantensequenzer kommenden Steuerpulses für dieses Potential. Wird eine definierte AMOS-Spannung angelegt, so werden auf der CCD-Anode eine definierte Ladungsmenge induziert (Abschnitt 3.2.1.1), die verwendet werden kann, um die Verstärkung zu testen. Wichtig dabei ist, daß

1. AMOS genau zu dem Zeitpunkt angelegt wird, zu dem auch beim normalen Schieben die Anode geladen würde,

2. AMOS anstelle der Phi-Pulse und nicht zusätzlich zu ihnen kommt, da der Ladevorgang sonst nicht definiert wäre,

3. AMOS bis zum Ende der Verarbeitung dieser Zeile, mindestens bis deutlich nach dem Signalsampling, anliegt.

3.2.3 Die restlichen CAMEX-Signale

Zusäztlich zu den bisher beschriebenen Signalen, werden noch weitere Steuerpulse am Kamerakopf gebraucht. In Abbildung 3.7 auf S. 22 sind dies die Eingänge CCD-RFGA und CMX-VB_ST, die vom Quadrantensequenzer gesteuert werden.

3.2.3.1 SEL_*i*

Um Leitungen einzusparen, gibt es die eben beschriebenen Signale INIT, Timing, CLK_D_IN und die Phi-Pulse nur einmal pro Quadrant, obwohl damit 3 unabhängige CCD-Systeme angesteuert werden sollen. Die Verteilung an das richtige System geschieht über ein weiteres Sequenzer-Signal: Das SELect, manchmal auch SEL_OUT genannt. Dieses Signal ist in 3-facher Ausführung vorhanden (SEL_0, SEL_1 und SEL_2), so daß das aktive CCD immer eindeutig definiert ist. Allerdings verteilt es nicht nur die einfach generierten Signale, sondern bestimmt auch, welcher CAMEX auf den Ausgang zur EPEA geschaltet wird, genauer welcher der 3 in Abschnitt 3.2.1.2 genannten Multiplexer arbeitet. Dies darf immer nur einer sein — schalteten mehrere CAMEXe ihre Daten auf die eine Leitung zur EPEA, würde das sonst unweigerlich zu „Datensalat“ führen.

3.2.3.2 VB_SEL_i

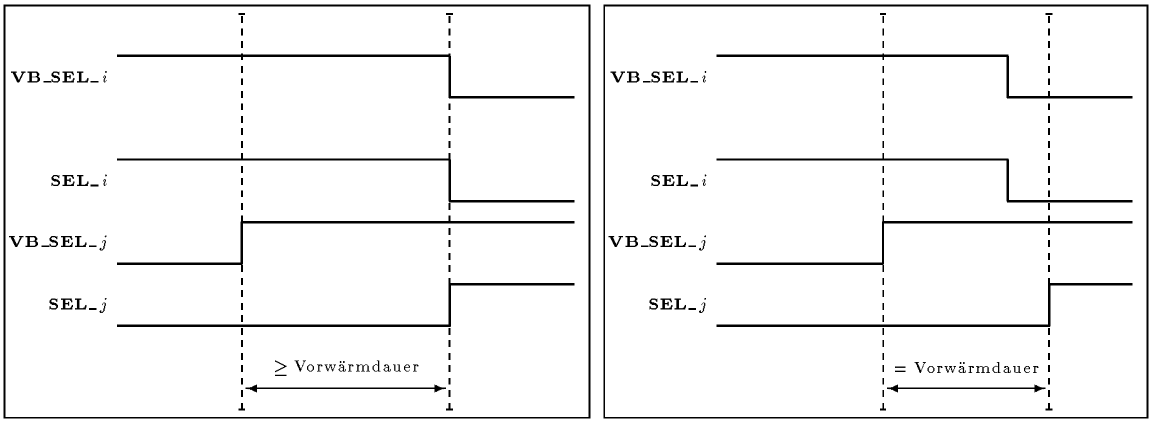
Das Verstärker-System jedes CCDs braucht einige Betriebsspannungen, wie z.B. CMX-VSSS. Nachdem pausierende Elemente auf einem Satelliten nur unnötig Strom verbrauchen und Wärme abgeben würden, ist es vorteilhaft, nicht benötigte Spannungen abzuschalten. Bei den pn-CCD-Systemen besorgt dies das VB_SELect-Signal, das vom Quadrantensequenzer für jedes CCD erzeugt wird. Allerdings müssen Spannungen in der Regel erst einschwingen, bevor sie stabil genug für den Betrieb sind. VB_SEL muß daher einige Zeit (ungefähr 1,2 ms) vor dem Beginn der eigentlichen CCD- und CAMEX-Ansteuerung eingeschaltet werden, und es muß sichergestellt sein, daß ein SEL_OUT

1. nicht ohne ein zugehöriges aktives VB_SEL

   und

2. erst nach Ablauf dieser „Vorwärm"-Zeit

gegeben wird. Dies wird in Abbildung 3.12 verdeutlicht.



(a) *CCD j wurde rechtzeitig vor CCD-Ende programmiert*

(b) *CCD j wurde zu kurz vor CCD-Ende programmiert*

Abbildung 3.12: *VB_SEL und SEL beim Umschalten zweier CCDs. Deutlich zu erkennen ist, daß ab dem Einschalten eines CCDs (Vorderflanke ↑ von VB_SEL_j) bis zum Einschalten von SEL_j die gesamte Vorwärmzeit verstrichen sein muß. Je nachdem, ob CCD j rechtzeitig programmiert wurde (a) oder so kurz vor CCD-Ende (b), daß die verbleibende Zeit nicht mehr ausreicht, schaltet SEL_j ↑ gleichzeitig mit SEL_i ↓ (a) oder erst später (b).*

Gemessen an einer angestrebten Auslesezeit von 4 bis 5 ms für ein ganzes CCD, sind 1 bis 2 ms jedoch nicht vernachlässigbar, weshalb diese Vorwärmzeit innerhalb eines Quadranten am besten CCD-überlappend geschaltet werden sollte. Außerdem muß die Wahl des als nächstes einzuschaltenden CCDs frei wählbar, also programmierbar sein.

3.2.3.3 RFGA_i

RFGA steht für **R**eset **FET**-**G**ate **A**node. Wie der Name schon sagt, setzt er die Anode zurück, auf die ja bei jedem Schiebevorgang neue Elektronen geschoben werden. Auf diese

Weise wird verhindert, daß das System in Sättigung geht. Sicherlich hat es Vorteile, wenn der Auslese-Zyklus eines CCDs mit einer leeren Anode begänne, weshalb ein RFGA am Anfang erwünscht ist. Dies läßt sich gut mit der erforderlichen Vorwärmzeit (Abschnitt 3.2.3.2) kombinieren: Während VB_SEL bereits ein-, SEL_OUT aber noch ausgeschaltet ist, kann die Anode rückgesetzt werden.

Für den Fall, daß damit Probleme auftreten, wurde gewünscht, daß dieses „Vorwärm-RFGA“ per Boden-Kommando an- und abschaltbar ist.

Außerdem ist — vor allem bei den Modi, die ein einzelnes CCD kontinuierlich betreiben (Timing-, S. 17, und Burst-Mode, S. 18) — zu erwarten, daß die Anode auch vor dem Abschalten des CCDs rückgesetzt werden muß, weshalb ein programmierbares „Zeilen-RFGA“ vor der Verarbeitung einer beliebigen Zeile notwendig ist.

3.2.4 Flush

Das letzte Signal, das vom Quadrantensequenzer an den Kamerakopf gegeben wird, wird wieder nur einfach erzeugt und in der EPCH an das aktive CCD geschaltet. Es handelt sich hierbei um das erwähnte Signal zur Datenverbesserung:

Depletierte Halbleiter tendieren dazu, an physikalischen Störstellen Elektronen einzufangen. Je größer der Weg ist, den ein durch ein Photon erzeugtes „Daten-Elektron“ im CCD zurücklegen muß, desto wahrscheinlicher wird dieser Einfang. Ein Maß dafür, wieviel ursprünglich erzeugte Elektronen auch am CAMEX ankommen, ist die CTE (**C**harge **T**ransfer **E**fficiency), die bereits Thema in mancher Diplomarbeit war [Bihler 1996 oder auch Kuster 1997]. Werden dem gesamten CCD zusätzliche Elektronen angeboten, so können die Störstellen durch diese abgesättigt werden, und es gehen dann (fast) keine Signal-Elektronen verloren.

Am sinnvollsten nutzt man dazu die Elektronen-Verschiebung beim Auslesen des CCDs: Nachdem die ersten paar Zeilen geschoben wurden, und sich somit keine validen Daten mehr in den zentralen Detektor-Zeilen befinden, wird diese innerste Zeile mit Elektronen geflutet. Daher auch der Signalname „Flush“. Wichtig für die Steuerung ist, daß genau während des Schiebens geflutet wird, das FLUSH-Signal also die Phi-Pulse vollständig überdeckt. Leider kann nicht sichergestellt werden, daß diese Elektronen nur in eine einzelne Zeile injektiert werden — typischerweise verteilen sie sich auf meherer Zeilen. Flush sollte daher erst in der 5. Zeile gegeben werden, und nach dem Auslesen der 200ten Zeile sollten noch weitere 10 weggeworfen werden, damit das nächste Bild nicht verfälscht wird.

Mindestens bei kontinuierlichen Modi (Timing-, S. 17, und Burst-Mode, S. 18) ist ein Fluten der inneren CCD-Zeilen jedoch nicht unbedingt notwendig oder auch sinnvoll. Sowohl Flush als auch das Wegwerfen der 10 Zeilen muß daher frei programmierbar sein.

3.3 Die Tübinger EPEA-Box

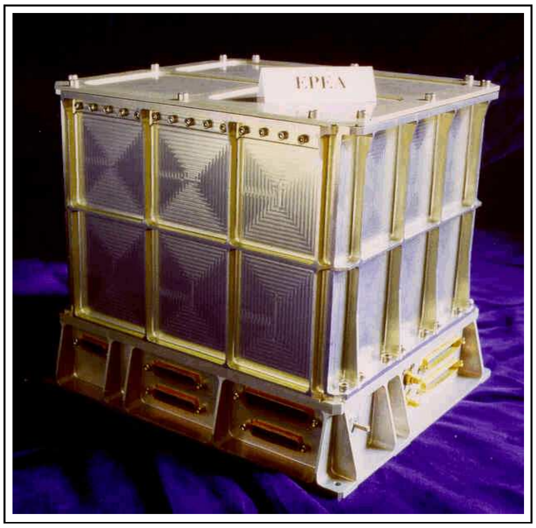


Abbildung 3.13: *Die EPEA-Box, wie sie im Satellit eingebaut werden wird (Das weiße Schild sollte den Box-Namen darstellen).*

In Abschnitt 2.1.4.1 wurde bereits erwähnt, daß der EPIC-pn-CCD-Event-Analyser EPEA für die gesamte Ablaufsteuerung der pn-CCD-Kamera zuständig ist. Es wurde daher großer Wert darauf gelegt, daß das System als ganzes weiterarbeiten kann auch wenn Teile davon ausfallen sollten. Dies war der Hauptgrund, warum das System in 4 unabhängige Quadranten aufgeteilt wurde, jeder mit seiner eigenen Steuerung und Datenverarbeitung und seinen eigenen 3 CCDs. Der Aufbau der EPEA-Box ist in Abbildung 3.14 schematisch dargestellt.

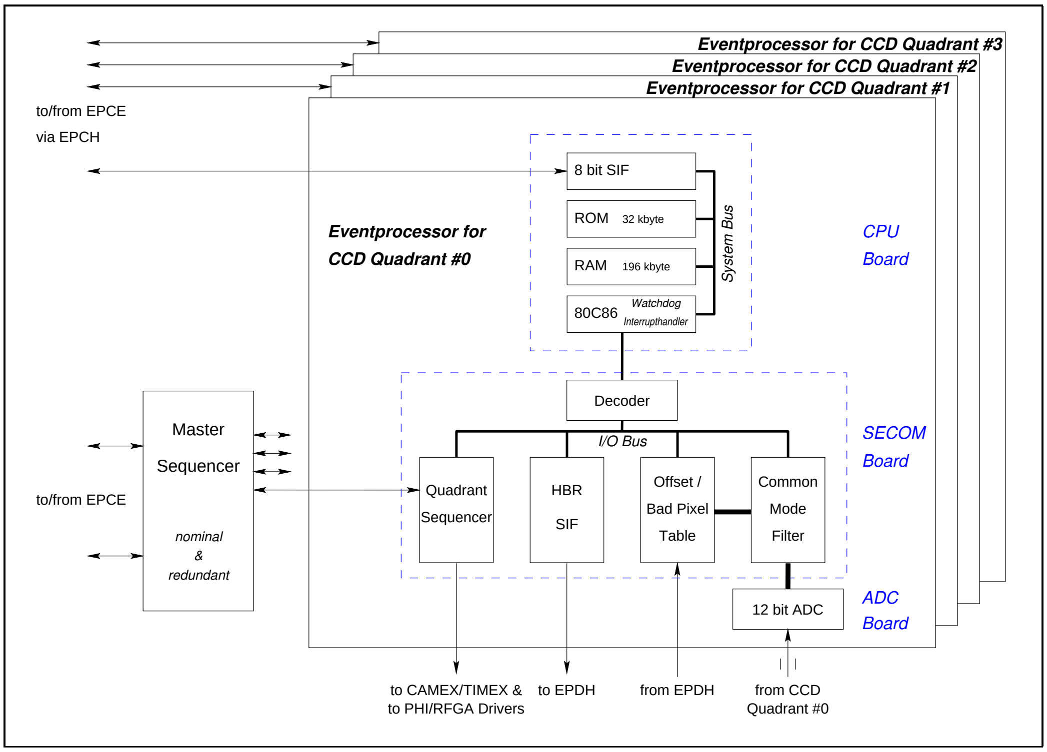


Abbildung 3.14: *Der schematische Aufbau der EPEA. Zu erkennen ist sowohl der nominelle wie auch der redundante Master-Sequenzer, sowie die 4 Quadranten-bezogenen Elemente.*

3.3.1 Der Master-Sequenzer

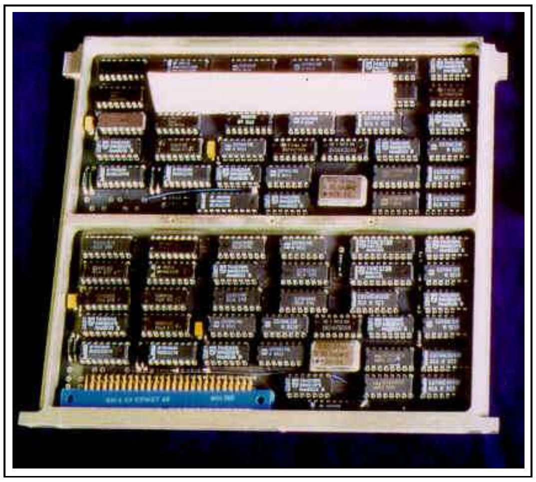


Abbildung 3.15: *Der redundante (oben, teilweise durch das Schild verdeckt) und der nominelle (unten) Master-Sequenzer.*

Dem Master-Sequenzer obliegt die Koordination der einzelnen Quadranten. Er legt den Zeitpunkt ihrer Aktivierung und damit auch die Integrationszeit in Einheiten von 10,48 ms fest, die ungefähr der Dauer eines Quadranten-Ablaufs entspricht. Dabei ist die minimal einstellbare Integrationszeit zwischen zwei Quadranten =0; die größte sind 16 Einheiten. Es können allerdings auch alle Quadranten gleichzeitig betrieben werden. Sollte ein ganzer Quadrant aus irgendwelchen Gründen vollständig ausfallen, so kann dies dem Master-Sequenzer mitgeteilt werden, und er überspringt ihn von da an in seiner Aktivierungs-Sequenz. Damit der Betrieb der pn-CCD-Kamera nicht an einem Master-Sequenzer-Ausfall scheitert, sind auf der „Master-Karte" zwei gleiche Master-Sequenzer untergebracht, ein nomineller und ein redundanter.

Die eben angesprochene Koordination erledigt der Master-Sequenzer indem er direkt mit dem Quadrantensequenzer in Verbindung tritt: Er liefert ihm den Takt (die — in der Flugversion — 25 MHz schnelle QCLK) und schaltet ihn mittels des Signales QSEQ (eine Art „Power-on") ein und aus. Natürlich gibt es jeweils das nominelle Signal (1QCLK, 1QSEQ), als auch dasjenige des redundanten Masters (2QCLK, 2QSEQ). Im Gegenzug dazu muß der Quadrantensequenzer dem Master-Sequenzer melden, wenn er mit seinem Quadranten fertig („ready") ist. Dies geschieht mittels des Signals QSEQ_RDY.

3.3.2 Die CPU

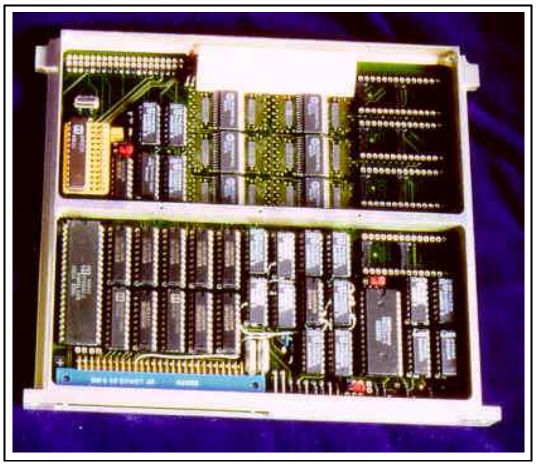


Abbildung 3.16: *Die CPU-Karte.*

Die CPU[3]-Karte ist eine der drei Karten, die in der EPEA-Box 4-fach vorhanden sind, für jeden Quadranten eine. Auf ihr befinden sich neben der eigentlichen CPU — ein strahlungsharter Intel® 80c86 Mikroprozessor (μP) — einige Speicherbausteine, in denen die gesamte Bord-Software untergebracht ist, die im Zusammenhang mit einem Quadranten benötigt wird.
Die CPU hat mehrere Aufgaben: Sie muß

1. „ihren" Quadranten gemäß der über die Data-Handling-Unit erhaltenen (siehe S. 10) Wünsche des Bodenpersonals richtig programmieren

[3]**C**entral **P**rocessing **U**nit, oft auch einfach „der Prozessor" genannt

2. die Meßdaten samt ihrer Ortsinformation und die dazwischengeschobenen Zeitworte beim Common-Mode-Filter (Abschnitt 3.3.4.1) abholen
3. die gefilterten Meßdaten von Spuren minimal ionisierender Teilchen befreien und danach über das High-Bitrate-Interface HBR an die EPDH weiterreichen
4. aus den ungefilterten Meßdaten eine ortsabhängige Korrekturtabelle (die „Offset-Tabelle") erstellen und für den Common-Mode-Filter (Abschnitt 3.3.4.1) im Offset-RAM („ORAM") speichern
5. ebenfalls aus ungefilterten Meßdaten auf Verlangen der Bodenstation eine ortsabhängige Rauschkarte berechnen und — in Telemetrie-Format verpackt — zum Senden bereitstellen
6. die Housekeeping-Daten verwalten und der EPCE zur Verfügung stellen, die sie über das Low-Bitrate-Interface LBR der EPDH weitergibt (Abschnitte 2.1.4.1/2.1.4.1).

3.3.3 Der ADC

Der ADC ist der AD872A, ein Flash-Analog-Digital-Wandler der Firma Analog Devices, der auf einer kleinen Extra-Karte realisiert ist, die am unteren Rand der SECOM-Karte (nächster Abschnitt 3.3.4) befestigt ist, wo er vor Strahlung gut geschützt ist. Der ADC872A digitalisiert die Analog-Daten iterativ in 3 Stufen. Sein Eingangs-Spannungs-Bereich beträgt $\pm\,1{,}024\,\mathrm{V}$, was eine Spannungs-Anpassung durch einen weiteren Operations-Verstärker notwendig macht, der sich noch im Kamerakopf befindet (der nicht-invertierende OPV an der rechten Seite von Abbildung 3.7).

Wie die CAMEX-Spannungen muß auch der ADC erst einschwingen bevor er stabil arbeitet. Dazu muß er für mindestens 6 Taktzyklen kontinuierlich mit einem symmetrischen Takt getaktet werden, obwohl er in eingeschwungenem Zustand auch Pausen von maximal $100\,\mu$s verträgt. Dies ergibt sich aus seiner minimalen Taktfrequenz von 10 kHz; seine maximale Taktrate beträgt 10 MHz, die jedoch wegen der Verarbeitungsdauer eines Pixels im Common-Mode-Filter nicht erreicht werden.

Eine weitere Anforderung an den Quadrantensequenzer außer diesem Einschwingen, das geschehen kann während die erste CCD-Zeile im CAMEX verstärkt wird (siehe die Abschnitte 3.2.1.1 und 3.4), ist

1. der korrekte Übernahme-Zeitpunkt des analogen Signals in den ADC

   und

2. die interne ADC-Pipeline.

Zur Pipeline: Durch die iterative Digitalisierung hat der AD872A 3 Analog-Werte gleichzeitig in Arbeit, allerdings in unterschiedlichen Konvertierungs-Stadien. Bis die 12 Bit des ersten Digital-Wertes in den Common-Mode-Filter übernommen werden können, vergehen demzufolge 3 „Pixel".

Zum korrekten Übernahmezeitpunkt: Wie in Abschnitt 3.2.1.2 bereits erwähnt wurde, wird das analoge Signal mit der Vorderflanke der Decoder-Clock (CLK_D_IN$\uparrow$) angefordert. Nach Ablauf der Signal-Laufzeit schwingt der analoge Signal-Pegel am ADC-Eingang ein und liegt solange an, bis der nächste Wert ankommt. Um sicher zu sein, daß auch der korrekte Wert konvertiert wird, darf die übernehmende Vorderflanke des ADC-Taktes (ADC_CLK ↑) erst im letzten Drittel dieses Signal-Plateaus gegeben werden. Dies geschieht, weil die Anstiegszeit des Analogsignals deutlich kürzer ist als die Abklingzeit, und damit sichergestellt wird, daß das Signal des vorherigen Pixels wirklich abgeklungen ist und das Signal den Wert des aktuellen Pixels ohne weitere Schwankungen angibt. Dieser Übernahmezeitpunkt in den ADC wird schematisch in Abbildung 3.17 dargestellt und in Abschnitt 4.3 genauer untersucht. Jegliche Übernahme während der langsamen Einstellung des Signals verfälscht den tatsächlichen Meßwert und führt zum sogenannten „Fahnen-Effekt", dem ein ganzer Abschnitt in der Diplomarbeit von Bihler (1996) gewidmet ist.

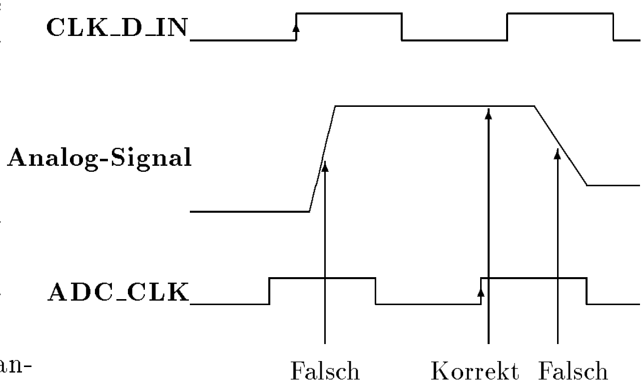


Abbildung 3.17: *Die schematische Darstellung des korrekten Übernahmezeitpunktes in den ADC.*

3.3.4 Die SECOM-Karte

„SECOM", das steht für **SE**quenzer-**CO**mmon-**M**ode, und zeigt deutlich, welches die zentralen Teile auf dieser Karte sind: der Quadrantensequenzer und der Common-Mode-Filter, der ab S. 34 beschrieben werden soll. Zu beiden Schaltungen gehören externe Speicherbausteine — das „Offset-RAM[4]" (ORAM), das die eben erwähnte Korrekturtabelle für den Common-Mode-Filter enthält, und das „Line-RAM" (LRAM), in dem das Sequenzer-Programm gespeichert wird — die sich ebenfalls auf der SECOM-Karte befinden. Außerdem sind in der SECOM-Schaltung eine Schnittstelle zur CPU (der „Decoder") und das erst im letzten Abschnitt wieder erwähnte HBR integriert. Auf letzteres soll hier nicht näher eingegangen werden, da der Quadrantensequenzer mit dieser EPEA-EPDH-Schnittstelle nichts zu tun hat; der Decoder wird in Abschnitt 3.3.4.3 näher erläutert.

Nachdem die SECOM-Schaltung sehr umfangreich und komplex ist, wie spätestens die Schaltpläne in Anhang D zeigen, wäre ein diskreter Aufbau aus gängigen ICs zu groß geworden, als daß man ihn hätte sinnvoll handhaben können. Es wurde daher beschlossen, auf Bausteine auszuweichen, die es ermöglichen, eine große Schaltung auf kleinem Raum unterzubringen:

[4] **R**andom **A**ccess **M**emory, oder auch **R**ead **A**nd Write **M**emory, ein Speicher aus dem nicht nur Daten in beliebiger Reihenfolge ausgelesen, sondern auch hineingeschrieben werden können.

In der Entwicklungsphase wurde mit sogenannten FPGAs[5] von XILINX® gearbeitet. Dies sind programmierbare Bausteine, die aus vielen untereinander verbundenen Zellen bestehen. In diesen Zellen sind alle möglichen Verknüpfungen von meist 4 Eingangs-Signalen samt eines Speichers für das Verknüpfungs-Ergebnis vorgesehen. Die Programmierung besteht darin, die Zellverbindungen und diese Verknüpfungen auszuwählen. Je nachdem, in welchem FPGA-Teil welcher Schaltungsteil realisiert ist, also wie groß die Signal-Laufzeiten sind, verkraftet die Schaltung schnellere oder langsamere Taktraten. Da die Gesamtheit von Decoder, Quadrantensequenzer, Common-Mode-Filter und High-Bitrate-Interface eine recht umfangreiche Schaltung ergibt, mußte sie für die Test-SECOM-Karte auf drei FPGAs verteilt werden. Es hat sich herausgestellt, daß der Quadrantensequenzer in der derzeitigen Version nur mit ca 20 MHz fehlerfrei arbeitet.

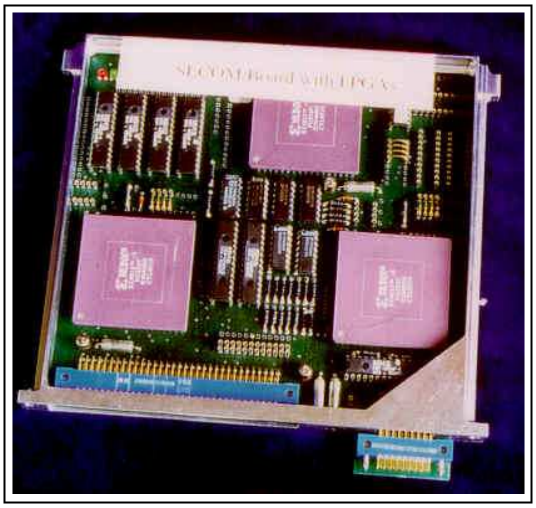


Abbildung 3.18: *Die SECOM-Karte mit den 3 FPGAs, Line- und Offset-RAM und dem CPU-FIFO. Am unteren rechten Rand ist die ADC-Karte zu erkennen.*

In der Flugversion der SECOM-Karte werden Decoder, Quadrantensequenzer, Common-Mode-Filter und das HBR gemeinsam in Form eines ASICs[6], also eines speziell hergestellten elektronischen Chips realisiert werden. In ihm sind die Signal-Laufzeiten deutlich geringer, so daß die erforderlichen 25 MHz kein Problem darstellen.

3.3.4.1 Der Common-Mode-Filter

Der Name dieses Schaltungsteils kommt von einem CAMEX-Effekt, dem sogenannten „Common-Mode-Rauschen“: Bei der Verarbeitung einer Zeile im CAMEX (Abschnitt 3.2.1.1) kann es vorkommen, daß dessen Versorgungs-Spannung leicht schwankt, was sich in einer Schwankung der Verstärkung ausdrückt. Da immer alle 64 Pixel einer Zeile gleichzeitig verarbeitet werden, wirkt sich diese Schwankung auf alle 64 Pixel gleichmäßig aus — dieses ist der „Common-Mode“.

Da wegen der beschränkten Telemetrie-Kapazität von < 40 kbit/sec nicht alle Pixel zur Erde übertragen werden können, muß diese Schwankung an Bord bestimmt und eliminiert werden, damit im Anschluß daran „informationslose“ Pixel ausgefiltert werden können, die zur Bestimmung des Common-Modes noch benötigt werden. Am besten geht dies, indem der Median einer Zeile bestimmt und abgezogen wird.

Zur Erinnerung: Der Median ist der mittlere Wert einer Werte-Menge, die nach ihrer Größe sortiert ist; besteht eine Menge aus m Elementen a_i, so ist der $\frac{m}{2}$-te Wert (im Zweifelsfall abgerundet) der Median. Im Gegensatz dazu ist der Mittelwert bekanntlich als $\frac{\sum_{i=1}^{m} a_i}{m}$ definiert.

Man nimmt den Median und nicht den Mittelwert, da die wenigen „Photonen-Pixel“ ja einen deutlich höheren Wert als die mehrheitlichen „Rausch-Pixel“ haben, und den abzuziehenden Wert nicht beeinflussen sollen.

[5]**F**ield **P**rogrammable **G**ate **A**rray

[6]**A**pplication **S**pecific **I**ntegrated **C**ircuit, bzw. **A**nwendungs-**S**pezifischer Schaltkreis (**IC**)

Nachdem eine der Hauptaufgaben des Quadrantensequenzer darin besteht, diesen Common-Mode-Filter zu steuern, soll er im folgenden recht genau erklärt werden. In Abbildung 3.23 ist eine grobe Übersicht über den Common-Mode-Filter („COM") enthalten, allerdings ohne die vom Quadrantensequenzer kommenden Signale. Ein Verständnis der Anforderungen des „COM" an den QSEQ läuft jedoch auf ein Verständnis dieser Signale hinaus, weshalb die Erklärung anhand des genaueren Blockschaltbildes aus Abbildung 3.19 erfolgen soll. In ihm sind mit einem S alle Signale gekennzeichnet, die der Quadrantensequenzer liefern muß. Zur besseren Übersicht sind sie mit ihren noch zu erklärenden Aufgaben in Tabelle 3.2 zusammengefaßt.

Signal	**Signal-Name**	**Signal-Aufgabe in Kurzform**
$\overline{\text{COMR}}$	Common-Mode-Read	Taktet in der einen Datenfluß-Pipeline-Stufe den Pixel-Zähler zur ORAM-Adressierung; in der nächsten Datenfluß-Pipeline-Stufe übernimmt es das Good-Event-Flag in sein Register.
$\overline{\text{NEWCHIP}}$	New Chip	Initialisiert am CCD-Anfang die Zeilen-Zähler,setzt die Median-Summe auf Null und schaltet ein von der CPU angemeldetes $\overline{\text{CLRFIF}}$ („Clear CPU-FIFO").
$\overline{\text{NEWLINE}}$	New Line	Taktet die beiden Zeilen-Zähler und initialisiert am Zeilen-Anfang den Pixel-Zähler, den Sortierer und das Line-FIFO.
$\overline{\text{COMW}}$	Common-Mode Write	Übernimmt in der einen Datenfluß-Pipeline-Stufe die Daten aus dem ADC in den Sortierer und das Line-FIFO; in der nächsten Datenfluß-Pipeline-Stufe setzt es das Good-Event-Register zurück.
$\overline{\text{MEDCLK}}$	Median Clock	Übernimmt den Median in ein Register und addiert ihn zur bisherigen Median-Summe dazu.
$\overline{\text{WFTIM}}$	Write-FIFO Time	Liefert den Schreib-Doppelpuls um ein Zeitwort in das CPU-FIFO zu schreiben.
$\overline{\text{WFENRG}}$	Write-FIFO Energy	Liefert den Schreib-Doppelpuls um Meßdaten und ihre Adresse in das CPU-FIFO zu schreiben.
QSEQ_RDY	QSEQ-Ready	Zeigt ein Quadranten-Ende an, mit dem der Chopper getaktet wird.
CCD (1:0)	CCD-ID	Kodiert das gerade aktive CCD zur Steuerung CCD-abhängiger Korrektur- und Auswahl-Werte.

Tabelle 3.2: *Ein Überblick über die Signale, die der Quadrantensequenzer an den Common-Mode-Filter liefern muß samt einer kurzen Beschreibung ihrer Funktion. Eine ausführliche Erläuterung soll diesem Abschnitt 3.3.4.1 entnommen werden. Der invertierende Balken über einem Signalnamen bedeutet generell, daß dieses Signal „low-aktiv" ist, also bei „Pegel 0" etwas bewirkt. Da dieser Balken auf keiner Tastatur direkt belegt ist, haben verschiedene Design-Programme ihn unterschiedlich definiert; so ist z.B.* $\overline{COMW}$ *gleichbedeutend mit COM_ (Mentor®-Notation) und COMW/ (ORCAD®-Notation). Im Gegensatz zu „low-aktiv" gibt es noch „high-aktive" Signale, die bei „Pegel 1" schalten, und die dazwischenliegenden positiven (↑) und negativen (↓) Flanken.*

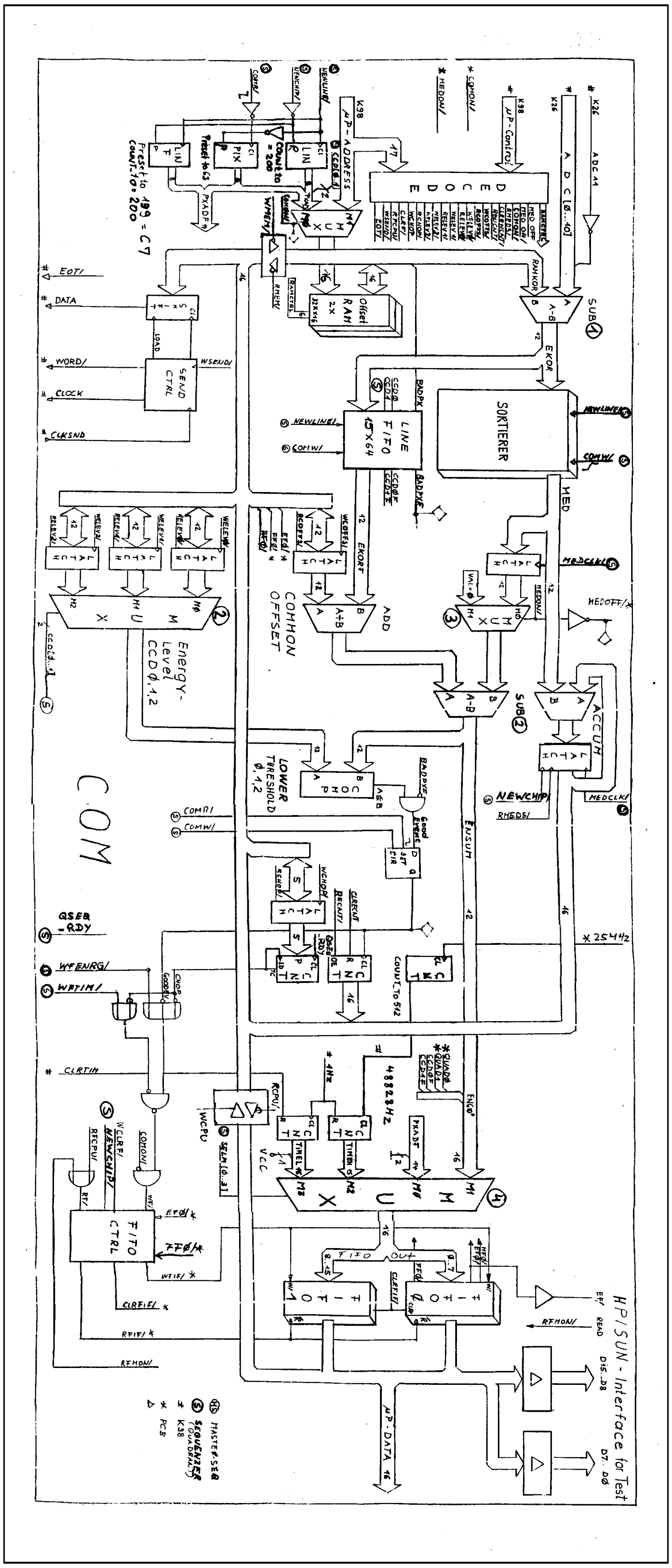


Abbildung 3.19: *Das Blockschaltbild des Common-Mode-Filters. Alle mit S gekennzeichneten Signale müssen vom Quadrantensequenzer generiert werden.*

Auffallend an diesem Operationswerk ist, daß es seinem Steuerwerk nichts über den „Stand der Dinge“ innerhalb des COM mitteilt. Damit ist der QSEQ gezwungen, die Steuer-Signale „auf Verdacht“ zu ändern, bzw. anhand eines zeitlich fest vorgegebenen Musters, das nur auf den Erfahrungswerten der Signal-Laufzeiten der Daten im Common-Mode-Filter beruht.

Damit der Median auch wirklich nur vom Common-Mode bestimmt wird, muß vor seiner Bestimmung der pixelspezifische Basiswert abgezogen werden („SUB 1“ in Abbildung 3.19). Dieser wird der in Abschnitt 3.3.2 erwähnten Tabelle entnommen, die die CPU im ORAM („Offset-RAM“ mitte links in Abbildung 3.19) gespeichert hat. Außerdem ist in der Offset-Tabelle auch eine Klassifizierung des jeweiligen Pixels (4 Bit, u.a. „Badpixel“, nachzulesen z.B. in Kuster 1997) abgelegt, die den Filter beeinflußt.

Nach der Offset-Korrektur gelangen die Daten zur Median-Bestimmung in den Sortierer (oben links in Abb. 3.19), der jedes Pixel mit der positiven Flanke des Signals $\overline{\text{COMW}}$ übernimmt. Sind alle Pixel sortiert, so wird der Median mit $\overline{\text{MEDCLK}}$ zum einen in einem Register für die nachfolgende Datenverarbeitung gespeichert und zum anderen über alle Zeilen eines CCDs als Kontrollwert aufsummiert („LATCH“ bzw. „ACCUM“ hinter dem Sortierer oben in der Mitte von Abb. 3.19). Der Sortierer wird mit $\overline{\text{NEWLINE}}$ am Anfang einer jeden Zeile gelöscht, während $\overline{\text{NEWCHIP}}$ die Median-Summe zu Beginn jedes Auslesens eines CCDs auf 0 setzt.

Nachdem der Median einer Zeile erst bestimmt sein muß, bevor er abgezogen werden kann, müssen alle 64 Pixel einer Zeile zusammen mit ihrer Bad-Pixel-Klassifizierung und CCD-Nummer während des Sortierens zwischengespeichert werden („Line-FIFO“, oben links in Abb. 3.19); die weitere Verarbeitung kann dann erfolgen, solange die nächste Zeile sortiert wird. Dieses ist die nächste Stufe der im CAMEX begonnenen Zeilen-Pipeline (Abschnitte 3.2.1.1 und 3.4).

Das Line-FIFO[7], in dem die Zeile zwischengespeichert wird, ist mittels eines RAMs realisiert, das wie der Sortierer eine spezielle Ansteuerung durch die Signale $\overline{\text{COMW}}$ und $\overline{\text{NEWLINE}}$ benötigt: Die RAM-Adresse wird mit $\overline{\text{NEWLINE}} = 0$ initialisiert und die in Adresse 0 gespeicherten Daten liegen am FIFO-Ausgang an. Wird $\overline{\text{COMW}}$ low, so werden die am FIFO-Eingang anliegenden Daten in das RAM übernommen, und danach, wenn $\overline{\text{COMW}}$ wieder auf 1 geht, wird die Adresse um 1 erhöht, und die Daten der nächsten Zelle liegen am Ausgang an. Dadurch ergibt sich ein komplexes Muster aus Pixel-Daten der alten Zeile n und neuen Zeile $(n+1)$ am FIFO-Ausgang (vergleiche Abbildung 3.20).

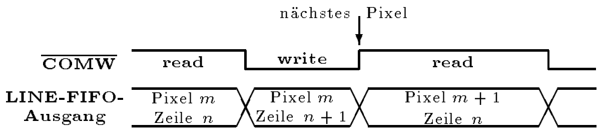


Abbildung 3.20: *Das Funktionsprinzip des LINE-FIFOs: Während* $\overline{COMW} = 1$ *kann gelesen werden, mit* $\overline{COMW} = 0$ *wird der neue Wert hineingeschrieben und* $\overline{COMW}\uparrow$ *schaltet um eine Adresse weiter.* $\overline{NEWLINE}$ *läßt bei Adresse 0 beginnen.*

Der Zwischenspeicher muß wie ein FIFO funktionieren, da die Daten ohne Ortsinformation aus dem ADC kommen und erst im Common-Mode-Filter die Pixel-Adresse aufgrund ihrer Reihenfolge zugeordnet bekommen. Das Zwischenspeichern der 14 Adress-Bits wäre sehr platzaufwendig; trotzdem müssen die Adressen sowohl vor dem Sortierer bei der Offset-Korrektur als auch nach dem Sortierer beim weiteren Verarbeiten (also eine Zeile

[7] **F**irst **I**n **F**irst **O**ut, eine Datenschlange analog einem Schieberegister oder auch einer Pipeline wie in Abschnitt 3.4 beschrieben.

später) korrekt sein, weshalb 2 Zeilen-Zähler vorhanden sind, die um eine Zeile versetzt mit $\overline{\text{NEWLINE}}$ zählen (links außen in Abb. 3.19). Damit beide Zeilen-Zähler auch immer die richtige Zeilen-Nummer angeben, müssen sie am Anfang eines CCDs (genauer gesagt, wenn die erste Zeile aus dem ADC kommt, also in der zweiten Stufe der Zeilen-Pipeline) mit dem Signal $\overline{\text{NEWCHIP}}$ auf einen Anfangswert gesetzt werden, und dürfen dann nur bis 199 zählen (Zeile 200 $\equiv$ Zeile 0) — bei kontinuierlichen Betriebsmodi soll die Zeile ja auch stimmen!

Um innerhalb einer Zeile die richtige Pixel-Nummer zu erhalten, ist ein dritter Zähler implementiert, mit dem die Zeilen-Zähler jeweils kombiniert werden — die Pixel-Nummern sind ja synchron in jeder Zeile (siehe Abb. 3.20). Der Pixel-Zähler wird mit der negativen Flanke des Signals $\overline{\text{COMR}}$ („Common-Mode-Read") getaktet, und am Anfang jeder Zeile mit $\overline{\text{NEWLINE}}$ auf seinen Anfangs-Wert gesetzt. $\overline{\text{COMR}}\downarrow$ muß dabei mindestens 50 ns vor $\overline{\text{COMW}}\downarrow$ kommen, schließlich muß das ORAM zur Offset-Korrektur vor der Daten-Übernahme in den Sortierer (bzw. das Line-FIFO) adressiert werden.

Ist der Median schließlich bestimmt und gespeichert, so kann er von jedem Pixel-Wert abgezogen werden („SUB 2" unter dem Akku in Abb. 3.19). Dabei könnten jedoch negative Werte entstehen, die im Folgenden Probleme bereiten würden. Daher wird vorher ein (von der Bodenstation über die CPU einstellbarer) fester Wert, der „Common Offset", addiert (rechte Mitte in Abb. 3.19), der in den Housekeeping-Daten übertragen wird und somit am Boden wieder abgezogen werden kann.

Nachdem die Daten nun Offset- und Common-Mode-bereinigt sind, kann der Filter zur Reduzierung der Datenmenge aktiv werden: Zunächst werden die Energie-Daten mit einem unteren Schwellwert verglichen („Lower Threshold 0, 1, 2 " in der Bildmitte von 3.19). Dieser „Energy-Level" ist (wieder von der Bodenstation über die CPU) für jedes CCD individuell einstellbar; welcher Vergleichs-Wert verwendet werden soll, bestimmt der Quadrantensequenzer über die von ihm generierte CCD-ID (CCD-Nummer) am Multiplexer, der den ELEVi-Registern nachgeschaltet ist (der „MUX", bzw. die 3 Latches unten links der Mitte in Abb. 3.19).

Das Ergebnis dieses Vergleichs und das oben erwähnte, mit zwischengespeicherte Bad-Pixel-Flag („BADPXF") bestimmen, ob das Pixel ein „Good Event" war. Wenn ja, dann wird das Pixel in dem „Event-Counter" gezählt (der mittlere der 3 untereinanderliegenden Zähler CNT rechts neben der Bildmitte von Abb. 3.19). In der Dokumentation oft mit „Above Threshold Counter" ATC bezeichnet, liefert dieser Zähler einen weiteren Kontrollwert. Damit er auch immer richtig zählt, wird der Good-Event-Pegel in einem Flip-Flop zwischengespeichert: Mit $\overline{\text{COMR}}\downarrow$ wird er übernommen, und $\overline{\text{COMW}}\uparrow$ setzt das Flip-Flop wieder auf 0.

Im Timing-Mode kann vor Erreichen des Pile-up-Limits die maximale Datenrate der pn-Kamera von 40 kbit/sec überschritten werden. Daher ist zur definierten Datenreduktion ein „elektronischer Chopper" eingebaut worden, der nur die Good-Events jedes k-ten Quadranten-Durchlaufs zur Übergabe an die CPU zuläßt. Dabei ist k in dem Latch vor dem unteren der bereits erwähnten 3 Zähler einstellbar: von CHOP = 0 (jeder) bis 31. Das Signal zum Zählen der Quadranten-Durchläufe wird wieder vom Quadrantensequenzer gegeben, indem er das für den Master-Sequenzer erzeugte Signal QSEQ_RDY (vergleiche Abschnitt 3.3.1) auch dem Common-Mode-Filter übermittelt.

Ein weiterer Vorteil dieses elektronischen Choppers ist die Möglichkeit für die CPU in den Modi ohne Integrationszeit (Abschnitt 3.1) die notwendige Zeit für die Berechnungen der Offset-Tabelle zu bekommen.

Neben den Pixel-Daten wird von den Wissenschaftlern zur Zeitanalyse auch eine Zeitinformation benötigt, die an jedem Quadranten-Ende zusätzlich zu den CCD-Daten übermittelt werden soll. Zu diesem Zweck sendet der Satellit an alle Instrumente jede Sekunde ein Signal, den 1 Hz-Puls, der einen 15-Bit-Sekunden-Zähler taktet (der untere der beiden übereinander liegenden Zähler in der rechten Bildmitte von Abb. 3.19). Daneben taktet die vom Master-Sequenzer gelieferte QCLK (siehe Abschnitt 3.3.1) zwei weitere, kaskadierte Zähler, die die 16 Bit Sekunden-Bruchteile ermitteln (die jeweils oberen der Zähler rechts der Mitte von Abb. 3.19). Dem Sekunden-Zähler-Ergebnis wird noch ein zusätzliches Bit mit Wert 1 als Erkennungsbit mitgegeben, so daß ein Zeitwort aus insgesamt 32 Bit besteht.

Die Meßdaten bilden ebenfalls ein 32-Bit-Wort: 12 bereinigte ADC-Bits mit je 2 Quadranten- und CCD-ID-Bits bilden die eine 16-Bit Gruppe (die Quadranten-IDs werden vom Steckplatz der jeweiligen SECOM-Karte in der EPEA-Box bestimmt), und die 14 Adress-Bits (8 Zeilen- und 6 Pixel-Bits) bilden zusammen mit 2 in diesem Fall auf 0 gesetzte Erkennungsbits die andere 16-Bit-Gruppe.

Nun sind alle Voraussetzungen gegeben, daß die Daten an die CPU zur weiteren Verarbeitung übergeben werden können (siehe Abschnitt 3.3.2): Die Daten sind korrigiert und ausgewählt, und die Zeitinformation steht auch zur Verfügung.

Die Schnittstelle zwischen Common-Mode-Filter und CPU bildet das CPU-FIFO (ganz rechts in Abb. 3.19). Allerdings müssen alle in ESA-Satelliten verwendeten Bauteile auch ESA-qualifiziert sein, und wegen des geplanten Orbits müssen für XMM zusätzlich strahlungsharte Teile ausgewählt werden. In diesem Bereich gibt es jedoch nur 8 Bit breite FIFOs mit diesen Qualifikationen, weswegen die insgesamt 64 Bit aufgeteilt werden müssen: Werden 2 dieser 8-Bit-FIFOs parallel geschaltet — bei mehr könnte die erforderliche Synchronität problematisch werden — dann sind es 4 Gruppen à 16 Bit, deren Zusammensetzung sich aus den letzten beiden Absätzen natürlich ergibt. Welche Bits an den FIFOs zur Übernahme anliegen, bestimmt wieder ein Multiplexer (der große MUX links in Abb. 3.19), der mittels der Signale SELM 0 bis SELM 3 (Select Multiplexer) vom Quadrantensequenzer gesteuert wird: Bei den Meßdaten wird zuerst der CCD-Wert (ENGID) und danach die Adresse (PXADF) durchgelassen; beim Zeitwort kommen zuerst die Bruchteile vor den ganzen Sekunden.

Die Kontrolle des CPU-FIFOs übernimmt die FIFO-Kontroll-Logik („FIFO CTRL", unten rechts in Abbildung 3.19). Sie hat mehrere Aufgaben: Zum einen verwaltet sie etwaige CPU-Anforderungen, das FIFO zu löschen ($\overline{\text{CLRFIF}}$). Dieses „Clear FIFO" benötigt die CPU bei der Erstellung der in Abschnitt 3.3.2 genannten Tabellen, und zwar am Anfang eines CCD-Zyklus, also mit $\overline{\text{NEWCHIP}}$. Zum anderen stellt sie sicher, daß nicht gleichzeitig von der CPU gelesen und vom Common-Mode-Filter geschrieben wird, und sie erzeugt die eigentlichen Schreib- und Lese-Pulse.

Letzteres geschieht zum einen unter Berücksichtigung des Füllstandes, der anhand von Status-Flags vom FIFO gemeldet wird — aus einem leeren FIFO kann schließlich nichts mehr abgeholt, bzw. in ein volles FIFO nichts mehr hineingestellt werden — zum anderen werden sie aus den Sequenzer-Signalen $\overline{\text{WFENRG}}$ (Write FIFO Energy) und $\overline{\text{WFTIM}}$ (Write FIFO Time) unter Hinzuziehen der Good-Event- und Chopper-Information generiert. Dabei gelten für die Sequenzer-Signale $\overline{\text{WF}xyz}$ die gleichen strengen Randbedingungen wie für den eigentlichen $\overline{\text{WFIF}}$-Puls, da die Kontroll-Logik sie einfach passieren läßt, wenn geschrieben werden darf. Abbildung 3.21 illustriert, wie der zu einem Pixel gehören-

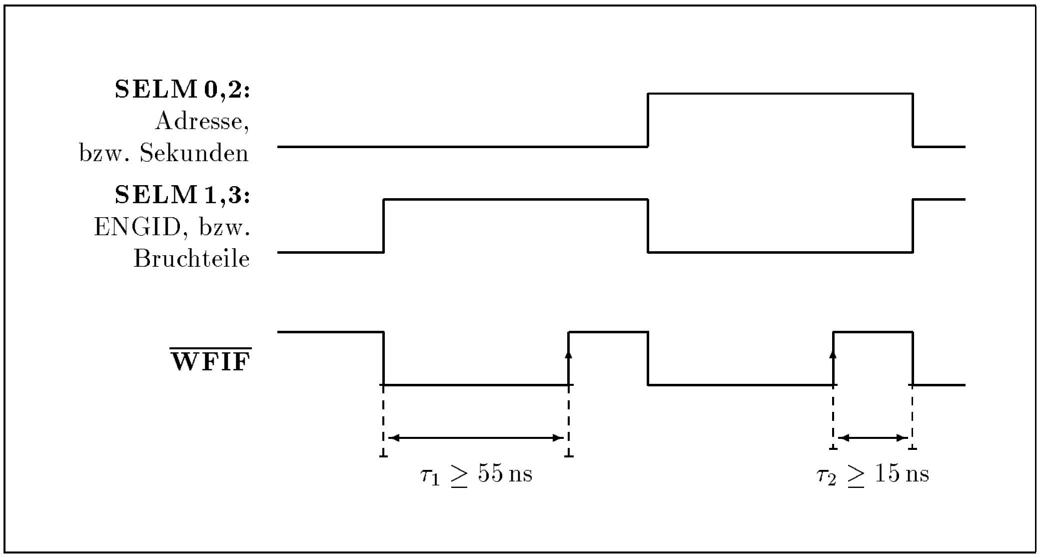


Abbildung 3.21: *Die Anforderungen an den FIFO-Schreib-Puls $\overline{WFIF}$. Neben den Mindestabständen zwischen den Flanken, muß die aktive Schreibflanke $\overline{WFIF}\uparrow$ sauber unter den SELM i-Signalen liegen.*

de Doppelpuls unter den SELM-Signalen liegen muß und welche Zeitabstände zwischen seinen Flanken eingehalten werden müssen.

In der genannten Abbildung ist allerdings nicht enthalten, daß für ein Pixel nur eine Gesamt-Filterzeit von ca 300 ns zur Verfügung steht. In dieser Zeit müssen mindestens 6 Signalwechsel erfolgen: Angefangen mit $\overline{\text{COMW}}$, der einen Pixelwert aus dem Line-FIFO holt und das Good-Event-Flip-Flop rücksetzt, über $\overline{\text{COMR}}$, der die zugehörige Ortsinformation generiert und der FIFO-Kontroll-Logik Good-Event zugänglich macht, bis zu $\overline{\text{WFENRG}}$ mit SELM 1 und SELM 0, die Energie und Adresse an die CPU-Schnittstelle übergeben.

Diese Anforderung, daß der Quadrantensequenzer spätestens alle 50 ns einen neuen Zustand liefern muß, ist vielleicht von allen Anforderungen diejenige gewesen, die sein „Aussehen“ am meisten bestimmt hat. Doch darauf soll in Kapitel 5 eingegangen werden, wo Abbildung 5.1 (a) auch einen zeitlichen Überblick über alle pixelrelevanten COM-Signale gibt.

3.3.4.2 Der Quadrantensequenzer in grober Struktur

Eine einfache Möglichkeit, einen festen Ablauf zu steuern, ist mit einem RAM. In diesem RAM ist für jedes Steuersignal ein Bit vorgesehen, und die Ablauf-Steuerung ergibt sich, indem das RAM Adresse für Adresse ausgelesen wird. Voraussetzung dafür ist natürlich, daß das RAM seinen Inhalt schnell genug Preis gibt, die Zugriffszeit also kurz genug ist. Im Fall des Quadrantensequenzer darf sie maximal 50 ns betragen, und dies ohne Sicherheits-Marge, wie aus dem vorherigen Absatz deutlich wird.

Wie bei den CPU-FIFOs des Common-Mode-Filters tauchen hier Schwierigkeiten auf: Es gibt keine strahlungsharten RAMs, die schnell genug wären die für die Verarbeitung eines Pixels notwendigen Signale auszugeben. Damit ist die kleinste Ausleseeinheit, die mit einem RAM gesteuert werden kann, eine Zeile. Alle Signale, die innerhalb einer Zeile

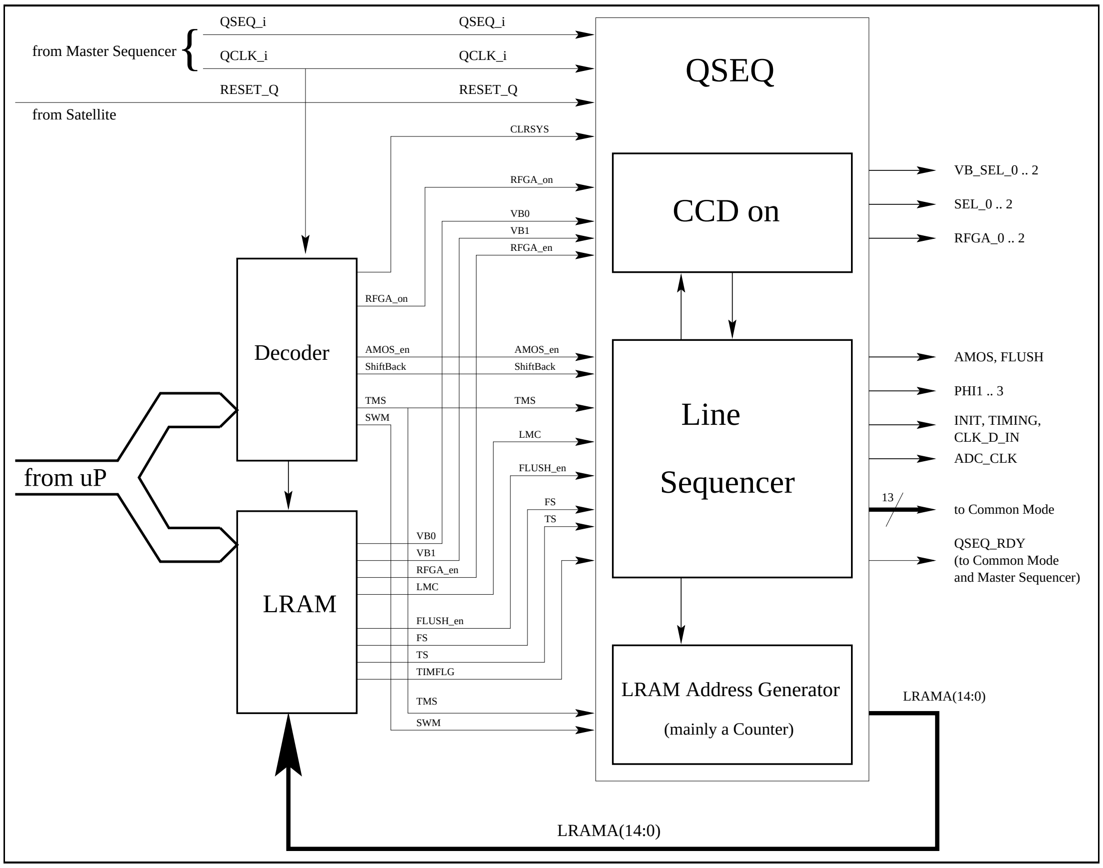


Abbildung 3.22: *Eine grobe Quadrantensequenzer-Übersicht*

mehrfach wechseln müssen, müssen durch eine diskrete Schaltung, einem „Line-Sequenzer“, erzeugt werden. Dahingegen kann jede Steuerung, die eine ganze Zeile betrifft, in einem RAM gespeichert werden, dem „Line-RAM“ oder „LRAM“, das von der CPU programmiert wird, und aus dem sich der Sequenzer während der Messungen die Steuerbits abholt.

Insgesamt hat der Quadrantensequenzer also eine Struktur, wie sie in Abbildung 3.22 dargestellt ist:

- Vom Master-Sequenzer wird er angeschaltet und bekommt seinen Takt — aus Gründen, wie sie in Zusammenhang mit Abbildung 5.1 (a) erläutert werden, soll die QCLK eine Frequenz von 25 MHz liefern, was einer Taktlänge von 40 ns entspricht;
- Er bekommt die Quadranten-übergreifende Steuerung von der CPU über einen Decoder, der im nächsten Abschnitt 3.3.4.3 bescheiben wird;
- Vom LRAM holt er sich die zeilenübergreifenden Steuerbits;
- Damit es immer die richtigen LRAM-Bits sind, generiert er sich während der Messungen die LRAM-Adresse selbst;
- Er hat einen Schaltungsteil, der das richtige Einschalten der CCDs übernimmt

  und
- Er hat einen Schaltungsteil, der die Signale für die Verarbeitung einer Zeile generiert.

3.3.4.3 Der Decoder

Der Decoder ist sowohl in Abbildung 3.19 des Common-Mode-Filter als auch in Abbildung 3.22 des QSEQ auf der linken Seite angedeutet. Er bildet die Schnittstelle zwischen CPU und SECOM: Von der CPU sind neben 17 Adress- und 16 (bidirektionalen) Daten-Leitungen auch 7 Kontroll-Signale an die SECOM-Schaltung geführt (siehe Tabelle 3.3). Über diese 40 Leitungen geht die gesamte Kommunikation zwischen der CPU einerseits, und HBR, Quadrantensequenzer und Common-Mode-Filter inklusive der zugehörigen RAMs andererseits.

Signal-Name	**Bezeichnung**	**Kurz-Erläuterung**
XA(16:0)	Adressbus	die 17 Adress-Bits der CPU
XD(15:0)	Datenbus (bidirektional)	die 16 Daten-Bits der CPU
$\overline{\text{XRD}}$	Read	die Lese-Anforderung der CPU
$\overline{\text{XWR}}$	Write	die Schreib-Anforderung der CPU
XM/$\overline{\text{IO}}$	Memory or I/O	das vorbereitende Signal für die durch die vorherigen beiden Signale angesprochene Peripherie
$\overline{\text{XMEM}}$	Memory	das enable-Signal für die Speicher-Bausteine
$\overline{\text{XDEN}}$	Data Enable	das Signal, das erklärt, wann gültige Daten auf einem Bus liegen
XDT$\overline{\text{R}}$	Data Transmit or Receive	das die Daten-Richtung anzeigt
$\overline{\text{XBHE}}$	Bit High Enable	das die Interpretation der CPU-Adressen mitbestimmt
CCD 1	CCD 1	das höherwertige Bit der CCD-ID gibt an, ob CCD 2 oder eines der beiden anderen aktiv ist

Tabelle 3.3: *Die Kontroll-Signale für den Decoder. Bis auf CCD 1, das vom QSEQ kommt, werden alle Signale von der CPU geliefert, was durch das* X *am Anfang des Signal-Namens gekennzeichnet ist.*

In der FPGA-Version gibt es zwei verschiedene Decoder, den „COM_DECODE“, der den für den Common-Mode-Filter relevanten Teil beherbergt, und den „SEQ_DECODE“, der die grundsätzliche QSEQ- und LRAM-Ansteuerung übernimmt. Dies ist aus Platz-Gründen geschehen, da der Quadrantensequenzer vom Common-Mode-Filter getrennt in einem eigenen FPGA untergebracht ist. In der ASIC-Version sind beide Teil-Decoder zu einem Gesamt-Decoder zusammengefaßt.

Abgesehen von der Unterteilung in COM_DECODE und SEQ_DECODE, gibt es eine weitere Aufteilungs-Möglichkeit: Der Teil, der die Line- und Offset-RAMs betrifft, also den Speicherbereich, und der Teil, der die Basiseinstellungen für Common-Mode-Filter und Sequenzer vornimmt, die sogenannten „Ports“ oder CPU-Befehle. Unterschieden wird beides durch die Adresse, die auf den CPU-Adress-Leitungen XA(16:0) anliegt: Die RAMs haben den Bereich 60000 bis 7ffff, wobei 60000 bis 77fff dem Offset und 78000 bis 7ffff dem

Line-RAM zur Verfügung steht; die Befehle belegen den Adressraum von E000 bis E01f für den COM, bzw. E100 bis E13f für den QSEQ. Alle Angaben sind hexadezimal.

Bei der Interpretation dieser Adressen ergibt sich eine Schwierigkeit: der Intel® 80c86 verwendet das unterste Adressbit A0 zur Unterscheidung paralleler Speicherbausteine („Odd“ und „Even“). Auf diese Weise ist ein schnellerer Speicherzugriff realisierbar. Da von dieser Möglichkeit für die SECOM-Karte jedoch kein Gebrauch gemacht wurde, gibt es nur jede zweite Adresse aus der Sicht der CPU; COM und QSEQ steuern „ihre“ RAMs hingegen fortlaufend an. Außerdem hat die CPU trotzdem noch ein Bit mehr zur Verfügung, als die RAMs verkraften. Es wird daher XA(15:1) der CPU auf RAMADR(14:0) der RAMs abgebildet. Diese Adress-Verschiebung muß bei der Programmierung von Line- und Offset-RAM berücksichtigt werden.

Neben der Festlegung, in welcher Richtung der bidirektionale Datenbus zwischen CPU und SECOM-Karte genutzt werden soll, besteht eine Hauptaufgabe des Decoders darin, festzulegen, wer die „Hoheit“ über die RAMs hat, SECOM oder CPU, und — daraus abgeleitet — das richtige Anlegen der Schreib-, Lese- und Chip-Select-Pulse. Hier gilt schließlich ähnliches wie bei dem CPU-FIFO aus Abschnitt 3.3.4.1: Es kann zum einen nicht gleichzeitig gelesen und geschrieben werden, zum anderen muß auch eindeutig sein, welche Adresse gemeint ist, diejenige, die die CPU anlegen würde, oder diejenige des Common-Mode-Filters bzw. Quadrantensequenzers. Dies geschieht mittels der Signale $\overline{\text{COMONC}}$ bzw. $\overline{\text{COMONS}}$ („Common-Mode-On“ für COM oder SEQ), die die CPU als Befehl einschalten kann: Sie steuern jeweils einen Multiplexer (Erklärung auf S. 24), der die CPU- oder SECOM-Adressen an „sein“ RAM weitergibt; Vorrang hat immer die CPU, die mit jedem Lese- oder Schreib-Zugriff auf den RAM-Adress-Bereich die Hoheit zurückerhält.

Was die erwähnten Befehle betrifft, so dienen sie beim Sequenzer dazu, eine Grund-Einstellung vorzunehmen: Die Einstellung des Betriebs- oder Test-Modus, sowie der Parameter, die aufgrund der Messungen aus Kapitel 4 eingeführt wurden. Außerdem lassen sich einige eingestellte Werte mittels solcher Befehle kontrollieren (eine Übersicht über die Sequenzer-CPU-Befehle findet sich im Anhang auf Seite 91).

Für die Common-Mode-CPU-Befehle gilt Änliches: Mit ihnen werden die in Abschnitt 3.3.4.1 erwähnten Schwellen, der Korrekturwert und der Chopper-Wert gesetzt und zur Kontrolle ausgelesen, aber auch ein Clear-FIFO angemeldet oder der Above-Threshold-Counter rückgesetzt und das High-Bitrate-Interface gesteuert.

3.4 Die Datenfluß-Pipeline

Um einen Großteil der Eigenheiten des Quadrantensequenzers zu verstehen, ist es wichtig die sogenannte Datenfluß-Pipeline des EPIC-pn-CCD-Kamera-Systems zu kennen, wie sie in Abbildung 3.23 (S. 45) dargestellt ist.

Eine Pipeline ist der elektronische Fachbegriff für eine Art Schlange, in der — ähnlich einem FIFO — Daten oder Befehle „anstehen“ um verarbeitet bzw. ausgeführt zu werden. Einzelne Einträge können in der Regel nicht aus dieser Schlange herausgenommen werden, ist ein falscher darunter, kann nur die gesamte Pipeline gelöscht werden. Trotzdem sind sie in der Datenverarbeitung sehr sinnvoll, da der Prozessor seltener auf die zeitraubenden Speicherzugriffe warten muß und somit Programmabläufe beschleunigt werden.

Im Prinzip wurde bereits (fast) jede Stufe der Datenfluß-Pipeline in den vorherigen Abschnitten erwähnt und erklärt, trotzdem sollen sie hier noch einmal vollständig vorgestellt werden:

Die erste Stufe entsteht dadurch, daß die Photonen zunächst in den CCD-Pixeln als Elektronen gespeichert und zeilenweise in den CAMEX übernommen werden (Abschnitt 3.2.1.1). Während die Analog-Daten der ersten Zeile (Zeile 0) aus dem Sample-and-Hold-Register Pixel für Pixel in den ADC geholt (Abschnitt 3.3.3) und in den Sortierer geschoben werden (Abschnitt 3.3.4.1), wird Zeile 1 im CAMEX verstärkt; die zweite Stufe der Datenfluß-Pipeline ist erreicht. In der dritten Stufe wird Zeile 0 aus dem Line-FIFO geholt, Median-korrigiert, gefiltert und in das CPU-FIFO geschrieben, aus dem die CPU die Daten beliebig abholen kann.

Damit in den verschiedenen Stufen nur die jeweils validen Daten behandelt werden, darf der Sequenzer während des Pipeline-Aufbaus bestimmte Steuersignale (COMR_, COMW_, Write-FIFO-Pulse) nur sukzessive zuschalten. Wenn dann alle Stufen mit validen Daten arbeiten, kann die Schaltung theoretisch beliebig viele Zeilen sukzessive bewältigen.

Allerdings sollte diese Zeilen-Pipeline am Ende eines CCDs auch definiert „abgebaut" werden. Denn wenn die letzte valide Zeile in Stufe 3 ist, sind natürlich irgendwelche „Müll-Daten" im Sortierer und ein „Müll-Median" wird bestimmt, der nicht in den Median-Akku übernommen werden soll. Außerdem sollte auch der Sequenzer wissen, wann ein CCD zu Ende ist und das nächste eingeschaltet werden kann. Daher besteht die Pipeline de facto aus 4 offensichtlich zeilenorientierten Stufen. Die einzelnen „Schlangen-Plätze" werden L 0 (Line 0), L 1 (Line 1), NL (Normal Line) und LL (Last Line) genannt; welche Signale konkret welche Pipeline-Stufe charakterisieren, zeigt Tabelle 3.4:

LMC	**Zeile**	**Abk.**	**Kennzeichnende Signale**
0	line 0	L0	CAMEX- und TIMEX-Signale
1	line 1	L1	COMR_, COMW_ für Sortierer, MEDCLK_ zur Median-Übernahme
2	normal line	NL	Write-FIFO-Pulse
3	last line	LL	internes EOCCD, evtl. QSEQ_RDY, kein MEDCLK_

Tabelle 3.4: *Charakteristische Signale jeder Zeilen-Pipeline-Stufe.*

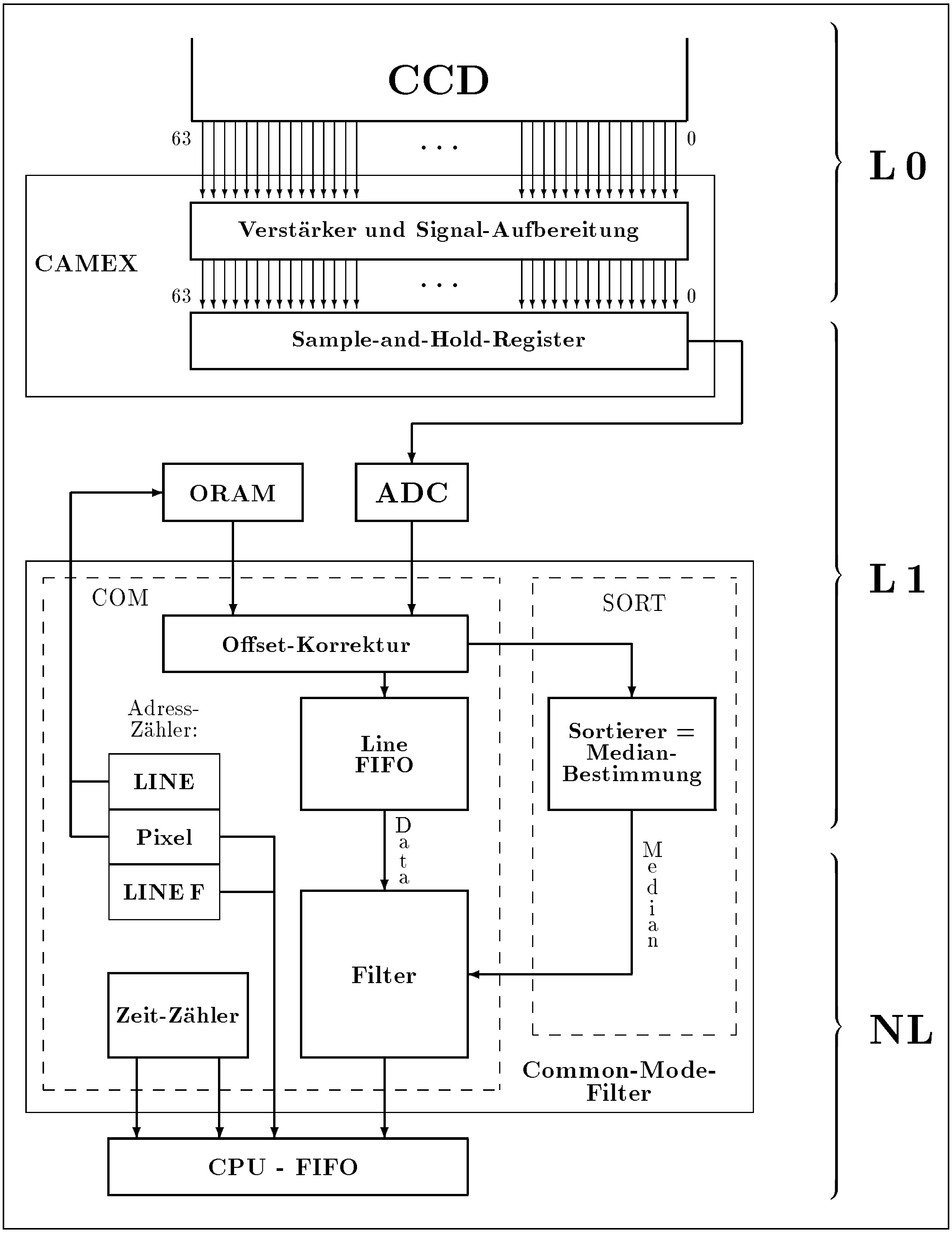


Abbildung 3.23: *Der Datenfluß der Daten der EPIC-pn-CCD-Kamera und die dadurch entstehende Pipeline.*

Kapitel 4

Messungen an der ZETA-Anlage und Auswirkungen auf den QSEQ

Nachdem ein Quadrantensequenzer entwickelt worden war, der allen in Kapitel 3 genannten Anforderungen genügte, und der zusammen mit den anderen Komponenten im Tübinger Labor auch korrekt funktionierte, wurden einige Messungen an der ZETA-Anlage des MPE in Garching gemacht. Dort ist der Kamerakopf in seinem jeweiligen Entwicklungs-Stadium installiert, so daß das Zusammenspiel zwischen EPEA und EPCH sinnvoll getestet werden kann.

Im Herbst 1996, als die Messungen gemacht wurden, war an der ZETA-Anlage ein Quadrant vollständig angeschlossen und betriebsbereit. Dies ermöglichte den Test eines Quadranten-Ablaufs, der in Tübingen in dieser Art nicht möglich war, da dort nur ein einziges CCD zur Verfügung steht; sollen am IAAT CCD-Wechsel getestet werden, so müssen sie mittels ODER-Gattern vorgetäuscht werden. Außerdem waren in Tübingen durch den anderen Aufbau auch andere Signal-Laufzeiten als in Garching wahrscheinlich. Bei den Messungen an der ZETA-Anlage wurde daher besonders auf die CCD-Wechsel und den zeitlichen Verlauf von Signalen zwischen EPEA und EPCH geachtet.

Zu diesem Zweck war sowohl am Ausgang der EPEA als auch nach der „Kink'schen Quadranten-Box", die aus den digitalen Steuersignalen des Quadrantensequenzers analoge Signale für die CCD-Platine generiert, jeweils eine sogenannte Breakout-Box installiert. Wird eine solche Box in den Signalweg gelegt, so ist es möglich, Signale direkt abzugreifen, sie unverändert durchzuschleifen oder auch verändert neu einzuspeisen. Auf diese Weise werden Systemtest, aber auch eine etwaige Fehlersuche sowie das Testen möglicher Lösungen deutlich erleichtert. Durch die zwei Breakout-Boxen konnten sowohl die digitalen Steuerpulse als auch die analogen Signale und Spannungen gleichzeitig untersucht werden.

Im folgenden soll auf die Messungen eingegangen werden, die unerwartete Ergebnisse gezeigt haben — Ergebnisse, die zu zusätzlichen Anforderungen und damit zu Änderungen im QSEQ geführt haben.

4.1 Die Phi-Pulse und S_{in}

Wie in Abschnitt 3.2.1.1 ausführlich erklärt, müssen die Schiebepulse vollständig unter dem S_{in}-Puls des Timing-Signals liegen. Dies gilt vor allem für die tatsächlich schiebenden Analog-Pulse, ist aber auch für die digitalen Steuerpulse wünschenswert, da Signal-Laufzeiten mit sinkender Temperatur eher kürzer werden und die Steuerpulse auch nicht zu früh kommen dürfen.

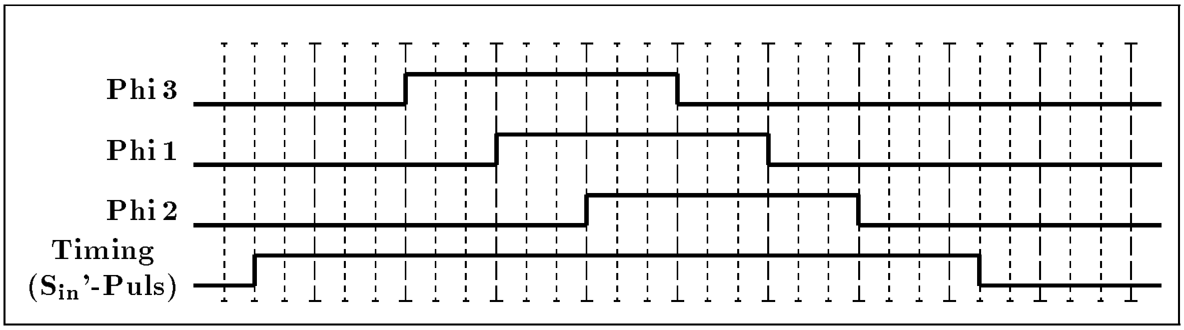
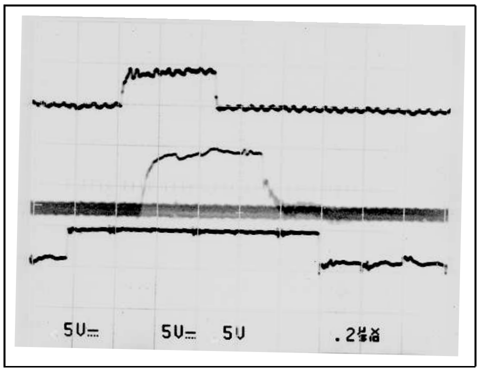
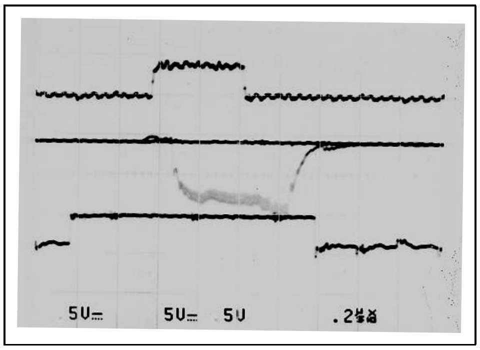


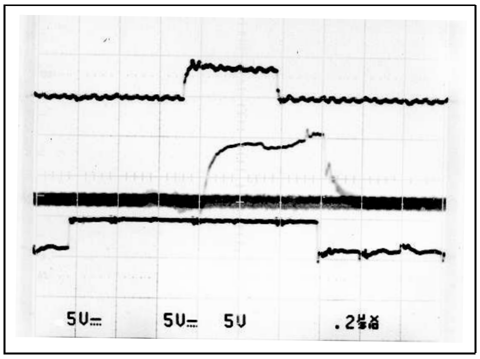
Abbildung 4.1: *Die alte Verteilung der Phi-Pulse unter dem S_{in}-Puls.*



(a) *Phi 3*



(b) *Phi 1*



(c) *Phi 2*

Abbildung 4.2: *Die drei Phi-Pulse unter dem S_{in}-Puls wie sie ursprünglich realisiert waren. Die Oszillographen-Aufnahmen der Messungen zeigen jeweils den digitalen Steuerpuls, den analogen Schiebepuls und den S_{in}-Puls des Timing-Signals. In allen drei Aufnahmen ist die Signal-Laufzeit von ca 240 ns deutlich zu erkennen, die dazu führt, daß vor allem Phi 2 die Forderung verletzt, vollständig unter dem Timing-Puls zu liegen.*

In Tübingen war vom Anlegen des digitalen Steuerpulses bis zum Schalten der analogen Schiebeschalter eine Zeitverzögerung von 120 ns gemessen worden. In QCLK-Taktzyklen ausgedrückt sind dies 4 Takte. Damit die Phi-Pulse also genau unter dem Timing-S_{in}-Puls liegen, sind mindestens 19 Takte für den S_{in}-Puls notwendig, wie in Abbildung 4.1 unschwer abzuzählen ist. Damit die Schalter keinesfalls zu früh schalten, waren für die Pulsdauer von S_{in} zusätzliche 5 Takte eingebaut, bis der erste Phi-Schalter gesteuert wurde. Damit ergibt sich ein Ablauf, wie in Abbildung 4.1 dargestellt.

Wie in Abbildung 4.2 deutlich zu sehen ist, wurden in Garching jedoch Signal-Laufzeiten von ca 240 ns gemessen, also gut doppelt so lang wie in Tübingen. Dadurch liegt der analoge Phi 2-Puls nicht mehr vollständig unter dem S_{in}-Puls, was zu unsauberen Daten führen kann.

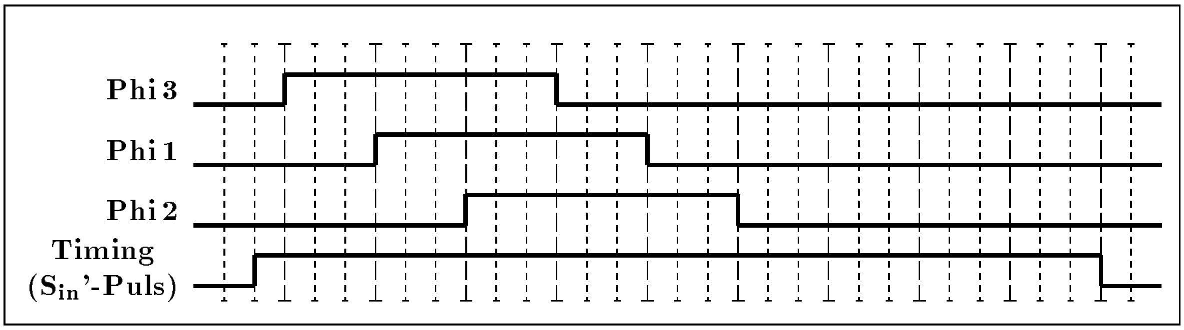
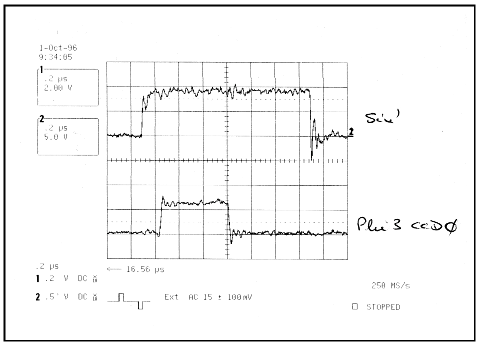
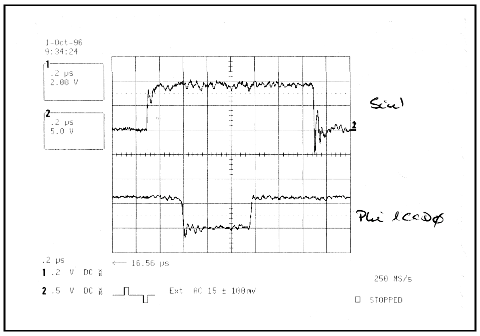


Abbildung 4.3: *Die aktuelle Verteilung der Phi-Pulse unter dem S_{in}-Puls.*

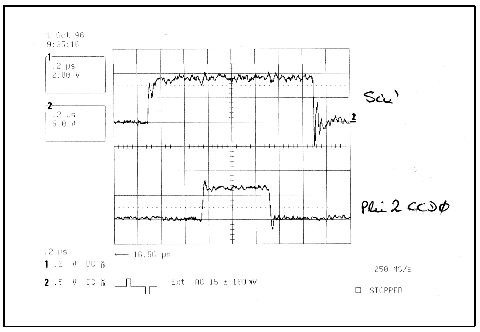
Daher mußte S_{in} von den bisherigen 24 auf 28 Takte und die Dauer einer Zeile damit von 568 auf 576 Takte verlängert werden. Außerdem mußten die Phi-Pulse unter dem Timing-Puls verschoben werden: Statt der bisherigen 5 Takte „vordere“ Sicherheit ist es jetzt nur noch einer, und anstelle der 4 Takte „hintere“ Sicherheit, geht S_{in} nun erst 12 Takte nach Phi 2 wieder auf 0 (siehe Abbildung 4.3). Die Messungen aus Abbildung 4.4 haben bestätigt, daß damit ausreichend Zeit für das Schalten der Analog-Pulse vorhanden ist.



(a) *Phi 3*



(b) *Phi 1*

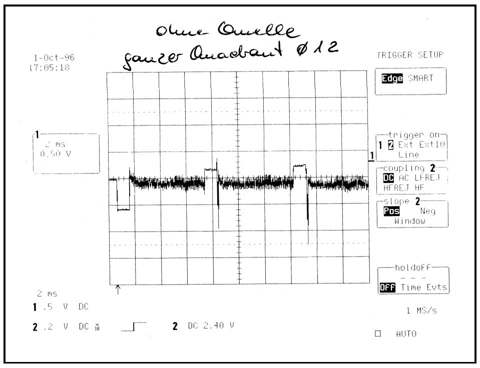


(c) *Phi 2*

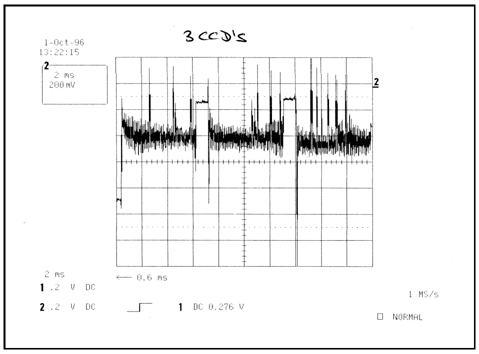
Abbildung 4.4: *Die drei Phi-Pulse unter dem S_{in}-Puls wie sie derzeit realisiert sind. Die Oszillographen-Aufnahmen der Messungen zeigen jeweils den analogen Schiebepuls und den S_{in}-Puls des Timing-Signals. Gut zu erkennen ist, daß jetzt alle Phi-Pulse deutlich unter dem S_{in}-Puls liegen.*

4.2 Das CCD-Einschaltverhalten

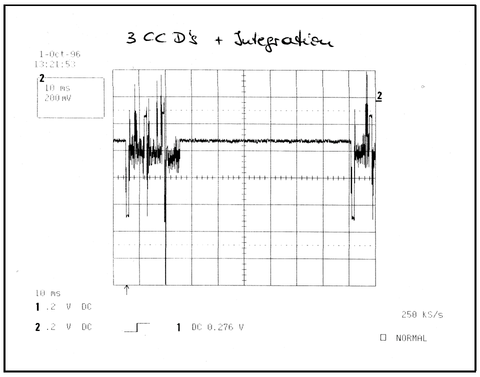
Eigentlich ist der Ausdruck „Einschalten der CCDs“ etwas unsauber; genau genommen werden die CAMEXe und nicht die CCDs aktiviert, wie Abschnitt 3.2.3 erläutert. Da sich jedoch im Sprachgebrauch das „Einschalten der CCDs“ eingebürgert hat, soll dies auch hier verwendet werden.



(a)



(b)



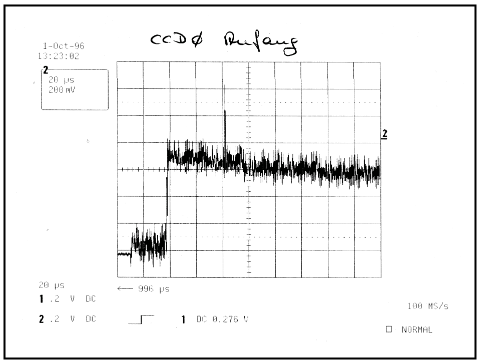
(c)

Abbildung 4.5: *Der CAMEX-Ausgang im Fullframe-Mode liefert einen guten Überblick über das Einschaltverhalten der 3 CCDs. Deutlich zu sehen sind die „Events“ der Eisen-Messung (Die Ausreißer nach oben in den Messungen 4.5 (b) und 4.5 (c)), aber auch das Einbrechen der Daten beim Einschalten des nächsten CCDs.*

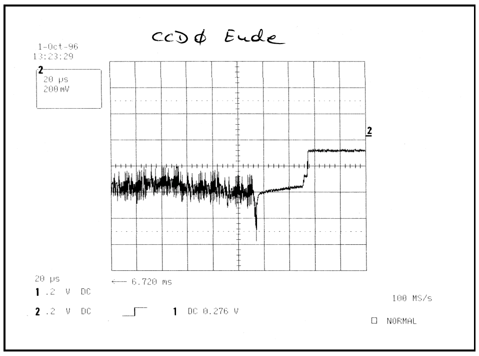
Bereits bei den Messungen zu den Phi-Pulsen waren einige Unregelmäßigkeiten beim sukzessiven Einschalten der einzelnen CCDs aufgefallen. Die Messungen wurden im Fullframe-Mode (siehe Seite 14) gemacht, so daß immer nach 3 CCDs die Pause für die Integrationszeit zu sehen war (Abbildung 4.5 (c); es wurde das analoge CAMEX-Ausgangs-Signal auf dem Oszilloskop betrachtet).

Auffallend war, daß die Daten bei jedem CCD-Wechsel große Sprünge mit teilweise exponentieller Normalisierung zeigten (Abbildung 4.5 (a), bzw. 4.6):

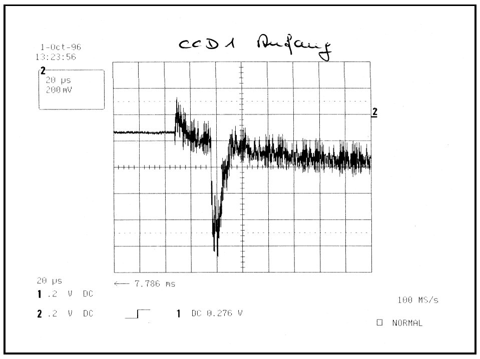
Der Effekt, daß das erste CCD erst nach Ablauf der ersten Zeile den korrekten Basiswert liefert (Abbildung 4.6 (a)), kann der Tatsache zugeschrieben werden, daß es sich dabei um die Daten aus dem Sample-and-Hold-Register handelt, dessen speichernder Kondensator sich während der Integrationszeit entladen haben könnte. Dies ist nicht weiter tragisch, da diese Zeile ja wegen der Pipeline sowieso nicht verarbeitet wird (siehe Abschnitt 3.4, Line 0).



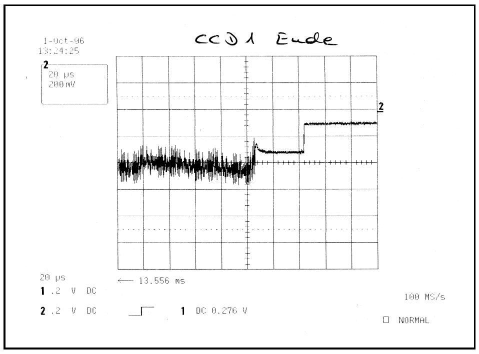
(a) *CCD 0-Anfang*



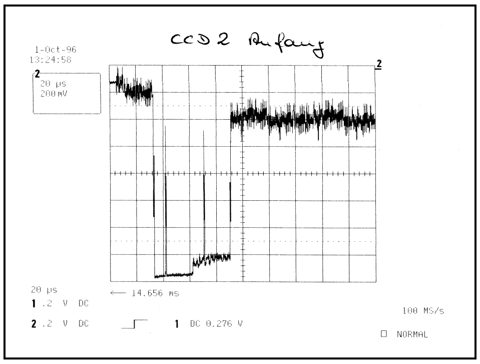
(b) *CCD 0-Ende*



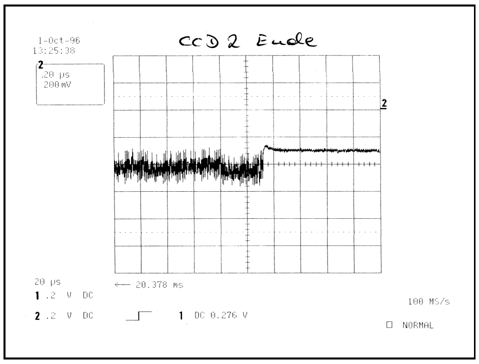
(c) *CCD 1-Anfang*



(d) *CCD 1-Ende*



(e) *CCD 2-Anfang*



(f) *CCD 2-Ende*

Abbildung 4.6: *Anfang und Ende jedes CCDs im Fullframe-Mode bei überlappendem Aktivieren der CAMEXe.*

Das doppelt-exponentielle Einpendeln beim Einschalten des mittleren CCDs (Abbildung 4.6 (c)) deutet auf ein nicht beendetes CAMEX-Einschwingen hin; offensichtlich ist sowohl das SEL_OUT als auch das erste INIT, das den Anfang der Auslesesequenz markiert, zu früh gegeben worden. Letzteres ist in Ansätzen auch bei den beiden anderen CCDs zu sehen, was obige Annahme untermauert und ein Verzögern von INIT nahelegt. Eine Pause zwischen SEL_OUT↑ und INIT bedeutet allerdings, daß das lückenlose Auslesen der CCDs während eines Quadranten nicht möglich ist.

Die Effekte beim Einschalten des letzten CCDs hingegen (Abbildung 4.6 (e)) konnte man sich nicht erklären.

Was das Ausschalten der einzelnen CCDs betrifft, so ist das letzte CCD das einzige, das den Erwartungen entspricht und sofort in „Ruheposition“ geht (Abbildung 4.6 (f)). Beide anderen CCDs zeigen Ausreißer mit exponentiellen Flanken, die zeitlich gesehen mit dem Vorwärmen des nächsten CCDs zusammenfallen. Dies legt ein Verbot sich überlappender VB_SEL-Signale nahe, das jedoch ein kontinuierliches Auslesen eines Quadranten erst recht unmöglich macht.

Zur Lösung dieses Problems wurde für jede weitere QSEQ-Variante ein CPU-Kommando (VB_EOCCD) eingeführt, das das Schalten von VB_SEL wahlweise während der Auslesezeit des vorherigen CCDs oder erst danach ermöglicht, wie in Abbildung 4.7 dargestellt. Auf diese Weise ist das pausenlose Auslesen eines Quadranten nicht grundsätzlich unterbunden.

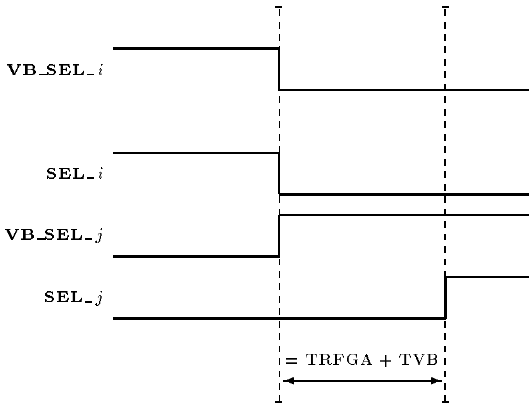


Abbildung 4.7: *Das Einschaltverhalten bei eingeschaltetem VB_EOCCD; das Verhalten bei ausgeschaltetem VB_EOCCD ist in Abbildung 3.12 dargestellt.*

Während der sich an die Messungen von Abbildung 4.6 anschließenden Testzeit sind einige Quadrantensequenzer entstanden. Sie wurden jeweils mit unterschiedlichen Kombinationen der in Abbildung 4.8 dargestellten Verzögerungszeiten entworfen: Es gab Versionen, in denen die Vorwärmzeit vollständig als TVB abgewartet wurde, und Versionen, in denen sie als TSEQRES eingestellt war (da RFGA am Kamerakopf nicht angeschlossen war, spielte TRFGA bei den Messungen keine Rolle); beide Varianten wurden mit Vorwärmzeiten von 0,45, 0,8 oder 1,2 ms (bei 40 ns Taktzyklus, bzw. 0,57, 1,0 oder 2,0 ms bei 20 MHz) kombiniert. Trotzdem war nach einigen Wochen keine wirklich gute Lösung gefunden. Um die Herstellung des ASICs nicht noch länger zu verzögern, wurde beschlossen, diese Vorwärmzeiten zu „parametrisieren“: Es wurden weitere CPU-Kommandos eingeführt, die es erlauben diese Zeiten als einstellbare Parameter zu programmieren. Zusammen mit VB_EOCCD kann das Einschaltverhalten somit noch im Flug optimiert werden.

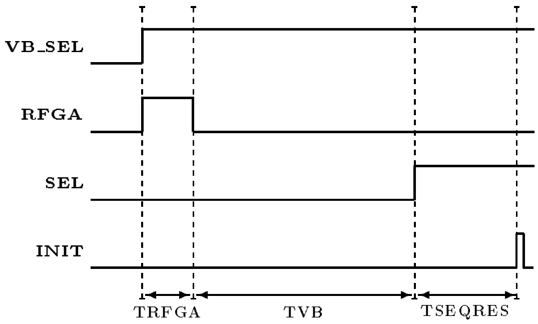


Abbildung 4.8: *Die Verzögerungsparameter.*

4.3 CLK_D_IN und ADC_CLK

Bereits in Abschnitt 3.3.3 wurde kurz auf die Auswirkungen eingegangen, die ein falscher Übernahmezeitpunkt der Analogdaten in den ADC hat. Damit dies nicht geschieht, müssen der Decoder-Clock CLK_D_IN, der bekanntlich die Analogdaten anfordert, und der ADC-Takt ADC_CLK, der sie in den ADC übernimmt, richtig zusammenarbeiten.

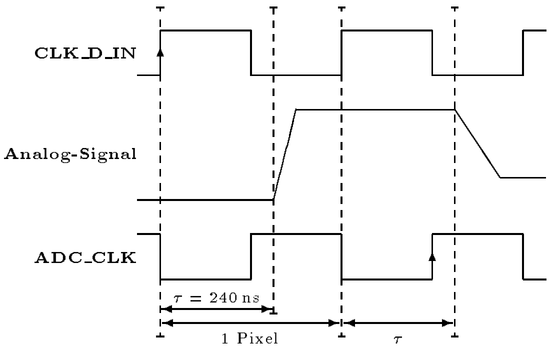


Abbildung 4.9: *Schematische Darstellung der analogen Signal-Laufzeit von der Anforderung bis zur Übernahme.*

Dabei müssen natürlich die Signal-Laufzeiten berücksichtigt werden, die sich in Garching wieder als deutlich länger als in Tübingen herausgestellt haben: Abbildung 4.10 (a) zeigt wieder 240 ns Verzögerung statt der erwarteten 120 ns. Nachdem die Daten erst im letzten Drittel in den ADC übernommen werden sollen, also am Besten einen Takt bevor sich der nächste Wert einstellt, sind zwischen der Anforderung und der Übernahme Zeiten von 500 bis 600 ns abzuwarten, je nach QCLK-Frequenz (ein Pixel + die 240 ns abzüglich eines Taktes). Die Messungen aus Abbildung 4.10 sind bereits mit einer verbesserten QSEQ-Version gemacht worden, in der dieser Umstand bereits berücksichtigt war (das ursprüngliche Verhältnis ist in Abbildung 3.17 dargestellt).

Allerdings wurde bei Messungen unter anderen Umgebungs-Bedingungen festgestellt, daß sich die Signal-Laufzeit vor allem mit der Temperatur ändert. Um während des Fluges nicht feststellen zu müssen, daß die Laufzeit-Abschätzungen falsch waren, wurde auch der Übernahmezeitpunkt parametrisiert: Unter der Voraussetzung, daß die Signal-Laufzeit nicht unter 160 ns fällt, war es möglich CLK_D_IN in 4 Stufen verschiebbar zu machen (Abbildung 4.11). Bei Signal-Laufzeiten von unter 160 ns wäre nicht mehr sichergestellt, daß auch die richtige aktive ADC_CLK-Flanke die Daten übernimmt, und daß sie dies nicht während der Einschwing-Zeit des Analog-Signals tut. Auch dieser Parameter wird über CPU-Kommandos eingestellt, und ermöglicht damit eine weitere Optimierung im Flug.

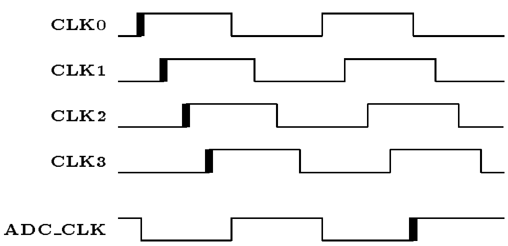
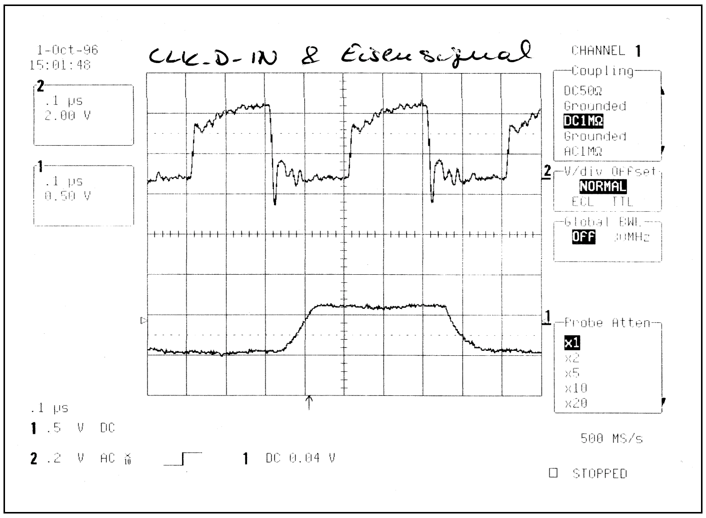
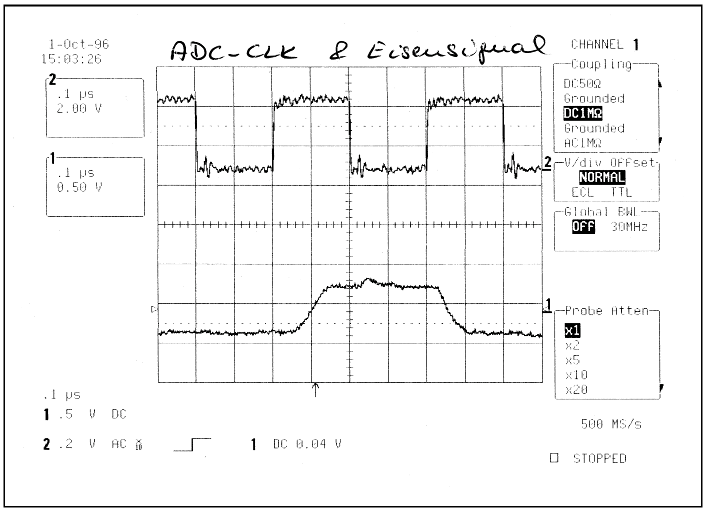


Abbildung 4.11: *Die verschiedenen Einstellungen von CLK_D_IN relativ zum ADC_CLK*



(a) *CLK_D_IN und das analoge Signal eines Eisen-Events*



(b) *Der ADC_CLK und das analoge Signal eines Eisen-Events*

Abbildung 4.10: *Anforderung (a) eines analogen Daten-Pixels (hier ein Event aus einer Eisen-Quelle) und Übernahme (b) in den ADC. Bei der Anforderung durch CLK_D_IN$\uparrow$ ist deutlich die Signal-Laufzeit von 240 ns zu sehen, was eine Übernahme durch ADC_CLK$\uparrow$ erst ca. 550 ns nach der Anforderung erlaubt.*

Kapitel 5

Die Realisierung des Quadrantensequenzers

Am Anfang einer Schaltungs-Entwicklung steht immer die sogenannte Spezifikation, die die Anforderungen, die an diese Schaltung gestellt werden, darlegt — sowohl alle Eingabe-Parameter, als auch die daraus resultierende Ausgabe. Normalerweise wird diese verbale Beschreibung danach in Form einer Tabelle und/oder eines Zustand-Übergangs-Diagramms umgeschrieben. Beides formalisiert eigentlich nur die Antwort auf die Frage „Wenn in dieser gegebenen Situation („Zustand“) folgende Eingabe gemacht wird („Input“), welches Ergebnis („Folgezustand“ und/oder „Output“) wird damit erzielt?“ Auf diese Weise kann bei gleichzeitiger Entwicklung von Operations- und Steuerwerk sichergestellt werden, daß eine gewisse Kommunikation zwischen beiden stattfindet, und das Steuerwerk in jedem Augenblick die richtigen „Kommandos“ gibt. Wichtig dafür ist, daß möglichst alle Kombinationen von Eingabeparametern und Zuständen (die durch eine gewisse Anzahl sogenannter „Zustands-Bits“ kodiert werden) durchdacht und definiert werden.

Sind diese Überlegungen getätigt, so ist die Schaltung (bis auf durchaus nicht zu vernachlässigende Optimierungen) im Prinzip fertig: Aus den Input-Signalen und den Zustandsbits, sind für jedes Output-Signal und für jedes Bit des Folgezustands recht schnell die Bool'schen Gleichungen und damit auch die Schaltung selbst erstellt.

Diese Vorgehensweise hat den Vorteil, daß problemlos mit einem einzigen Systemtakt gearbeitet werden kann, der periodisch (z.B. mit jeder Vorderflanke) Zustand und Ergebnis für einen Takt lang einfriert. Damit ist es zum einen möglich, die Folgezustände und -Signale auf der Basis von stabilen Signalen zu erzeugen, und zum anderen muß bei späteren Hardware-Tests nur die System-Clock angehalten bzw. „durchgestepped“ werden, um die korrekte Zuordnung von Input, Zustand und Output zu überprüfen.

Der Nachteil besteht darin, daß es möglicherweise schwieriger ist, auf Änderungen in den Anforderungen zu reagieren. Außerdem kann dieses Entwurfs-Schema zu aufwendig sein, z.B. wenn es sich um kleine Testschaltungen handelt, bei denen die Realisierung offensichtlich oder schnell geändert ist. Letzteres gilt vor allem für die Analog-Elektronik; bei ASICs ist es nicht mehr möglich einzelne Schaltungsteile nach der Fertigung zu verändern.

Bei der Entwicklung des hier beschriebenen aktuellen Quadrantensequenzers wurde nicht danach vorgegangen — zum einen war das Operationswerk (Common-Mode-Filter) bereits voll entwickelt und zum anderen existierten bereits einige Überlegungen zur Realisierung seines Kernstücks, die sich aus den Erfahrungen mit den Vorläufern ergeben hatten, und daher auch in der neuen Schaltung berücksichtigt werden sollten. Diese Vorläufer waren reine Zeilen-Sequenzer ohne die Möglichkeit die Pipeline oder die Betriebsmodi zu berücksichtigen, weswegen auch die Überlegungen auf den Sequenzer-Teil beschränkt waren, der den Ablauf einer Zeile steuert:

1. Die Verwendung eines RAMs für die zeilenübergreifende Steuerung (das LRAM, in Abschnitt 3.3.4.2 bereits eingeführt),
2. Die Erzeugung der entsprechenden RAM-Adresse und der Phi-Pulse (Abschnitte 5.5.2 und 5.5),
3. das Prinzip des „SEQADR“ und der „Tor-Flip-Flops“ (Abschnitt 5.1).

5.1 Der SEQADR und die an ihm hängenden Signale

Bereits in Abschnitt 3.3.4.2 wurde erläutert, daß der QSEQ spätestens alle 50 ns einen neuen Steuerzustand liefern muß. Es wurde ebenfalls erwähnt, daß dies nur durch einen eigenen Schaltungsteil, dem Line-Sequenzer machbar ist. Kernstück dieses Zeilen-Steuerwerks ist ein Zähler, der „SEQADR“ (SEQenzer ADRess), der die für eine Zeile benötigten Takte einfach abzählt und je nach Zählerstand („Adresse“) die zeilen-relevanten Signale schaltet. Allerdings sollten diese Signale nur dann kommen, wenn auch ein CCD aktiv ist und tatsächlich Zeilen zumindest geschoben werden sollen. Außerdem sollte der CAMEX seine Arbeit nicht gerade dann aufnehmen, wenn die Anode zwischen CCD und CAMEX rückgesetzt wird (Abschnitt 3.2.3.3). Dieser Zähler darf also nur dann zählen, wenn beide Bedingungen erfüllt sind, wie es im Schaltplan[1] SEQADR (S. 126) oben links deutlich zu sehen ist.

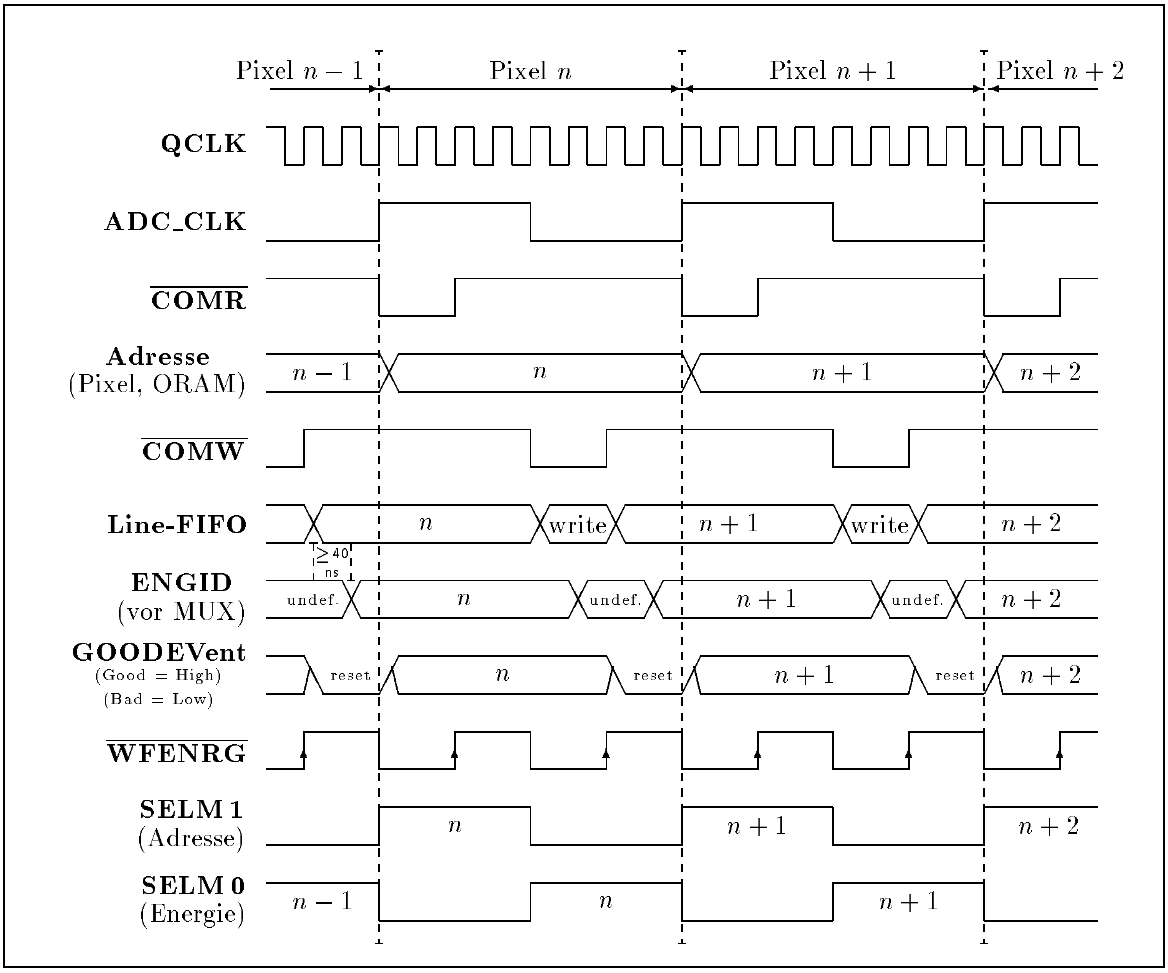
Die meisten dieser Signale — diejenigen die die Verarbeitung eines Pixels steuern — werden sich sehr häufig mit der Periode eines Pixels wiederholen müssen. Daher ist es sinnvoll, ebenfalls periodische Signale zur Erzeugung dieser Steuersignale zu verwenden.

D_3	D_2	D_1	D_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
	⋮		

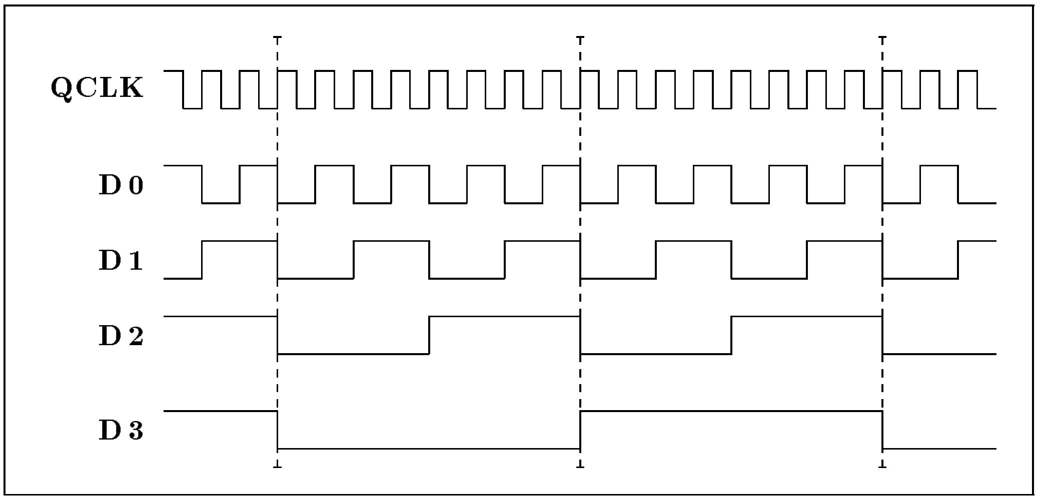
Ein Zähler liefert mit seinen einzelnen Stellen solche periodischen Signale. Allerdings zählt weder ein Dezimal- noch ein Binär-Zähler modulo 6, was den 6 Signalwechseln pro Pixel entspräche und die 50 ns Taktzyklus bewirkt! Ein Binärzähler wiederholt sich immer nach 2^n Schritten, wie nebenstehend verdeutlicht wird, was als nächstliegende Anzahl 4 oder 8 Signalwechsel bedeutet. Da weniger als 6 Signalwechsel nicht erlaubt waren (und mir die naheliegende Idee mit kaskadierten Zählern nicht kam), wurden 8 Signalwechsel pro Pixel implementiert, wie in Abbildung 5.1 (a) dargestellt ist. Allerdings wird nun für ein Pixel etwas mehr als 300 ns, nämlich 320 ns benötigt, da die QCLK für die strahlungsharten Bauteile sonst zu schnell geworden wäre.

[1] Einige Hinweise zum Verständnis der Schaltpläne: Bei den Gattern steht ihre Verknüpfungsart immer dabei. „and“ und „or“ sind dabei selbserklärend; „nand“ und „nor“ sind einfach ihre Negation (in Hardware-Kreisen fast die beliebtesten Gatter, da jede logische Funktion auch nur aus NANDs, bzw. NORs aufgebaut werden kann). „inv“ steht für „Inverter“, der ein Signal negiert, und damit z.B. auch aus einer bisher positiven Flanke eine negative macht. Eine Kurzform für Negationen sind kleine Kreise; bei Eingängen invertieren sie diesen Eingang und bei Ausgängen das Ergebnis (daher das Symbol der NAND- und NOR-Gatter als AND-/OR-Gatter mit einem Kreis am Ausgang).

Gleiches gilt für den Takteingang von Flip-Flops („f...“) und Binär-Zähler („cb...“): Er ist mit „C“ bezeichnet und hat bei negativ-taktflanken gesteuerten Bauteilen einen Kreis davor. Dieser Eingang wird durch den Clock-enable-Eingang CE aktiviert, was die Verwendung einer einzelnen Systemclock oftmals erst ermöglicht.



(a) *Der zeitliche Ablauf der Verarbeitung zweier Pixel. Außer den Daten sind vor allem die pixelrelevanten Steuersignale dargestellt. Allerdings fehlt CLK_D_IN, da sich dieses Signal in seiner Lage ändern kann (siehe Abbildung 4.11).*



(b) *Die unteren 4 Bit eines Binär-Zählers samt des „Zähl-Taktes", also die QCLK und Bit 0 bis 3 des SEQADR, aus denen die Steuersignale erzeugt werden.*

Abbildung 5.1: *Zeitdiagramme zweier Pixel.*

$$
\begin{aligned}
\mathrm{ADC\_CLK} &= \overline{\mathrm{D}_2} \\
\overline{\mathrm{COMR}} &= \mathrm{D}_1\overline{\mathrm{D}_2} \\
\overline{\mathrm{COMW}} &= \mathrm{D}_1\mathrm{D}_2 \\
\overline{\mathrm{WFENRG}} &= \mathrm{D}_1 \\
\mathrm{SELM}\,1 &= \overline{\mathrm{D}_2} \\
\mathrm{SELM}\,0 &= \mathrm{D}_2
\end{aligned}
$$

Auf diese Weise können die unteren 3 SEQADR-Bits für die Erzeugung der Steuersignale verwendet werden: Vergleicht man die Abbildungen 5.1 (a) und 5.1 (b), so sind die nebenstehenden Zuordnungen leicht verständlich. Dabei gilt (ab) ist gleichbedeutend mit (a und b), $(a \,\&\, b)$ oder auch $(a \wedge b)$. Analog bedeutet $(a+b)$ (a oder b), bzw. $(a \vee b)$. Auch die Äquivalenz, also die Gleichheit zweier Signale hat unterschiedliche Schreibweisen: $(a = b)$, $(a \iff b)$, $(a \equiv b)$.

Für die einzelnen CLK_D_IN-Varianten werden die Gleichungen etwas komplexer. Es wurde jeweils die letzte der nebenstehenden Formen realisiert, da sie durch das gemeinsame $(\mathrm{D}_1 \equiv \mathrm{D}_2)$ die geringste Gatteranzahl ermöglicht (CLK 0 braucht sowieso keine weitere Verknüpfung).

$$
\begin{aligned}
\mathrm{CLK}\,0 &= \mathrm{D}_2 \\
\mathrm{CLK}\,1 &= \mathrm{D}_0\mathrm{D}_2 \vee \overline{\mathrm{D}_0}\mathrm{D}_1\mathrm{D}_2 \vee \overline{\mathrm{D}_0\mathrm{D}_1\mathrm{D}_2} \\
&= \mathrm{D}_0\mathrm{D}_2 \vee \overline{\mathrm{D}_0}(\mathrm{D}_1 \equiv \mathrm{D}_2) \\
\mathrm{CLK}\,2 &= \mathrm{D}_1\mathrm{D}_2 \vee \overline{\mathrm{D}_1\mathrm{D}_2} \\
&= \mathrm{D}_1 \equiv \mathrm{D}_2 \\
\mathrm{CLK}\,3 &= \overline{\mathrm{D}_1\mathrm{D}_2} \vee \mathrm{D}_0(\mathrm{D}_1 \equiv \mathrm{D}_2) \\
&= (\overline{\mathrm{D}_0 \vee \mathrm{D}_2}) \vee \mathrm{D}_0(\mathrm{D}_1 \equiv \mathrm{D}_2)
\end{aligned}
$$

Bis jetzt kommen diese Signale allerdings alle während der gesamten Zeile, ohne Rücksicht auf ADC- oder Zeilen-Pipeline, und auch während der letzten Timing-Pulse (siehe Tabelle 3.1), wo sie ebenfalls störend sind. Es wird also noch eine Art Tor benötigt, das die erzeugten Signale hindurchläßt, oder pausieren läßt, je nachdem ob sie gebraucht werden oder nicht. Dies wird mittels eines Flip-Flops erreicht: Abhängig von der SEQADR-Adresse und dem Line-Mode (siehe Abschnitte 3.4, bzw. 5.2) wird dieses Register gesetzt und die Pixel-Steuerung damit durchgelassen, oder rückgesetzt, womit das entsprechende Signal inaktiv ist.

Durch unterschiedliche Signal-Laufzeiten können sogenannte „Glitches“ oder „Hazards“ entstehen. Dies sind zusätzliche und unerwünschte Signal-„Wackler“, die bei Steuersignalen das Operationswerk deutlich aus dem Tritt bringen würden. Um sie zu verhindern, werden ebenfalls Flip-Flops verwendet, die die Signale oder auch nur irgendwelche Zwischenergebnisse zwischenspeichern („latchen“).

Dies geschieht immer zu bestimmten Zeitpunkten, die durch die Systemclock vorgegeben werden. Und natürlich muß dabei darauf geachtet werden, daß auch genügend Zeit für die Verknüpfungen, bzw. SEQADR-Abfragen verbleibt: Der SEQADR zählt mit der positiven Taktflanke (QCLK$\uparrow$), daher übernehmen die „Tor-Flip-Flops“ ihr Abfrage-Ergebnis, aber auch die zur Signal-Erzeugung verwendeten unteren 3 SEQADR-Bits D 0 bis D 2, mit den negativen Taktflanken (QCLK$\downarrow$), und die fertigen Signale werden mit jeder positiven Taktflanke (QCLK$\uparrow$) gelatched. Damit ist sichergestellt, daß nur stabile Signale zur tatsächlichen Steuerung verwendet werden. Dieses Prinzip ist im Schaltplan „COMCONT“ (S. 129) deutlich zu erkennen.

Diese Takthalbierung geschieht zum einen um die dadurch entstehende interne Pipeline nicht zu lang zu machen und zum anderen um den benötigten Strom etwas zu reduzieren. Da bei jedem Pegelwechsel einiges an Strom fließt, kann die Stromstärke auf diese

Weise vermindert werden. Allerdings ist es ebenso Usus nur die positiven Taktflanken zu verwenden, ja, einige ASIC-Hersteller verlangen dies sogar.

Nachdem alle pixelrelevanten Signale auch immer während ganzer Pixel benötigt werden, können einige Signale bezüglich des Tor-Flip-Flops zusammengefaßt werden. Zum besseren Überblick ist dies in folgender Aufstellung aller am SEQADR hängender Signale ebenfalls geschehen.

- $\overline{\text{COMR}}$ und $\overline{\text{COMW}}$ werden beide immer gleichzeitig gebraucht; sowohl was die Pixelsteuerung betrifft, als auch in welchem Line-Mode sie angeschaltet werden.

- SELM 1, SELM 0 und $\overline{\text{WFENRG}}$ werden nur gebraucht, wenn Daten an die CPU übergeben werden sollen; ihr gemeinsames „enable"-Signal ist als „Data" in Schaltplan COMCONT (S. 129 teilweise noch mit älteren Signalnamen) zu erkennen.

- Analog zu Data, gibt es in genanntem Schaltplan ein „Time", das die Signale zusammenfaßt, die für das Zeitwort verantwortlich sind (SELM 0, SELM 2 und $\overline{\text{WFTIM}}$). Ebenfalls an Time hängt $\overline{\text{MEDCLK}}$. Dieses Signal kommt — was das Zeitverhalten innerhalb einer Zeile betrifft — gleichzeitig zum Zeitwort, wird aber im CCD-Ablauf durch zusätzliche Signale (Line-Mode und das LRAM-Bit TIMFLG, siehe Abschnitt 5.9.1) von diesem unterschieden.

- Ähnliches gilt für $\overline{\text{NEWCHIP}}$, $\overline{\text{NEWLINE}}$ und INIT. Auch sie kommen während einer Zeile gleichzeitig, nämlich ganz am Anfang, und werden durch weitere Signale voneinander unterschieden. Da dies während der Entwicklungszeit jedoch nicht immer eindeutig war, hat hier trotzdem jedes Signal sein eigenes Tor. Anders als INIT, das den TIMEX gerne jede Zeile initialisieren darf, ist es bei den beiden anderen Signalen wichtig, daß sie immer in der physikalischen Zeile 1 des CCDs kommen, da an ihnen die Zuordnung der Pixeladresse zu den Daten hängt. Daher werden beide nach dem ersten Schiebevorgang aktiviert, wie Schaltplan „LMC_200" (S. 120) zeigt. NC_en für $\overline{\text{NEWCHIP}}$ wird mit dem nächsten Schieben, das ja nach dem Anfang von Zeile 1 kommt, wieder abgeschaltet, während NLIN_en erst durch einen CCD-Wechsel (SEQRES, Abschnitt 5.8) oder mit dem Quadranten-Ende bei Betriebsmodi ohne CCD-Wechsel deaktiviert wird.

- Die ADC_CLK unterliegt eigentlich ebenfalls zeitlichen Einschränkungen. Allerdings wurde in Abschnitt 3.3.3 bereits darauf hingewiesen, daß der ADC einschwingen muß. Da er die ihm angebotenen Analog-Daten nicht beeinflußt und auch kleinere Pausen verträgt, läuft er einfach mit dem SEQADR durch; in den Zeiten, in denen die anderen Signale warten (Anfang und Ende einer Zeile) konvertiert er auf diese Weise die anliegenden Analog-Werte. Diese sind abhängig davon, welcher Timing-Puls zuletzt gegeben wurde: Vor dem RD-Puls ist es einfach noch immer Pixel 63 der aktuellen Zeile; zwischen RD- und SH-Puls ist es nochmals Pixel 0 dieser Zeile und nach dem SH-Puls, der ja die neue Zeile zum Abholen bereitstellt, ist es Pixel 0 der nächsten Zeile.

- CLK_D_IN hat ein eigenes Tor-Flip-Flop, das für alle CLK i gültig ist. Allerdings ist vor das dahinterliegende Register ein Multiplexer geschaltet, der aus den 4 möglichen (und auch stets realisierten) Varianten diejenige auswählt, die erwünscht ist (siehe Schaltplan „ADC_CLK_D_IN“ auf Seite 128). Nebenstehende Tabelle zeigt die Kodierung der 2 Leitungen, die diesen MUX steuern. Ihr Pegel wird durch CPU-Kommandos definiert (Abschnitt 5.9.3.6). Auch der CLK_D_IN arbeitet bereits während Line 0, obwohl er zu diesem Zeitpunkt noch keine validen Daten abholen kann. Damit ist sichergestellt, daß die erste Datenzeile nicht unter anderen Bedingungen verstärkt wird, als die anderen.

CLKMUX 1	**CLKMUX 0**	**CLK** i
0	0	CLK 0
0	1	CLK 1
1	0	CLK 2
1	1	CLK 3

Tabelle 5.1: *Die Kodierung der CLK i.*

- Ein Signal, das das 2-Flanken-Prinzip extrem ausnutzt ist das Timing-Signal. Schaltplan „Timing“ (S. 127) zeigt deutlich, wie für jeden Timing-Puls ein eigenes rückflankengesteuertes Flip-Flop benötigt wird, deren Ausgänge über ein großes ODER auf das vorderflankengesteuerte Flip-Flop gehen. Dadurch ist jedoch die Realisierung von FLUSH recht einfach geworden: Da dieses Signal zeitlich identisch mit dem zweiten S_{in}-Puls sein soll, kann als Tor dessen Abfrage-Flip-Flop verwendet werden. Für das zweite Flip-Flop muß dann nur noch die grundsätzliche „Erlaubnis“ (FLUSH_en aus dem LRAM) berücksichtigt werden.

- Weitere vom SEQADR abhängige Signale sind die drei Phi-Pulse und AMOS. Ihnen ist ein eigener Abschnitt gewidmet (ab Seite 65).

- Als letzte Signale sind QSEQ_RDY, EOCCD und EOLINE zu nennen, die jeweils das Ende eines Quadranten, eines CCDs und einer Zeile markieren. Die beiden letztgenannten sind bisher zwar nicht erwähnt worden und haben außerhalb des QSEQ auch keinerlei Bedeutung, sind aber für den internen Ablauf sehr wichtig: Von ihnen hängen sowohl der Line-Mode als auch das Schalten der SEL_i-Signale ab, weshalb das Zeitverhalten, das diese Signale untereinander haben, von großer Bedeutung ist. Schaltplan „SEQADR“ (S. 126) zeigt ihre Realisierung. Deutlich zu erkennen ist, daß

  1. EOLINE direkt am Tor nach der negativen Taktflanke abgegriffen wird,
  2. EOCCD mit der Last-Line-Bedingung (LL) verknüpft einen halben Takt später kommt,
  3. QSEQ_RDY aus der Verknüpfung dieses EOCCDs mit einem um einen ganzen Takt verzögerten (Flip-Flop$\downarrow$ und Flip-Flop$\uparrow$) TIMFLG entsteht. Es wird also noch einen halben Takt später erzeugt, wenn das letzte CCD sicher fertig ist, und liegt dann 4 Takte lang an (paralleles setzen von 4 Flip-Flops, die als Schieberegister hintereinandergeschaltet sind). Die UND-Verknüpfung von EOCCD und TIMFLG macht allerdings auch deutlich, daß QSEQ_RDY nur dann kommt, wenn beide Bedingungen erfüllt sind: TIMFLG und LL!

     Weitere Aufgaben von QSEQ_RDY sind das Rücksetzen des LRAMCNT in jedem Betriebsmodus außer im Timing-Mode (Abschnitt 5.5.2), das Schalten des Chopper-Zählers im Common-Mode-Filter (Abschnitt 3.3.4.1) und das Aufwecken der CPU, die sich nach getaner Arbeit während der Integrationszeit schlafen legt um Strom zu sparen.

5.2 Der Line-Mode

Anders als $\overline{\text{NEWCHIP}}$ und $\overline{\text{NEWLINE}}$, die immer am physikalischen CCD-Anfang kommen müssen, orientiert sich der Line-Mode wie er in Abschnitt 3.4 erklärt wurde an den tatsächlich zu verarbeitenden CCD-Zeilen, also an der Fenstergröße. Da diese variabel ist, muß auch der Line-Mode variabel gehalten werden. Nachdem er sich während eines Quadranten-Ablaufs ändert, er aber für eine Zeile konstant ist, ist für seine Steuerung ein Bit im Line-RAM vorgesehen: Bit 3, das LMC-Bit.

Sein Name ist auf die Realisierung des Line-Modes mittels eines Zählers, des „Line-Mode-Counters", zurückzuführen: Es handelt sich dabei um einen 2-Bit-Zähler, der am Anfang auf 0 gesetzt wird, was soviel wie Line 0 bedeutet. Danach zählt er bei jedem gesetztem LMC-Bit zyklisch um einen Mode weiter; nach 3, was der Last-Line entspricht, kommt wieder L 0.

Allerdings hat die Programmierung des Line-Modes eine Tücke: Damit der Zähler für die ganze Zeile, insbesondere bereits am Zeilenanfang, den richtigen Mode anzeigt, muß er bereits am Ende der vorhergehenden Zeile zählen! Seine „Zählbedingung" lautet daher (LMC $\wedge$ EOLINE), wie es Schaltplan „LineMode" (S. 121) deutlich zeigt.

Noch eine Bemerkung am Rande: Der Schaltplan zeigt übrigens auch, warum die EOLINE mit QCLK$\downarrow$ erzeugt werden muß: Der Zähler (cb2re) zählt mit QCLK$\uparrow$, und diese Flanke muß durch das Enable-Signal (CE-Eingang) sauber überdeckt werden. Dieses Prinzip mußte bei allen Zählern angewendet werden. Für Enable-Signale, die länger als einen Takt anliegen und den Zähler daher mehrfach zum zählen auffordern würden, sind sogenannte Flankendetektoren („FD_...") davor geschaltet worden. Je nachdem, ob der Zähler mit der Vorder- oder Rückflanke des Enable-Signals zählen soll, gibt es FD_ups oder FD_downs. Damit diese Detektoren jedoch ordentlich funktionieren, müssen sie auch die Taktflanke berücksichtigen, mit der das Enable-Signal erzeugt wird: „..._1" zeigt Erzeugung mit QCLK$\downarrow$ an, wie die Schaltpläne ab S. 148 verdeutlichen. Der Zähler muß dann natürlich mit der anderen Flanke zählen. Diese Flankendetektoren werden unnötig, wenn der Zählpuls direkt an den Takteingang angeschlossen wird. Dies wird von ASIC-Herstellern jedoch äußerst ungern gesehen, da nur ein einziger Systemtakt ein Testen durch durchsteppen der Takte ermöglicht.

5.3 Die Realisierung des Timing-Modes

Der Timing-Mode (Abschnitt 3.1.3) ist von allen Betriebsmodi der einzige, der (bis auf den Line-Mode, das Zeilen-RFGA und FLUSH, Abschnitte 3.2.3.3 bzw. 3.2.4) vollkommen hartkodiert ist; sobald die CPU das Kommando TMS_ON gegeben und den QSEQ eingeschaltet hat kommen QSEQ_RDY und das Zeitwort automatisch alle 200 Makro-Zeilen. Das bedeutet allerdings auch, daß die CCD-Größe was die Zeilen-Anzahl betrifft nicht wie in den anderen Modi verändert werden kann.

Damit dies funktioniert, sind vor allem 2 Zähler notwendig: Der eine zählt die 10 Schiebevorgänge ab, die im Timing-Mode ja eine Makro-Zeile definieren (siehe Schaltplan „TMS_Phi", S. 131), und der andere zählt die 200 Zeilen-Enden eines (Makro-)CCDs (Schaltpläne „CNT_200", S. 122, bzw. „LMC_200", S. 120).

Nachdem es im Timing-Mode kein einzustellendes Fenster gibt, sind auch alle Zeilen in ihrem Ablauf identisch, sieht man einmal von der Datenfluß-Pipeline ab. Dadurch ist es aber auch nicht notwendig, eine einmal aufgebaute Zeilen-Pipeline während der Messung wieder abzubauen — ist die „Normal Line“ erreicht, muß der Line-Mode nicht mehr geändert werden und der Ablauf wiederholt sich zyklisch.

Das Anhalten des Line-Mode-Counters mit NL bei eingeschaltetem TMS ist recht einfach, wie Schaltplan „LineMode“ (S. 121) zeigt; für die Realisierung des zyklischen Ablaufs sind einige Überlegungen vonnöten:

Zeile im	**Zeilen-Nummer**											
Zeilen-Zähler	0	1	2	3 ··· 199	0	1	2	3 ··· 199	0	1	2 ···
Line-FIFO		0	1	2 ··· 198	199	0	1	2 ··· 198	199	0	1 ···
Filter			0	1 ··· 197	198	199	0	1 ··· 197	198	199	0 ···
Beachte:				···		↑		···		↑	···

Tabelle 5.2: *Überlegungen zu $\overline{\text{NEWCHIP}}$ und QSEQ_RDY im Timing-Mode.*

Damit nach dem Pipeline-Aufbau die Zusatz-Signale wie QSEQ_RDY und das Zeitwort oder auch $\overline{\text{NEWCHIP}}$ zur Adress-Initialisierung zum richtigen Zeitpunkt kommen, muß man sich klar machen, wann welche Zeile wo in der Pipeline ist. Betrachtet man Tabelle 5.2, so fällt auf, daß die letzte Zeile verarbeitet wird, wenn die erste Zeile Offset-korrigiert werden soll. Das bedeutet also, daß QSEQ_RDY mit dem Zeitwort und $\overline{\text{NEWCHIP}}$ in der selben Zeile kommen müssen, nämlich in Zeile 1. Diese Zeile „Timing-Mode 1“ ist also eine besondere Zeile, die erkannt werden muß, was mit dem Signal TM1 in Schaltplan CNT_200 (S. 122) geschieht. Allerdings soll während des Pipeline-Aufbaus, also das allererste Mal, in Zeile 1 kein QSEQ_RDY samt Zeitwort kommen — das CCD hat ja gerade eben erst angefangen. Daher wird ein Signal benötigt, daß auch dieser Zusatz-Bedingung „und nicht L 1 aus der Pipeline“ gehorcht. In Schaltplan LMC_200 (S. 120) ist dies als TMS1 realisiert, ein Signal, das auch schon in Schaltplan SEQADR (S. 126) bei der Erzeugung von QSEQ_RDY mitgemischt hat. Außerdem ist in LMC_200 auch deutlich zu erkennen, daß $\overline{\text{NEWCHIP}}$ über NC_en bereits mit TM1 aktiviert wird.

Eine weitere besondere Zeile im Timing-Mode ist Zeile 199: Damit der Modus zyklisch arbeitet, muß natürlich auch das LRAM zyklisch adressiert werden. Nachdem es nur verwirrend ist, wenn der Zeilen-Zähler, der die 200 Makro-Zeilen abzählt, und der LRAM-Adress-Zähler (siehe Abschnitt 5.5.2) unterschiedlich zählen, muß der Adress-Zähler zwischen dem Beginn von Zeile 199 und Zeile 0, also in Zeile 199, auf 0 gestellt werden. Dies wird durch das Signal „Res-RAMADR“ in den Schaltplänen CNT_200 (S. 122), LMC_200 (S. 120) und Phi (S. 130) erreicht.

5.4 Das Schnelle Schieben

In der Erklärung der Betriebsmodi (Abschnitt 3.1) wird mehrfach von einem Fenster in der Detektor-Mitte ausgegangen, das dadurch erreicht wird, daß die Daten aller äußeren Zeilen schnell geschoben und weggeworfen werden. Zu diesem Zweck sollten alle pixelrelevanten Signale angehalten, und nur die Phi-Pulse gegeben werden. Dies ginge am einfachsten, wenn der SEQADR angehalten würde. Dies ist jedoch nicht durchführbar, da die Zeilen-Nummern trotzdem hochgezählt werden müssen um die Pixel-Adresse der Fensterlage korrekt zuordnen zu können.

Eine andere, auch realisierte Möglichkeit, die betreffenden Signale zu deaktivieren, ist die Verwendung des „Fast-Shift-Kommandos". Das Schnelle Schieben bezieht sich immer auf ganze Zeilen innerhalb eines Quadranten-Ablaufs; seine Aktivierung ist daher mittels eines LRAM-Bits namens FS implementiert, dessen Pegel zur Beeinflussung der „Tor-Flip-Flops" verwendet werden kann: Bei allen Signalen, die während des Fast Shifts nicht kommen dürfen, übernimmt das Tor zum Zeitpunkt an dem es normalerweise öffnet den Wert $\overline{\mathrm{FS}}$ — ist FS eingeschaltet, ist dies 0, was soviel wie ein geschlossenes Tor bedeutet, soll die Zeile normal verarbeitet werden, ist FS=0, also $\overline{\mathrm{FS}}$=1, und das Tor wird geöffnet. Dieses Prinzip ist im Schaltplan Timing (S. 127) gut zu sehen.

Umgekehrt kann FS das Tor für die Phi-Pulse überbrücken und sie unabhängig vom SEQADR am Anfang der Fast-Shift-Zeile aktivieren, wie es Schaltplan Phi (S. 130) verdeutlicht; sie werden erst wieder ausgeschaltet, wenn FS = 0 aus dem LRAM gelesen wird.

Das Weiterzählen der Zeilen-Nummer hat allerdings auch seine Tücken: Werden nur äußere Zeilen weggeworfen, so funktioniert alles richtig, und die Zuordnung von Adresse zu den Daten bleibt korrekt erhalten. Sollte jedoch am CCD-Ende ein Fast Shift gegeben werden, so führt dies zu einer falschen Adress-Zuordnung der darauf folgenden normalen Zeile: Auch während schnell geschoben wird, sind wegen der Datenfluß-Pipeline (Abschnitt 3.4) Daten im Line-FIFO und im Sample-and-Hold-Register, die zu den letzten beiden normal ausgelesenen Zeilen vor dem Schnellen Schieben gehören. Da für einen CCD- oder Quadranten-Wechsel aber eine normale „Last Line" unbedingt notwendig ist (EOCCD, Seite 62), werden zumindest die Daten aus dem Line-FIFO noch verarbeitet und mit der dann gültigen (späteren) Zeilen-Nummer an die CPU übergeben!

5.5 Die Phi-Pulse und was von ihnen abhängt

Dieser Schaltungsteil war der erste, der entwickelt wurde. Das in Abschnitt 5.1 vorgestellte 2-Flanken-Prinzip für die vom SEQADR abhängigen Signale („Tor-Flip-Flop" mit QCLK↓ und „Stabilisierungs-Flip-Flop"mit QCLK↑) war zu diesem Zeitpunkt noch nicht entstanden, weshalb hier sowohl das Tor als auch die Stabilisierung mit QCLK↑ arbeiten. Außerdem wird dieser Abschnitt zeigen, daß — anders als bei den anderen Signalen, die in dem erwähnten Abschnitt aufgelistet sind — mindestens eine weitere Taktflanke zur eigentlichen Erzeugung der Phi-Pulse benötigt wird, ein weiterer Punkt, der eine strenge Anwendung des genannten Prinzips nicht möglich macht.

Der zeitliche Ablauf der Phi-Pulse wurde bereits in den Abschnitten 3.2.2 und 4.1 diskutiert. Besonders Abbildung 4.3 zeigt deutlich, daß pro Puls 9 Takte und für alle 3 Pulse mindestens 15 Takte benötigt werden. Für ununterbrochenes Schnelles Schieben ist es allerdings von Vorteil, wenn der letzte und der erste Puls nicht bündig, also mit der selben Taktflanke schalten, sondern eine kleine Pause zwischen ihnen besteht.

Dieses geforderte Verhalten läßt sich recht einfach mit einem sogenannten 9-Bit-„Johnson-Zähler“ erreichen. Dieses ist ein Schieberegister aus 9 Flip-Flops mit einer negativen Rückkopplung wie Schaltplan CJ9RE (S. 147) zeigt. Dadurch daß Flip-Flop 0 immer das negative Ergebnis von Flip-Flop 8 übernimmt, ergibt sich ein zyklischer Ablauf, wie er in Tabelle 5.3 dargestellt ist. Bei genauerer Betrachtung der mit einem Pfeil gekennzeichneten Zeilen ist unschwer festzustellen, daß sie genau das Pulsmuster der Phi-Pulse liefern: Q 0 entspricht Phi 3, Q 3 hat das zeitliche Verhalten von Phi 1 und Q 6 bildet Phi 2 ab; selbst eine Pause von 3 Takten bei einem zyklischen Ablauf ist gegeben.

Bit	Sein Wert mit der Zeit (taktweise)
Q 0	111111111000000000 1111··· ⟵
Q 1	011111111100000000 0111···
Q 2	001111111110000000 0011···
Q 3	000111111111000000 0001··· ⟵
Q 4	000011111111100000 0000···
Q 5	000001111111110000 0000···
Q 6	000000111111111000 0000··· ⟵
Q 7	000000011111111100 0000···
Q 8	000000001111111110 0000···

Tabelle 5.3: *Der zeitliche Ablauf in dem 9-Bit-Johnson-Zähler, wie er für die Erzeugung der Phi-Pulse verwendet wird.*

5.5.1 Das „Timing Shift“

Neben dem Fast Shift, das bereits am Zeilenanfang schieben läßt, gibt es das Timing Shift (TS) aus dem LRAM, das die Schiebe-Pulse nicht in ihrer Lage innerhalb einer Zeile verändert, sondern aus dem einfachen Schieben ein 10-faches Schieben macht, wie es im Timing-Mode automatisch geschieht (daher der Name). Diese Möglichkeit 10 Zeilen auf einmal zu verarbeiten ohne in den Timing-Mode gehen zu müssen, wurde durch die Anforderungen von FLUSH gefordert (siehe Abschnitt 3.2.4).

Die Realisierung dieses 10-fachen Schiebens ist etwas komplexer; vor allem bei der später notwendig gewordenen Wahlfreiheit zwischen 4 verschiedenen CLK_D_IN (Abschnitt 4.3) war hier die größte Schwierigkeit.

Wie bereits mehrfach betont wurde, müssen alle Phi-Pulse vollständig unter dem zweiten S_{in}-Puls des Timing-Signals liegen. Dies gilt für die Phi-Pulse eines n-fachen Schiebevorgangs ebenso wie bei einem einfachen. Will man für das Timing Shift, bzw. den Timing-Mode, nicht jeden Puls des Timing-Signals ab dem S_{in}' (Abbildung 3.9) einer neuen SEQADR-Adresse zuordnen, ist die einfachste Art dies zu erreichen, den SEQADR anzuhalten, wie es die Schaltpläne Phi (S. 130) und SEQADR (S. 126) zeigen.

Dies bedeutet jedoch auch, daß alle Pixel-relevanten Signale angehalten werden! Daher muß sowohl der Zeitpunkt des Anhaltens als auch der des Weiterzählens wohlüberlegt werden:

Damit aufgrund der in Abschnitt 4.3 ausgewerteten Messungen zwischen Daten-Anforderung an den CAMEX und Daten-Anliegen am ADC keine falsche Adress-Zuordnung durch diese Pause entsteht, hält man den SEQADR am sinnvollsten nach einer ADC-Übernahme an: Abbildung 4.11 macht deutlich, daß damit zwar bereits ein neues Pixel angefordert wurde, das dann aber die ganze Pause über anliegt, da kein weiteres CLK_D_IN↑ gegeben wird. Aus Abbildung 5.1 (a) geht hervor, daß dies ebenfalls ein guter Zeitpunkt für den Common-Mode-Filter ist: Die letzte Taktflanke, mit der der SEQADR zählt, ist entweder die durch eine der gestrichelten Linien gegebene oder die darauf folgende. Für beide Zeitpunkte gilt: Die Daten sind bereits korrigiert, der Filter hat mit dieser Flanke entschieden, ob das Pixel „good“ oder „bad“ ist, und auch die zugehörige Adresse wurde eben erzeugt. Damit entstehen keinerlei Probleme, wenn der SEQADR weiterzählt und die Daten in das CPU-FIFO geschrieben werden.

Die Entscheidung, welche der beiden möglichen Takte genommen wird, wird durch die Phi-Pulse gegeben: Relativ zu ihrem Ablauf muß der SEQADR im letzten Schiebevorgang an der gleichen Stelle wieder eingeschaltet werden, wie er im ersten Schieben angehalten wurde. Trotzdem muß der Zähler, der die Anzahl der geschobenen Zeilen zählt, Gelegenheit gehabt haben, den 10. Vorgang zu erkennen, Dies geschieht am Besten durch das Zählen der ersten Schiebeflanke, also Phi 3↑, wie es Schaltplan TMS_Phi (S. 131) zeigt. Hier tritt das beim Line-Mode (Abschnitt 5.2) erklärte Prinzip mit den Flanken-Detektoren voll in Aktion. Nachdem Phi 3↑ einen ganzen Takt nach einer aktiven ADC_CLK-Flanke kommt, kann der SEQADR frühestens 1,5 Takte nach dieser Übernahme wieder aktiviert werden, indem das „StopSEQADR“ in Schaltplan Phi (S. 130) durch das „Start-SEQADR“ aus Schaltplan TMS_Phi wieder aufgehoben wird. Die Entscheidung für das Anhalten muß also „einen Takt nach Übernahme“ lauten.

5.5.2 Die LRAM-Ansteuerung und das RAMREG

Das Line-RAM wurde bereits in Abschnitt 3.3.4.2 eingeführt. In ihm sind die Steuerbits für eine ganze Zeile gespeichert, wie sie ab Seite 74 zusammenfassend erklärt sind. Für die Programmierung, aber auch für das Auslesen, dieses LRAMs ist es am einfachsten, wenn die einzelnen Einträge für einen Quadranten-Ablauf in aufeinander folgende Speicherzellen geschrieben werden. Noch einfacher wäre es, wenn jedes CCD immer bei Null anfangen könnte; statt sich die Zuordnung von Inhalt zu Adresse für 3 CCDs überlegen zu müssen, wäre es mit einem Mal getan. Nachdem die CPU jede RAM-Zelle mit dem selben Aufwand adressiert, reduziert sich die Frage nach der Realisierbarkeit dieser Idee auf die Frage, ob es möglich ist, während eines Quadranten-Ablaufs trotzdem immer die richtige Adresse zu generieren.

Dies ist möglich: Ein Zähler übernimmt die Erzeugung der unteren Adress-Bits, während die oberen zwei Bit durch das aktive CCD gegeben werden. Betrachtet man nebenstehende Tabelle, so wird schnell klar, daß SEL_2 und SEL_1 genau die binäre Kodierung des aktiven CCDs liefern, wie sie gewünscht ist. Diese beiden Signale bilden also die Bits 12 und 13 der LRAM-Adresse.

SEL_2	SEL_1	SEL_0	aktives CCD
0	0	1	CCD 0
0	1	0	CCD 1
1	0	0	CCD 2

Tabelle 5.4: *Die Kodierung des aktiven CCDs.*

Dies hat mehrere Auswirkungen:

1. Das LRAM wird in 4 Blöcke aufgeteilt, von denen nur die Anfangsteile von Block 0, 1 und 2 genutzt werden.

2. In der Adressierung kann nicht unterschieden werden zwischen den Fällen „derzeit kein aktives CCD“ und “CCD 0 ist aktiv“; in beiden Fällen wird immer Block 0 adressiert werden (Abbildung 5.3, S. 79)!

3. Bei einem CCD-Wechsel muß auch ein Block-Wechsel erfolgen.

Was Punkt 1 betrifft, so gilt dies nicht immer. Sowohl im Small-Window-Mode als auch im Timing-Mode werden die Blöcke anders genutzt, bzw. verschwinden ganz. Beides hat mit einzelnen LRAM-Bits zu tun:

Zum Small-Window-Mode: Eine der Forderungen war bekanntlich, daß auch die Wahl der CCD-Reihenfolge programmierbar sein muß, damit ein nicht richtig arbeitendes CCD die Messungen nicht unnötig belastet. Dies bedeutet aber, daß das als nächstes einzuschaltende CCD im Line-RAM kodiert sein muß. Betrachtet man die CCD-Reihenfolge des Small-Window-Modes wie sie auf Seite 20 dargestellt ist, so fällt auf, daß nach CCD 0 nicht immer dasselbe CCD eingeschaltet werden soll, sondern daß es abwechselnd CCD 1 und 2 sind. Bei der Programmierung eines CCDs muß aber immer eindeutig sein, welches CCD das nächste ist, daher muß CCD 0 in diesem Betriebsmodus doppelt programmiert werden. Wird das LRAM durch das aktive CCD adressiert, kann zwischen diesen beiden Sequenzen für CCD 0 jedoch nicht unterschieden werden. Daher ist dies der einzige Betriebsmodus, in der das Programm bei Adresse 0 beginnt und der Ablauf des ganzen Quadranten „am Stück“ in aufeinanderfolgende Zellen eingetragen ist. Bit 12 und 13 bleiben also während der ganzen Zeit auf dem Wert, der durch den erwähnten Zähler vorgegeben wird; der Zähler selber wird mit einem TIMFLG, das ja ein Quadranten-Ende markiert, wieder von vorne anfangen.

Zum Timing-Mode: In Abschnitt 5.3 wurde ausführlich darauf eingegangen, wie der Ablauf in diesem Modus zyklisch erfolgen soll. Eine Voraussetzung dafür, daß dieser Zyklus korrekt arbeitet, ist jedoch, daß der Aufbau der Pipeline (Abschnitt 3.4) nicht jedesmal neu durchlaufen wird und 2 Extra-Zeilen produziert. Daher holt sich der Sequenzer die Steuerbits für den Pipeline-Aufbau und den zyklischen Ablauf von unterschiedlichen LRAM-Bereichen: Der Aufbau muß in Block 0 programmiert sein, und der Zyklus in Block 1.

TMS	SWM	Modus
1	1	verboten
1	0	Timing-Mode (Blöcke 0 und 1)
0	1	Small-Window-Mode (LRAM bildet einen einzigen Block)
0	0	jeder andere Modus (4 Blöcke, von denen 0 bis 2 verwendet werden)

Tabelle 5.5: *Die Kodierung der Betriebsmodi, bzw. der grundsätzlich unterschiedlichen LRAM-Adressierungs-Arten.*

Diese 3 grundsätzlich unterschiedlichen LRAM-Adressierungen werden durch die CPU gemäß der Kodierung aus Tabelle 5.5 eingeschaltet. Ihre Realisierung kann in Schaltplan VB (S. 124) nachvollzogen werden.[2]

[2]Sollte jemals eine verbesserte Version des SECOM-ASICs oder auch nur des QSEQs entstehen, so möchte ich hier auf eine mögliche Verbesserung hinweisen: Indem der Line-Mode-Counter mit TMS $\wedge$ NL angehalten wird (Schaltplan „LineMode“, S. 121), ist bereits sichergestellt, daß die Pipeline nicht über die Normale Zeile hinausgeht, selbst wenn das LMC-Bit im Line-RAM gesetzt wäre. Demzufolge ist es auch nicht erforderlich den Pipeline-Aufbau und den zyklischen Ablauf im LRAM zu trennen.

In der normalen blockartigen LRAM-Adressierung wird Block 3 niemals verwendet. Es wäre daher möglich, den kompletten Timing-Mode in diesem obersten Block zu programmieren. Dies hätte den Vorteil, daß für das Einschalten des Timing-Modes nur TMS_ON gegeben werden müßte; das Programm kann bei der Programmierung jedes sonstigen Modus in Block 3 geschrieben werden, da selbst der Small-Window-Mode diesen Bereich nicht benötigt. Die einzige QSEQ-Änderung, die zu machen wäre, wäre bei der Modifizierung der Bits 12 und 13 in Schaltplan VB (S. 124). Es müßte realisiert werden:

$$\text{Bit } 12 = \text{TMS} \vee \overline{\text{SWM}}\&\text{SEL\_1} \vee \text{SWM}\&\text{LRMCNT}(12)$$

und $$\text{Bit } 13 = \text{TMS} \vee \overline{\text{SWM}}\&\text{SEL\_2} \vee \text{SWM}\&\text{LRMCNT}(13)$$

Was die beiden letzten Punkte in der Liste der Auswirkungen betrifft, so sind einige Reset-Signale eingeführt worden, damit alles richtig funktioniert. Ihnen ist ein eigener Abschnitt 5.8 gewidmet.

Bisher wurde ausführlich auf die oberen beiden Bits der LRAM-Adresse eingegangen; nun sollen die Zusammenhänge um den Zähler erklärt werden:

Ebenfalls in Abschnitt 3.3.4.2 wurde auf die relativ lange Zugriffszeit für die RAMs hingewiesen. Nachdem sich der QSEQ für jede neue Zeile die Steuerbits aus dem Line-RAM holen muß, ist es wichtig, daß diese zum Anfang einer Zeile auch sicher anliegen.

Ein Signal, das sicherlich jede Zeile kommt und das weit genug vor dem Zeilenende liegt, ist der erste Phi-Puls Phi 3. Selbst wenn schnell geschoben wird, reicht die Zeit vom ersten Puls bis zum letzten aus, das RAM zu adressieren. Es wird daher ein analoges Prinzip angewandt, wie es schon in Abschnitt 5.5.1 erklärt wurde, als es darum ging, wann der SEQADR angehalten werden sollte: Gezählt wird die positive Flanke von Phi 3 — Bei einem einfachen Schieben jedes Mal und bei einem Timing-Schieben nur jedes 10. Mal, da dies ja einem LRAM-Eintrag entspricht. Damit das Ergebnis des „Schiebezählers“ beim Timing Shift auch berücksichtigt werden kann, muß die Flanke von Phi 3 um einen halben Takt verschoben werden, wie es Schaltplan TMS_Phi (S. 131) zeigt. Nachdem der Flanken-Detektor in disem Schaltplan ein „Zähle-Jetzt-Ergebnis“ erkannt hat, wird mit der nächsten QCLK↑ der LRAM-Zähler in Schaltplan Phi (S. 130) geschaltet. Dieses „Schalten“ wird in der Regel ein „Erhöhen um 1“ sein; es kann aber auch ein Rücksetzen des Zähler sein, wenn der Timing-Mode dies mit Res-RAMADR oder der Small-Window-Mode es mit einem TIMFLG anfordern. Weitere Anlässe den LRMCNT wieder auf 0 zu stellen, ist ein Quadranten-Ende (außer im Timing-Mode, siehe Abschnitt 5.3) oder ein CCD-Wechsel. Letzteres ist durch die Kombination von CCDactive und SEQRES gekennzeichnet, und wird in Abschnitt 5.8 näher erläutert.

Bisher wurde stillschweigend davon ausgegangen, daß sich eine Änderung der LRAM-Adresse nicht sofort auf den Sequenzer auswirkt, sondern daß dieser die Steuerbits immer eine ganze Zeile zur Verfügung hat. Damit dies auch tatsächlich der Fall ist, wird ein Register benötigt, das die LRAM-Bits immer zum Anfang jeder Zeile übernimmt und sie damit für die gesamte Zeilen-Dauer stabil hält. Dieses ist das „RAMREG“ in Schaltplan SEQADR (S. 126), das in einer normalen Zeile einen Takt nach EOLINE mit „load“ aktiviert wird. Dieser späte Zeitpunkt nach der Anzeige eines Zeilenendes ist notwendig, damit dem Line-Mode-Counter, der ja einen halben Takt nach EOLINE↑ zählt, nicht das LMC-Bit entzogen wird, bevor er arbeiten konnte.

Im Falle eines Schnellen Schiebens, wird der Zeitpunkt dieses loads jedoch nicht erreicht. Hier muß die hintere Flanke des letzten Phi-Pulses, also Phi 2↓, das RAMREG aktivieren. Nachdem bis zum ersten Phi-Puls eines weiteren Fast Shifts noch 3 Takte verstreichen, bleibt genügend Zeit die kontinuierliche Phi-Puls-Erzeugung abzuschalten, sollte ein FS=0 für eine normale Verarbeitungs-Zeile übernommen worden sein.

Dieses Prinzip, daß de facto am Ende einer Zeile die Bits für die nächste geladen werden, birgt allerdings wieder eine Stolperfalle für den LRAM-Programmierer: Außer im Small-Window- und im Timing-Mode, wo der Adresszähler bereits zum Zeitpunkt der Phi-Pulse mitgeteilt bekommen hat, ob er auf 0 gehen oder hochzählen soll, zählt der LRAMCNT grundsätzlich mit jedem Schiebevorgang um eins weiter. Dies tut er auch in der letzten Zeile eines CCDs und dieser Wert wird dann auch in das RAMREG übernommen werden.

Bei einem CCD-Wechsel ist dies nicht weiter schlimm, da damit auch ein Block-Wechsel verbunden ist, und die für die erste Zeile des neuen CCDs gültigen Werte daher sowieso nochmal geladen werden müssen. Handelt es sich jedoch um einen Neuanfang ohne CCD-Wechsel, wie er z.B. beim Burst-Mode erfolgt (Abschnitt 3.1.4), so geht der Sequenzer korrekterweise davon aus, daß er sich im richtigen Block befindet, unterläßt ein neuerliches Laden, und der LRAMCNT zählt nach dem Rücksetzen mit QSEQ_RDY in der ersten CCD-Zeile von 0 auf 1. Dies hat allerdings zur Folge, daß der LRAM-Eintrag in Adresse 0 dieses Blockes nur beim allerersten CCD-Durchlauf gelesen wird; für jeden weiteren muß dieser Eintrag zusätzlich an das Ende des Programms gehängt werden!

Es empfiehlt sich nicht nur bei einem CCD-Neuanfang sondern auch bei einem CCD-Wechsel den ersten Eintrag des nächsten Ablaufs an diese „Zusatz-Adresse“ zu schreiben, deren Inhalt ja auf jeden Fall in den QSEQ übernommen wird. Damit wird sichergestellt, daß er zumindest nichts falsches lädt; wird diese Vorsorge unterlassen, so kann dies zu einem zusätzlichen VB_SEL führen, das zumindest zu diesem Zeitpunkt falsch wäre!

5.5.3 Das Rückwärts-Schieben und AMOS

Die Realisierung des Rückwärts-Schiebens ist bereits in Abschnitt 3.2.2 angesprochen: Es werden einfach Phi 3 und Phi 2 in ihrer Funktion als der erster und der letzter Schiebepuls vertauscht, sobald „ShiftBack“ durch die CPU eingeschaltet wurde. Dabei muß allerdings sichergestellt sein, daß die beiden Zähler, die mit dem ersten Puls arbeiten (Abschnitte 5.5.1 und 5.9.1), und auch das RAMREG im Falle eines Fast Shifts, alle mit dem richtigen Phi-Puls angesteuert werden. Dies wird erreicht, indem das Vertauschen erst direkt vor den Stabilisierungs-Flip-Flops geschieht, wie es in Schaltplan Phi (S. 130) dargestellt ist.

Gleiches gilt für den Testpuls AMOS. Er soll anstelle der Phi-Pulse kommen, sofern die CPU AMOS_en eingeschaltet hat. Auch hier wird dies mittels dieser letzten Flip-Flops erreicht: Sie übernehmen die — im Zweifelsfall bereits vertauschten — Phi-Pulse nur, wenn AMOS_en ausgeschaltet ist, andernfalls übernehmen sie eine Null, wie es ebenfalls in Schaltplan Phi abzulesen ist.

5.6 Das Zeilen-RFGA

Obwohl das Zeilen-RFGA die Übernahme-Anode am CCD mit der selben Dauer zurücksetzt wie das Vorwärm-RFGA, und auch die selbe Leitung dafür benutzt, so wurde doch hierfür ein eigener Schaltungsteil generiert. Der Grund liegt in der Aufteilung der Vorwärmzeit in RFGA- und Zusatz-Zeit, wie es die Messungen aus Abschnitt 4.2 erforderlich gemacht haben (Abbildung 4.8). Vorher kam das Vorwärm-RFGA während der gesamten Vorwärmzeit und war daher direkt an VB_SEL und SEL_OUT gekoppelt; für das deutlich kürzere Zeilen-RFGA war eine eigene Schaltung notwendig. Als beschlossen wurde, den Einschaltvorgang zu parametrisieren und die RFGA-Dauer nur mit einem Parameter einzustellen (TRFGA, S. 77), war es einfacher, diese Aufteilung beizubehalten. Außerdem ist damit eine saubere Trennung in zeilenbezogene und CCD-bezogene Signale gegeben.

Das Zeilen-RFGA wird natürlich mit einem LRAM-Bit, dem RFGA_en (S. 75) aktiviert. Ist dieses Bit gesetzt, so wird ein Zähler („RFGACNT“) eingeschaltet, der mit jedem QCLK$\downarrow$ zählt. Ist der Zählerstand erreicht, der durch die CPU als RFGA-Dauer eingestellt wurde, so wird der Zähler wieder angehalten, nicht aber rückgesetzt. In Schaltplan RFGA (S. 123) ist leicht abzulesen, daß das RFGA-Signal solange aktiv ist, bis der Komparator eine Gleichheit der Bitmuster des Zählers und der Einstellung erkennt. Dies ist auch die Zeit, während der der SEQADR und damit der gesamte Zeilen-Sequenzer angehalten wird (Schaltplan SEQADR, S. 126), damit der CAMEX nicht mit einer „inaktiven" Anode arbeitet (Abschnitt 3.2.1.1).

Der RFGACNT wird erst zurückgesetzt und die Schaltung damit wieder betriebsbereit gemacht, wenn die aktuelle CCD-Zeile beendet ist. Dies geschieht allerdings nicht mit EOLINE (S. 62), sondern mit dem ebenfalls in Schaltplan SEQADR generierten Signal NEXTLINE, das das RAMREG mit neuen Steuerbits lädt. Auf diese Weise ist sichergestellt, daß auch das Ende einer Fast-Shift-Zeile ein neuerliches RFGA erlaubt.

Allerdings wird das in der RFGA-Schaltung erzeugte RFGA nicht direkt an den Kamerakopf geleitet — damit es dem aktiven CCD zugeordnet werden kann, wird es an die VB-Schaltung übergeben, die für das Einschalten der CCDs verantwortlich ist. Dort wird es in Schaltplan VB_ON (S. 125) mit dem VB_SEL jedes CCDs verknüpft; nur ein aktives VB_SEL erlaubt auch ein RFGA, sei es das Zeilen-RFGA oder das ebenfalls in diesem Schaltplan erzeugte Vorwärm-RFGA, auf das im nächsten Abschnitt 5.7 eingegangen werden soll.

5.7 Das Einschalten der CCDs

In diesem Schaltungsteil werden die Vorteile modularer Entwurfstechnik besonders deutlich: Der zeitliche Ablauf innerhalb eines CCDs sollte während des Einschaltens für alle gleich sein — lediglich der Anfangszeitpunkt ändert sich. Dies ermöglicht es, das Kernstück dieser Schaltung nur einmal zu entwerfen, aber 3 mal zu implementieren: In Schaltplan VB (S. 124) ist daher die Schaltung VB_ON (S. 125) dreifach vorhanden, und außer der Erzeugung der oberen LRAM-Adressbits (Abschnitt 5.5.2), wird in VB nur noch das Zusammenspiel der eigentlichen Einschalt-Schaltungen VB_ON definiert.

In der Erklärung soll zunächst von einem ausgeschalteten VB_EOCCD (Abschnitte 4.2 und 5.9.3.5) ausgegangen werden, so daß sich die CCD-Kodierung aus dem LRAM (S. 75) mit der Übernahme in das RAMREG bemerkbar macht: Einen Takt später wird ein Flip-Flop synchron gesetzt, das entsprechende VB_SEL geht an und aktiviert damit diese VB_ON-Schaltung. In ihr ist das Abwarten der einstellbaren Signaldauern und Verzögerungszeiten nach dem selben Prinzip wie in der eben erklärten RFGA-Schaltung realisiert:

Zunächst wird durch das Einschalten mittels VB_on der RFGACNT aktiviert, der die Takte der Vorwärm-RFGA-Dauer abzählt (S. 77) und der natürlich nur bei ausgeschaltetem SEL_OUT arbeiten darf. Vorausgesetzt die CPU hat RFGA_ON nicht abgeschaltet (S. 77), wird mit der nächsten QCLK$\downarrow$ das RFGA eingeschaltet und bleibt aktiv bis der Komparator festgestellt hat, daß die eingestellte Zeit verstrichen ist; er hält den RFGACNT an, wobei er auch RFGA wieder deaktiviert, und schaltet den nächsten Zähler ein, der die zusätzlichen Takte abzählt, die bis zu SEL_OUT$\uparrow$ noch mindestens abgewartet werden müssen. Auch hier wird wieder durch einen Vergleich mit dem eingestellten TVB-Bitmuster (S. 78) ermittelt, wann der Zeitpunkt erreicht ist, bei dem SEL_OUT frühestens schalten darf; hat der Komparator eine Gleichheit erkannt, so wird dies als „CCDon"-Anforderung einer Ebene höher gemeldet und auch der VBCNT wird angehalten.

In VB wird nun geprüft, ob das entsprechende SEL_OUT auch eingeschaltet werden darf, schließlich ist es strikt verboten, daß zwei derartige Signale gleichzeitig aktiv sind (Abschnitt 3.2.3.1):

Zuerst wird anhand eines Flip-Flops ermittelt, ob es sich dabei um das erste CCD handelt: Das Flip-Flop wird gleichzeitig durch das Einschalt-Signal vom Master-Sequenzer (QSEQ) auf 1 gesetzt und von jedem aktiven SEL_OUT auf 0 gehalten; dabei hat „CCDactive“ die Priorität. Handelt es sich bei der Anforderung um ein „1st CCD“, so wird das SEL_OUT mit der nächsten positiven Taktflanke eingeschaltet; da während des Einschaltvorgangs des ersten CCDs keine weiteren Daten in das RAMREG übernommen werden, besteht auch keine Gefahr, daß ein anderes CCD ebenfalls eingeschaltet wird.

Handelt es sich nicht um das erste CCD, so muß das Ende des noch arbeitenden CCDs abgewartet werden. Mit EOCCD (S. 62) werden dann die Signale in beiden betroffenen VB_ON-Schaltungen geändert — die einen werden ausgeschaltet und das nächste SEL_OUT gleichzeitig angeschaltet. Dazu muß allerdings vorher sichergestellt sein, daß nur ein VB_ON sein SEL_OUT einschalten will. Zu diesem Zweck werden alle 3 CCDon-Anforderungen abgefragt und das Ergebnis als „Only1“ jeder VB_ON-Schaltung mitgeteilt. Damit dies funktioniert, werden mit der Anforderung aus dem LRAM ein neues CCD vorzuwärmen die Zähler der anderen beiden über die R0- und R1-Eingänge von VB_ON rückgesetzt und auf Null gehalten. Andernfalls würde der letzte Komparator des aktiven CCDs weiterhin CCDon = 1 melden und damit jedes weitere SEL_OUT blockieren.

Auch diese Realisierung der Einschaltsequenz hat eine Tücke: Dadurch daß die beiden unteren LRAM-Bits das entsprechende VB_SEL sofort mit ihrer Übernahme in den Quadrantensequenzer einschalten, eine Änderung des entsprechenden Flip-Flops aber erst mit EOCCD geschieht, ist es möglich alle VB_SEL-Signale gleichzeitig aktiv zu haben: In der einen Zeile wird das erste eingeschaltet, das dann auch das tatsächlich aktive CCD wird, in einer späteren Zeile wird das nächste eingeschaltet, dessen SEL_OUT allerdings durch das erste verhindert wird, und wieder später wird die Kodierung für das dritte CCD aus dem LRAM übernommen. Mit letzterem werden alle Zähler der beiden anderen rückgesetzt, etwaige berechtigte CCDon-Anforderungen dadurch gelöscht, nicht aber ihre VB_SEL deaktiviert, da sonst ein Vorwärmen des einen CCDs bei aktivem anderen ja nicht möglich wäre! Erst mit EOCCD wird entschieden, welches VB_SEL im folgenden aktiv bleiben soll; ist es eines der beiden ersten, so fängt die Vorwärm-Sequenz erst jetzt wieder an!

Diese Falle existiert bei eingeschaltetem VB_EOCCD (S. 79) nicht: Hier wird das Setzen dieses Flip-Flops nur durch EOCCD erlaubt oder wenn es sich um das erste CCD handelt.

5.8 Die Auswirkungen der Reset-Signale

Innerhalb des Quadrantensequenzers werden 4 verschiedene Rücksetz-Signale verwendet, deren Erzeugung und Aufgaben nun besprochen werden sollen.

Bei einer Schaltung ist es immer wünschenswert, daß sie schon beim Einschalten der Versorgungs-Spannung in einem ganz definierten Zustand ist; daß das CCD z.B. nicht gerade schiebt ohne angeschaltet zu sein oder ähnliches.

Der Quadrantensequenzer hält im „Einschalt“-Zustand alle Steuersignale auf ihrem inaktiven Pegel und alle durch die CPU programmierbaren Betriebs-Parameter (Abschnitt 5.9.3) auf ihrem jeweiligen Default-Wert. Dieser Zustand kann außer durch das Einschalten auch erreicht werden, indem die EPDH den Befehl „Reset_Q“ („Reset Quadrant“) gibt.

Von dem Einschaltzustand unterscheidet sich der Grundzustand insofern, als daß er die bisherigen Betriebs-Parameter beibehält, aber wie im Einschalt-Zustand alle Steuersignale inaktiv hält. Dieser Zustand wird durch das Signal „MRES“ („Master Reset“) erreicht, das durch vier recht unterschiedliche Situationen erzeugt wird (Schaltplan SEQRES, S. 119):

1. Wenn der Master-Sequenzer den Quadrantensequenzer abschaltet, also durch QSEQ = 0. Dies geschieht, damit jeder Quadrant auch tatsächlich am Anfang anfängt.

2. Wenn die CPU auf das „LRAM“ zugreift, also durch ein inaktives (=1!) $\overline{\text{COMONS}}$, wie auch schon in Abschnitt 3.3.4.3 beschrieben: Damit die zeilen-spezifischen Einstellungen in das Line-RAM programmiert werden können, muß die CPU Zugriff auf dieses LRAM haben. Andererseits muß der Quadrantensequenzer das LRAM adressieren können um sich für jede Zeile die gültige Einstellung zu holen. Damit sich CPU und Sequenzer nicht um das LRAM „streiten“, muß der Zugriff sauber geregelt sein, was durch den Befehl „COMONS“ geschieht. Gibt die CPU dieses Kommando, so bekommt der Sequenzer den Zugriff; er wird ihm wieder entzogen, sobald die CPU versucht das Line-RAM zu adressieren. Als Voreinstellung hat die CPU den Zugriff.

3. Wenn die CPU den Quadrantensequenzer bewußt in den Grundzustand setzt, indem sie den Befehl „CLRSYS“ gibt.

4. Wenn die EPDH die mittels Reset_Q = 1 verlangt.

Bis auf die Parameter-Register, die ausschließlich von Reset_Q rückgesetzt werden können (Schaltplan PARAMETER, S. 118), erreicht MRES jedes einzelne Flip-Flop im QSEQ, sei es direkt oder durch die von ihm eingeschalteten beiden restlichen Reset-Signale VBRES und SEQRES. Seine Aufgabe besteht darin, den inaktiven Quadrantensequenzer betriebsbereit zu halten. Dazu gehört vor allem, die richtige Anfangs-Adresse für das LRAM parat zu haben:

Eine große Anforderung an das System als ganzes ist bekanntlich, daß es möglich sein muß jeden Teil, der nicht mehr ordnungsgemäß funktioniert, zu ersetzen (Master-Sequenzer, Abschnitt 3.3.1) oder im Ablauf zu überspringen. Dies gilt natürlich auch für einzelne CCDs. Durch die blockweise Adressierung (Abschnitt 5.5.2 kann die Anfangs-Adresse nicht einfach aus lauter Nullen bestehen; dies würde in den meisten Betriebsmodi ja bedeuten, daß CCD 0 angeschaltet werden soll. Es wird daher eine zusätzlich Adresse benötigt, deren Inhalt bestimmt, welche die erste richtige „Verarbeitungs“-Adresse sein soll, indem das entsprechende CCD eingeschaltet wird. Eine Adresse, die sich hierfür anbietet, ist die letzte Adresse in Block 0, die für die blockartigen Programme nicht benötigt wird. Sie besteht aus 2 Nullen (Bits 12 und 13) und ansonsten lauter Einsen und ist daher recht einfach zu generieren: Die beiden Nullen, werden durch die inaktiven CCDs gegeben, während der LRAMCNT in Schaltplan Phi (S. 130) mit MRES die Einsen lädt.

Allein im Small-Window-Mode ist dies nicht möglich; durch das Programmieren am Stück, wird Adresse 79FFE (in CPU-Notation, vergleiche Abschnitt 3.3.4.3) für eine Zeile eines späteren CCDs benötigt. Allerdings ist durch eben dieses Programmieren am Stück eine Zusatz-Adresse auch nicht notwendig — die erste Verarbeitungs-Adresse steht mit 78000 sowieso schon fest.

Nachdem MRES auch das RAMREG rückgesetzt hält, muß es ein Signal geben, das nach MRES↓ die LRAM-Bits in das Register übernimmt. Unter anderem dazu ist das Signal SEQRES („Sequenzer Reset“) eingeführt worden: Schaltplan SEQRES (S. 119) zeigt, wie dieses Signal durch MRES (oder die SEL_OUT, aber davon später) aktiviert wird, aber erst einen Takt nach MRES↓ inaktiv wird. Die Flanke SEQRES↓ wird daher zum Laden des

RAMREG verwendet (Schaltplan SEQADR, S. 126), natürlich wieder nach dem Flanken-Detektor-Prinzip (Abschnitt 5.2), weshalb ein halber Takt zwischen SEQRES↓ und dem tatsächlichen Laden des RAMREGs verstreicht.

Sind die Bits geladen, so weiß auch die VB-Schaltung, die für das Einschalten der CCDs zuständig ist (Abschnitt 5.7), welches VB_SEL sie einschalten soll. Hier nun kommt das letzte Reset-Signal VBRES in's Spiel: Würde die VB-Schaltung direkt mit MRES oder SEQRES inaktiv gehalten werden, so ginge sie in der kurzen Zeit, die es braucht, das RAMREG zu laden und den Inhalt in der Schaltung zu verbreiten, davon aus daß sie CCD 0 einschalten soll, was auch geschähe. VBRES ist daher noch einmal 1,5 Takte länger als SEQRES — das tatsächlich adressierte CCD wird eingeschaltet.

SEQRES kommt erst wieder zum Zuge, wenn nach dem VB_SEL das erste SEL_OUT geschaltet hat und damit die LRAM-Adressierung auf die erste Verarbeitungs-Adresse dieses CCDs wechselt. Der Inhalt dieser Adresse kann jedoch erst übernommen werden, wenn die Zugriffszeit des RAMs verstrichen ist. Dies ist der Hauptgrund, warum SEQRES den Zeilen-Sequenzer noch für die Dauer von TSEQRES (Abbildung 4.8) ruhig halten muß. TSEQRES ist durch die beiden CPU-Befehle WTSEQRESH und WTSEQRESL einstellbar und sollte 2 nicht unterschreiten — schließlich muß die Zugriffszeit des LRAMs abgewartet werden. Schaltplan SEQRES (S. 119) zeigt, daß dabei wieder nach dem selben Prinzip vorgegangen wurde, wie es schon in Abschnitt 5.6 bei der Signaldauer von RFGA und später auch beim Einschalten der CCDs angewandt wurde: Ein Zähler wird zusammen mit dem Signal angeschaltet, und der Zählerstand jeden Takt mit der Parameter-Einstellung verglichen. Ist die Gleichheit erreicht, so wird der Zähler angehalten und geht gleichzeitig mit SEQRES einen Takt später wieder auf 0. Dabei werden die richtigen Steuerbits wie oben beschrieben von SEQRES↓ in das RAMREG übernommen.

Handelt es sich nicht um das erste CCD nach einem MRES, sondern um einen CCD-Wechsel, so wird durch SEQRES zusätzlich sichergestellt, daß der Zeilen-Sequenzer inklusive des Line-Mode-Counters (Abschnitt 5.2) wieder von vorne anfängt; der eben beschriebene LRAM-Adress-Wechsel mit SEL_OUT↑ gilt hier natürlich auch.

5.9 Die Eingreifmöglichkeiten der CPU

Obwohl alle LRAM-Bits und (fast) alle CPU-Befehle in den vorherigen Abschnitten bereits mit ihren Auswirkungen erklärt worden sind, und obwohl sich ein tabellarischer Überblick über alle Befehle im Anhang ab S. 91 befindet, soll in diesem Abschnitt eine zusammenhängende Kurzerläuterung aller Einstellmöglichkeiten gegeben werden. Dies soll einem Anwender der Schaltung den Umgang mit dem QSEQ erleichtern.

5.9.1 Die LRAM-Bits

Die zeilenspezifischen Parameter werden in 8 Bit codiert im Line-RAM abgespeichert, das durch den Quadrantensequenzer hauptsächlich mittels eines Zählers adressiert wird (Abschnitt 5.5.2). Die einzelnen Bit haben folgende Bedeutung:

5.9.1.1 Bit 0 und 1: VB0 und VB1

Diese beiden Bit codieren das Einschalten der CCDs. Je nachdem wie VB_EOCCD geschaltet ist (Abschnitt 5.9.3.5), wird das betreffende VB_SEL am Anfang dieser CCD-Zeile gegeben oder erst am Ende des aktuellen CCDs. Wichtig ist allerdings, daß zum Zeitpunkt des EOCCDs am Ende der letzten Zeile eines CCDs das nächste auf jeden Fall codiert ist, da dies der Zeitpunkt ist, zu dem die Einstellung in die Vorwärmschaltung übernommen wird (Abschnitt 5.7).

VB1	**VB0**	**CCD**
0	0	0
0	1	1
1	0	2
1	1	keins

Tabelle 5.6: *Die Kodierung des nächsten CCDs.*

5.9.1.2 Bit 2: RFGA_en (Reset-Fet-Gate-Anode-enable)

Wenn am Anfang einer CCD-Zeile ein RFGA (Abschnitt 3.2.3.3) gegeben werden soll, so muß dieses Bit gesetzt sein. Die Dauer ist identisch wie bei dem Vorwärm-RFGA und mittels des Befehls TRFGA einstellbar (S. 77).

5.9.1.3 Bit 3: LMC (Line Mode Counter)

Der Aufbau der Zeilen-Pipeline (Abschnitt 3.4) wird durch einen Zähler, den Line-Mode-Counter gesteuert, der wiederum durch LMC=1 aktiviert wird. Damit der Zeilen-Modus auch wirklich bereits am Anfang der entsprechenden Zeile richtig ist, muß er jedoch am Ende der vorherigen zählen. Dieses „Zählen am Zeilenende für die nächste Zeile“ sorgt zusammen mit dem zusätzlichen LRAM-Eintrag am CCD-Ende (Abschnitt 5.5.2) bei der Line-RAM-Programmierung erfahrungsgemäß immer wieder für Verwirrung.

5.9.1.4 Bit 4: FL_en (Flush-enable)

Um die innerste CCD-Zeile mit Elektronen zu fluten, die die Halbleiter-Fehlstellen absättigen sollen (Abschnitt 3.2.4), muß dieses Bit auf 1 gesetzt werden. Nicht vergessen werden sollte die Tatsache, daß die Injektions-Elektrode beim FLUSH die Elektronen auf bis zu 5 Zeilen vor und nach der eigentlich gefluteten Zeile verschmiert. Als Konsequenz sollte FL_en=1 daher in Zeile 4 gegeben werden, und am CCD-Ende müssen zusätzlich 10 Zeilen geschoben werden. Diese Zeilen sollten nicht mehr im Common-Mode-Filter verarbeitet werden, was der Hauptgrund zur Einführung von Bit 6 war.

5.9.1.5 Bit 5: FS (Fast Shift)

Sollen ganze CCD-Zeilen „weggeworfen“ werden, so werden sie auf die Anode geschoben ohne übernommen zu werden (Abschnitt 5.4). Das bedeutet allerdings gleichzeitig, daß keine anderweitige Zeilen-Verarbeitung vorgenommen wird (CAMEX-Ansteuerung, Median-Bestimmung, Common-Mode-Filter) wenn dieses Bit gesetzt ist. Dadurch wird natürlich wesentlich weniger Zeit für eine CCD-Zeile benötigt (18 Takte), daher der Name für dieses Bit.

5.9.1.6 Bit 6: TS (Timing Shift)

Dieses Bit erlaubt es, in jedem beliebigen Modus 10 normalen CCD-Zeilen wie beim Timing-Mode zu einer Makrozeile zusammenzuschieben, und die integrierten Daten zu verarbeiten (Abschnitt 5.5.1). TS=1 wird vor allem benötigt, wenn mittels Bit 4 ein Flush gegeben wurde, um alle 10 überschüssigen Zeilen auf einmal wegzuschieben. Damit keine „Injektions"-Daten verarbeitet werden, sollte dies in der letzten CCD-Zeile geschehen, also in derjenige die in der Daten-Pipeline stecken bleibt.

5.9.1.7 Bit 7: TIMFLG (Time Flag)

Wie der Name schon sagt, bewirkt ein gesetztes 7. Bit, daß ein Zeitwort in den Datenstrom zur CPU geschleust wird. Allerdings hat TIMFLG noch weitere Aufgaben: Da ein Zeitwort immer nur am Ende eines Quadranten-Durchlaufs kommen soll, signalisiert dieses Bit dem Master-Sequenzer am Zeilenende auch, daß der Quadrant fertig ist, und weckt eine eventuell schlafende CPU auf (4 Takte QSEQ_RDY, S. 62). Vorausetzung dafür ist, daß sich die Schaltung tatsächlich in einer „letzten Zeile" befindet. Außerdem stellt es sicher, daß im Small-Window-Mode von vorne angefangen wird, indem es die LRAM-Adresse wieder auf 0 setzt.

5.9.2 Die CPU-Kommandos

Um verschiedene Einstellungen über einen längeren Zeitraum aktiv halten zu können (was zumindest bei den Basis-Einstellungen ja der Fall sein muß), müssen die zuständigen Signale gespeichert werden. Dazu werden in der Regel Flip-Flops gesetzt — für jedes Signal ein Flip-Flop. Soll die Einstellung geändert werden, muß man diese Flip-Flops auch rücksetzen können, weswegen die meisten dieser Befehle paarweise auftreten: „Befehl_ON" und „Befehl_OFF". Eine weitere Charakteristik dieser „Flip-Flop"-Befehle ist die Tatsache, daß es völlig egal ist, was dabei auf dem 16 Bit breiten CPU-Datenbus XD (Abschnitt 3.3.4.3) anliegt; um ein Flip-Flop zu bedienen, bedarf es lediglich zweier Leitungen, eine die es setzt und eine die es rücksetzt. Beide sind allein durch die jeweilige Kommando-Adresse („Port") gegeben.

Eine Ausnahme bilden die Befehle, die die mehrere Bit breiten, variabel einstellbaren Zeiten betreffen, sowie die bereits im Zusammenhang mit den verschiedenen Reset-Möglichkeiten des Quadrantensequenzers (Abschnitt 5.8) erklärten Befehle $\overline{\text{COMONS}}$, der den LRAM-Zugriff regelt, und CLRSYS, der den Sequenzer aus dem laufenden Betrieb heraus in den Anfangszustand dieses Betriebsmodus setzt: Bei den Verzögerungs-Befehlen werden natürlich außer der Adresse, die den Befehl definiert, auch die Daten benötigt, und die Befehle $\overline{\text{COMONS}}$ (Port E104) und CLRSYS (Port E10a) haben keinen „Partner-Befehl".

Welche Einstellung beim Einschalten des Quadrantensequenzers gilt, ist mit „default" gekennzeichnet.

5.9.3 Die einstellbaren Betriebs-Parameter

Die Messungen aus Kapitel 4 haben gezeigt, daß das gesamte Detektor-System unterschiedlich reagiert, je nachdem in welchen Umweltbedingungen (Druck, Temperatur) es sich befindet. Daher mußten einige einstellbare Parameter geschaffen werden, die es erlauben, ein möglichst optimales Verhalten zu erzielen. Dies betrifft vor allem das sukzessive Einschalten der CCDs, aber auch das Rücksetzen der Anode vor den analogen Verstärkern und die Übernahme der einzelnen Pixel in den ADC.

Die meisten CPU-Befehle funktionieren nur mittels der Adresse (Port), ohne daß die Daten, die während dessen auf der CPU-Datenleitung anliegen, ein Rolle spielen. Dies sind die Befehle, bei denen nur ein Flip-Flop gesetzt oder rückgesetzt werden muß. Allein wenn die Dauer von Signalen eingestellt werden soll (Abbildung 4.8, Seite 53), geschieht dies, indem die Signal-Länge in Sequenzer-Taktzyklen umgerechnet wird, und diese Taktanzahl über den Datenbus in die Schaltung geladen wird: Wie Schaltplan PARAMETER (S. 118) zeigt, wird das entsprechende Register mit einem Schreib-Befehl aufgefordert, die in dem Moment auf dem Datenbus anliegenden Daten zu übernehmen (Wxyz am CE-Eingang); ein Lese-Befehl ermöglicht die Kontrolle des eingetragenen Wertes. Dabei stellt der Decoder (Abschnitt 3.3.4.3, Schaltplan DECODE S. 111) sicher, daß die Daten aus dem richtigen Register an die CPU geschickt werden.

Schaltplan PARAMETER verdeutlicht allerdings auch eine „Krücke", die aus der Entwicklung mittels dreier FPGAs entstanden ist (Abschnitt 3.3.4): Die Kommandos, die 16 Bit Daten als Parameter übergeben sollten, mußten alle in 2 8-Bit-Kommandos — „...L" für Low-Byte und „...H" für High-Byte — aufgeteilt werden. Der Grund dafür besteht darin, daß das QSEQ-FPGA ursprünglich nur die unteren 8 Bit des Datenbusses zur Programmierung des LRAMs benötigt hat; die oberen 8 Bit wurden erst nach den Messungen aus Abschnitt 4.2 notwendig. Zu diesem Zeitpunkt lagen die Eingänge zu diesem FPGA allerdings bereits fest — das High-Byte war dabei weggelassen worden, da seine Verbindung zum QSEQ nur zusätzlichen Aufwand beim Festlegen der Verbindungen auf der SECOM-Karte bedeutet hätte. Bei der Umsetzung auf das ASIC-Design (Kapitel 6) hat dies dann allerdings wieder zu Schwierigkeiten geführt: Sobald diese QSEQ-Parameter ausgelesen werden sollten, waren die oberen Bit des Datenbusses natürlich undefiniert. Da ein Bus, der „floatet" jedoch zu unvorhersehbaren Effekten führt (z.B. zu ein zu starker Strombedarf), ist dies verboten. Schaltplan „toplevel" (S. 110) zeigt daher, wie diese Bits mittels „Dummy-XD-Treibern" auf 0 gezogen wurden.

Alle Befehle, die in diesem Abschnitt beschrieben werden, betreffen im Grunde genommen Einstellungen, die weder mit dem laufenden Normalbetrieb, noch mit irgendeinem Funktionstest zusammenhängen. Die Werte dieser Parameter werden am Missionsanfang möglichst optimal bestimmt, um dann (hoffentlich) nicht mehr geändert zu werden.

5.9.3.1 RFGA_ON und _OFF

Dieser Parameter bestimmt, ob ein RFGA während der Vorwärmzeit sicherstellt, daß der CCD-Ablauf mit einer leeren Anode beginnt (Abschnitt 3.2.3.3). Das durch Bit 2 des LRAMs programmierbare Zeilen-RFGA ist davon nicht betroffen.

- RFGA_ON (Port E10c) erlaubt das Vorwärm-RFGA (default);
- RFGA_OFF (Port E10e) schaltet das Vorwärm-RFGA ab.

5.9.3.2 TRFGA

TRFGA stellt die RFGA-Dauer (Time, Taktanzahl) ein, unabhängig ob es sich dabei um das Vorwärm- oder Zeilen-RFGA handelt. Er kann sowohl in den Sequenzer geschrieben

(WTRFGA), als auch zur Kontrolle gelesen werden (RTRFGA). Für diesen Parameter sind nur 8 Bit vorgesehen, die intern um 2 Bit höher interpretiert werden, damit ein größerer Wertebereich erzielt wird:

- WTRFGA (Port E128) lädt den Sequenzer mit der RFGA-Dauer, wobei gilt TRFGA $=$ (Dauer $-$ 1) $\div$ 4 Taktzyklen (default ist 0);
- RTRFGA (Port E128) liest den eingestellten Wert, nach dem sich die RFGA-Dauer in Taktzyklen berechnet zu Dauer $=$ (TRFGA $\cdot$ 4) $+$ 1.

5.9.3.3 TVBH und TVBL

Dies ist einer der beiden Parameter, der in High- und Low-Byte verarbeitet werden muß (Abschnitt 5.9.3). TVBH und TVBL bilden gemeinsam einen 16-Bit-Wert, der den Teil der Vorwärmzeit bestimmt, der nicht durch RFGA abgedeckt ist, also von RFGA$\downarrow$ bis SEL_OUT$\uparrow$. Die Angabe erfolgt ebenfalls in Sequenzer-Taktzyklen, und es kann wieder gelesen und geschrieben werden:

- WTVBH (Port E122) lädt das High-Byte der Vorwärmzeit abzüglich der RFGA-Dauer in den Sequenzer (default ist 0);
- RTVBH (Port E122) liest den eingestellten Wert;
- WTVBL (Port E120) lädt das Low-Byte der Vorwärmzeit abzüglich der RFGA-Dauer in den Sequenzer (default ist 0);
- RTVBL (Port E120) liest den eingestellten Wert;

5.9.3.4 TSEQRESH und TSEQRESL

Dies ist der andere Parameter, der in High- und Low-Byte verarbeitet werden muß (Abschnitt 5.9.3). Auch TSEQRESH und TSEQRESL bilden gemeinsam einen 16-Bit-Wert. In diesem Falle bestimmt er die Dauer (in Taktzyklen), die der Sequenzer bei jedem CCD-Anfang im Grundzustand verbleibt (SEQuenzerRESet), also von SEL_OUT$\uparrow$ bis INIT$\uparrow$.

Seine Dauer in Taktzyklen berechnet sich ähnlich wie bei TRFGA als

$$\text{Dauer} = \text{TSEQRES} + 1.$$

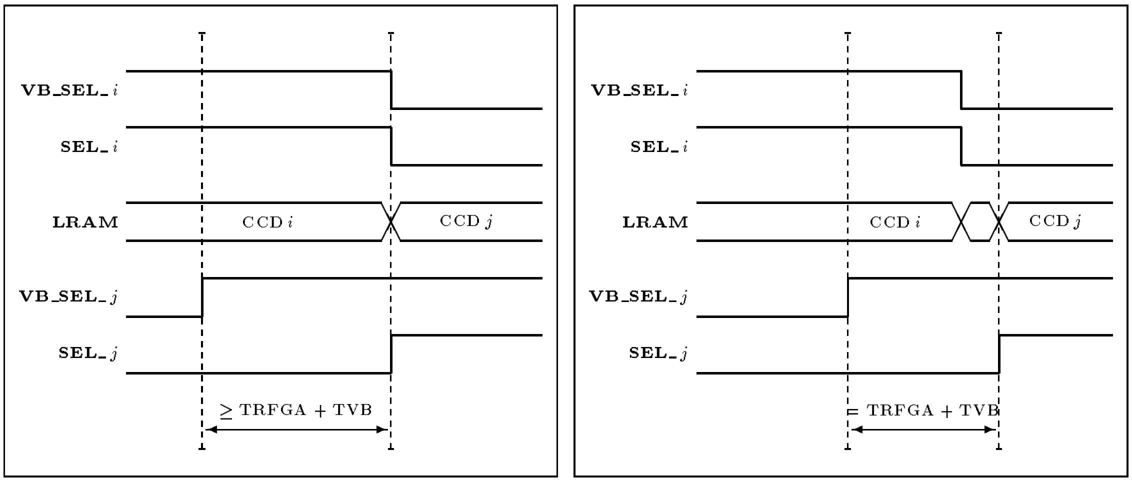
Umgekehrt gilt natürlich TSEQRES $=$ Dauer $-$ 1. Allerdings können keine negativen Werte eingestellt werden, weshalb zwischen SEL_OUT$\uparrow$ und INIT$\uparrow$ mindestens ein Takt abgewartet wird.

Dieser Reset betrifft nur die Zeilen- Signale; er stellt sicher, daß bei einem CCD-Wechsel und am Quadranten-Anfang die richtigen LRAM-Bits anliegen und die Daten-Pipeline richtig aufgebaut wird (Abschnitt 5.8). TSEQRES sollte daher zur Sicherheit auf wenigstens 4 stehen (Zugriffszeit des LRAMs!). Auch hier kann wieder gelesen und geschrieben werden:

- WTSEQRESH (Port E126) lädt das High-Byte der INIT-Verzögerungszeit abzüglich eines Taktes in den Sequenzer (default ist 0);
- RTSEQRESH (Port E126) liest den eingestellten Wert;

- WTSEQRESL (Port E124) lädt das Low-Byte der INIT-Verzögerungszeit abzüglich eines Taktes in den Sequenzer (default ist 0);
- RTSEQRESL (Port E124) liest den eingestellten Wert;

5.9.3.5 VB_EOCCD_ON und _OFF



(a) *CCD j wurde rechtzeitig vor CCD-Ende programmiert.*

(b) *CCD j wurde zu kurz vor CCD-Ende programmiert.*

Abbildung 5.2: *Das Einschaltverhalten zweier CCDs bei ausgeschaltetem VB_EOCCD. Das VB_SEL_j kommt sofort mit der Zeile, in der es im LRAM auftaucht, allerdings muß bis zum Einschalten von SEL_j die gesamte Vorwärmzeit verstrichen sein. Je nachdem, ob CCD j rechtzeitig programmiert wurde (a) oder so kurz vor CCD-Ende (b), daß die verbleibende Zeit nicht mehr ausreicht, schaltet SEL_j↑ gleichzeitig mit SEL_i↓ (a) oder erst später (b).*

Die Messungen aus Abschnitt 4.2 haben deutlich gezeigt, daß das Einschalten von VB_SEL_j während des Abarbeitens von CCD i zu Störungen führt. Um die Möglichkeit des zeitsparenden überlappenden Einschaltens nicht grundsätzlich zu verlieren, wurde dieser Befehl geschaffen: Er legt fest, ob VB_SEL_j sofort mit der Übernahme aus dem LRAM gegeben wird (Abbildung 5.2), oder ob es erst beim Ausschalten von SEL_i angeschaltet wird (Abbildung 5.3). In letzterem Falle muß sichergestellt sein, daß die richtige Kodierung von CCD j (Abschnitt 5.9.1.1) auch in der letzten Zeile anliegt, wenn VB_SEL_j mit EOCCD aktiviert werden soll (Abschnitte 5.7 und Seite 62). In jedem Falle muß die vollständige Vorwärm-

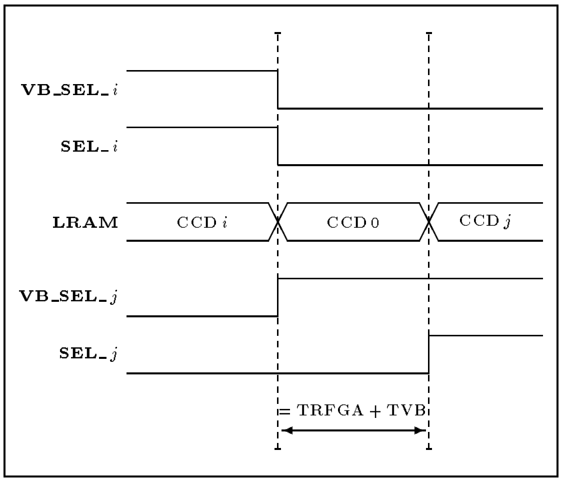


Abbildung 5.3: *Das Einschaltverhalten zweier CCDs bei eingeschaltetem VB_EOCCD.*

zeit abgelaufen sein, bevor SEL_j angeschaltet wird.

- VB_EOCCD_ON (Port E118) schaltet das nächste CCD erst mit dem CCD-Ende (EOCCD = End of CCD) des vorherigen ein;
- VB_EOCCD_OFF (Port E11a) schaltet das nachfolgende VB_SEL bereits mit dem Zeilenanfang der CCD-Zeile an, in der es erstmalig im LRAM auftaucht.

5.9.3.6 Die CLKMUX-Signale

Diese vier Befehle dienen dazu, den Übernahme-Zeitpunkt des Analogsignals in den ADC zu regeln. Es hat sich herausgestellt, daß die Signallaufzeit vom CAMEX zum ADC mit sinkender Temperatur kürzer wird. Daher ist es notwendig, das CLK_D_IN-Signal auch während des Fluges in 4 Stufen verzögern zu können, wie Abbildung 4.11 (S. 54) verdeutlicht. Absatz CLK_D_IN auf Seite 61 erklärt, daß dies geschieht, indem man eine der vier Varianten (CLK 0 bis 3) an einem Multiplexer (MUX) auswählt, dessen Auswahlsignale CLKMUX 0 und CLKMUX 1 wie folgt kommandiert werden:

- CLKMUX0_ON (Port E12c) schaltet CLKMUX0 auf 1;
- CLKMUX0_OFF (Port E12e) schaltet CLKMUX0 auf 0 (default).
- CLKMUX1_ON (Port E130) schaltet CLKMUX1 auf 1;
- CLKMUX1_OFF (Port E132) schaltet CLKMUX1 auf 0 (default).

5.9.3.7 Das Einstellen der Betriebsmodi

In Abschnitt 5.5.2 wurde ausführlich darauf eingegagen, wie sich die Einstellung des Betriebs-Modus vor allem auf die Adressierung des LRAMs auswirkt. Analog zu den CLKMUX-Signalen gibt es zwei Modus-Signale, deren in Tabelle 5.5 dargestellte Kombinationen den Betriebs-Modus bestimmen. Dies wird wieder duch An- bzw. Ausschalten der beiden Modus-Signale erreicht:

- SWM_ON (Port E110) Schaltet den Small-Window-Mode ein;
- SWM_OFF (Port E112) Schaltet den Small-Window-Mode aus (default).
- TMS_ON (Port E114) Schaltet den Timing-Mode ein;
- TMS_OFF (Port E116) Schaltet den Timing-Mode aus (default).

5.9.4 Die unterschiedlichen Test-Einstellungen

Damit das System auch während des Fluges wenigstens teilweise getestet werden kann, gibt es zwei weitere Einstellungs-Möglichkeiten:

5.9.4.1 SB_ON und _OFF

Für einen reinen Rauschtest der Elektronik müssen die Phi-Pulse auch jegliche Ladung von der Anode abziehen (Abschnitt 3.2.2). Dies zu erreichen ist Ziel dieses Befehls, dessen Voreinstellung die Daten natürlich vorwärts in Richtung auf die Anode und den CAMEX schiebt:

- SB_ON (Port E106) schaltet Shift Back an;
- SB_OFF (Port E108) schaltet Shift Back aus (default).

5.9.4.2 AMOS_en

Dieser Befehl dient zum Testen der Verstärkung (Abschnitt 3.2.2): Ist AMOS eingeschaltet, so wird anstelle der Phi-Pulse bis zum Ende der Zeile ein Steuerpuls gegeben, der eine definierte Ladungsmenge in die Übernahme-Anode zum CAMEX induziert. Diese „Testdaten“ werden durch die Elektronik (CAMEX, ADC, COM) geschleust, wodurch vor allem die ordnungsgemäße Funktion des CAMEX überprüft werden kann.

- AMOS_ON (Port E100) schaltet statt der Phi-Pulse den AMOS-Steuerpuls bis zum Zeilenende;
- AMOS_OFF (Port E102) läßt das CCD normal vor- oder rückwärts schieben (default).

Kapitel 6

Das Referenz-Design

Bereits in der Einführung zu Kapitel 5 wurde darauf hingewiesen, daß es bei der Entwicklung von ASICs nach ihrer Herstellung — außer durch eine neue Produktionsserie — nicht möglich ist, Änderungen am Design vorzunehmen. Daher ist es unumgänglich, die Schaltung vor der Produktion ausgiebig zu testen. Dies kann prinzipiell auf zwei verschiedene Arten geschehen:

1. Durch den Aufbau einer gleichwertigen Schaltung aus diskreten Bauelementen

und

2. durch Simulationen.

Beide Arten haben ihre Vor- und Nachteile: Eine diskrete Schaltung kann unter realen Bedingungen getestet werden, hat aber durch den deutlich größeren physikalischen Umfang deutlich größere Signal-Laufzeiten. Bei einer Simulation können die Laufzeiten hingegen den typischen ASIC-Signal-Laufzeiten angepaßt werden, aber es werden sicherlich nicht alle realen Bedingungen des gesamten Systems berücksichtigt.

Bei der Entwicklung des SECOM-ASICs wurden daher beide Testmöglichkeiten genutzt: Es wurde simuliert und es wurde eine Referenz-Version für diese Simulationen aus 3 FPGAs (Erklärung auf Seite 34) aufgebaut, die sich identisch verhalten sollte wie der spätere ASIC.

Die Verwendung dreier FPGAs für den diskreten Aufbau ergab sich aus der Komplexität und dem Umfang der Schaltung: Ein Aufbau aus normalen IC-„Käfern“ hätte eine Größe erreicht, die nicht mehr sinnvoll zu handhaben gewesen wäre. Dies gilt vor allem da während einer Testphase einige Änderungen zu erwarten sind, die bei einer derartigen Realisierung jedesmal ein Umlöten erforderlich gemacht hätten — bei einer Schaltung in dieser Größenordnung undenkbar! Die Wahl fiel daher auf die größten damals verfügbaren FPGAs, die XC 4013 – PG 223 von XILINX®. Trotzdem wurden davon noch 3 Stück benötigt: Der Quadrantensequenzer ist in dem QSEQ-FPGA untergebracht, während der Common-Mode-Filter auf die FPGAs COM und SORT verteilt wurde, wie es in Abbildung 3.23 bereits angedeutet ist.

Die durch die Verwendung dieser programmierbaren Bausteine gegebene Möglichkeit, Änderungen im Design einfach als neues Programm in diese FPGAs zu laden, haben die Arbeit während der Testphase allerdings ungemein erleichtert: Die gewünschte Design-Änderung wurde auf dem Rechner in die Schaltung eingebaut, das Programm-Paket, das die Übersetzung in den binären Programm-Code für das FPGA erledigt, mit der neuen Schaltung gefüttert und eine Weile später war die neue Version fertig zum Testen. Dabei hing diese „Weile“ deutlich von der verwendeten Entwicklungs-Umgebung ab:

Der Common-Mode-Filter war bereits für frühere Test-Aufbauten verwendet und daher als erster entwickelt worden. Nachdem das ORFEUS-Team am Institut bereits Erfahrungen mit dem ORCAD®-Programm-Paket gesammelt hatte, das XILINX® für FPGA-

Schaltungs-Entwicklung auf PCs zur Verfügung stellt, war dieser Teil der SECOM-Schaltung mit ORCAD® entwickelt worden. Hier sind mehrere Stunden für das „Routen" eines FPGAs notwendig, wie der Prozess genannt wird, der die benötigten Bauteile in dem FPGA verteilt („plazieren") und miteinander verbindet („routen"). Für COM- oder SORT-Änderungen wurde das Programm normalerweise vor Feierabend gestartet und über Nacht laufen gelassen; am nächsten Morgen konnten dann erste Tests ohne den CCD-Aufbau unternommen werden, nachdem das Bit-File mittels des „XCHECKERs" — ein Programm mit Spezialkabel zum Laden der FPGAs — direkt vom Rechner in das entsprechende FPGA geladen wurde. Für Tests mit dem CCD wurde das Bit-File typischerweise in ein PROM[1] gebrannt, aus dem das FPGA schon beim Anschalten der Versorgungs-Spannung geladen wird.

Vor allem im Hinblick auf die notwendigen Simulationen (Abschnitt 6.2), wurde der Quadrantensequenzer auf einer Workstation mit dem Programm-Paket Mentor® von Mentor Graphics Corporation MGC entwickelt, das sowohl zu den XILINX-Programmen und -Bibliotheken als auch zu den für den ASIC benötigten Paketen eine Schnittstelle besitzt, und neben dem „Design-Architekt" zum Erstellen der Schaltungen unter anderem auch den leistungsfähigen Simulator „quicksim" zur Verfügung stellt. Hier wurden in der Anfangsphase der QSEQ-Entwicklung ca. 45 Minuten, für die komplexere Endversion ca. 1,5 bis 2,5 Stunden für das Erstellen des Bit-Files benötigt.

Auf Seite 34 wurden die Auswirkungen der oben erwähnten Nachteile von Tests mit einer diskreten Schaltung bereits beschrieben; das fehlerfreie Arbeiten bei 20 MHz anstelle der erforderlichen 25 MHz wurde allerdings auch nur durch einen Trick erreicht, den sogenannten „Clock-Loop":

Nachdem Digital-Schaltungen typischerweise einen einzigen System-Takt haben (siehe die Einführung zu Kapitel 5), werden dafür in den Chips spezielle Leitungen vorgesehen. Diese Clock-Leitungen sind wesentlich leistungsfähiger ausgelegt als die normalen Signal-Leitungen, da sie ja alle getakteten Bausteine alle paar Nanosekunden mit einem neuen Pegel versorgen müssen. Auch die FPGAs haben solche Clock-Leitungen. Diese haben allerdings keine Verbindung zu den Gatter-Ausgängen, sondern können nur direkt von außen gespeist werden. Nun ist es aber unumgänglich, daß die QCLK das Ergebnis einer Veroderung von 1QCLK und 2QCLK ist, schließlich muß sichergestellt sein, daß sowohl der nominelle als auch der redundante Master-Sequenzer den Quadranten steuern können (Abschnitt 3.3.1). Es wurde schon relativ bald klar, daß der Ausgang eines ODERs nicht in der Lage ist, alle getakteten QSEQ-Bauteile mit der nötigen Geschwindigkeit zu steuern. Als Zwischenlösung wurde vorübergehend auf den redundanten Master verzichtet, bis die endgültige Lösung gefunden war. Sie ist am besten in Schaltplan „qseq_fpga_clk_loop" (S. 115) zu sehen: Die Taktsignale beider Master-Sequenzer gehen wie schon früher in dem Quadrantensequenzer auf ein ODER-Gatter. Dessen Ausgang wird nun allerdings als „FPGA_CLK wieder aus dem FPGA hinausgeführt, um dann als QCLK auf COM und QSEQ verteilt wieder reinzukommen, diesmal natürlich auf einer schnellen Clock-Leitung.

Dieser Trick ist allerdings nur im FPGA-Referenz-Design notwending; im ASIC kann dafür gesorgt werden, daß bereits die FPGA_CLK eine schnelle und leistungsfähige Clock-Leitung ist.

[1] **P**rogrammable **R**ead **O**nly **M**emory, ein Speicher-Baustein, der nur einmal geladen, aus dem aber beliebig oft gelesen werden kann.

6.1 Das reale und das virtuelle Referenz-Design

Nachdem für das FPGA- und das ASIC-Design völlig unterschiedliche Bauteile-Bibliotheken, teilweise mit unterschiedlichen Rahmenbedingungen verwendet werden müssen, ist es notwendig, beide Schaltungen in ihrer Gesamtheit zu simulieren und die Simulations-Ergebnisse des ASIC-Designs anhand derjenigen des FPGA-Referenz-Designs zu qualifizieren. Dies kann jedoch nur sinnvoll sein, wenn auch sichergestellt ist, daß sich das virtuelle, für die Simulationen verwendete Referenz-Design, genau wie sein reales Gegenstück verhält.

Um das Referenz-Design mit „quicksim“ simulieren zu können, muß es allerdings vollständig auf der Workstation vorliegen. Zu diesem Zweck wurden die ausgedruckten ORCAD®-Schaltpläne des COM- und des SORT-FPGAs von Hand im Design-Architekt abgemalt, und die 3 FPGAs sodann mühsam miteinander verbunden. „Mühsam“ zum einen, weil die Struktur aus 3 FPGAs erhalten bleiben mußte, da nur so auch die entsprechenden Bitfiles für den Vergleich zwischen virtuellem und realem Referenz-Design erzeugt werden können. Und „mühsam“ zum anderen, weil nicht alle FPGA-Schaltungen bereits während des Entwurfs für dieses Zusammenschalten vorbereitet worden waren, indem alle erforderlichen Anschlüsse auf oberster Ebene zur Verfügung gestellt worden wären.

Die Verwendung der XILINX-Bibliothek „XBLOX“ in allen Entwürfen hat zusätzliche Probleme bereitet: So angenehm es ist, während des Entwurfs nicht darauf achten zu müssen, welche Bitbreite ein Bus hat, sondern sich auf die wesentlichen Charakteristika der Signale konzentrieren zu können, so lästig ist es, daß Mentor® diese Busse daher als nur ein Bit breit interpretiert, bis die tatsächlich benötigte Busbreite bei den Vorbereitungen zur Erzeugung der Bitfiles definiert wird. Und selbst dann ist es noch notwendig ein zusätzliches Programm ablaufen zu lassen, das aus dem verwirrenden „flachen“ Design, in dem ein Bauteil neben das andere gesetzt ist und die Verbindungen nur durch die an die Ein- und Ausgänge geschriebenen Signal-Kürzel gegeben sind, wieder einen für Menschen lesbaren Schaltplan generiert. Von der zukünftigen Verwendung von XBLOX sei daher dringend abgeraten, vor allem, da XILINX diese Bibliothek auch nicht weiter unterstützen will!

Nachdem die FPGAs alle in Mentor® vorhanden und auch verbunden waren, wurden die wichtigsten ASIC-externen Bauteile der SECOM-Karte — das CPU-FIFO und die Line- und Offest-RAMs — hinzugefügt, sowie ein weiteres RAM, „ADC_RAM“, das die Roh-Daten aus dem ADC simuliert. Es wird mit der ORAM-Adresse angesteuert, nachdem diese für die Offset-Korrektur ja immer mit den gleichen Pixeln arbeitet, wie sie der ADC liefert (Abschnitt 3.3.4.1). Zu erkennen ist dies alles in Schaltplan „toplevel“ auf Seite 110. Damit waren alle Voraussetzungen geschaffen, so realitätsnah wie möglich zu simulieren.

6.2 Die Simulationen

Für die Simulationen wurde der weiter oben erwähnte, im Mentor®-Paket mitgelieferte Simulator „quicksim“ verwendet. Er erlaubt sowohl rein funktionale Simulationen, wo allein der logische Zusammenhang ohne die Signal-Laufzeiten berücksichtigt wird, als auch zeitliche Simulationen mit minimalen, maximalen oder typischen Gatter-Laufzeiten zusätzlich zu den Laufzeiten zwischen den Bauteilen. Dabei wird mit einer maximalen Zeitauflösung von 0,1 ns für die Simulationsdaten gearbeitet, was zwar eine Vervielfachung der Simulationszeit um mehrere Größenordnungen im Vergleich zur realen Zeit zur Folge hat —

schließlich muß er jedes Signal durch die gesamte Schaltung verfolgen und das Zusammenspiel der Signale berechnen — aber Schwachpunkte der Schaltung recht sicher sichtbar macht.

Außerdem bietet „quicksim" die Möglichkeit, den Wert jeder beliebigen Leitung zu jedem beliebigen Zeitpunkt nicht nur anzuschauen, sondern auch vorzugeben oder Unbestimmtheiten zu ihrem Ursprung zurückzuverfolgen, egal ob es sich dabei um eine interne oder um eine nach außen geführte Leitung handelt. Dies wird unterstützt durch eine umfangreiche, vor allem farbliche Kodierung der Ausgabe von Schaltplänen, Zeitdiagrammen oder Wertelisten der Signale, wobei die ersten beiden vornehmlich bei der interaktiven Simulation verwendet werden, wohingegen die in Dateien geschriebenen Listen die Grundlage für den Vergleich von Referenz- und ASIC-Design bildeten.

Um sowohl die Simulationszeit möglichst kurz als auch die List-files möglichst klein zu halten, war es nicht sinnvoll die Hardware-Tests aus dem Labor unverändert für die Simulationen zu übernehmen: Während ein Quadranten-Durchlauf im Labor nur Bruchteile von Sekunden dauert, benötigt der „quicksim" für einen Quadranten-Ablauf im Timing-Mode ca. 2, im Small-Window-Mode sogar um die 4 Stunden. Erklärtes Ziel war es daher Simulations-Files zu erstellen, die alle Bereiche der gesamten Schaltung mit möglichst wenig Zeitaufwand möglichst gründlich testen. Dabei sollten auch (in ihrer Kombination) sinnlose Eingangswerte angenommen werden, wie sie beim Einschalten oder durch eine Fehlbedienung entstehen können, damit eine Zerstörung irgendeines Bauteils des EPIC-Systems in diesen Fällen möglichst ausgeschlossen werden kann.

Vorteilhaft wäre es außerdem, wenn die Simulationen ohne die interaktive Simulations-Oberfläche ablaufen könnten, da dies dem Rechner zusätzliche Kapazitäten frei macht, und die Simulation damit beschleunigt. Dazu sind allerdings einige Vorarbeiten notwendig: Zunächst muß jeder CPU-Befehl in ein sogenanntes „force"-File übersetzt werden, in dem die Abfolge der CPU-Signale vorgegeben wird. Sodann müssen die RAM-Inhalte in den sogenannten „Model"-Files zusammengestellt werden, was vor allem für das LRAM mehrere Dateien erfordert. Und zuletzt müssen diese Elemente in den „Batch"-Files zu ganzen Simulations-Abläufen zusammengestellt werden, die vollkommen selbständig simulieren und die Ergebnisse der erwünschten Simulations-Zeitpunkte (typischerweise 0,1 ns vor QCLK$\uparrow$) in die richtigen List-Files schreiben. Ein Überblick über diese Files und eine grobe Beschreibung ihrer jeweiligen Test-Aufgaben findet sich in Anhang C.

Die Ergebnisse der Simulationen haben für den Quadrantensequenzer nur wenige Überraschungen gebracht. Die einzig erwähnenswerte tritt bei der Kombination der beiden LRAM-Bits TS und FS (Abschnitte 5.5.1 und 5.4, bzw. 5.9.1), eine Kombination, die während des normalen Betriebes nicht unbedingt sinnvoll ist: Wird ein schnell geschobenes Timing Shift programmiert (TS = FS = 1), so arbeitet die Phi-Schaltung wie man es intuitiv erwartet: Sie schiebt kontinuierlich und zählt den LRAM-Adresszähler auch wirklich nur jedes 10. Mal um eine Adresse weiter (Abschnitte 5.5 und 5.5.2). Allerdings holt er sich trotzdem am Ende jeden Schiebevorgangs die Steuerbits erneut in das RAMREG, und läßt dabei auch den SEQADR wieder von vorne beginnen. Dies hat allerdings zur Folge, daß mit jedem Schiebevorgang auch ein $\overline{\text{NEWLINE}}$ gegeben wird, und die Zeilenzähler im Common-Mode-Filter (Abschnitt 3.3.4.1) daher die einfachen CCD-Zeilen und nicht die Makro-Zeilen zählen. Dies kann zwar für manche Anwendungen nützlich sein, man darf dabei aber nie vergessen, daß auf diese Weise LRAM- und Pixel-Adresse mit einem Faktor 10 auseinanderlaufen. Ist zudem das RFGA_en-Bit gesetzt, so gilt das gleich für das Rücksetzen der CCD-Anode: Sie wird vor jedem einfachen und nicht vor jedem 10. Schiebevorgang rückgesetzt, da das RAMREG ja auch mit jedem Schiebevorgang geladen wird (Abschnitt 5.6).

Kapitel 7

Zusammenfassung

Inhalt dieser Diplomarbeit war die Entwicklung des Quadrantensequenzers für die Flugversion des pn-CCD-Kamera-Systems an Bord der Röntgenastronomie-Satelliten XMM und ABRIXAS.

Als Voraussetzung dafür wurde auf die spezifischen Anforderungen eingegangen, die an die Schaltung des Quadrantensequenzers (QSEQ) gestellt wurden. Dabei wurden zwei Schwerpunkte herausgearbeitet: Der eine Schwerpunkt besteht aus einer Beschreibung der Betriebsmodi der pn-CCD-Kamera; der andere in der Darstellung der Zusammenhänge des Quadrantensequenzers mit den anderen Komponenten der Röntgenkamera. Besonderer Wert wurde auf die Erklärung der Funktionalität der vom QSEQ zu steuernden Hardware gelegt.

Die endgültige Sequenzer-Schaltung wurde daraufhin in einer Ausführlichkeit beschrieben, die es auch noch in einigen Jahren ermöglichen soll, die Motivationen für diese Realisierung nachzuvollziehen.

Mit der ersten Version dieses Quadrantensequenzers konnte zum ersten Mal ein ganzer Quadrant angesteuert und ausgelesen werden. Bei den dann erfolgten Tests und Messungen sind einige Probleme aufgetreten; es waren dies zu einem Großteil Laufzeitprobleme zwischen dem Event-Analyzer EPEA und Kamerakopf, aber auch unerwartete Störungen beim Einschalten der CCDs. Beides hatte zur Folge, daß weitere Parameter eingeführt werden mußten, die auch während des Fluges vom Boden aus eingestellt werden können.

Ein weiteres Thema dieser Arbeit war die Umsetzung des FPGA-Testdesigns auf die Bibliothek des ASIC-Herstellers Temic (Matra Harris). Neben den Schwierigkeiten, die bei dieser Umsetzung auftauchten, wurde vor allem auf die Notwendigkeit und die Durchführung der Simulationen eingegangen.

Während alle bisherigen Tests mit der FPGA-Version der SECOM-Karte durchgeführt wurden, hat in diesen Tagen der Herstellungs-Prozess der ASICs begonnen. Die ersten Chips werden Mitte August 1997 erwartet und sollen dann bereits bei den Tests im September des selben Jahres verwendet werden.

Als Schlußbemerkung sei erwähnt, daß die in dieser Arbeit beschriebene SECOM-Karte nicht nur auf den Satelliten XMM und ABRIXAS fliegen wird, sondern auch in anderen Bereichen der Physik eingesetzt werden kann. So verwendet z.B. die Quanten-Optik, ein Max-Planck-Institut in München, die FPGA-Realisierung der SECOM-Karte bereits seit einigen Monaten bei ihren Messungen.

Anhang A

Aufstellung der CPU-Befehle und LRAM-Bits

Name	**Port**	**Parameter**	**Erläuterungen**
AMOS_ON	E100	—	Schaltet Testpuls AMOS an Stelle der Phi-Pulse bis zum Zeilenende ein
AMOS_OFF	E102	—	bzw. aus (default)
CLKMUX0_ON	E12c	—	Diese 4 Befehle kodieren die
CLKMUX0_OFF	E12e	—	Verzögerung von CLK_D_IN.
CLKMUX1_ON	E130	—	Ihr Bit-Muster ergibt die Taktanzahl
CLKMUX1_OFF	E132	—	der Verzögerung (ON = 1; OFF = 0)
CLRSYS	E10a	—	Läßt den Sequenzer von vorne anfangen, schaltet ihn also in den Grundzustand
COMONS	E104	—	Schaltet LRAM-Zugriff von CPU (default) auf Sequenzer. Während die CPU den Zugriff hat, wartet der Sequenzer im Grundzustand
RFGA_ON	E10c	—	Erlaubt RFGA während Vorwärmphase (default)
RFGA_OFF	E10e	—	Schaltet Vorwärm-RFGA ab
RTRFGA	E128	—	Liest RFGA-Taktanzahl $\div$ 4
RTSEQRESH	E126	—	Liest die oberen 8 Bit der Taktanzahl zwischen SEL_OUT$\uparrow$ und INIT$\uparrow$
RTSEQRESL	E124	—	Liest die unteren 8 Bit der Taktanzahl zwischen SEL_OUT$\uparrow$ und INIT$\uparrow$. Sollte auf mindestens 4 stehen
RTVBH	E122	—	Liest die oberen 8 Bit der Taktanzahl zwischen Vorwärm-RFGA$\downarrow$ und SEL_OUT$\uparrow$
RTVBL	E120	—	Liest die unteren 8 Bit der Taktanzahl zwischen Vorwärm-RFGA$\downarrow$ und SEL_OUT$\uparrow$
SB_ON	E106	—	Schaltet Rückwärtsschieben (Shift Back) ein
SB_OFF	E108	—	bzw. aus (default)
SWM_ON	E110	—	Schaltet Small-Window-Mode ein
SWM_OFF	E112	—	bzw. aus (default)
TMS_ON	E114	—	Schaltet Timing-Mode ein
TMS_OFF	E116	—	bzw. aus (default)
VB_EOCCD_ON	E118	—	Läßt VB_SEL erst am Ende des vorherigen CCDs einschalten

Name	**Port**	**Parameter**	**Erläuterungen**
VB_EOCCD_OFF	E11a	—	VB_SEL des nächsten CCDs kommt am Anfang der CCD-Zeile, die zu der LRAM-Adresse gehört, wo das nächste VB_SEL auftaucht (default)
WTRFGA	E128	Bit 7:0	Lädt Taktanzahl jeglichen RFGA- Pulses $\div$ 4 in den Sequenzer
WTSEQRESH	E126	Bit 7:0	Lädt oberen 8 Bit der Taktanzahl zwischen SEL_OUT$\uparrow$ und INIT$\uparrow$ in den Sequenzer
WTSEQRESL	E124	Bit 7:0	Lädt unteren 8 Bit der Taktanzahl zwischen SEL_OUT$\uparrow$ und INIT$\uparrow$ in den Sequenzer. Die Gesamtdauer sollte mindestens 4 Takte sein
WTVBH	E122	Bit 7:0	Lädt oberen 8 Bit der Taktanzahl zwischen Vorwärm-RFGA$\downarrow$ und SEL_OUT$\uparrow$ in den Sequenzer
WTVBL	E120	Bit 7:0	Lädt unteren 8 Bit der Taktanzahl zwischen Vorwärm-RFGA$\downarrow$ und SEL_OUT$\uparrow$ in den Sequenzer.

Tabelle A.1: *Die CPU-Befehle im Überblick.*

Bit	**Name**	**Erklärung**
0	VB_0	Kodiert zusammen mit VB_1, welches CCD als nächstes eingeschaltet werden soll
1	VB_1	Kodiert zusammen mit VB_0, welches CCD als nächstes eingeschaltet werden soll
2	RFGA_en	schaltet das zusätzliche „Zeilen-“RFGA an
3	LMC	Schaltet am Ende dieser Zeile den Line-Mode- Counter für die nächste Zeile (Pipeline-Auf- und Abbau)
4	FLUSH_en	FLUSH in dieser Zeile während des Timing-S_{in}'-Pulses
5	FS	Fast-Shift für das schnelle Schieben im Burst- oder einem Window-Modus
6	TS	Timing-Shift erlaubt das Verarbeiten der gesammelten Daten aus 10 Zeilen, ohne in den Timing-Mode gehen zu müssen
7	TIMFLG	TIMeFLaG setzt die LRAM-Adressierung zurück, schiebt die Zeitinformation in den Datenfluß und meldet dem Master-Sequenzer, daß dieser Quadrant fertig ist

Tabelle A.2: *Die LRAM-Bits im Überblick.*

Anhang B

Die verschiedenen Betriebsmodi

B.1 Die Programmierung des LRAMs

Als zusätzliche Hilfe für den LRAM-Programmierer, sollen vor der Darstellung des LRAM-Inhalts und der für den Quadrantensequenzer notwendigen CPU-Befehle noch einige Tips gegeben werden, die sich bei der Programmierung bewährt haben:

- **Pipeline** (Abschnitte 3.4 und 5.2): Eines der Hauptaugenmerke ist auf den richtigen Pipeline-Auf- und -Abbau zu richten: Grundsätzlich ist am Anfang jedes normalen CCD-Auslesens in den ersten 2 Zeilen ein LMC zu geben. Außerdem sind zwei zusätzliche Zeilen zu programmieren, in denen ebenfalls das LMC-Bit gesetzt sein muß (Bei einem CCD-Wechsel muß in der allerletzten Zeile nicht unbedingt in L0 geschaltet werden, da dies auch mit dem SEQRES, Abschnitt 5.8, geschieht. Bei einem Neuanfang desselben CCDs ist dies außer im SWM und Timing-Mode unbedingt notwendig damit der nächst Ablauf auf jeden Fall im richten Line-Mode beginnt). Wichtig ist in jedem Falle, daß das CCD nicht mit einer Fast-Shift-Zeile endet, da sonst das normale Zeilen- und damit CCD-Ende fehlen würde, das das nächste CCD einschaltet bzw. dem Master-Sequenzer das Quadrantenende mitteilt (Abschnitt 5.4).

- **CCD-Wechsel/CCD-Neuanfang** (Abschnitt 5.5.2 Ende): Ein weiterer Punkt, der nicht vergessen werden sollte, ist der zusätzliche LRAM-Eintrag am CCD-Ende, mindestens wenn es sich um einen Neuanfang handelt.

- **Erstes CCD** (Abschnitt 5.8): Außer im Small-Window-Mode, wo das als erstes einzuschaltende CCD implizit im ersten LRAM-Eintrag enthalten ist, muß die Codierung für das erste CCD (vergl. 5.9.1.1) an CPU-Adresse 79ffe, also der LRAM-Zelle direkt vor Block 2 stehen. Dies ermöglicht es, in allen Modi jedes beliebige CCD als erstes einzuschalten.

- **Programmier-Strategie:** Es hat sich bewährt, bei der Programmierung des Line-RAMs CCD-weise vorzugehen: Zunächst werden alle das CCD i betreffenden Speicherzellen mit $0\,i$ gefüllt, und erst danach die LMCs, das TIMFLG und etwaige optionale Bits gesetzt, wobei die Codierung für das jeweilige CCD nicht vergessen werden darf.

B.1.1 Der Fullframe-Mode

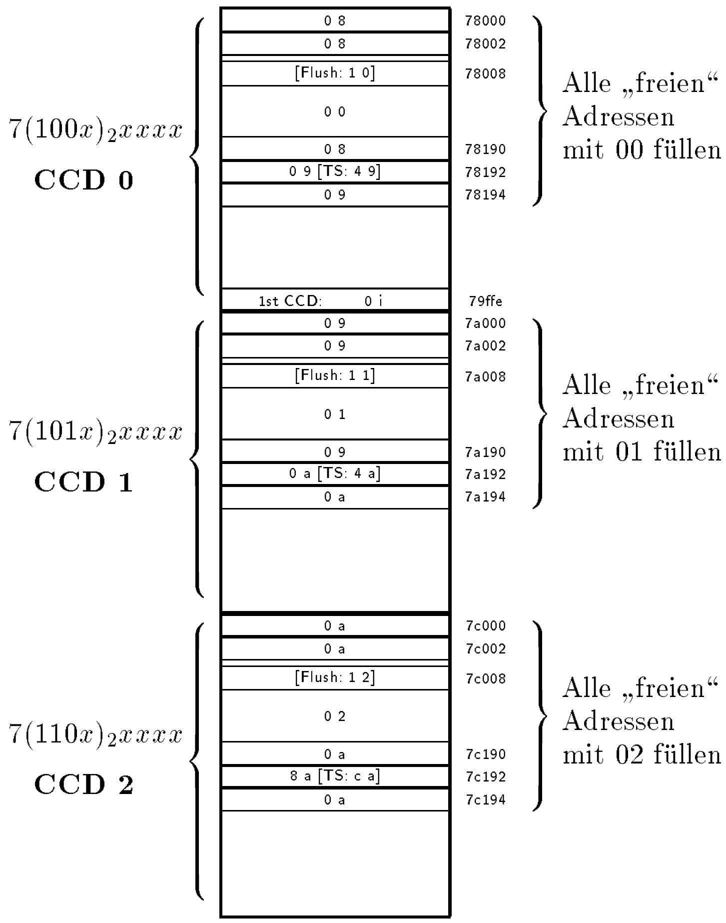


Abbildung B.1: *Die vollständige Programmierung des Fullframe-Modes. Sollte mit VB_EOCCD_OFF gearbeitet werden, so muß spätestens ab* $7(8+i)180$ *das nächste CCD* j *eingeschaltet sein, indem statt* $0i$ *eben* $0j$ *bzw* $0(j+8)$ *geschrieben wird. Der Master-Sequenzer ist auf serielles Auslesen eingestellt.*

WTRFGA WTSEQRESH WTSEQRESL WTVBH WTVBL	VB_EOCCD_ON SWM_OFF *(default)* TMS_OFF *(default)* RFGA_ON *(default)*	*Die Programmierung des Common-Mode-Filter samt ORAM (untere* Schwellen *für die verschiedenen CCDs, Common-Mode-Offset,* MEDON *für Mediankorrektur und zuletzt* COMONC)
CLKMUX0_ON/_OFF	SB_OFF *(default)*	
CLKMUX1_ON/_OFF	AMOS_OFF *(default)*	COMONS

Tabelle B.1: *Die CPU-Befehle des Fullframe-Modes*

B.1.2 Der Timing-Mode

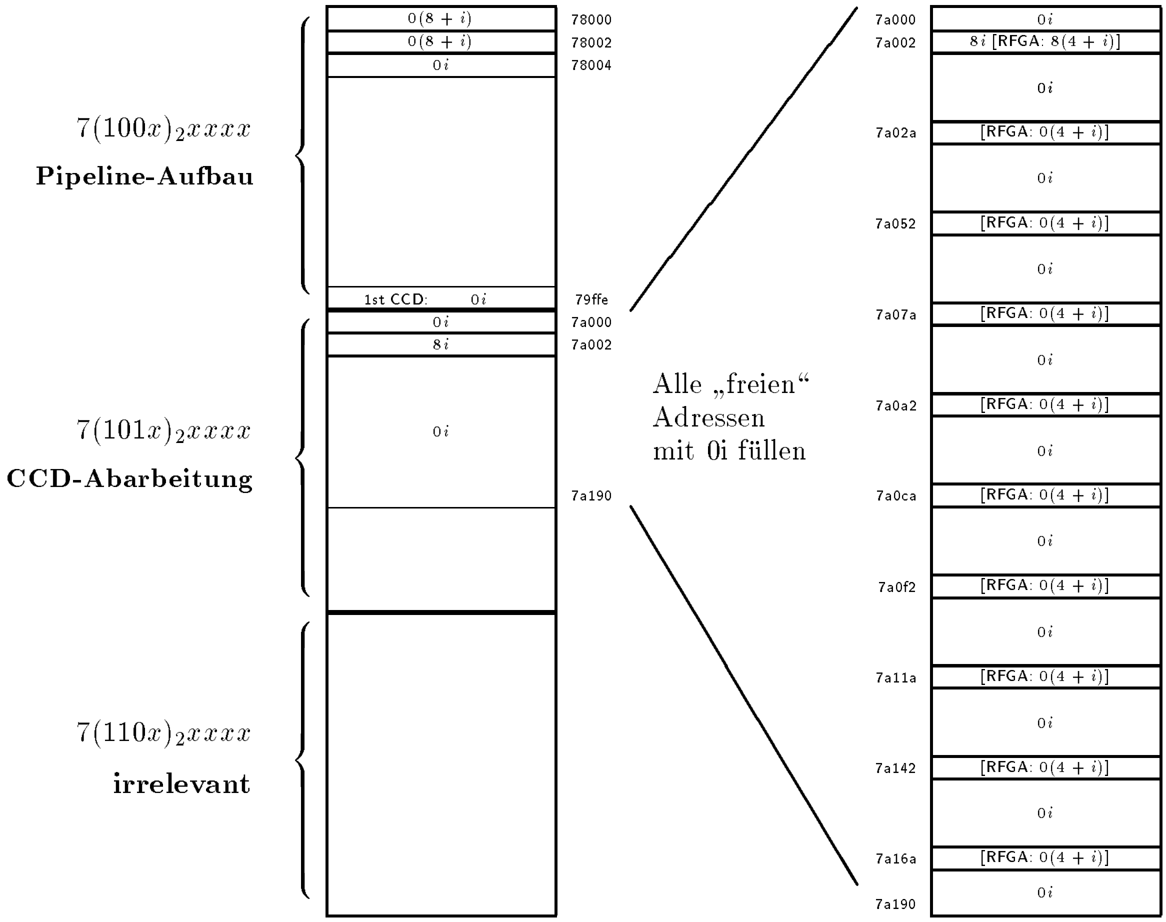
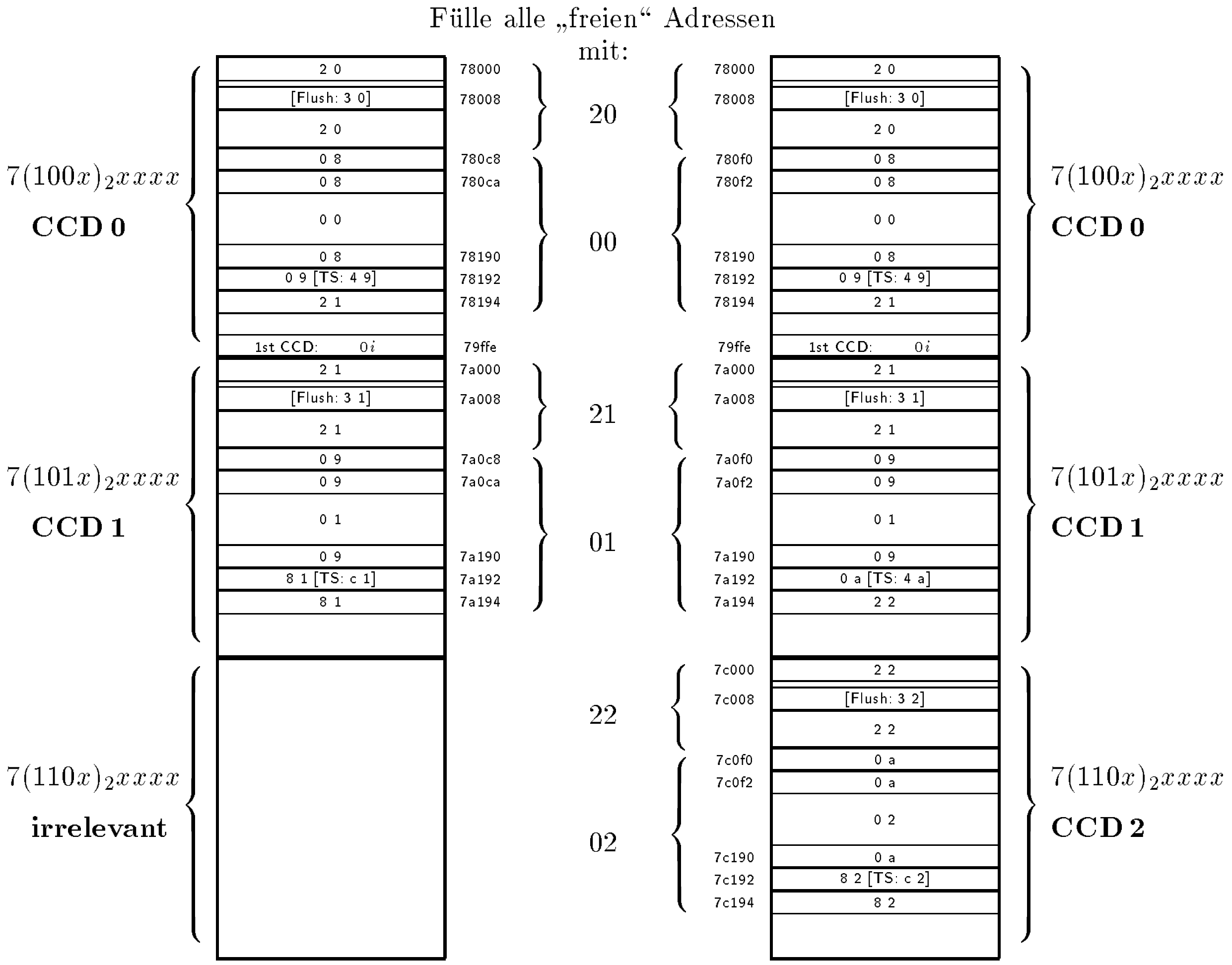


Abbildung B.2: *Die vollständige Programmierung des Timing-Modes. Die Pipeline wird in Block 0 aufgebaut, während das zyklische Auslesen in Block 1 programmiert ist. Die optionalen zusätzlichen RFGAs, die über das CCD verteilt gegeben werden können, sind im rechten Bildteil aufgezeigt. Der Master-Sequenzer ist auf paralleles Auslesen eingestellt.*

WTRFGA	VB_EOCCD_ON	*Die Programmierung des Common-Mode-Filter samt ORAM (untere* Schwellen *für die verschiedenen CCDs, Common-Mode-Offset,* MEDON *für Mediankorrektur und zuletzt* COMONC)
WTSEQRESH		
WTSEQRESL	SWM_OFF *(default)*	
WTVBH	**TMS_ON**	
WTVBL		
	RFGA_ON *(default)*	
CLKMUX0_ON/_OFF	SB_OFF *(default)*	
CLKMUX1_ON/_OFF	AMOS_OFF *(default)*	COMONS

Tabelle B.2: *Die CPU-Befehle des Timing-Modes.*

B.1.3 Der Large-Window-Mode



(a) *Das Line-RAM im Large-Window-Mode ohne CCD 2*

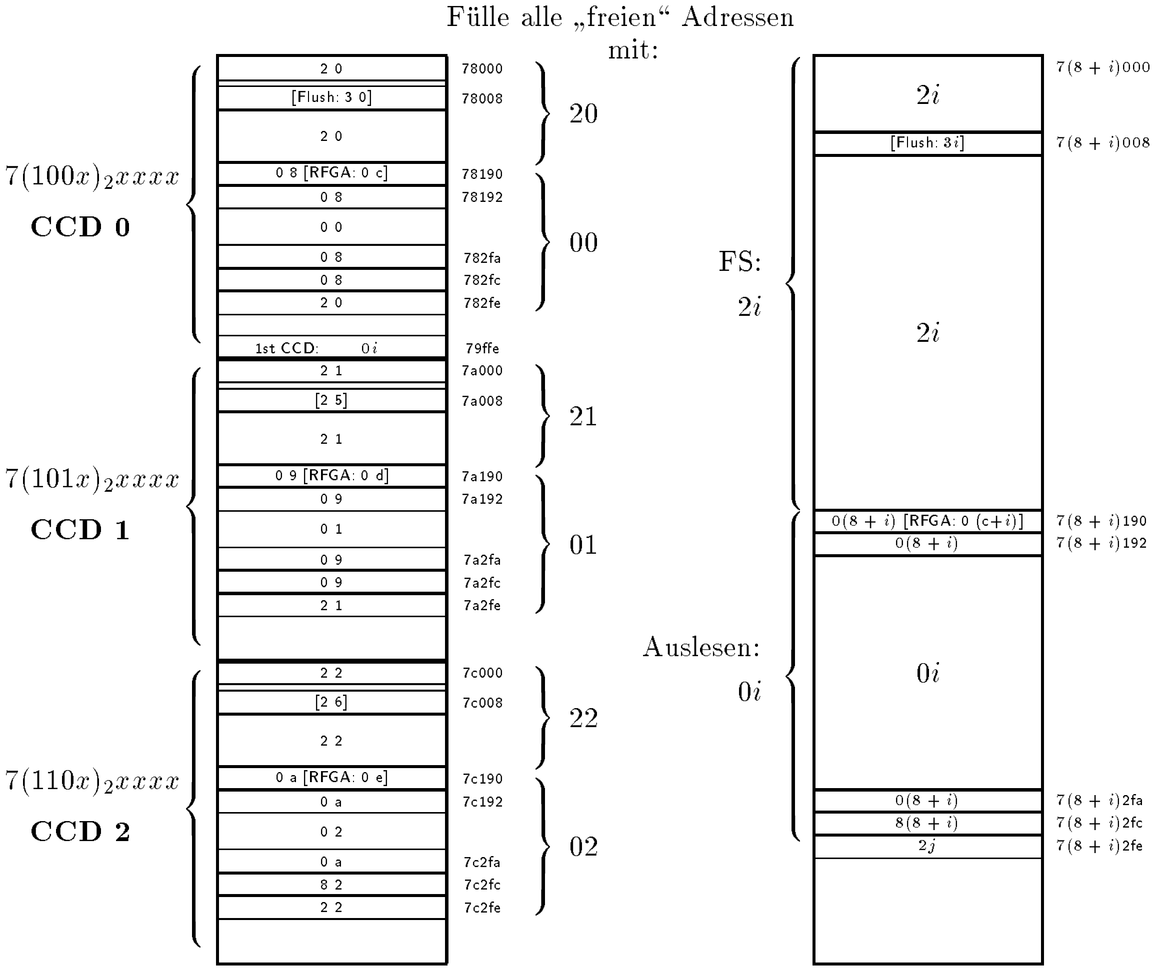
(b) *Das Line-RAM im Large-Window-Mode mit CCD 2*

Abbildung B.3: *Die vollständige Programmierung des Large-Window-Modes. Rechts ohne und links mit Verwendung von CCD 2. a: Das Fenster ist 2 CCDs mal 100 Zeilen groß; b: Das Fenster ist 3 CCDs mal 80 Zeilen groß. Der Master-Sequenzer ist auf serielles Auslesen eingestellt.*

WTRFGA	VB_EOCCD_ON	*Die Programmierung des Common-Mode-Filter samt ORAM (untere* Schwellen *für die verschiedenen CCDs, Common-Mode-Offset,* MEDON *für Mediankorrektur und zuletzt* COMONC)
WTSEQRESH		
WTSEQRESL	SWM_OFF *(default)*	
WTVBH	TMS_OFF *(default)*	
WTVBL		
	RFGA_ON *(default)*	
CLKMUX0_ON/_OFF	SB_OFF *(default)*	
CLKMUX1_ON/_OFF	AMOS_OFF *(default)*	COMONS

Tabelle B.3: *Die CPU-Befehle des Large-Window-Modes*

B.1.4 Der Burst-Mode



(a) *Das Line-RAM im Burst-Mode, wobei alle CCDs programmiert sind*

(b) *Das Line-RAM eines CCDs im Burst-Mode*

Abbildung B.4: *Die vollständige Programmierung des Burst-Modes. Da jeweils nur ein CCD eingeschaltet ist, muß auch nur ein Block programmiert werden; welcher Block dies ist bestimmt das aktive CCD, das an Adresse 79ffe codiert ist. Es wurde davon ausgegangen, daß die letzten 20 Zeilen überbelichtet sind und daher nicht ausgewertet werden. Der Master-Sequenzer ist auf paralleles Auslesen eingestellt.*

WTRFGA WTSEQRESH	VB_EOCCD_ON	*Die Programmierung des Common-Mode-Filter samt ORAM (untere* Schwellen *für die verschiedenen CCDs, Common-Mode-Offset,* MEDON *für Mediankorrektur und zuletzt* COMONC)
WTSEQRESL	SWM_OFF *(default)*	
WTVBH	TMS_OFF *(default)*	
WTVBL		
	RFGA_ON *(default)*	
CLKMUX0_ON/_OFF	SB_OFF *(default)*	
CLKMUX1_ON/_OFF	AMOS_OFF *(default)*	COMONS

Tabelle B.4: *Die CPU-Befehle des Burst-Modes.*

B.1.5 Der Small-Window-Mode

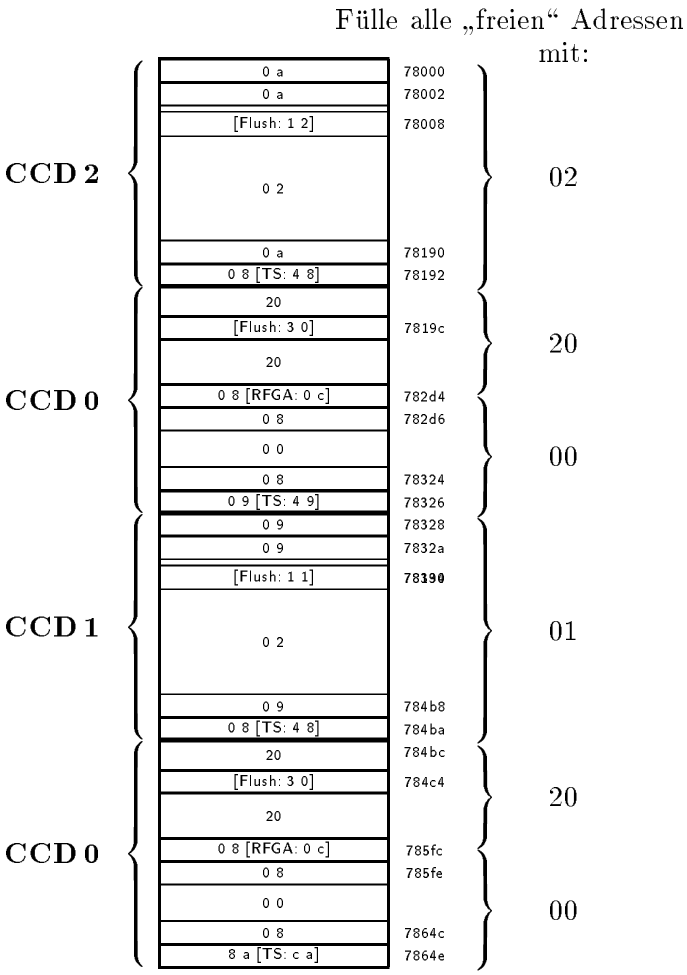


Abbildung B.5: *Die vollständige Programmierung des Small-Window-Modes. Man beachte, daß dies der einzige Modus ist, in dem das Line-RAM nicht blockweise programmiert wird. In diesem Beispiel ist das Fenster in CCD 0 40 Zeilen groß. Durch die zyklische Abarbeitung der CCDs ist es eigentlich egal mit welchem CCD begonnen wird; hier ist es CCD 2. Der Master-Sequenzer ist auf paralleles Auslesen eingestellt.*

WTRFGA	VB_EOCCD_ON	*Die Programmierung des Common-Mode-Filter samt ORAM (untere* Schwellen *für die verschiedenen CCDs, Common-Mode-Offset,* MEDON *für Mediankorrektur und zuletzt* COMONC)
WTSEQRESH		
WTSEQRESL	**SWM_ON**	
WTVBH	TMS_OFF *(default)*	
WTVBL		
	RFGA_ON *(default)*	
CLKMUX0_ON/_OFF	SB_OFF *(default)*	
CLKMUX1_ON/_OFF	AMOS_OFF *(default)*	COMONS

Tabelle B.5: *Die CPU-Befehle des Small-Window-Modes*

B.2 Die Auslesezeiten der verschiedenen Betriebsmodi

Fläche	**Berechnung**	**Takte**	**25 MHz**	**20 MHz**
Die Basiseinheiten:				
Normale Zeile (NL)		576	23,05 μs	28,80 μs
Fast-Shift (FS)		18	0,72 μs	9,00 μs
Timing-Shift (TS)	$NL + 9FS$	738	29,52 μs	37,64 μs
Der Fullframe-Mode:				
CCD	202 NL	116 352	4 654,08 μs	5 817,60 μs
Quadrant	3 CCDs	349 056	13,962 ms	17,453 ms
Detektor	4 Quadranten	4 349 056	**55,85 ms**	69,81 ms
Der Large-Window-Mode (Je nach Fensterlage):				
CCD	$100FS + 102NL$	60 552	2 422,08 μs	3 027,60 μs
Quadrant	2 CCDs	121 104	4 844,16 μs	6 055,20 μs
Detektor	4 Quadranten	484 416	**19,38 ms**	24,22 ms
Der Small-Window-Mode (Je nach CCD 0-Fenster):				
CCD 0	$160FS + 42NL$	27 072	1 082,88 μs	1 353,60 μs
CCD 1/2	202 NL	116 352	4 654,08 μs	5 817,60 μs
Quadrant	$2CCD\,0 + 2CCD\,1$	286 848	**11,474 ms**	14,342 ms
Detektor = Quadrant				
Der Timing-Mode (ohne Pipeline-Aufbau):				
CCD	200 TS	147 600	**5,904 ms**	7,380 ms
Quadrant = Detektor = CCD				
Der Burst-Mode:				
CCD	$200FS + 202NL$	108 432	**4,337 ms**	5,422 ms
Quadrant = Detektor = CCD				

Tabelle B.6: *Die Auslesezeiten bei 20 und 25 MHz. Bei den Berechnungen wurden die variablen Pausen zwischen den CCDs und zwischen den Quadranten nicht berücksichtigt!*

Anhang C

Die Simulations-Files

Zunächst seien die erwähnten CPU-Files aufgelistet, die es ermöglichen die CPU-Befehle analog wie bei den Hardware-Tests auch in den Simulations-Abläufen zu kommandieren. Die Dauer jedes Befehls beläuft sich auf 2000 ns; eine Ausnahme bilden die Befehle WCOMONC_ und WCOMONS_, die jeweils 3000 ns Simulationszeit benötigen.

AMOS_OFF	RELEV0_	VB_EOCCD_OFF
AMOS_ON	RELEV1_	VB_EOCCD_ON
CLKMUX0_OFF	RELEV2_	WCHOP_
CLKMUX0_ON	RFCPU_	WCLRECNT
CLKMUX1_OFF	RFGA_OFF	WCLRF
CLKMUX1_ON	RFGA_ON	WCOFFS_
CLRSYS	RMEDS_	WCOMONC_
EOT_WRITE_READ	RTRFGA_	WCOMONS_
FIFOFLG	RTSEQRESH_	WELEV0_
LRAMRD	RTSEQRESL_	WELEV1_
LRAMWR	RTVBH_	WELEV2_
MEDOFF_	RTVBL_	WSEND_
MEDON_	SB_OFF	WTRFGA_
ORAMRD	SB_ON	WTSEQRESH_
ORAMWR	SWM_OFF	WTSEQRESL_
RCHOP_	SWM_ON	WTVBH_
RCOFFS_	TMS_OFF	WTVBL_
RECNT_	TMS_ON	

Als nächstes sollen die Batch-Files und damit der eigentliche Inhalt der Simulationen zusammengestellt werden. Sie unterteilen sich in drei Testgruppen, von denen nur auf die QSEQ-Gruppe etwas näher eingegangen werden soll:

1. Die Datenpfade (testet hauptsächlich die Funktionalität des CPU-Busses und damit des Decoders):

   datapath.bat   datapath_asic.bat

2. Die Common Mode Funktionalität:

accu.bat	accu_asic.bat
ccd_id.bat	ccd_id_asic.bat
encnt.bat	encnt_asic.bat
fifsyn.bat	fifsyn_asic.bat
send.bat	send_asic.bat
transparent.bat	transparent.bat
chop.bat	chop_asic.bat
linecounter.bat	linecounter_asic.bat
timestamp.bat	timestamp_asic.bat

3. Die Quadrantensequenzer-Funktionalität unterteilt sich in:

   (a) Das Testen der Parameter-Zähler bei der kritischen Parameter-Einstellung einer „1 mit lauter Nullen“:
   ccd_on.bat

   (b) Die verschiedenen Reset-Möglichkeiten, einschließlich einer Umprogrammierung des LRAMs und der Verwendung beider Master-Sequenzer:
   reset.bat

   (c) Verschiedene Fast-Shift-Kombinationen. Dabei ist das LRAM für jedes CCD anders programmiert ist, wie folgende Auszüge der entsprechenden Model-Files zeigen:
   CCD0:
   4000 / 20 ; # FS
   4001 / 24 ; # FS, RFGA
   4002 / 30 ; # FS, FLUSH
   4003 / 62 ; # FS, TS, CCD2 vorwärmen
   4004 / 34 ; # FS, FLUSH, RFGA
   4005 / 64 ; # FS, TS, RFGA
   4006 / 70 ; # FS, TS, FLUSH
   4007 / 74 ; # FS, TS, FLUSH, RFGA
   4008 / 20 ; # FS
   CCD1:
   5000 / 31 ; # FS, FLUSH
   5001 / 65 ; # FS, TS, RFGA
   5002 / 71 ; # FS, TS, FLUSH
   5003 / 75 ; # FS, TS, FLUSH, RFGA
   5004 / 20 ; # FS, CCD0 vorwärmen
   5005 / 25 ; # FS, RFGA
   5006 / 35 ; # FS, FLUSH, RFGA
   5007 / 61 ; # FS, TS
   5008 / 21 ; # FS
   CCD2:
   6000 / 22 ; # FS
   6001 / 32 ; # FS, FLUSH
   6002 / 62 ; # FS, TS

6003 / 35 ; # FS, FLUSH, RFGA, CCD1 vorwärmen
6004 / 66 ; # FS, TS, RFGA
6005 / 72 ; # FS, TS, FLUSH
6006 / 76 ; # FS, TS, FLUSH, RFGA
6007 / 26 ; # FS, RFGA
6008 / 22 ; # FS

Hier gibt es folgende Testfiles, wobei der Zusatz angibt, welches CPU-Kommando zusätzlich gegeben wurde, welcher Parameter also vom Default abweicht. Die _swm.bat-files sollten identische Ergebnisse bei allen CCDs liefern, weshalb _swm_rfga nur für CCD2 ausgeführt wurde. Ähnlich verhält es sich mit AMOS: Dieser Schaltungsteil existiert nur einmal im QSEQ und arbeitet unabhängig davon, welches CCD eingeschaltet ist, muß also auch nur einmal getestet werden.

fs0.bat	fs1.bat	fs2.bat	fs2_amos.bat
fs0_rfga_sb.bat	fs1_rfga_sb.bat	fs2_rfga_sb.bat	fs2_swm_rfga.bat
fs0_swm.bat	fs1_swm.bat	fs2_swm.bat	
fs0_tms.bat	fs1_tms.bat	fs2_tms.bat	
fs0_vbe.bat	fs1_vbe.bat	fs2_vbe.bat	

(d) Ähnliche Files wie bei (c) aber für normale Verarbeitungs-Zeilen: Auch hier ist das LRAM für die verschiedenen CCDs unterschiedlich programmiert. Grundsätzlich wird nur ein CCD abgearbeitet, auch wenn ein weiteres vorgewärmt wurde, weshalb es keinen Sinn macht VB_EOCCD einzuschalten – dies wird unter (e) getestet.

CCD0 alleine:
4000 / 08 ; # LMC
4001 / 0a ; # LMC, CCD2
4002 / 04 ; # RFGA
4003 / 18 ; # LMC, FLUSH
4004 / c8 ; # LMC, TS, TIMFLG, CCD0!!
4005 / 08 ; # LMC, CCD0!!
CCD1 alleine:
5000 / 5c ; # LMC, FLUSH, TS, RFGA, CCD0
5001 / 89 ; # LMC, TIMFLG
5002 / 01 ; # Mal gar nichts zusätzliches
5003 / 09 ; # LMC
5004 / 89 ; # LMC, TIMFLG, CCD1!!
5005 / 09 ; # LMC, CCD1!!
CCD2 alleine:
6000 / 9a ; # LMC, FLUSH, TIMFLG
6001 / 4a ; # LMC, TS
6002 / 81 ; # CCD1, TIMFLG
6003 / 0e ; # LMC, RFGA
6004 / 8a ; # LMC, TIMFLG, CCD2!!
6005 / 0a ; # LMC, CCD2!!

Die Testfiles sind hier eine „wilde" Kombination aller Parameter, in der Hoffnung, möglichst viele Betriebsvarianten mit möglichst wenig Testzeit abzudecken:

ff0_clk2.bat	ff1_clk1.bat	ff2_clk0.bat
ff0_swm_rfga_clk1.bat	ff1_swm_rfga_clk1.bat	ff2_swm_rfga_clk1.bat
	ff1_sb_rfga_clk3.bat	ff2_amos_rfga_clk2.bat

(e) Die Kombination aus (c) und (d) indem ein Quadrant aus CCD 1 → 0 → 2 zyklisch abgearbeitet werden kann. In jedem CCD wird zunächst schnell geschoben und dann normal ausgelesen um zu überprüfen, ob $\overline{\text{NEWCHIP}}$ auch richtig kommt. Das LineRAM sieht wie folgt aus (lineram.ffs102):

4000 / 20 ; # FS
4001 / 22 ; # FS, CCD2
4002 / 20 ; # FS
4003 / 08 ; # LMC
4004 / 0a ; # LMC, CCD2
4005 / 04 ; # RFGA
4006 / 18 ; # LMC, FLUSH
4007 / ca ; # LMC, TS, CCD2, TIMFLG zu Testzwecken
4008 / 0a ; # CCD2, LMC
⋮
4fff / 01 ; # CCD1 = 1. CCD
5000 / 21 ; # FS
5001 / 21 ; # FS
5002 / 21 ; # FS
5003 / 5c ; # LMC, FLUSH, TS, RFGA, CCD0
5004 / 89 ; # LMC, TIMFLG
5005 / 01 ; # Mal gar nichts zusätzliches
5006 / 09 ; # LMC
5007 / 00 ; # kein LMC, CCD0
5008 / 00 ; # CCD0
⋮
6000 / 22 ; # FS
6001 / 22 ; # FS
6002 / 22 ; # FS
6003 / 9a ; # LMC, FLUSH, TIMFLG
6004 / 4a ; # LMC, TS
6005 / 81 ; # CCD1, TIMFLG
6006 / 0e ; # LMC, RFGA
6007 / 81 ; # kein LMC, CCD1, TIMFLG als Quadrantenende
6008 / 01 ; # CCD1

Die Testfiles sind hier lediglich 2 Stück, da die wichtigsten Tests bereits unter (d) abgehandelt wurden:

ffs102.bat
ffs102_vbe.bat

(f) Für den Timing-Mode noch ein Test, der auch überprüft, ob ein evtl falsch vorgewärmtes weiteres CCD tatsächlich kein SEL_OUT bekommt (lineram.tms_CCD2):
4000 / 0a ; #LMC
4001 / 0a ; #LMC
4002 / 02 ;
⋮
4fff / 02 ; # CCD2 = 1. CCD
5000 / 02 ;
5001 / 82 ; # TIMFLG in Zeile TMS1 $\Longrightarrow$ virtuelles CCD-Ende
5002 / 05 ; # RFGA, CCD1
5003 / 92 ; # FLUSH, TIMFLG für Zeitwort ohne QSEQ_RDY
5004 / 02 ;

Das Testfile heißt:
tms2.bat

Die Durchführung der Tests geschah mittels der Befehle:

- Für das Referenz-Design:
  Start der Simulation mit

  „quicksim referenz_testbench -ts 0.01 -nodisplay < /home/secom/testmuster/xxx.bat"

  Die Simulations-Ergebnisse stehen dann in der Datei
  /home/secom/testmuster/results/xxx.list

- Für das ASIC-Design:
  Start von mhs_quicksim aus mhs_dmgr, dann „force from file xxx_a.bat"

  Die Simulations-Ergebnisse stehen dann in der Datei
  /home/secom/testmuster/results/xxx_a.list

Anhang D

Schaltpläne

Nachdem die Schaltung des Quadrantensequenzers ohne den Common-Mode-Filter etwas aus ihrem Zusammenhang gerissen ist, sind in diesem Anhang alle Schaltpläne aufgeführt, die den SECOM-ASIC betreffen. Es handelt sich dabei ausschließlich um die Versionen, die für das Referenz-Design (Kapitel 6) verwendet wurden, also um die unter Mentor® entstandenen XILINX®-FPGA-Schaltungen.

Die Schaltpläne sind in ihrer Reihenfolge gemäß der SECOM-Struktur angeordet: Im ersten Teil das Referenz-Design mit den Schaltungsteilen, wie sie nur dort verwendet wurden, danach der „QSEQ“ und der Common-Mode-Filter in ihrer jeweiligen FPGA-Variante. Dabei ist der Common-Mode-Filter in COM“ und „SORT“ aufgeteilt, wie in Abbildung 3.23 angedeutet. Den Abschluß bilden die Hilfsmittel, wie die Flanken-Detektoren (Abschnitt 5.2) und die Bauteile, die in der verwendeten xc4000-Bibliothek nicht enthalten waren.

D.1 Das Referenz-Design

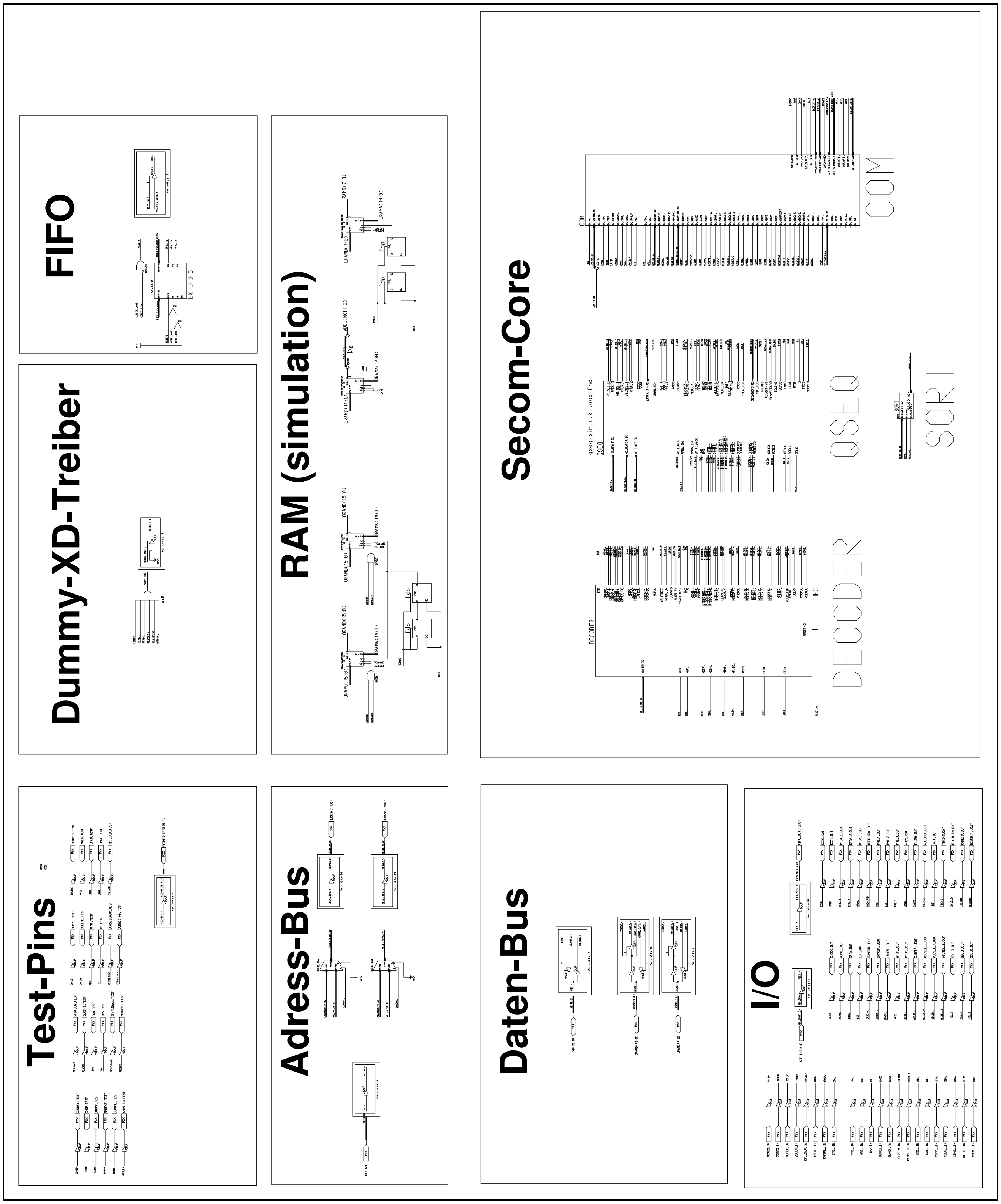


Abbildung D.1: „toplevel“

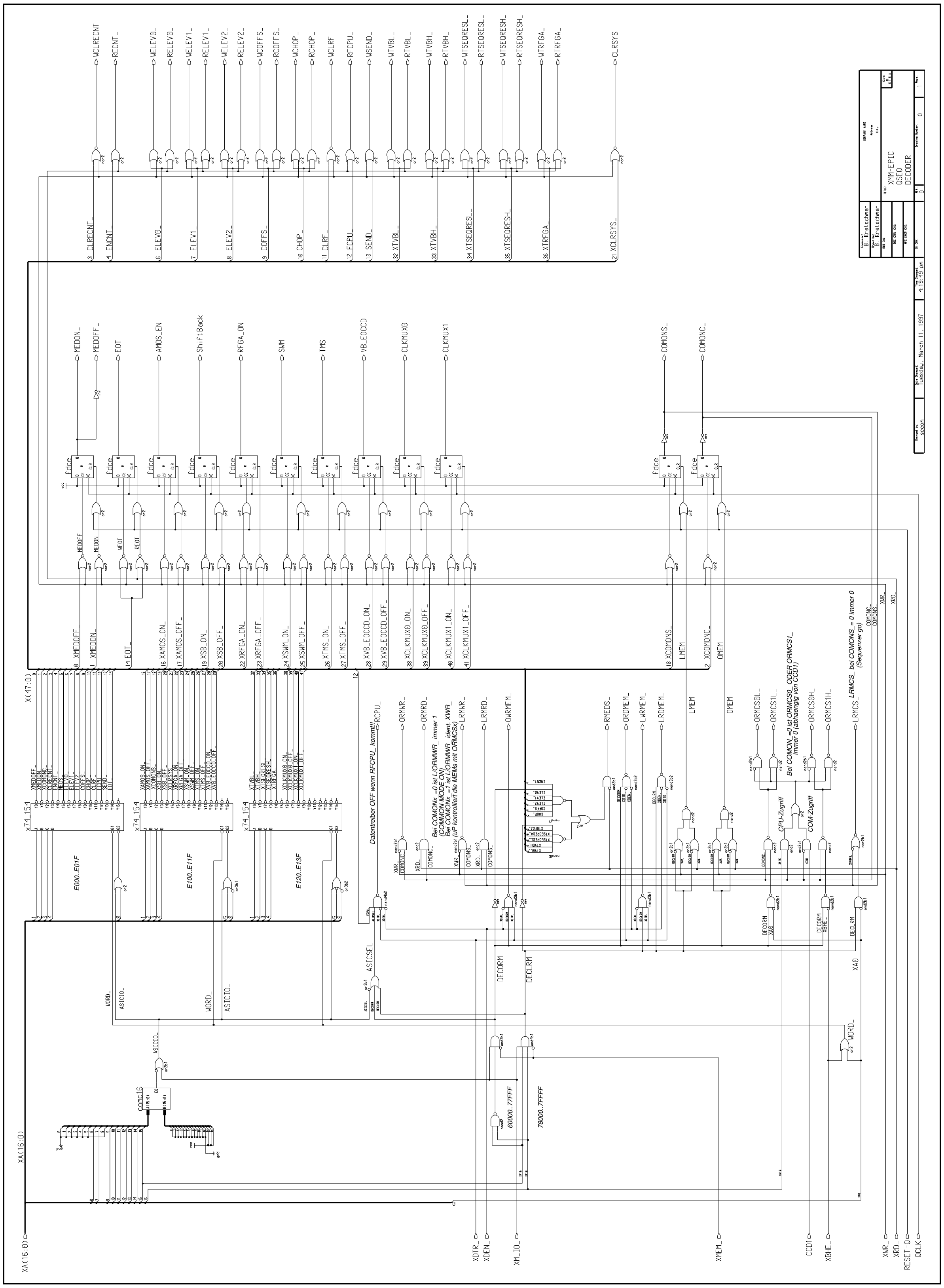


Abbildung D.2: *„DECODE“*

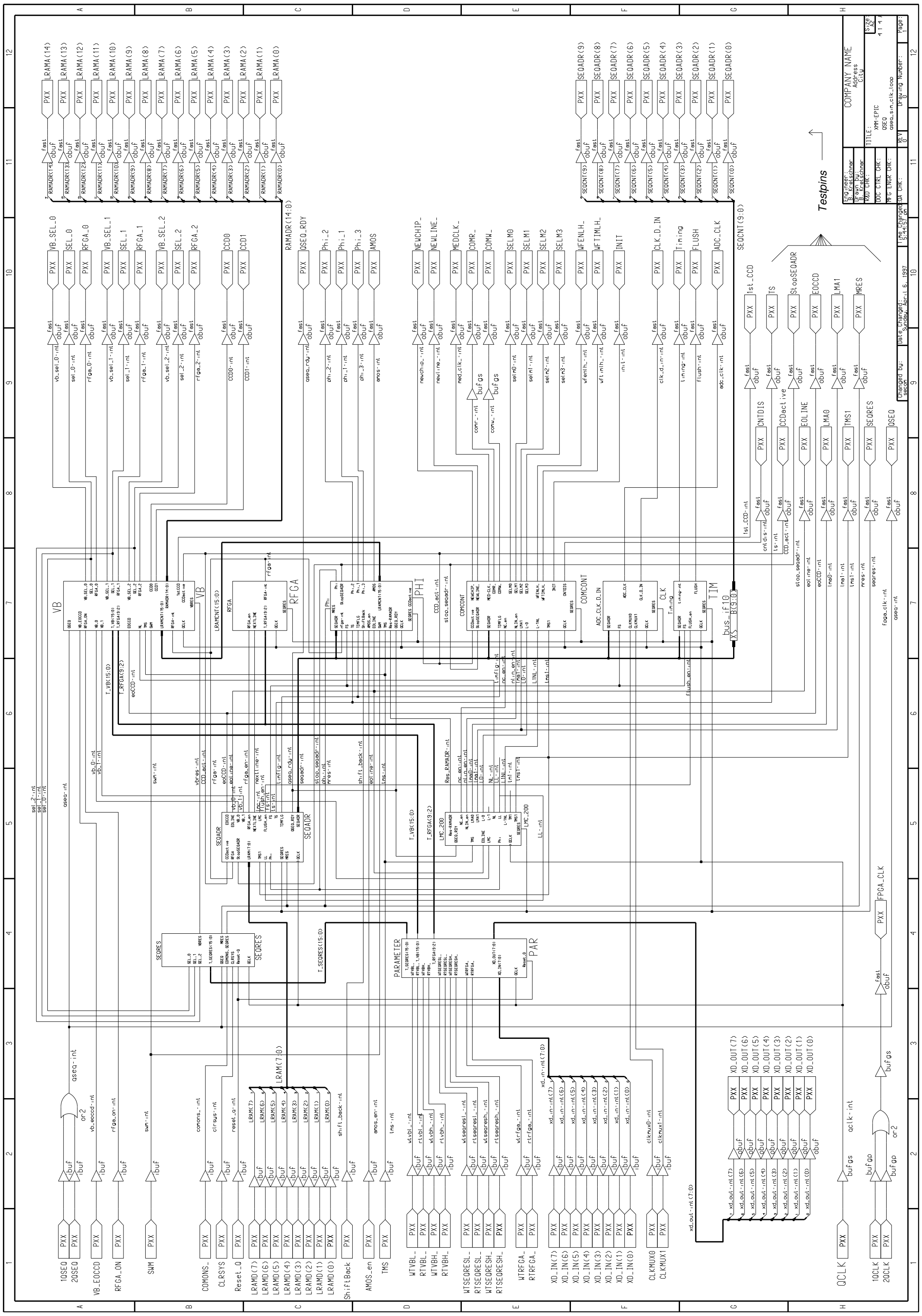


Abbildung D.3: *„qseq_sim_clk_loop"*

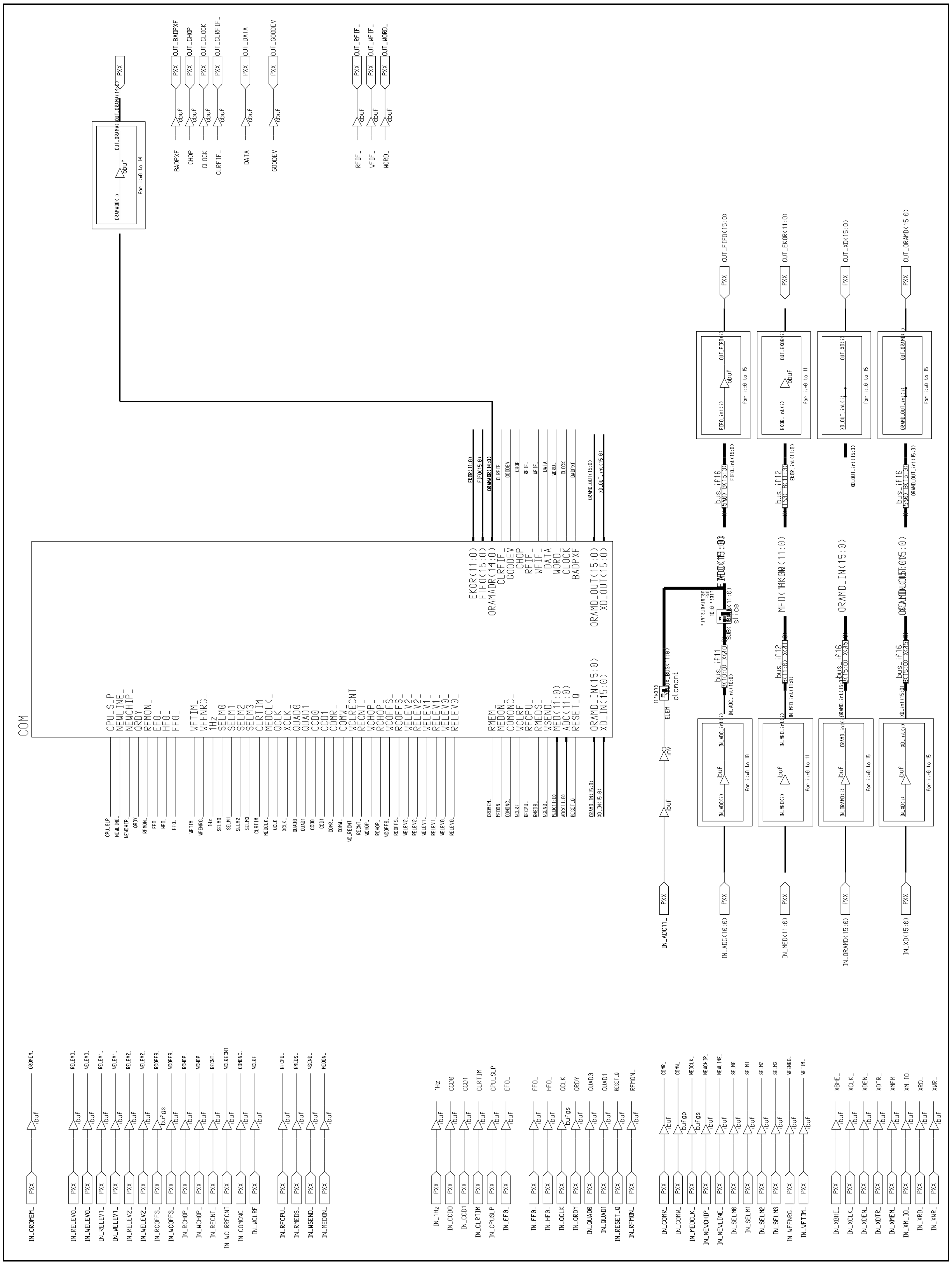


Abbildung D.4: *„com_fpga_ref“*

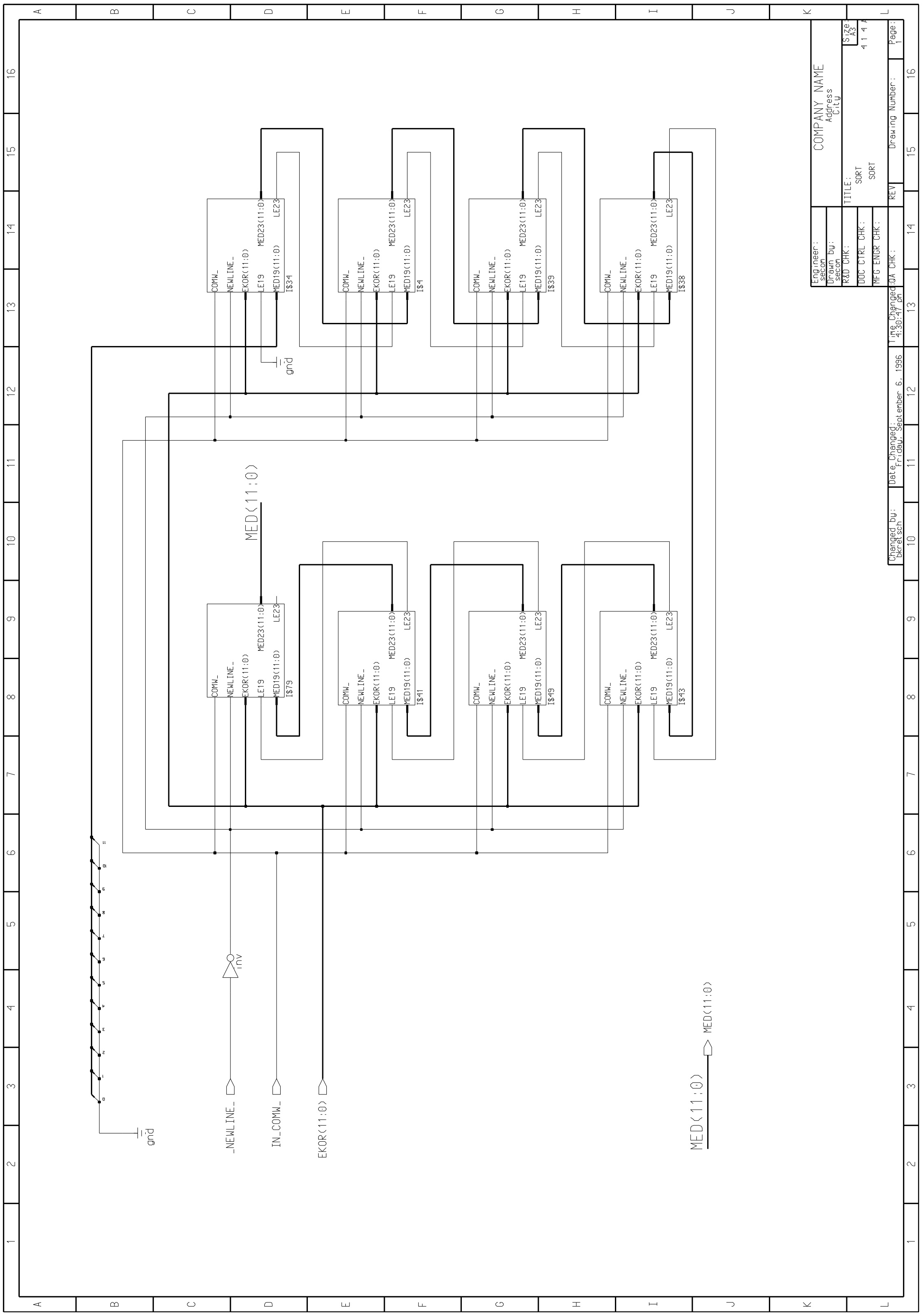


Abbildung D.5: *„SORT_ref“*

D.2 Das FPGA des Quadrantensequenzers

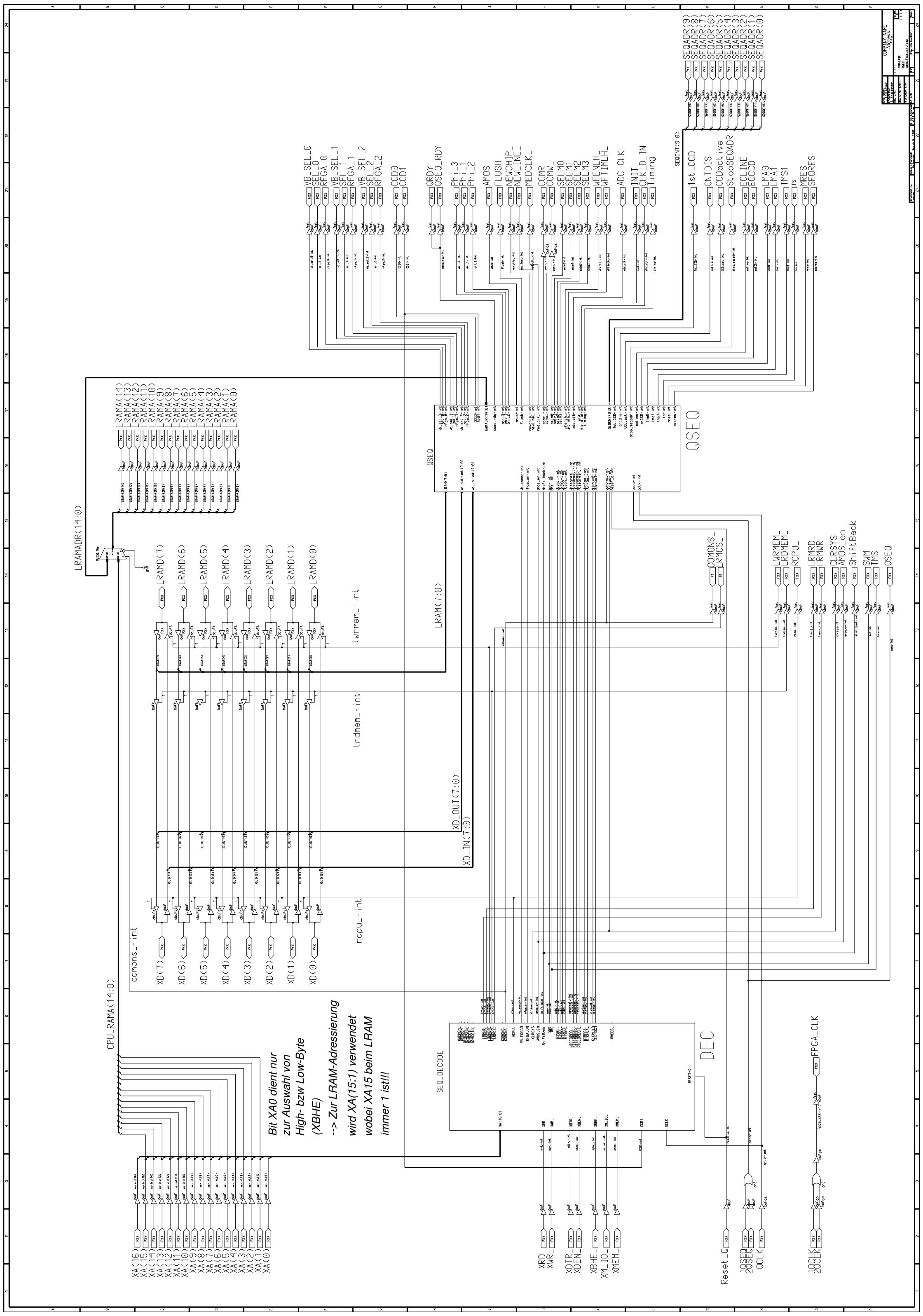


Abbildung D.6: *„qseq_fpga_clk_loop“*

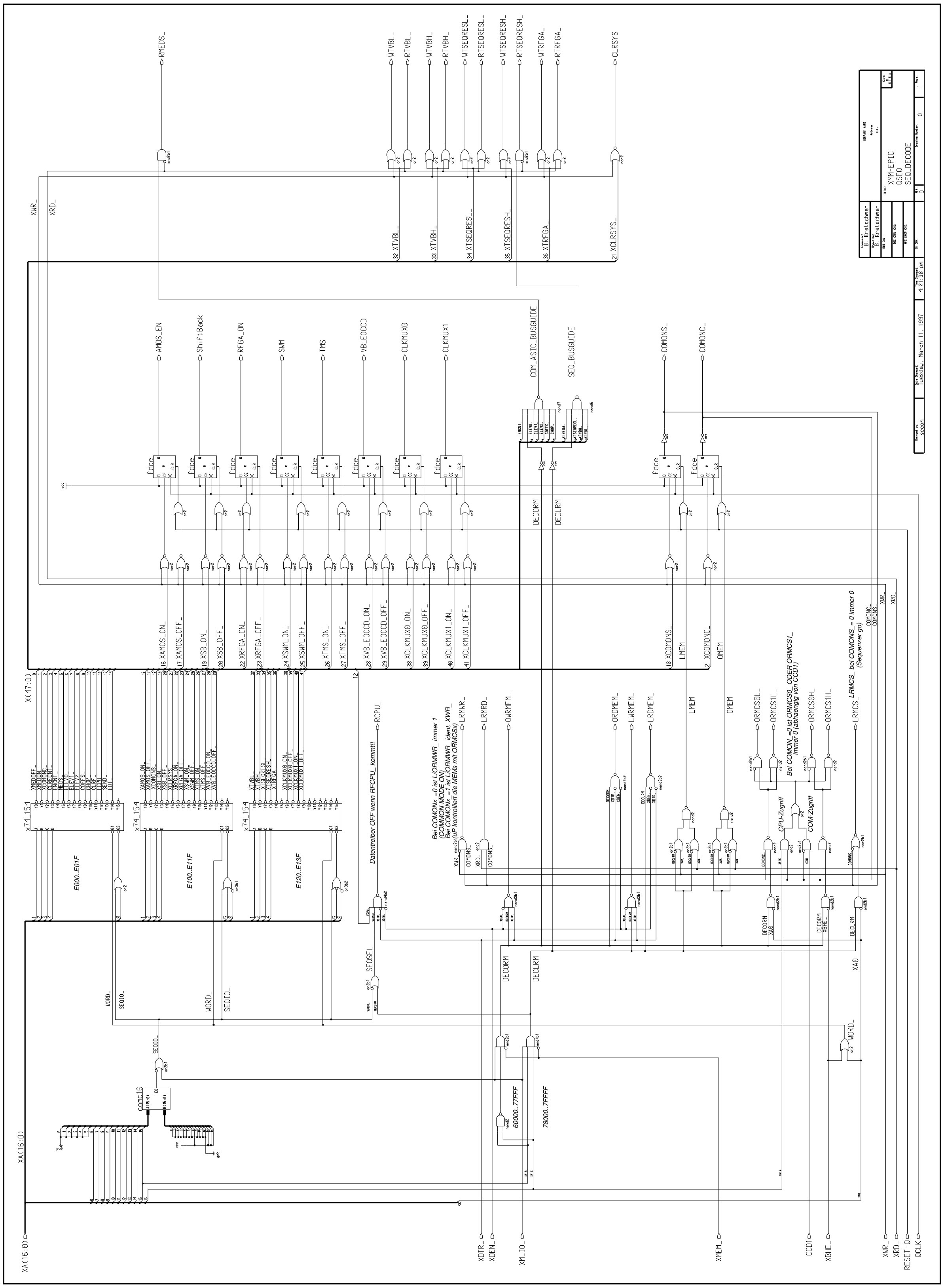


Abbildung D.7: *„SEQ_DECODE“*

D.2.1 Der eigentliche QSEQ

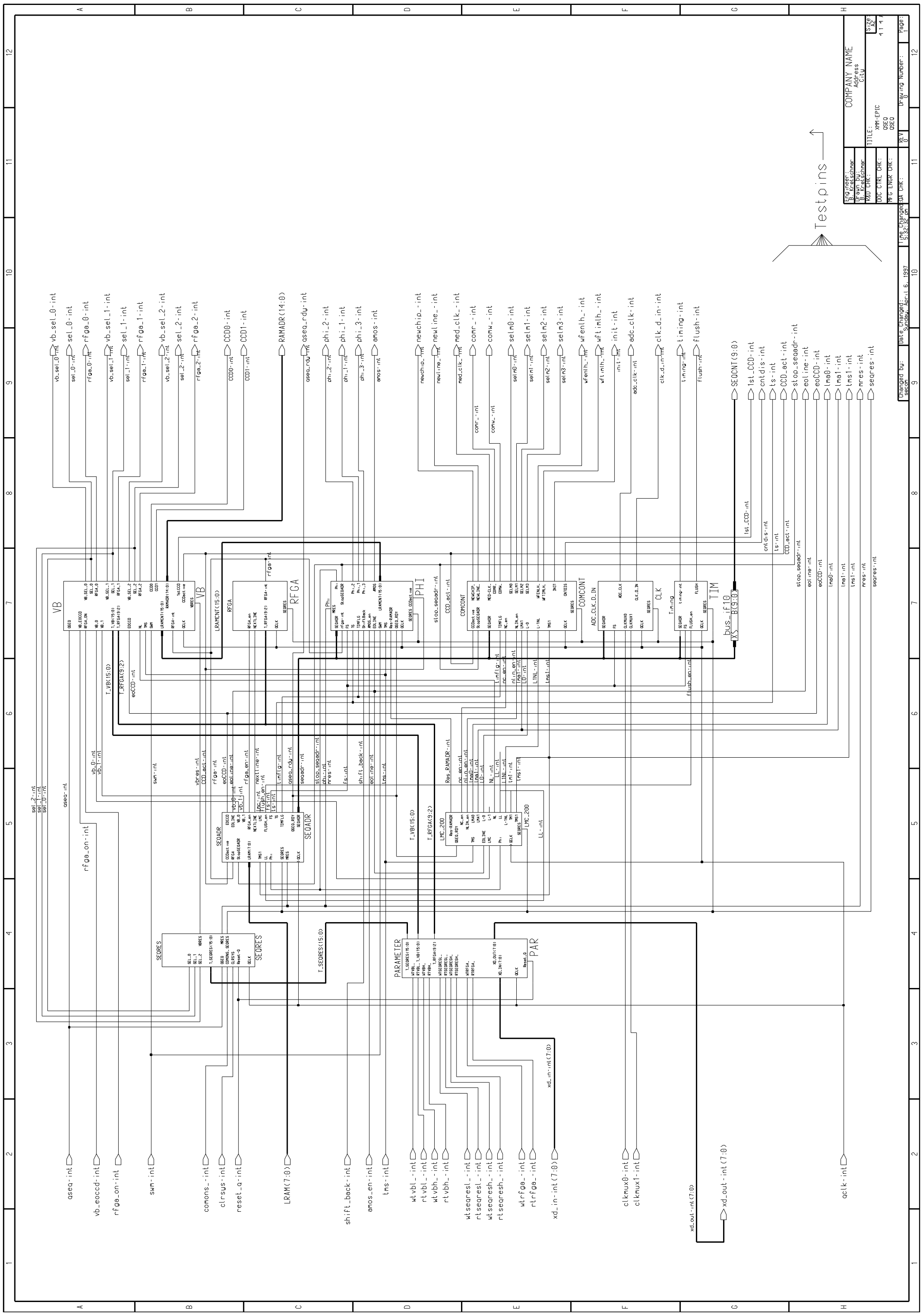


Abbildung D.8: *„QSEQ“*

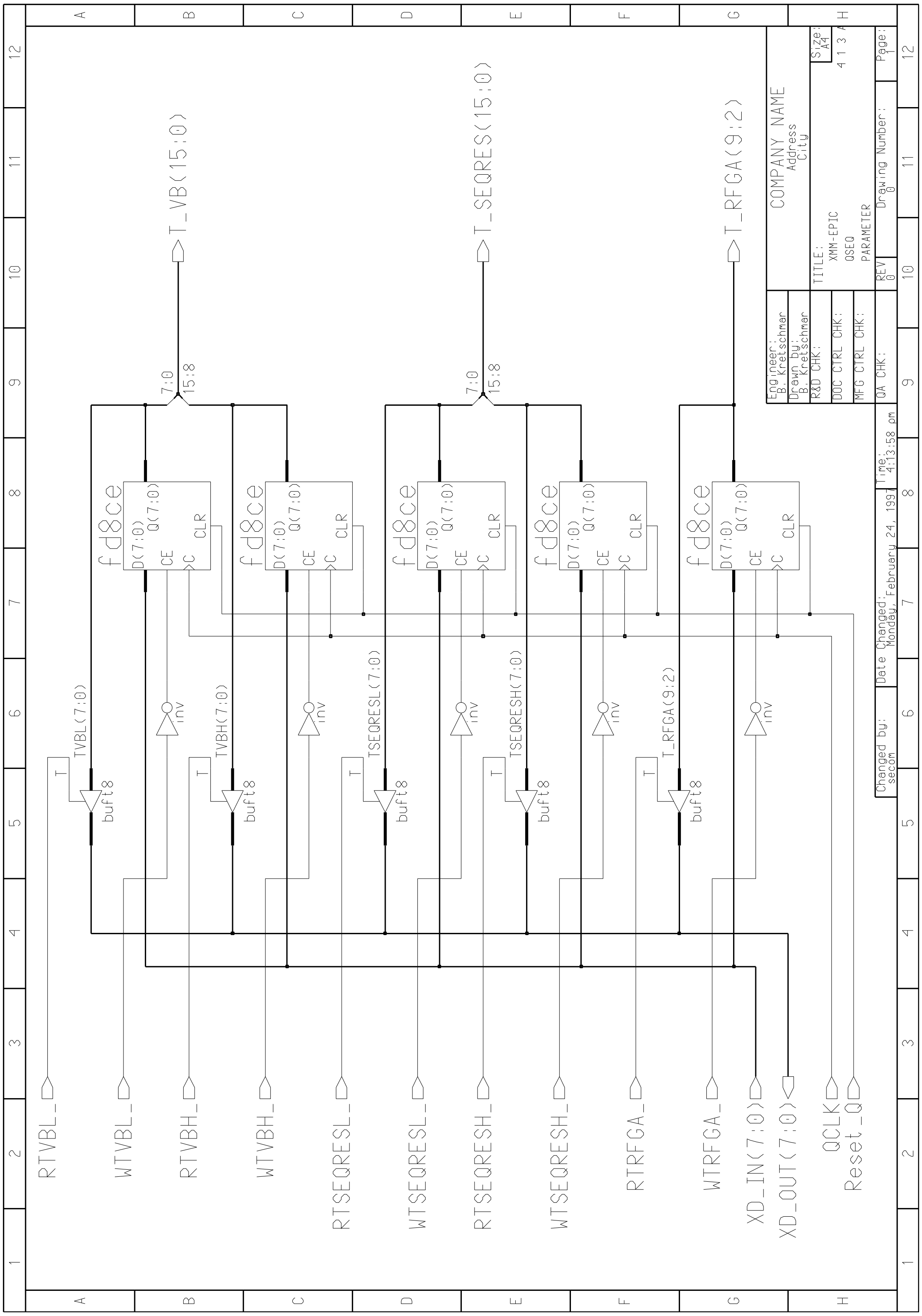


Abbildung D.9: *„PARAMETER“*

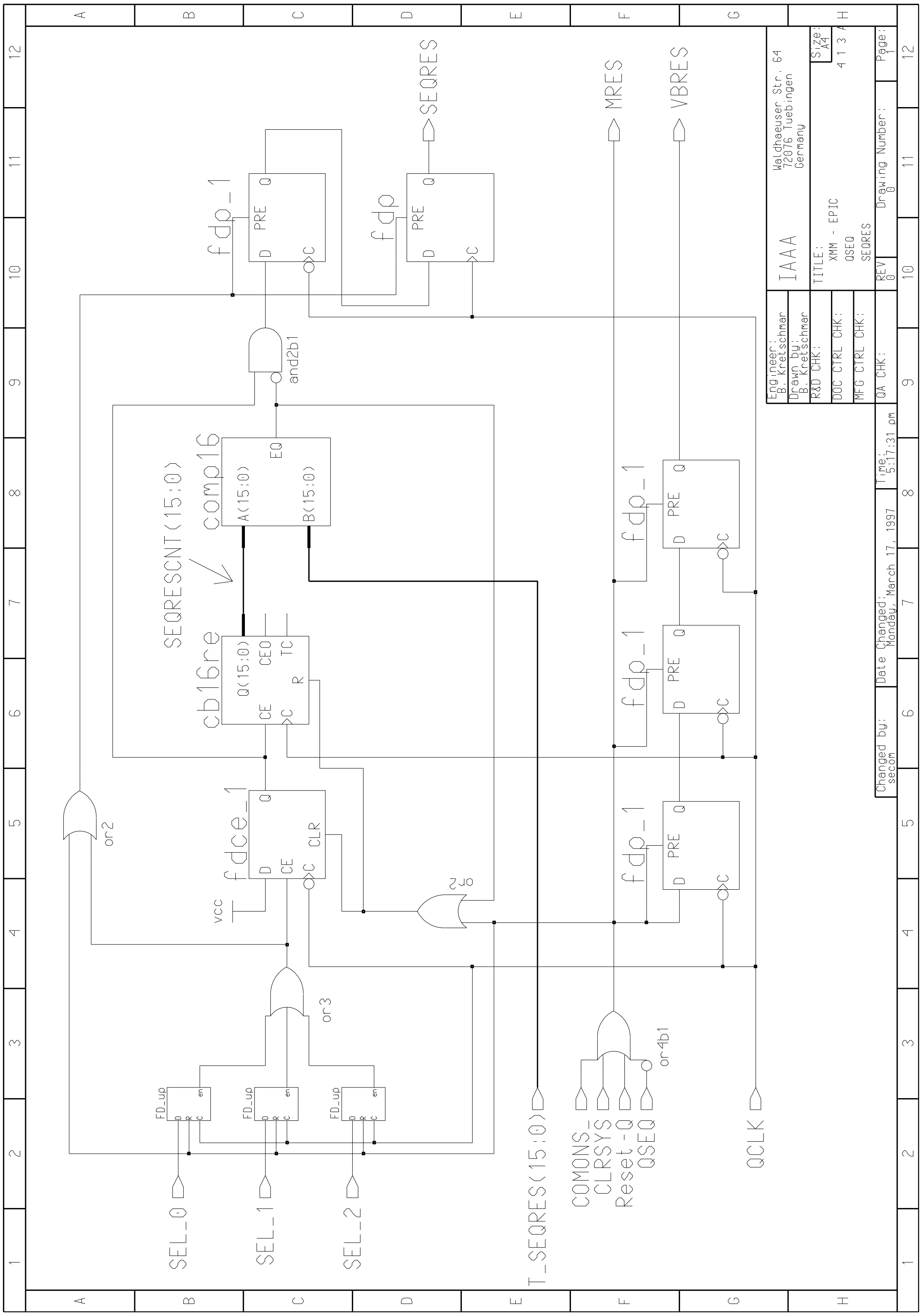


Abbildung D.10: *„SEQRES“*

D.2.2 Die Erkennung von Sonderzeilen

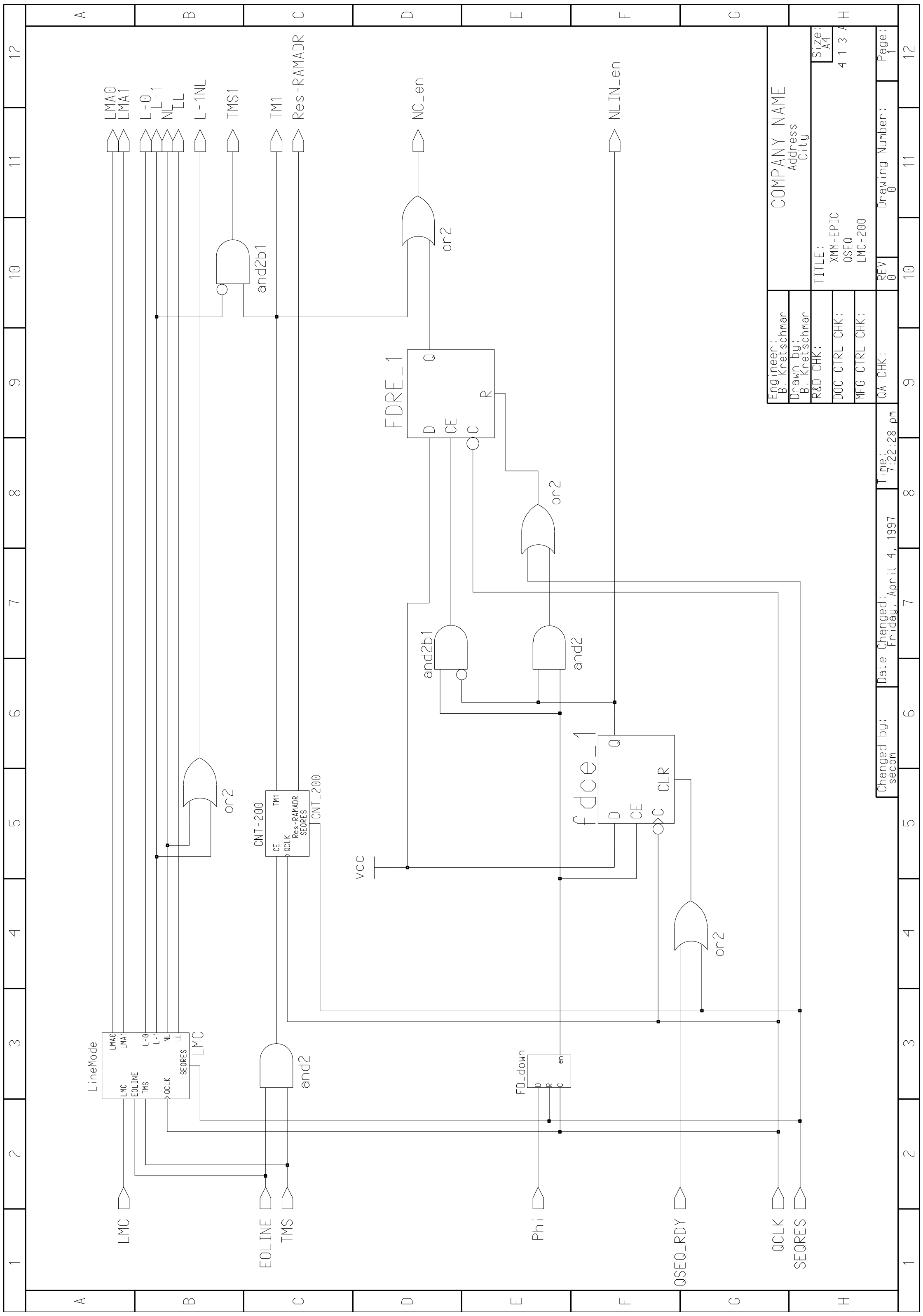


Abbildung D.11: *„LMC_200“*

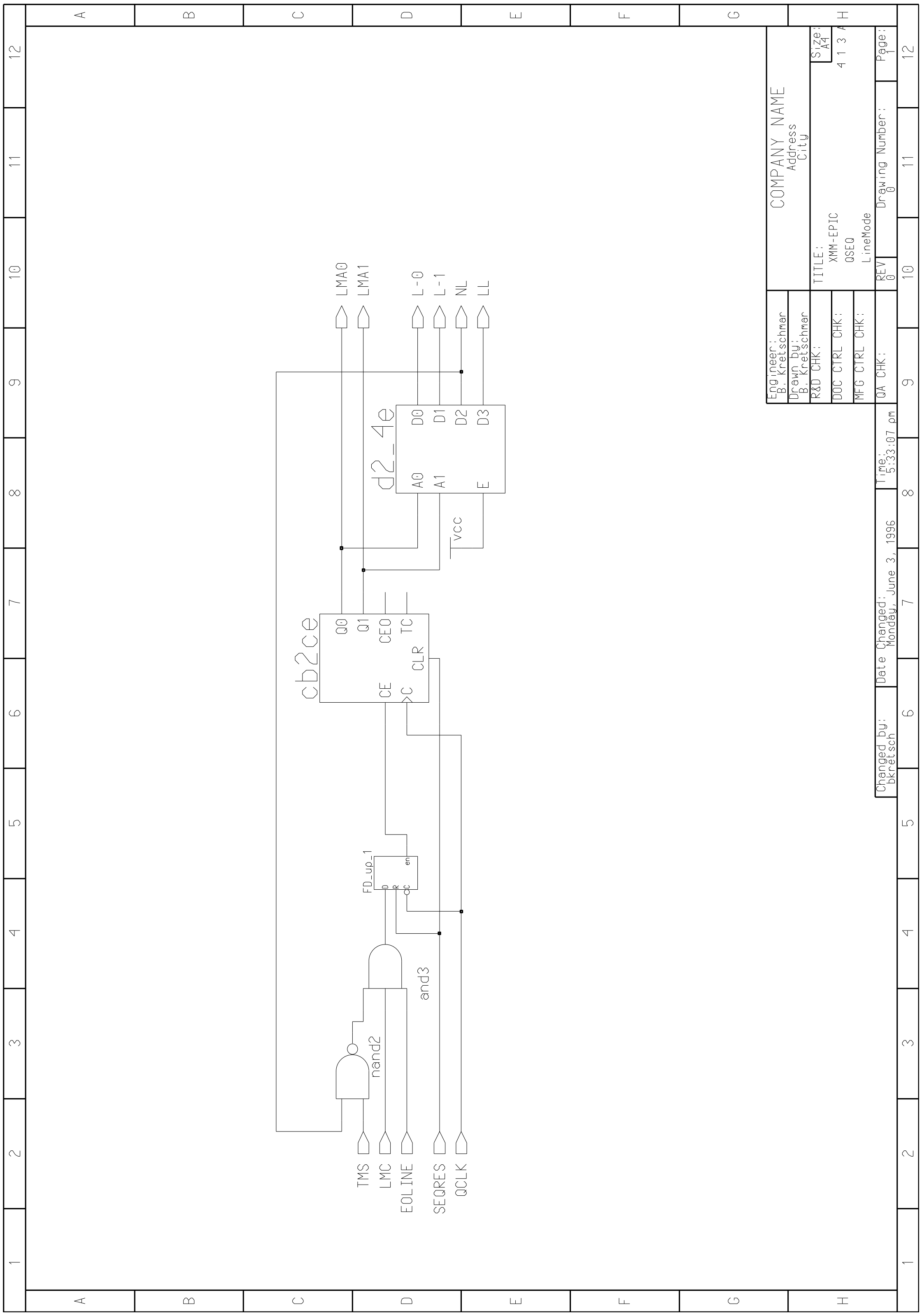


Abbildung D.12: *„LineMode“*

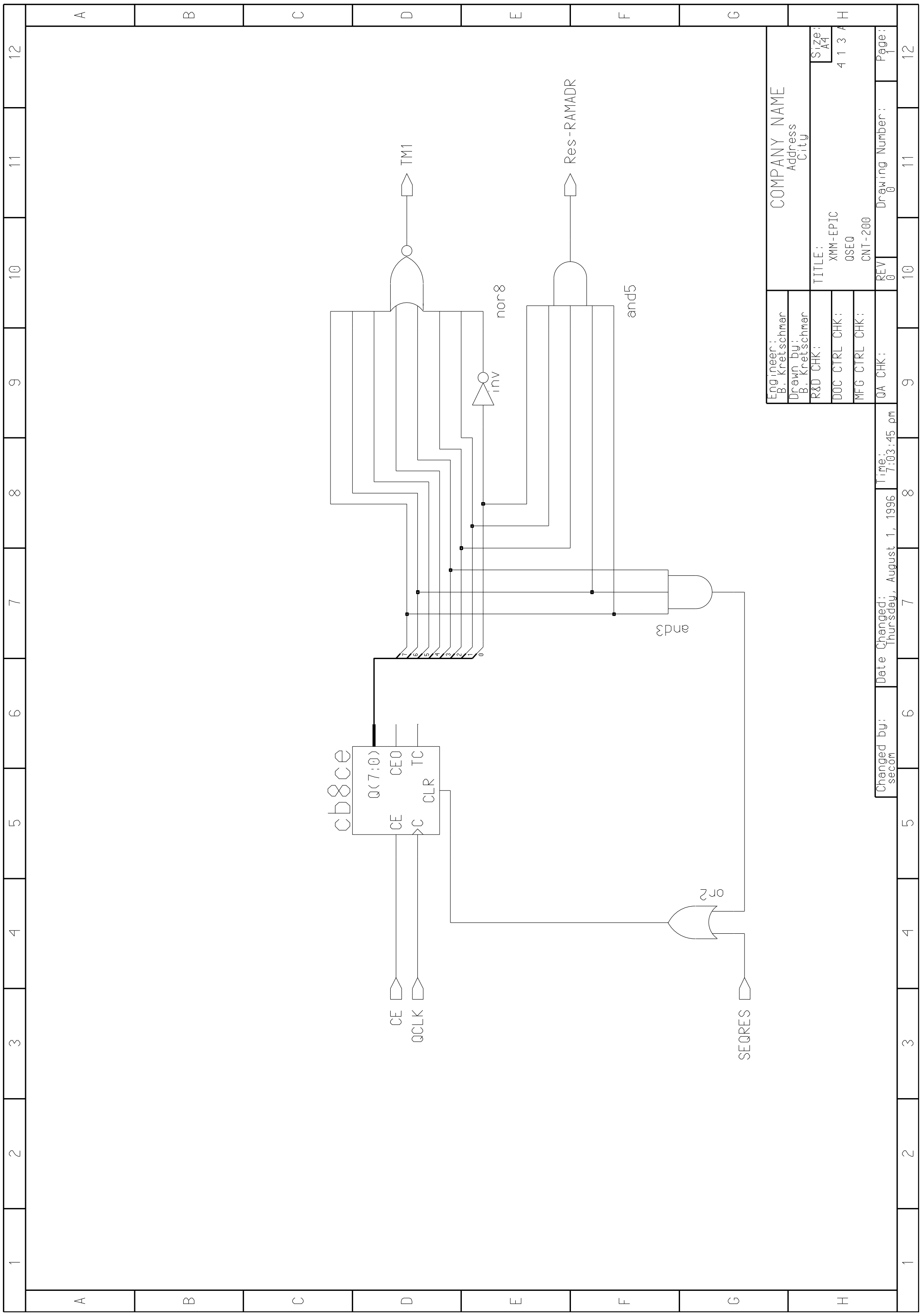


Abbildung D.13: *„CNT_200"*

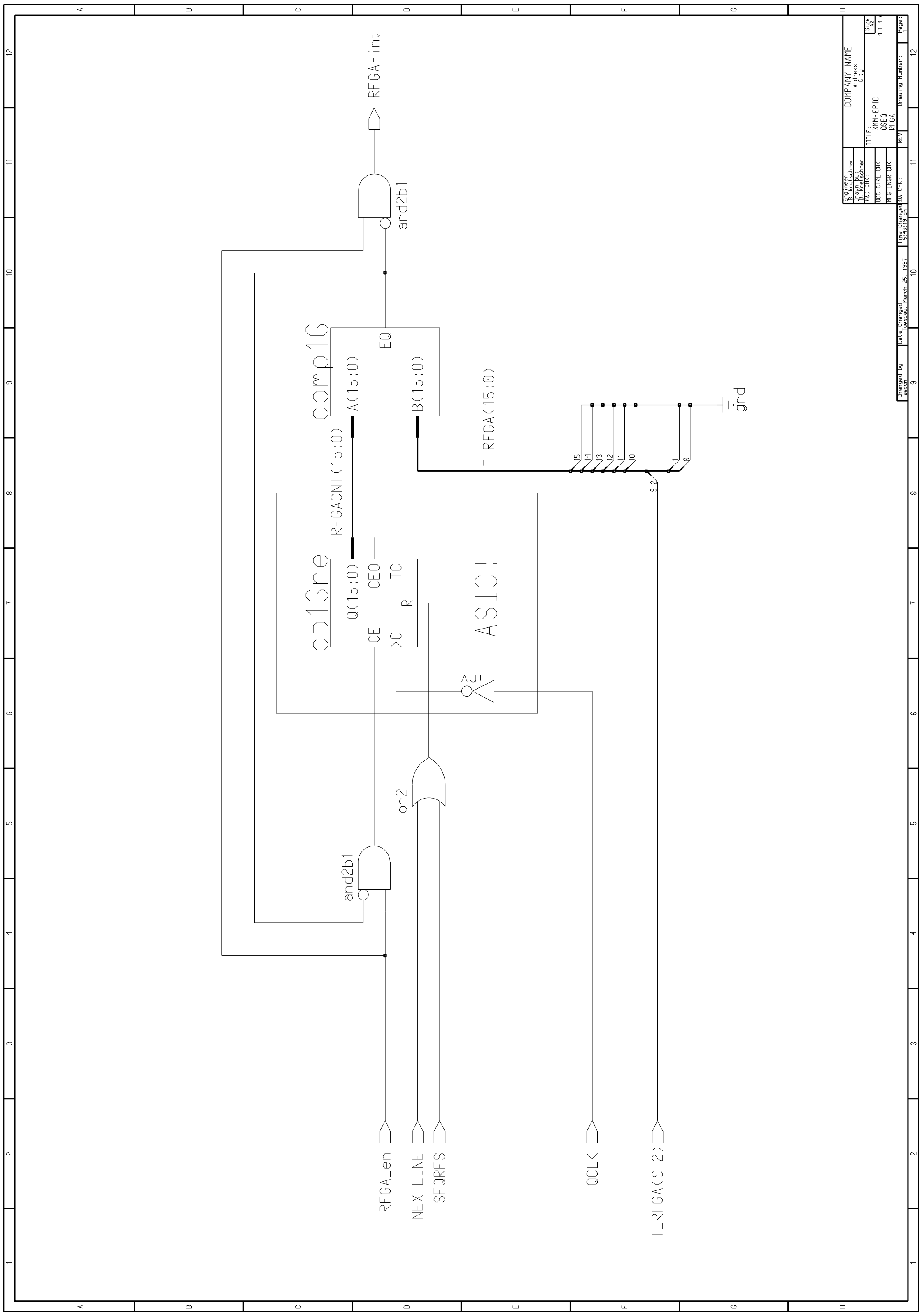


Abbildung D.14: *„RFGA“*

D.2.3 Das Einschalten der CCDs

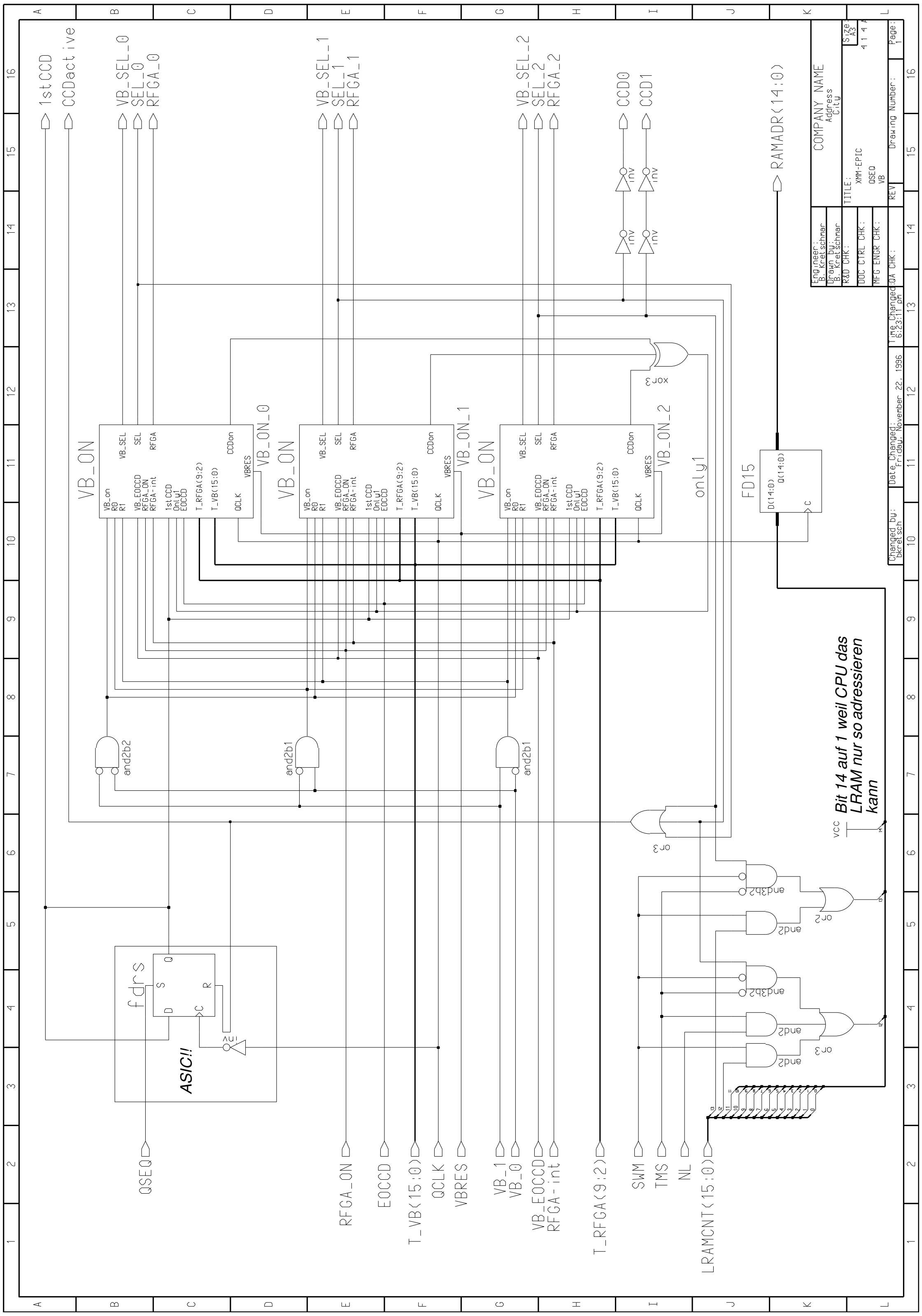


Abbildung D.15: *„VB“*

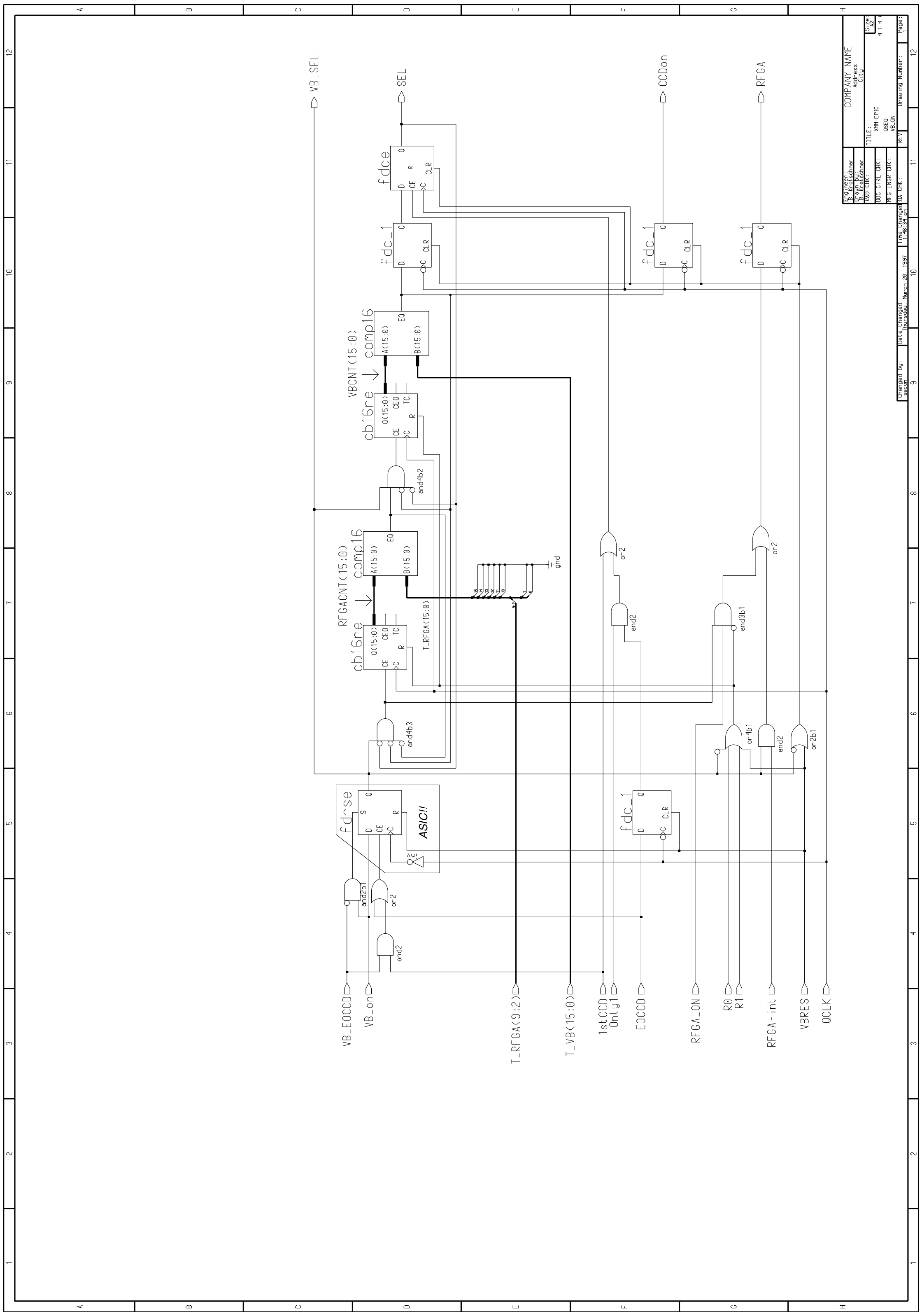


Abbildung D.16: *„VB_ON“*

D.2.4 Die SEQADR-abhängigen Schaltungen

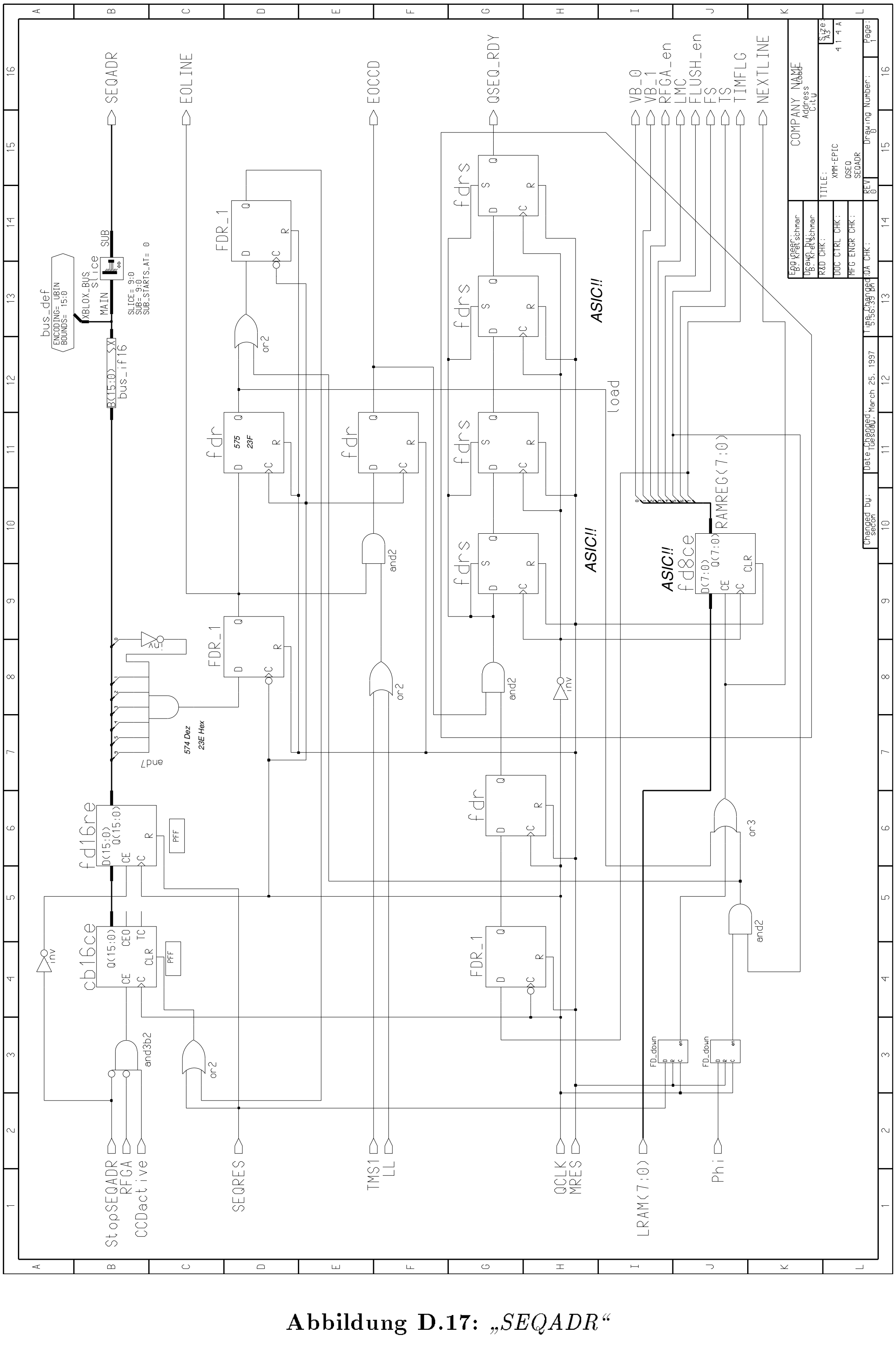


Abbildung D.17: *„SEQADR“*

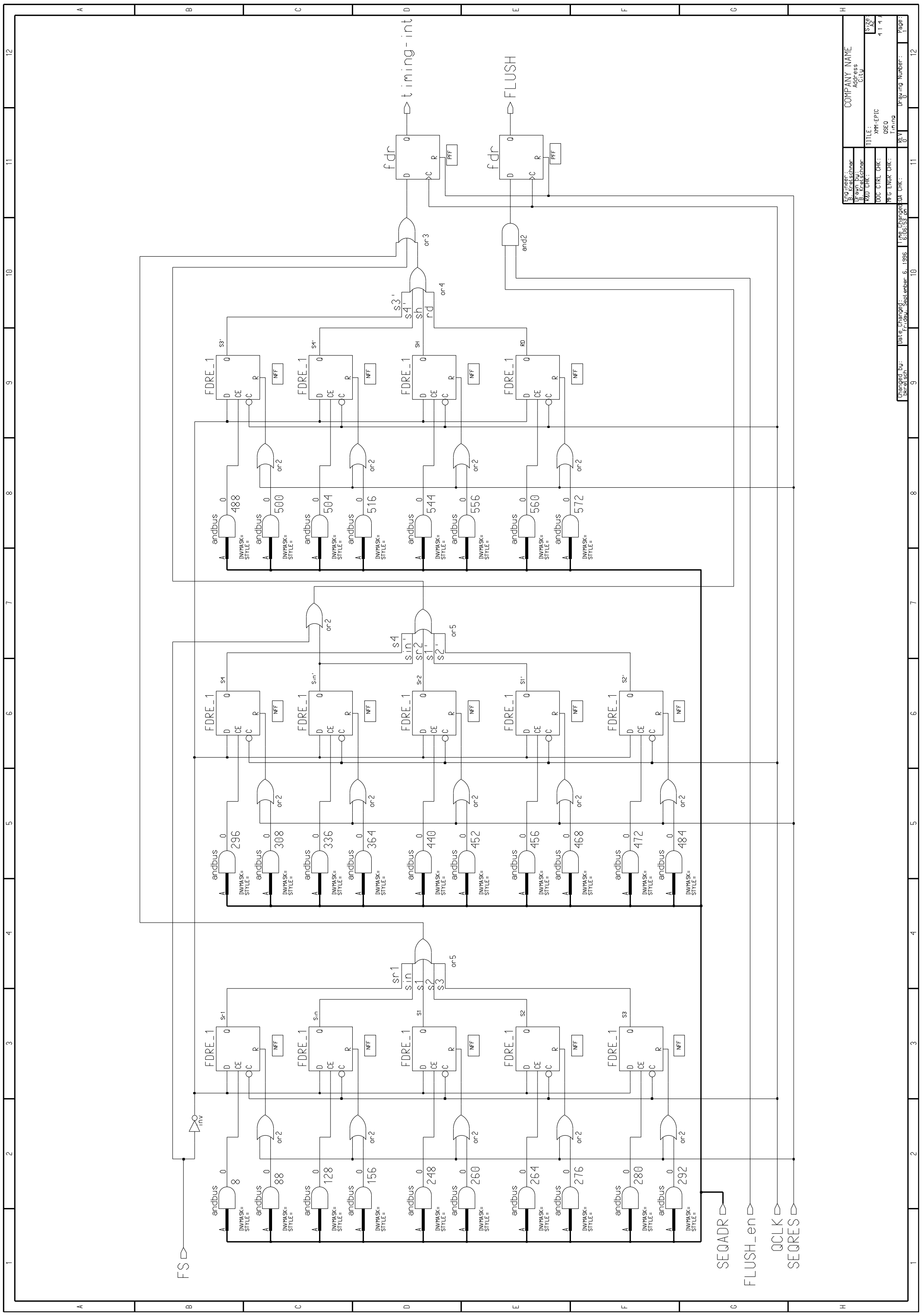


Abbildung D.18: *„Timing“*

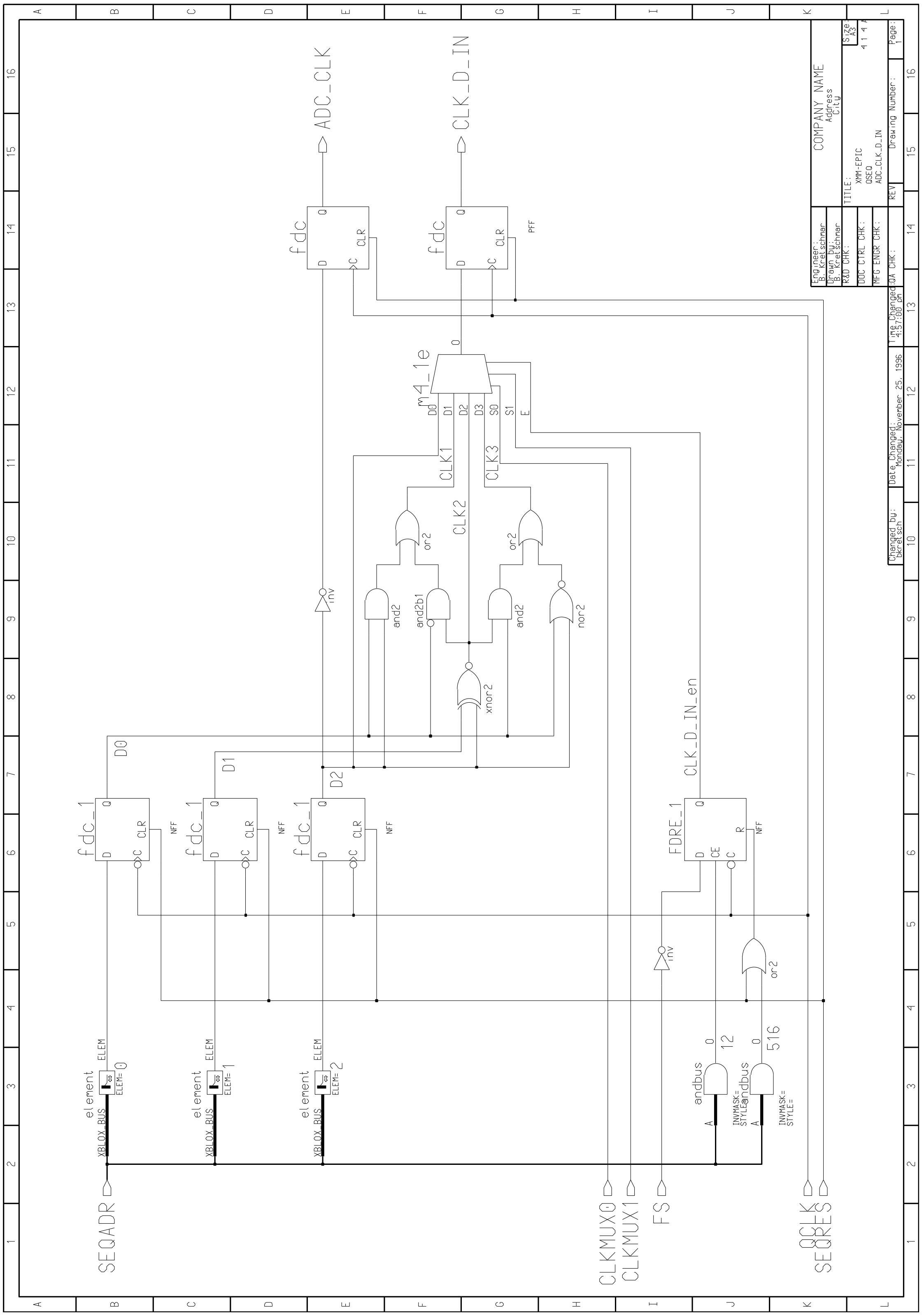


Abbildung D.19: *„ADC_CLK_D_IN“*

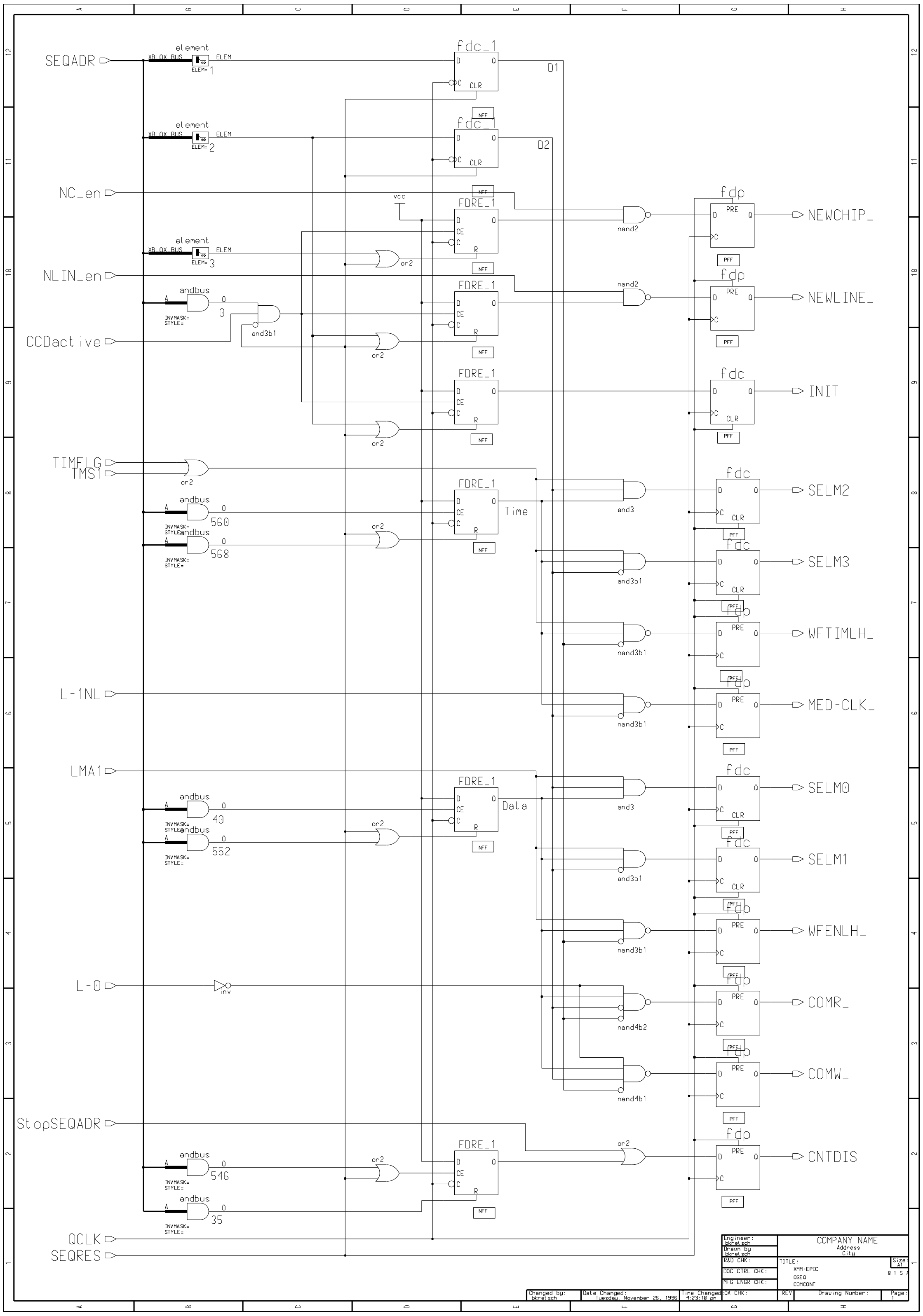


Abbildung D.20: *Im „COMCONT“ sind teilweise noch alte Signal-Namen verwendet, so heißt* $\overline{WFTIM}$ *hier noch WFTIMLH_ und* $\overline{WFENRG}$ *WFENLH_.*

D.2.5 Die Phi-Schaltungen

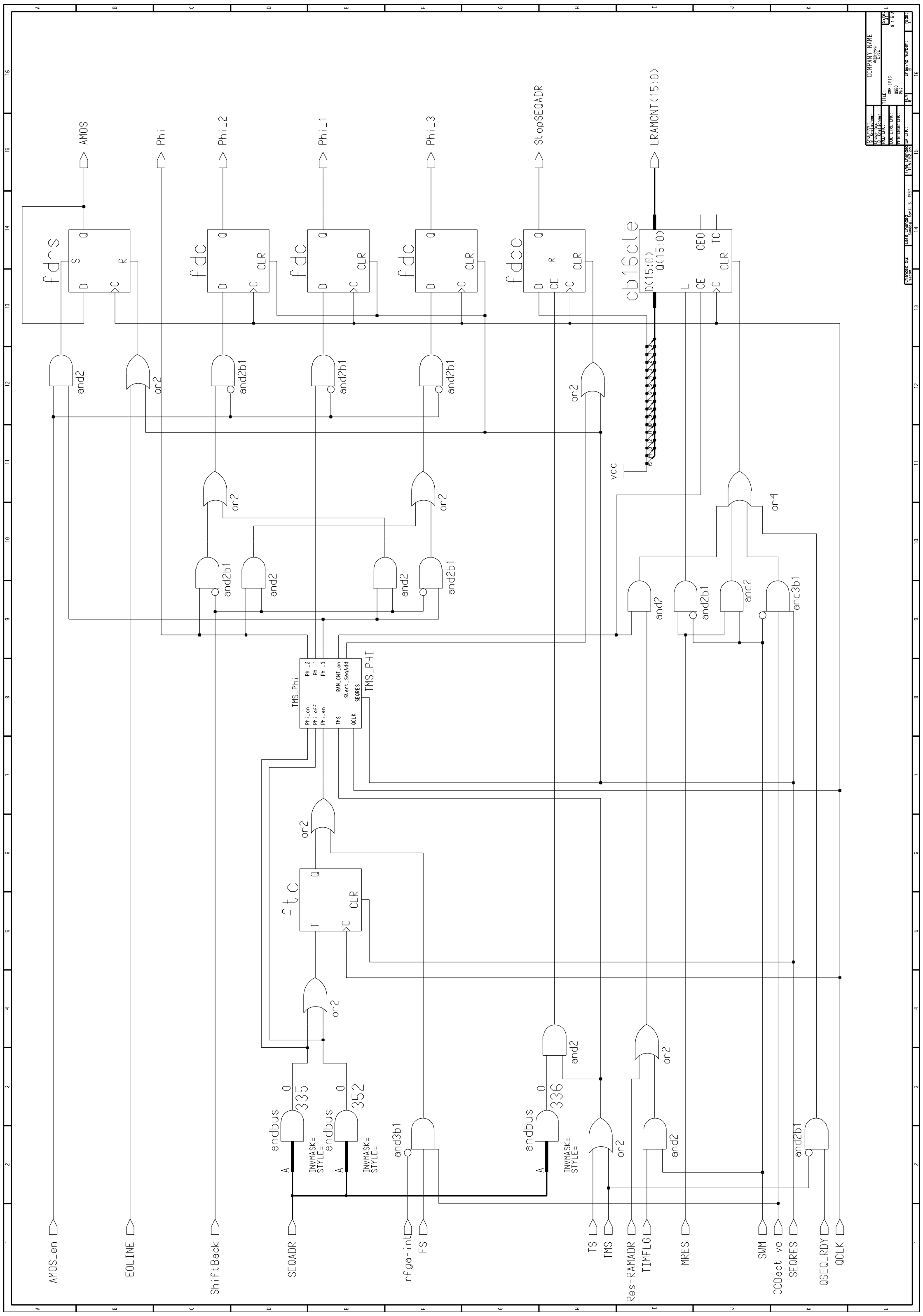


Abbildung D.21: *„Phi“*

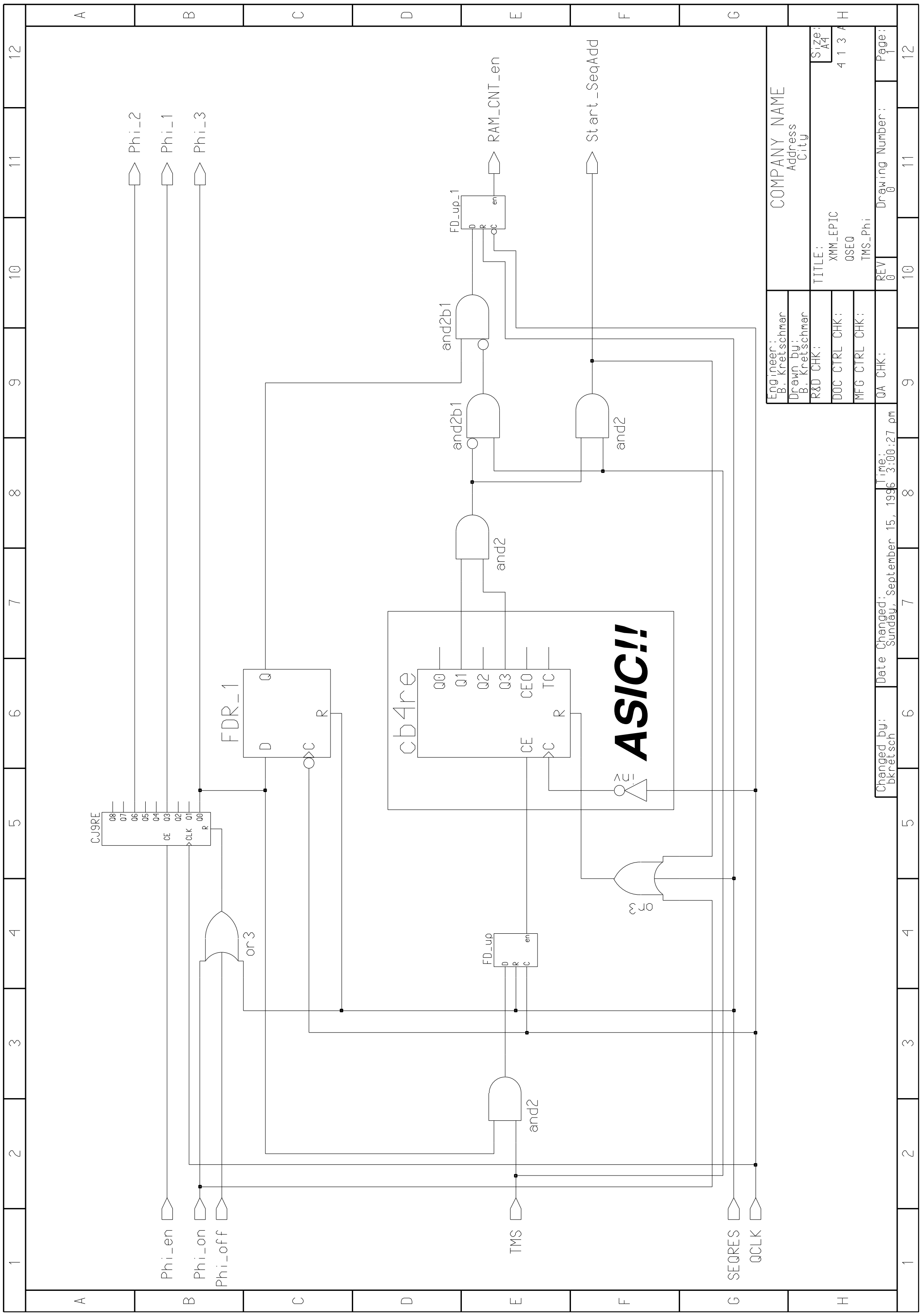


Abbildung D.22: *„TMS_Phi“*

D.3 Das COM-FPGA

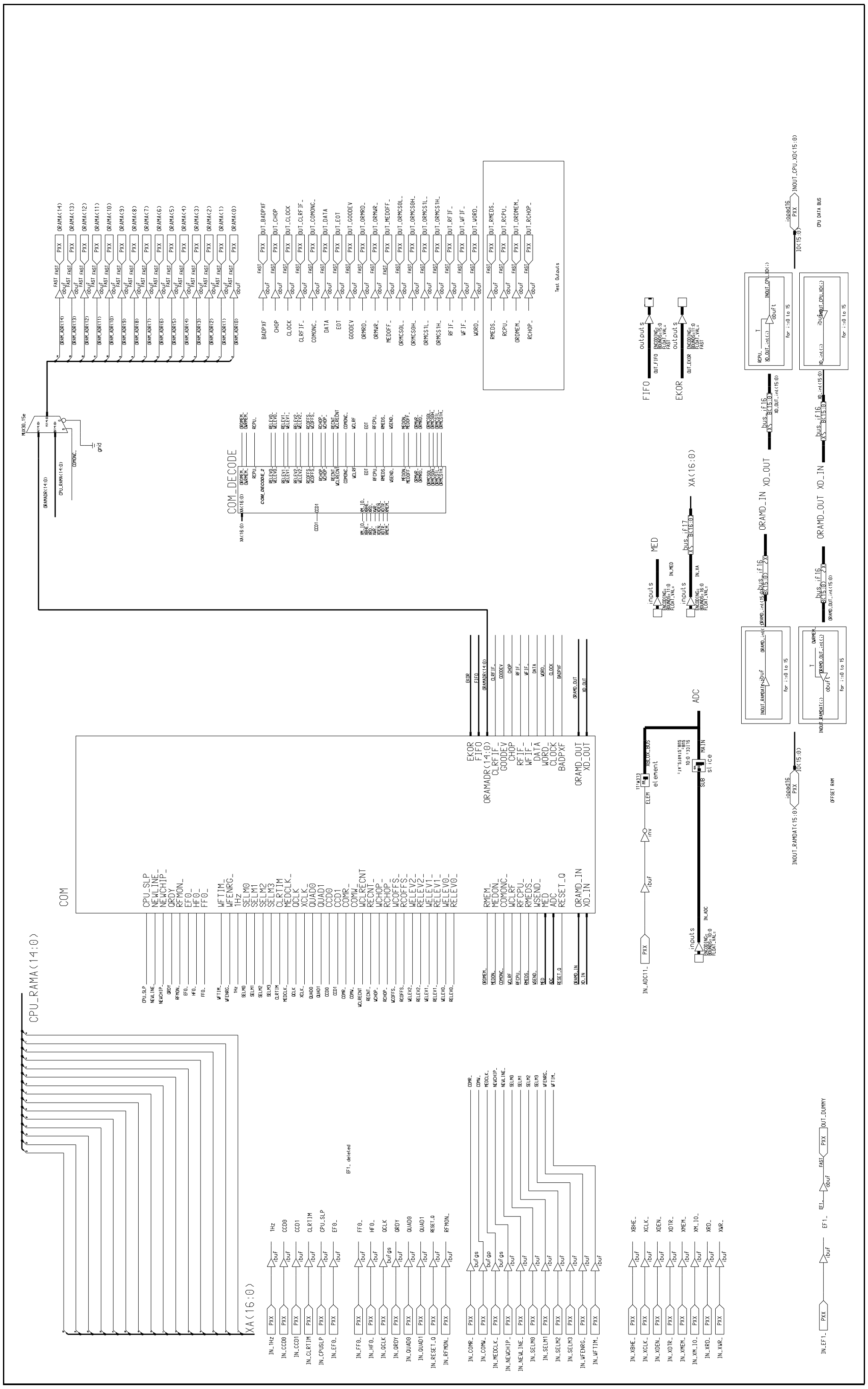


Abbildung D.23: *„com_fpga“*

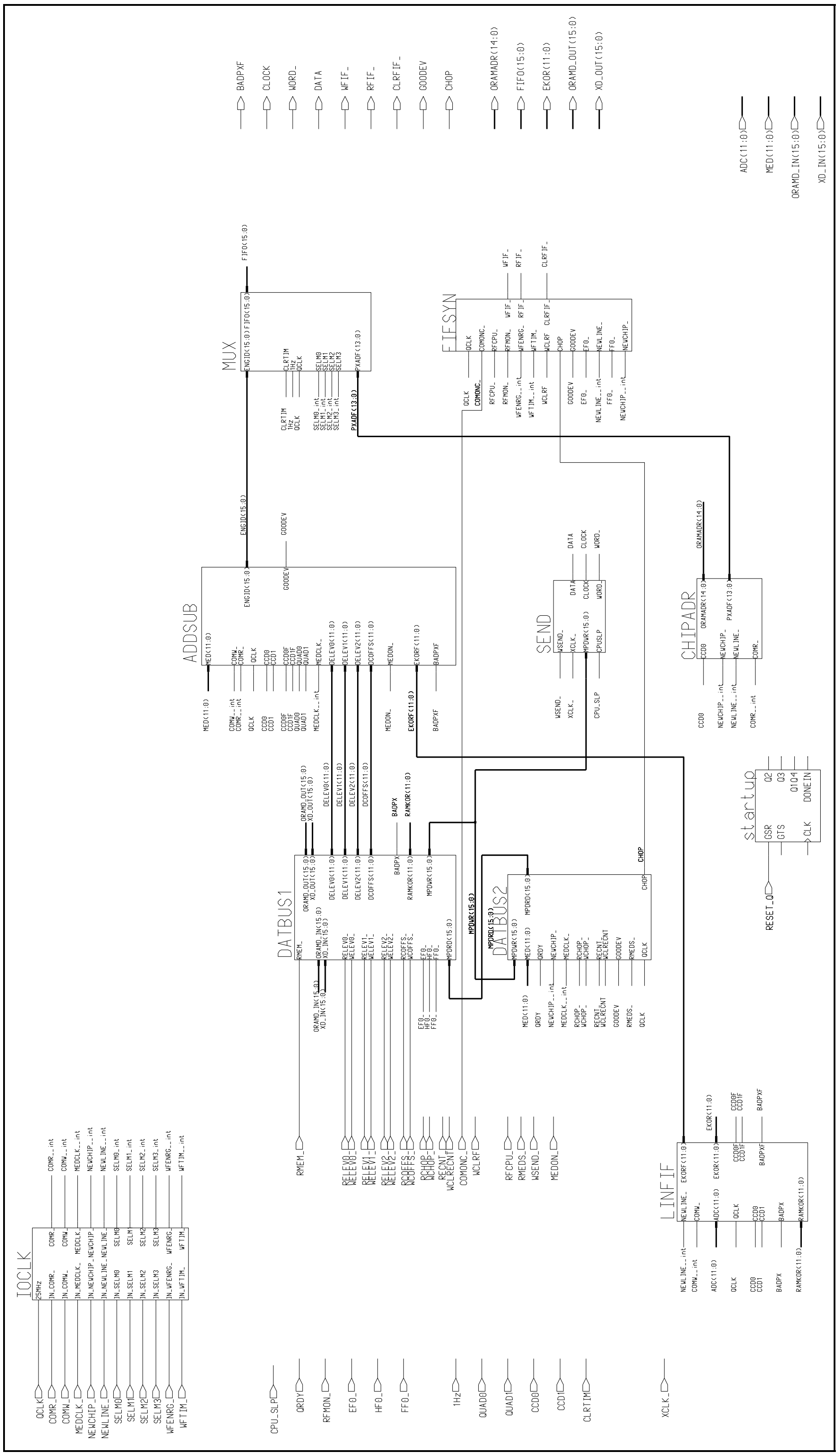


Abbildung D.24: *„COM“*

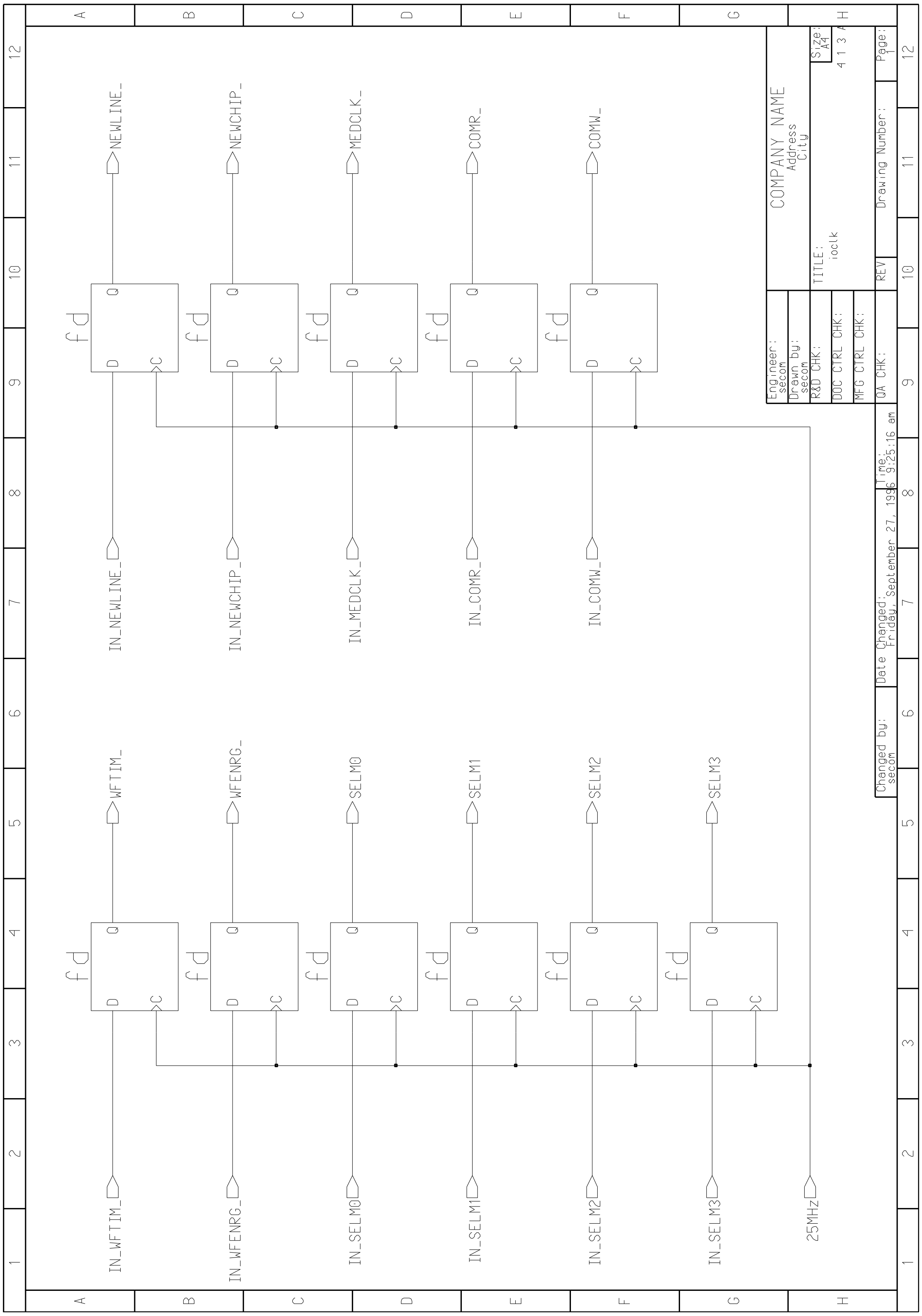


Abbildung D.25: *In „ioclk“ werden die Sequenzer-Signale nochmals gelatched, damit etwaige Laufzeitunterschiede zwischen den FPGAs nicht zu Schwierigkeiten führen.*

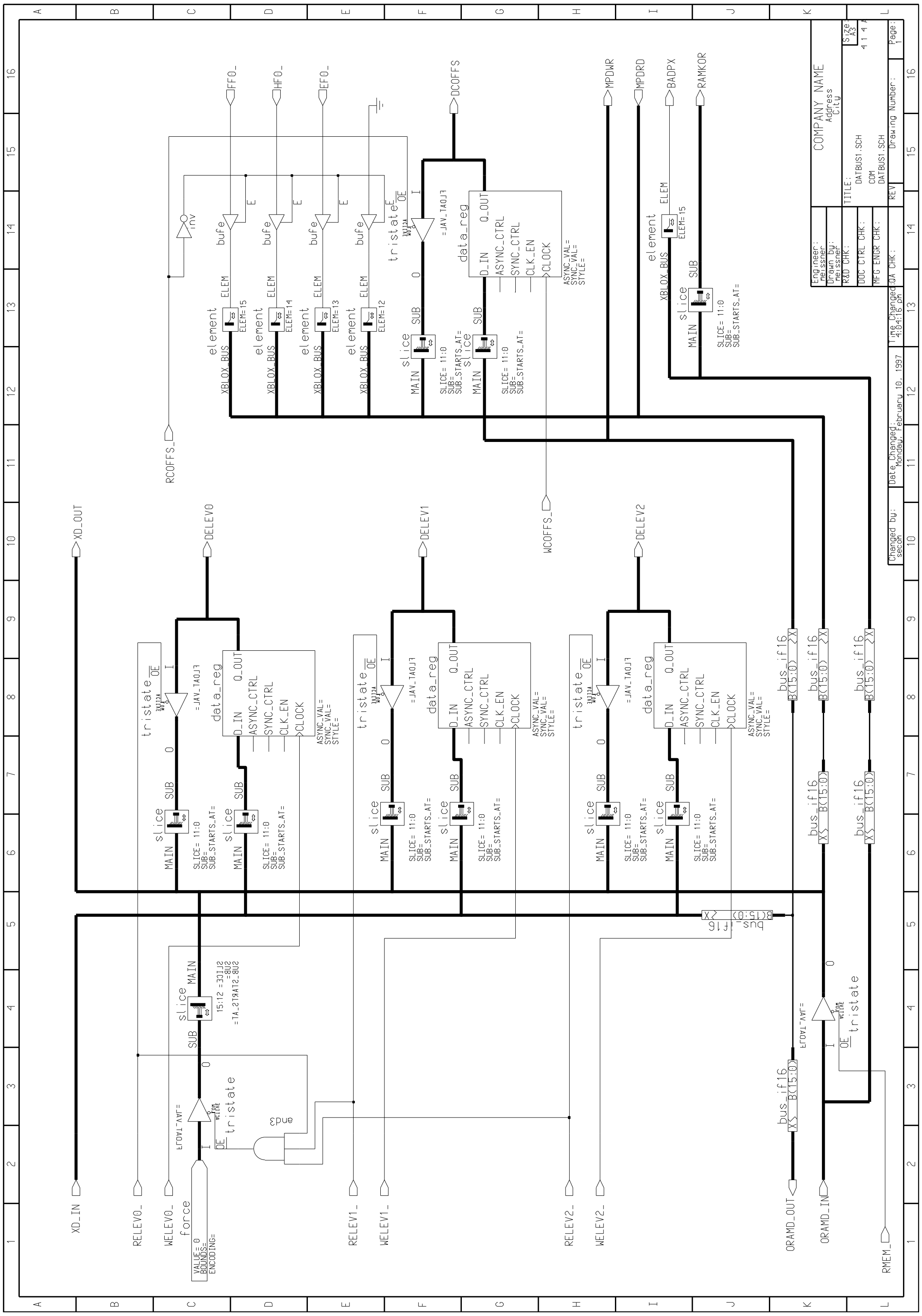


Abbildung D.26: *In „datbus1“ werden die Schwellwerte und der Common-Offset in den Common-Mode-Filter geladen*

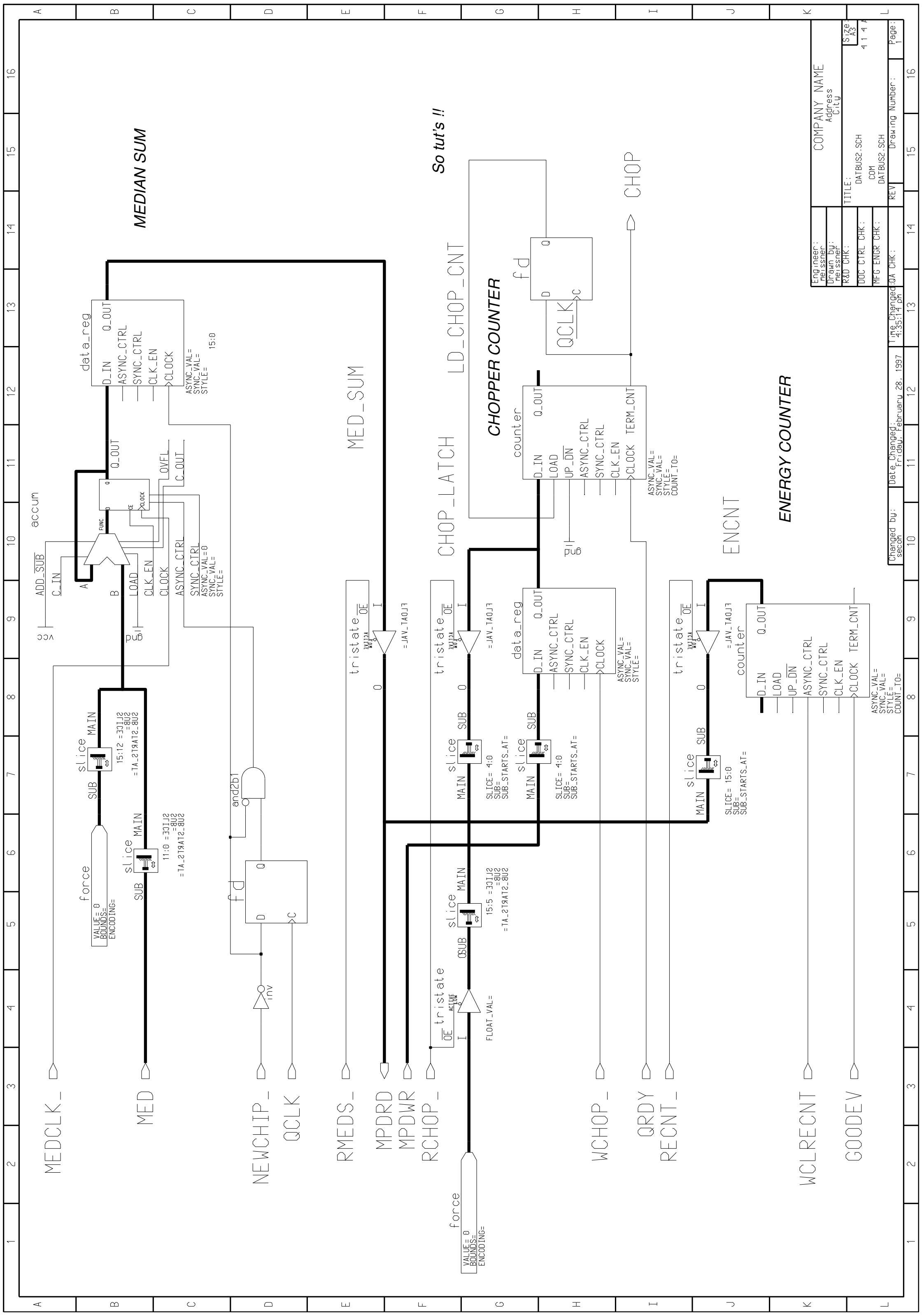


Abbildung D.27: *In „datbus2“ wird der elektronische Chopper eingestellt und bedient, der Kontrollwert Median-Summe ermittelt und die „Goodevents“ gezählt (der Above-Threshold-Counter ATC heißt hier noch Energy Counter ENCNT).*

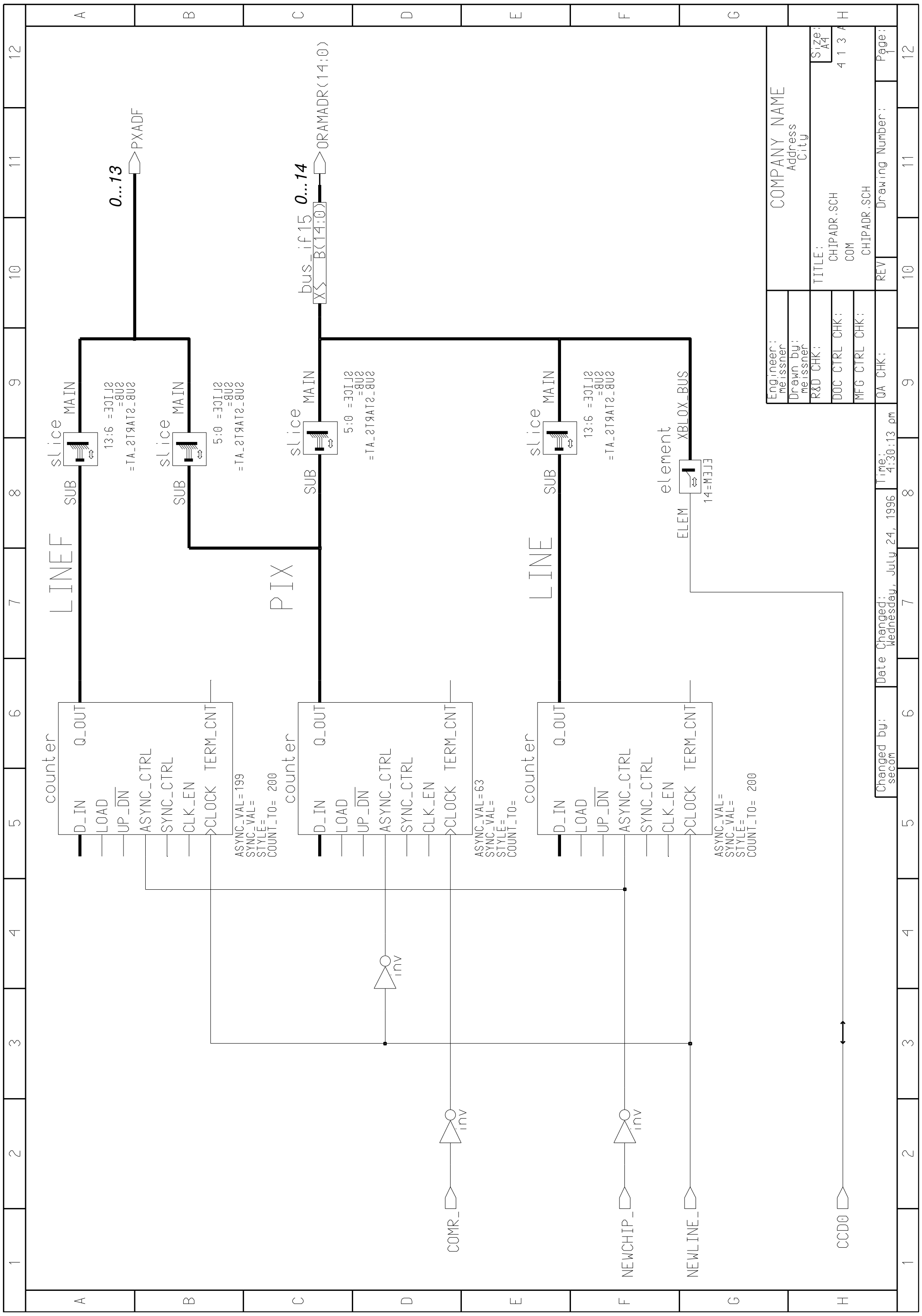


Abbildung D.28: *In „chipadr“ werden die Adressen generiert, sowohl zur Ansteuerung des Offset-RAMs als auch die den Daten zugeordnete Pixeladresse.*

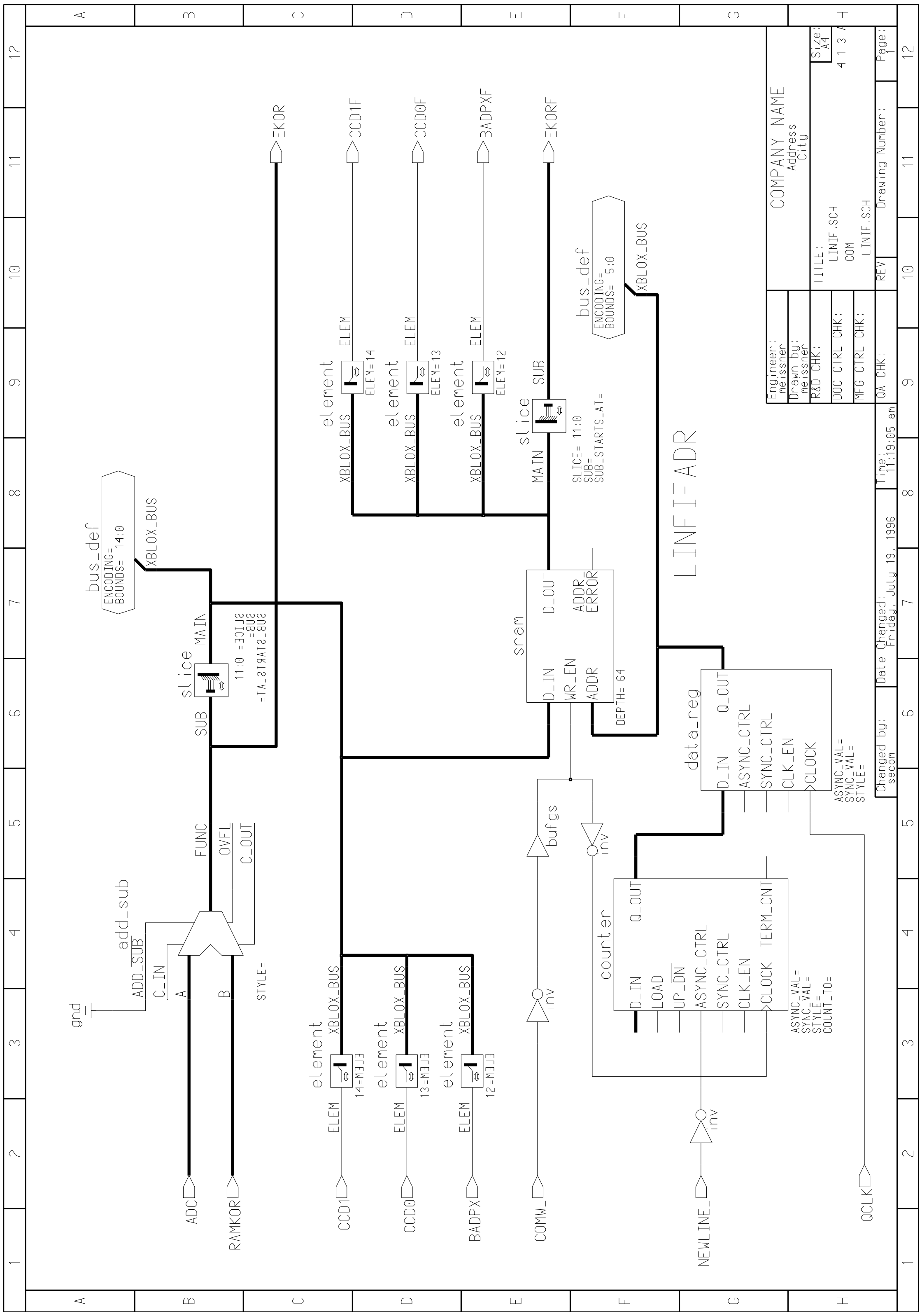


Abbildung D.29: *In „linfif“ werden die digitalisierten Rohdaten mit dem Offset-Wert aus dem ORAM korrigiert und zusammen mit ihrem Bad-Pixel-Wert und der CCD-ID im Line-FIFO zwischengespeichert.*

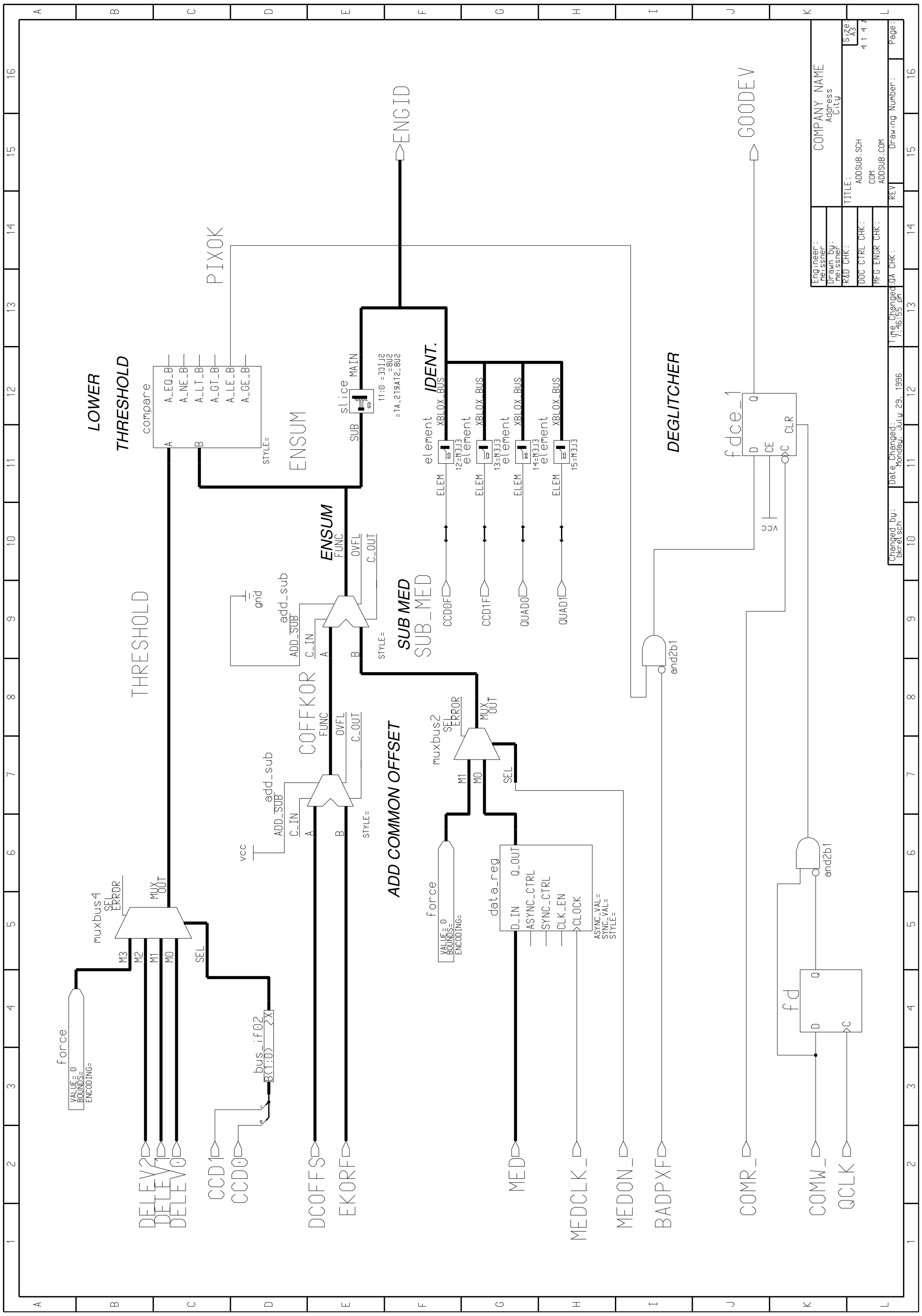


Abbildung D.30: *Im „addsub“ werden die Offset-korrigierten Daten um eine Konstante erhöht, der Median abgezogen und ermittelt, ob es sich um ein „Good“- oder „Bad“-Pixel handelt.*

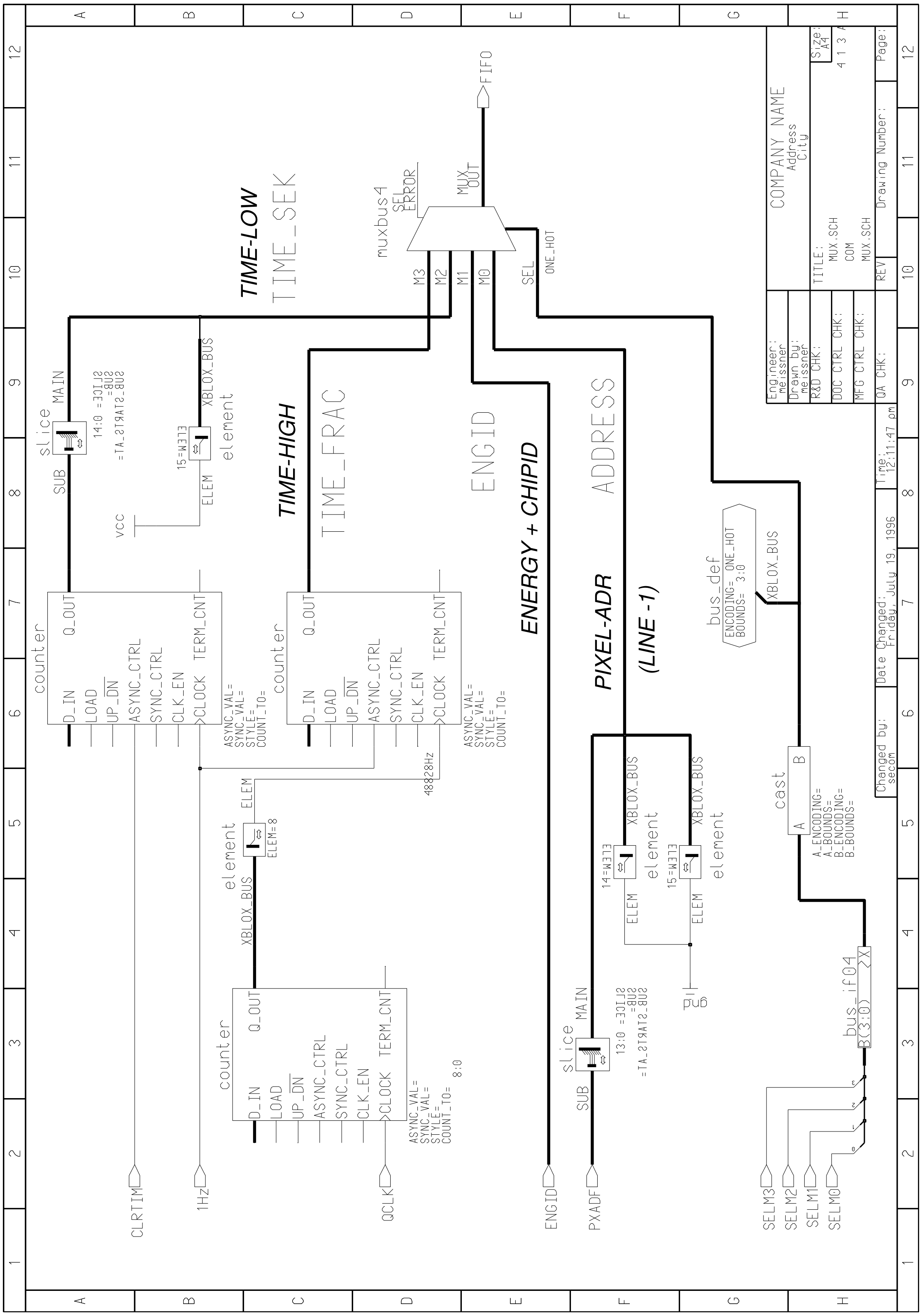


Abbildung D.31: *Der „mux“ bildet das Tor zum CPU-FIFO. Außerdem werden in diesem Schaltungsteil die Zeiteinheiten abgezählt.*

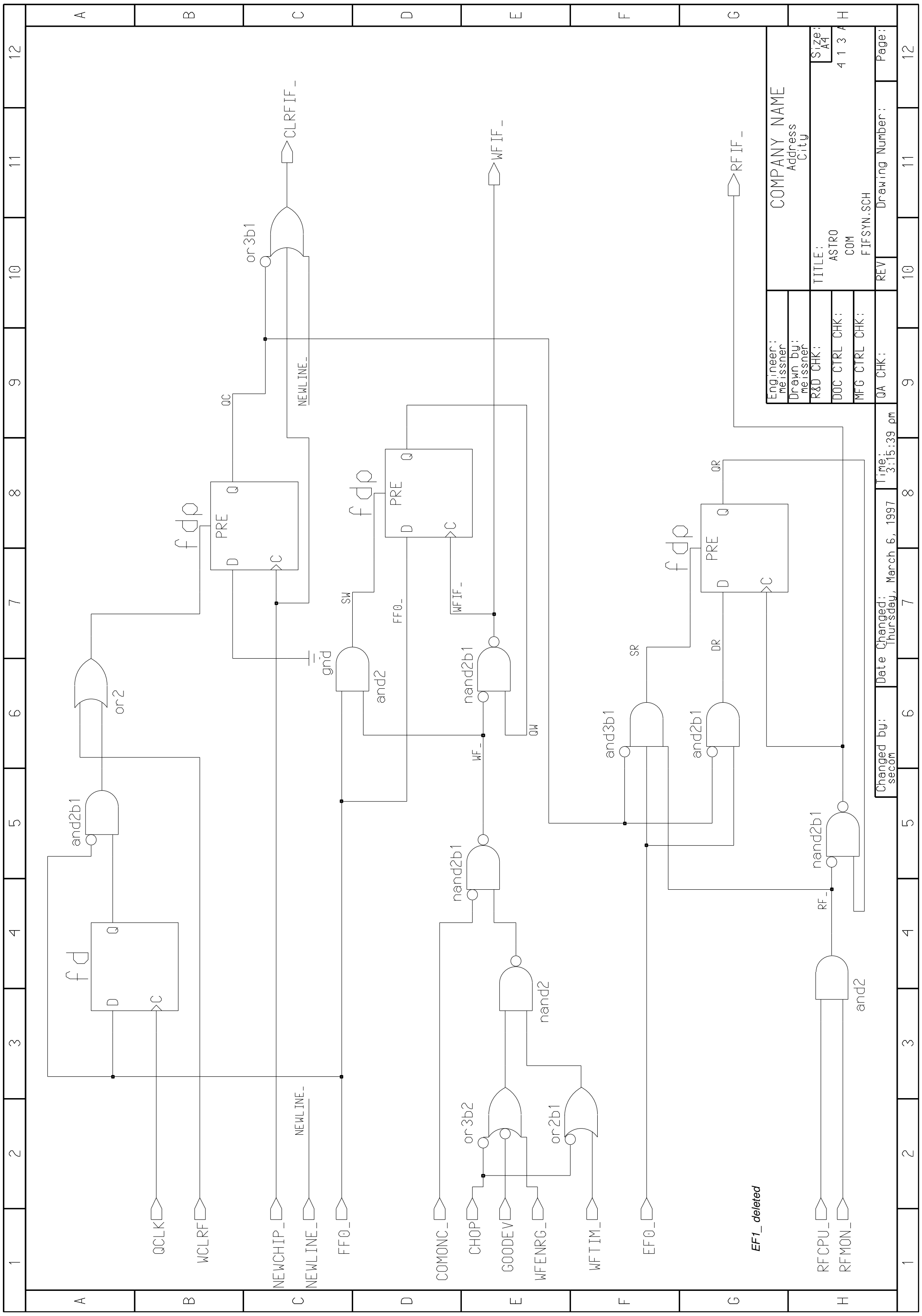


Abbildung D.32: *„fifsyn" übernimmt die Steuerung des CPU-FIFOs.*

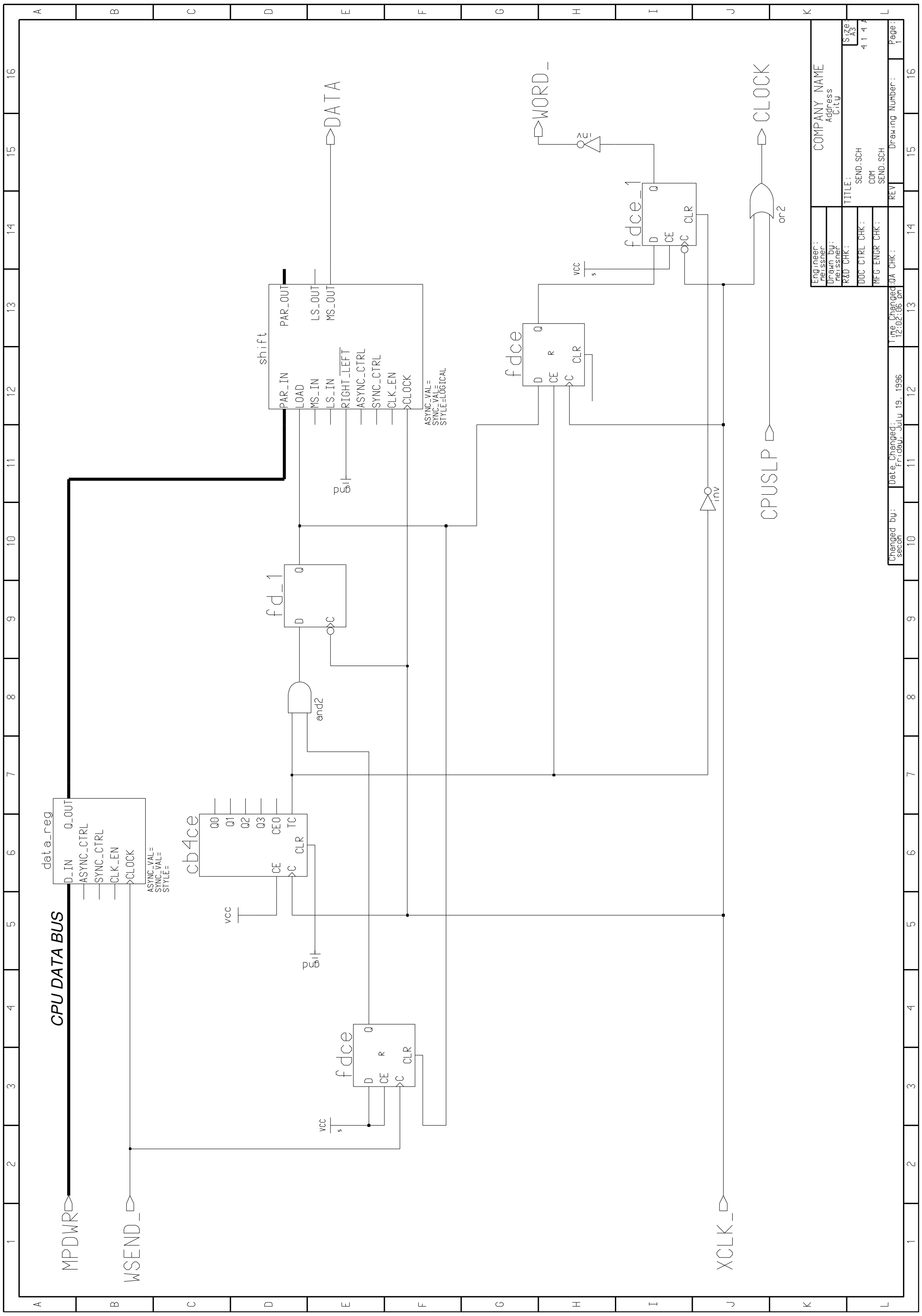


Abbildung D.33: *„send“ ist das High-Bit-Rate-Interface HBR.*

D.4 Das SORT-FPGA

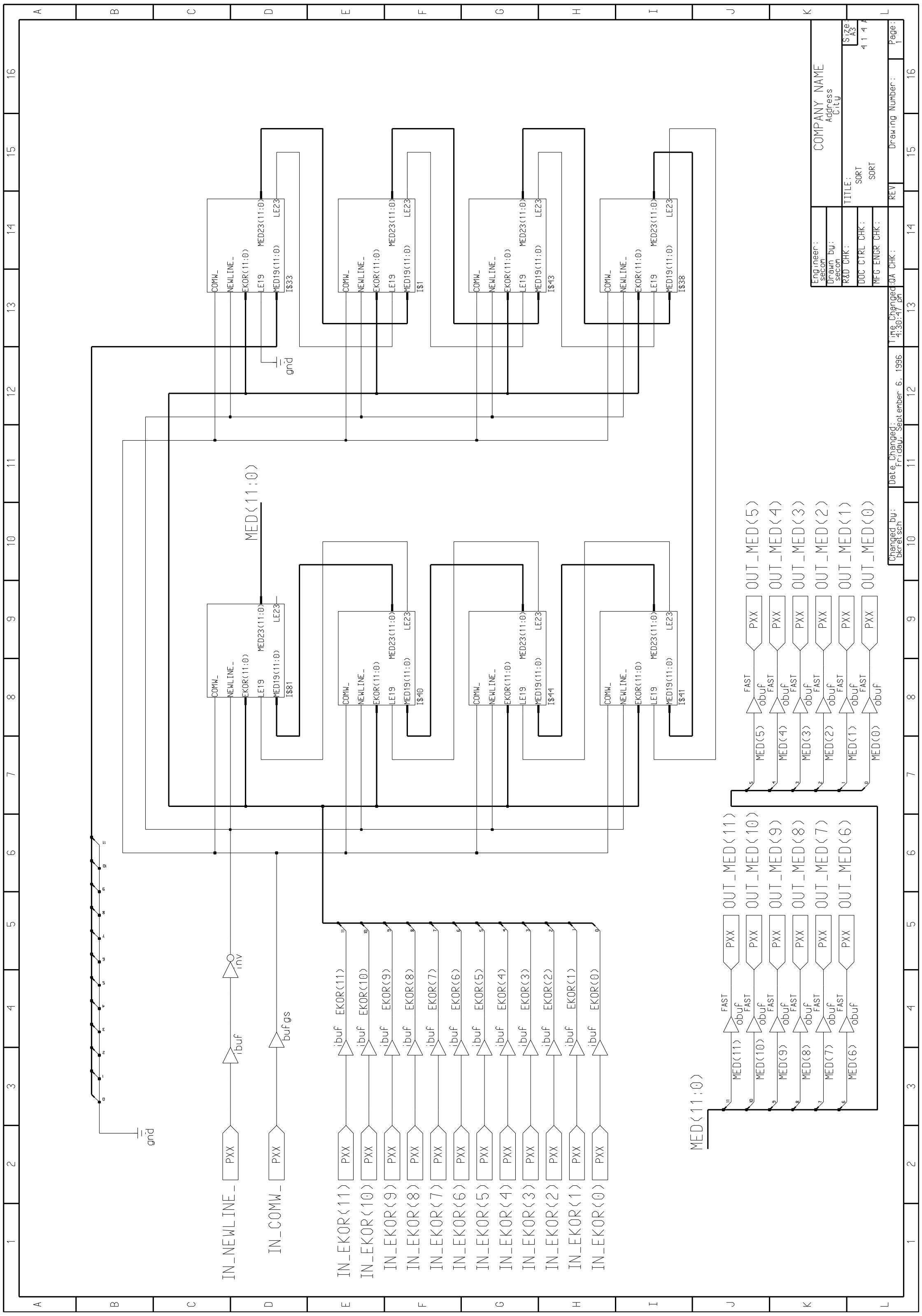


Abbildung D.34: *Im „sort_fpga“ werden die Daten sortiert und der Median bestimmt. Zur besseren Übersicht besteht er aus mehreren identischen Einheiten.*

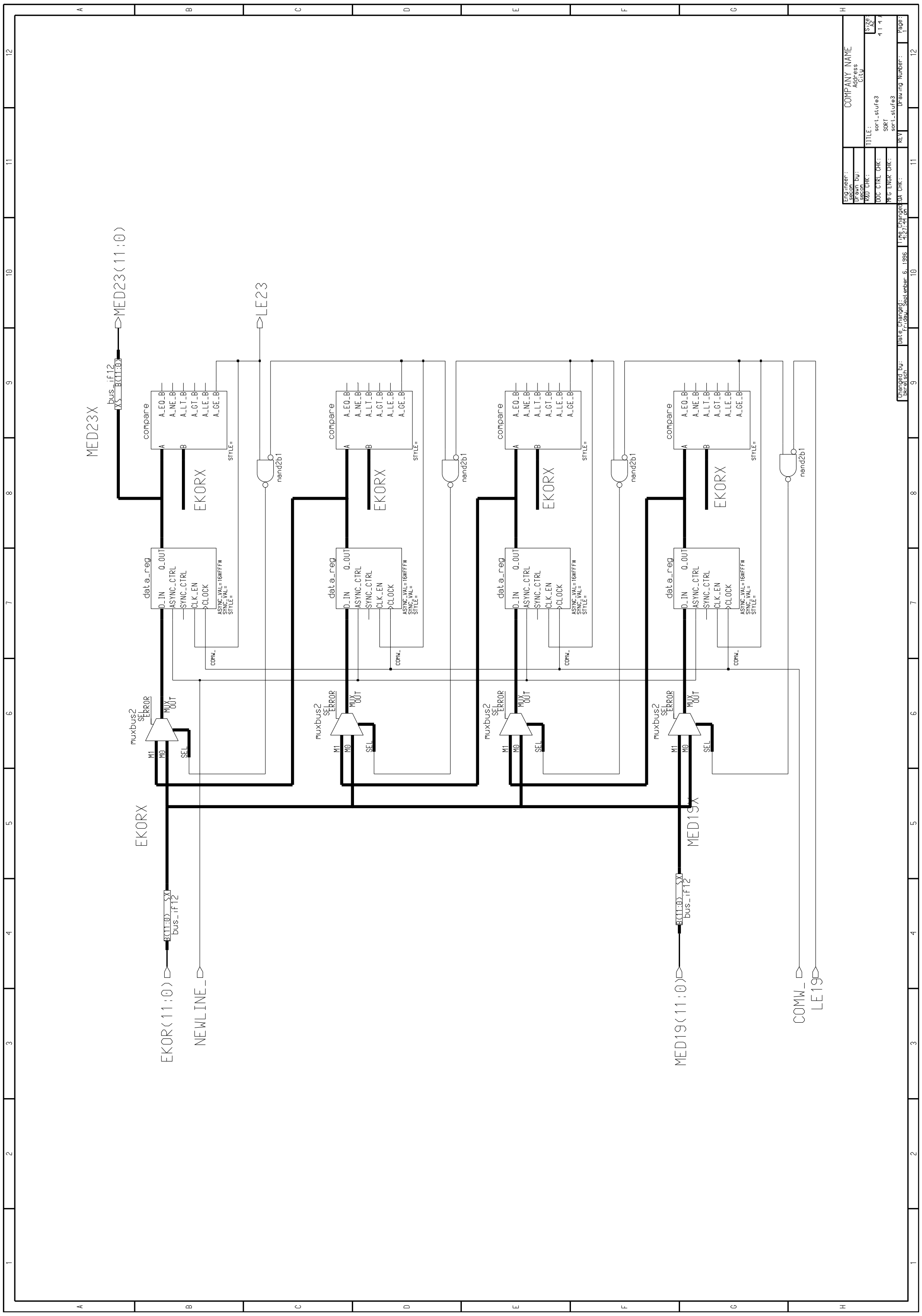


Abbildung D.35: *„sort_unit“ bildet eine der erwähnten Einheiten, in denen bestimmt wird, an welcher Stelle die neu angekommenen Daten einsortiert werden müssen. Die anderen werden im Zweifelsfall verschoben, um dem „Neuankömmling„ Platz zu machen.*

D.5 Selbsterzeugte Bauteile

D.5.1 Die 15-Bit-Bauteile für die LRAM-Adresse

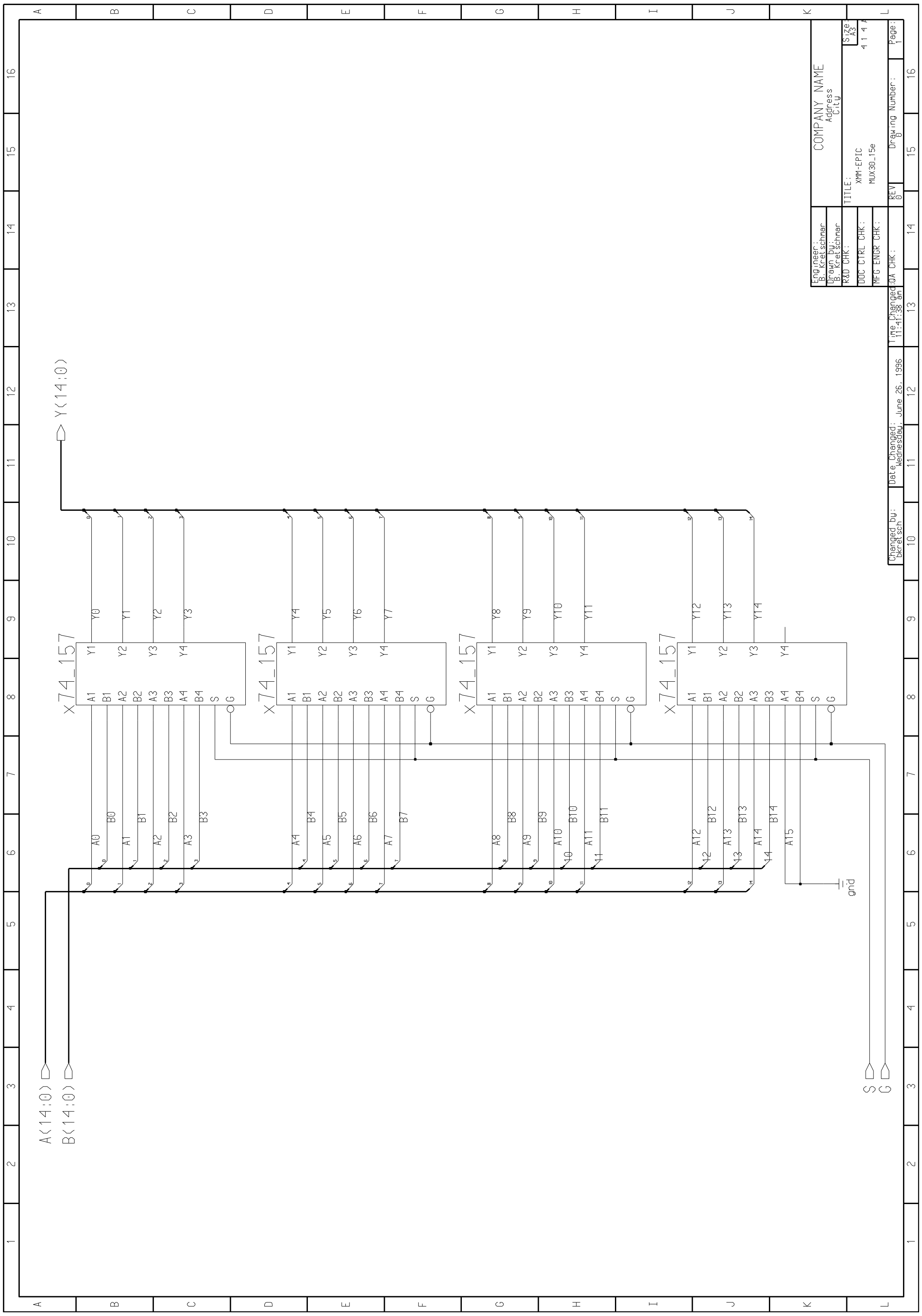


Abbildung D.36: *„MUX30_15e“*

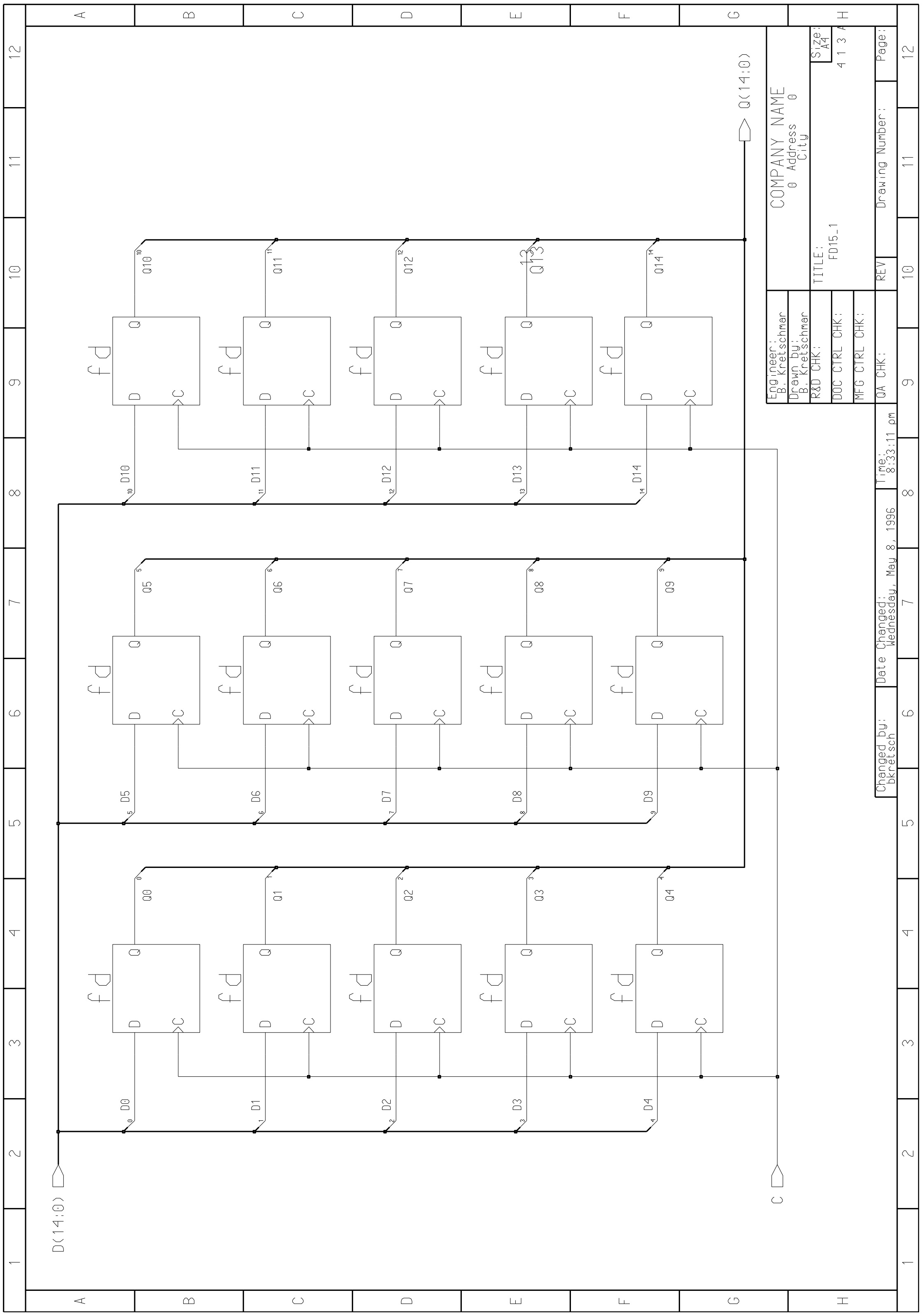


Abbildung D.37: *„FD15_1“*

D.5.2 Der 9-Bit-Johnson-Zähler für die Phi-Puls-Erzeugung

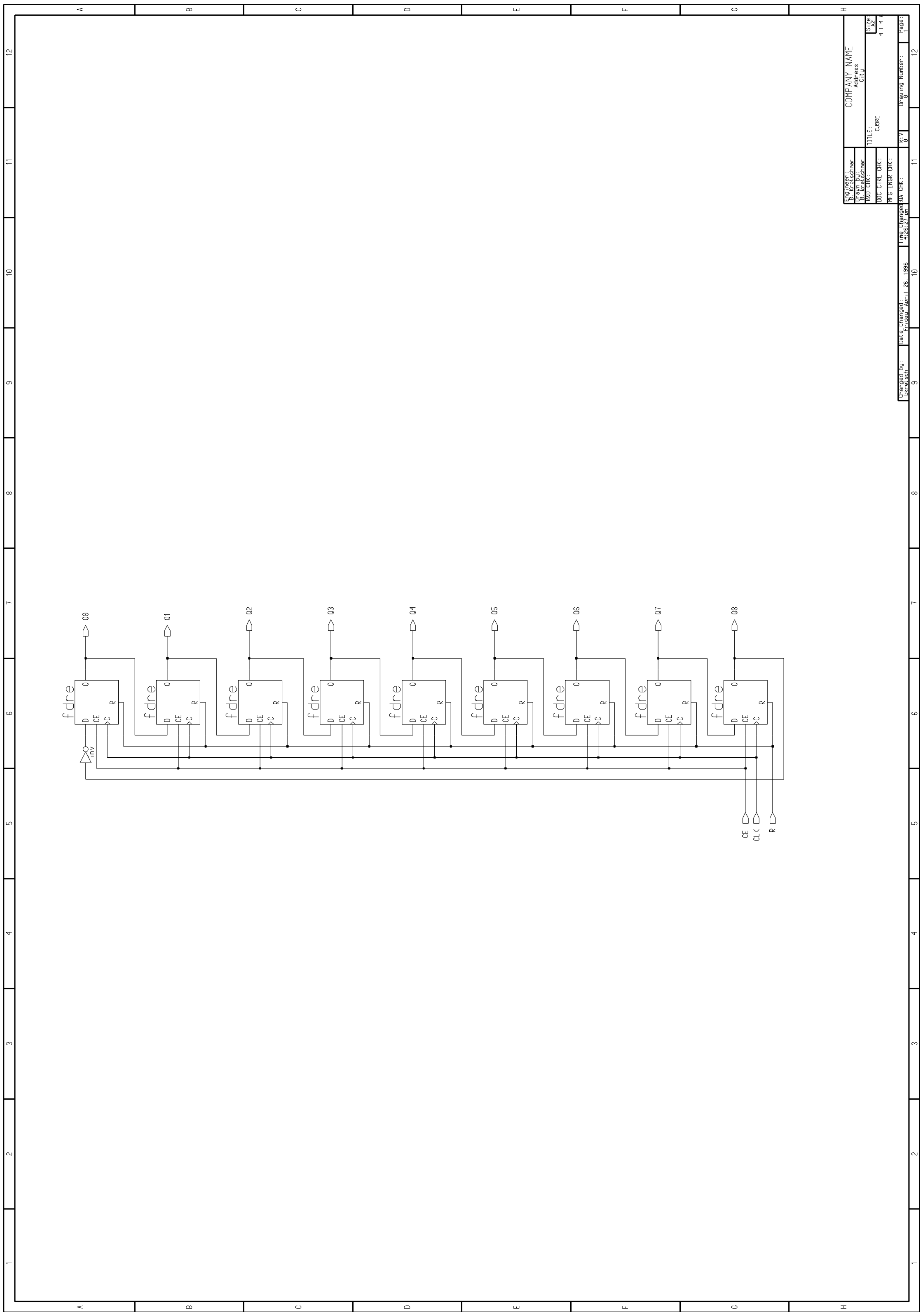


Abbildung D.38: *„CJ9RE“*

D.5.3 Die Flankendetektoren

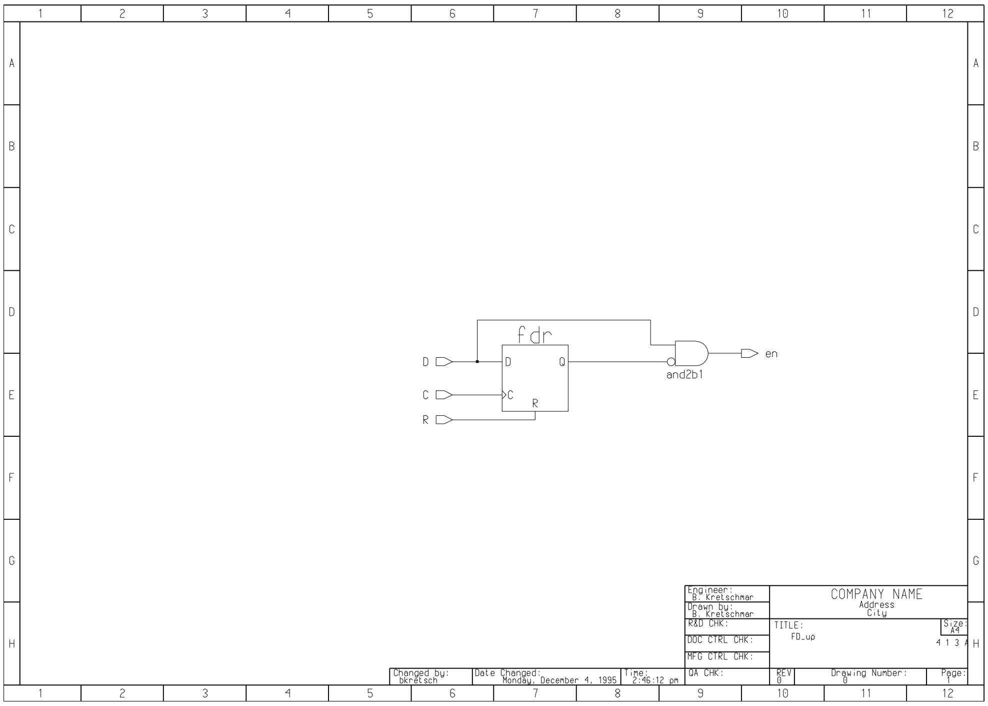


Abbildung D.39: *Der „FD_up" erkennt Signal-Vorderflanken, die mit QCLK↑ erzeugt wurden.*

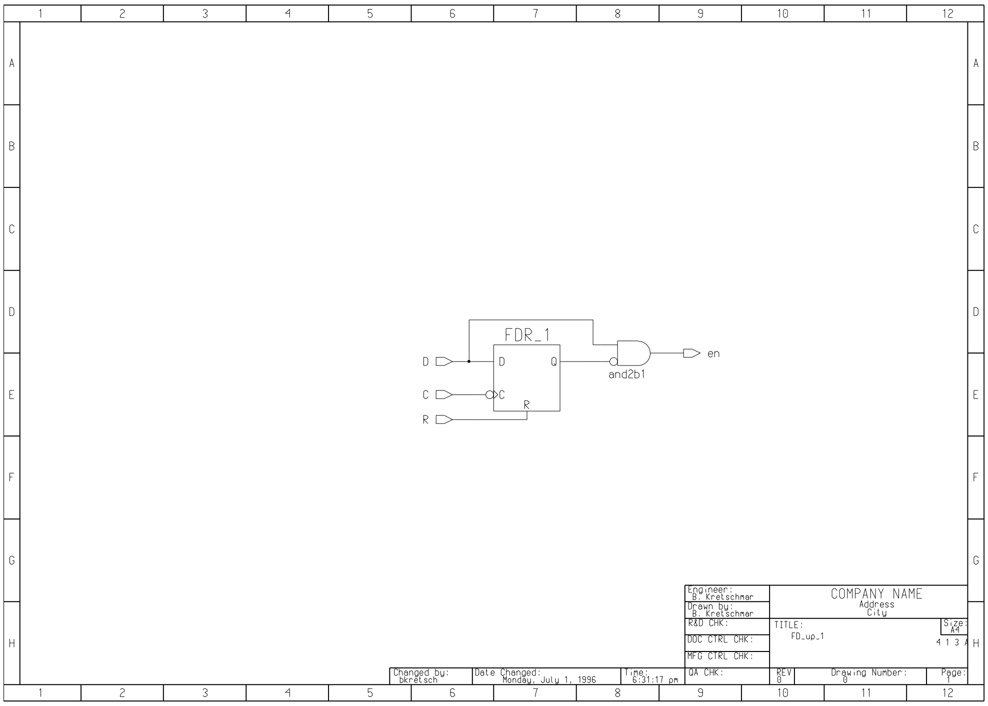


Abbildung D.40: *Der „FD_up_1" erkennt Signal-Vorderflanken, die mit QCLK↓ erzeugt wurden.*

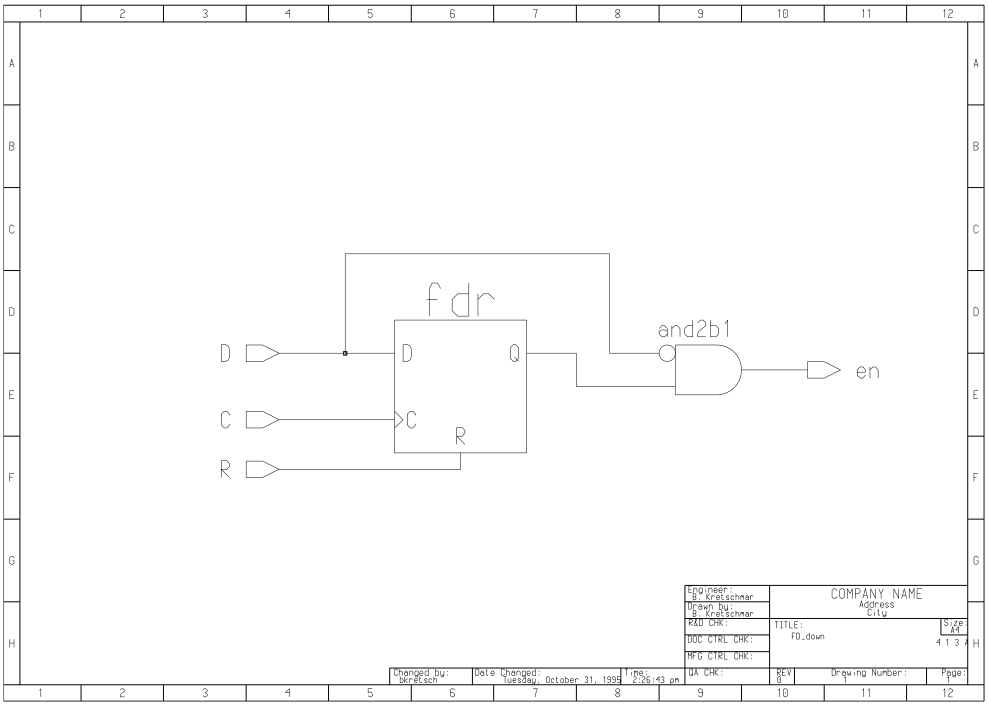


Abbildung D.41: *Der „FD_down“ erkennt Signal-Rückflanken, die mit QCLK↑ erzeugt wurden.*

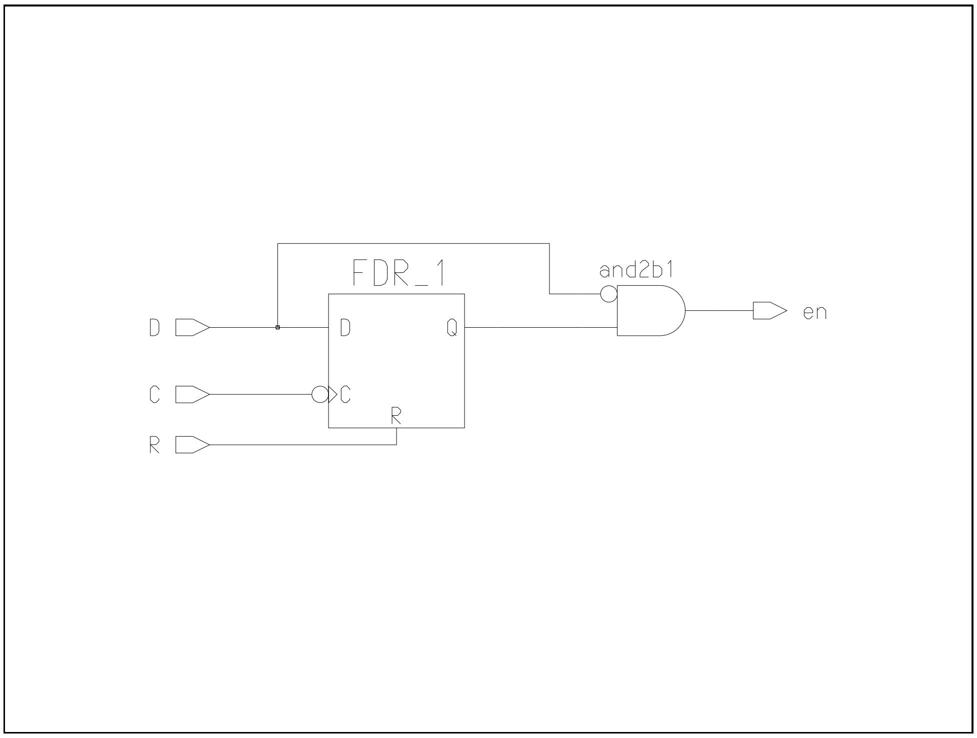


Abbildung D.42: *Der „FD_down_1“ erkennt Signal-Rückflanken, die mit QCLK↓ erzeugt wurden.*

D.5.4 Negativ-Taktflanken-gesteuerte Flip-Flops

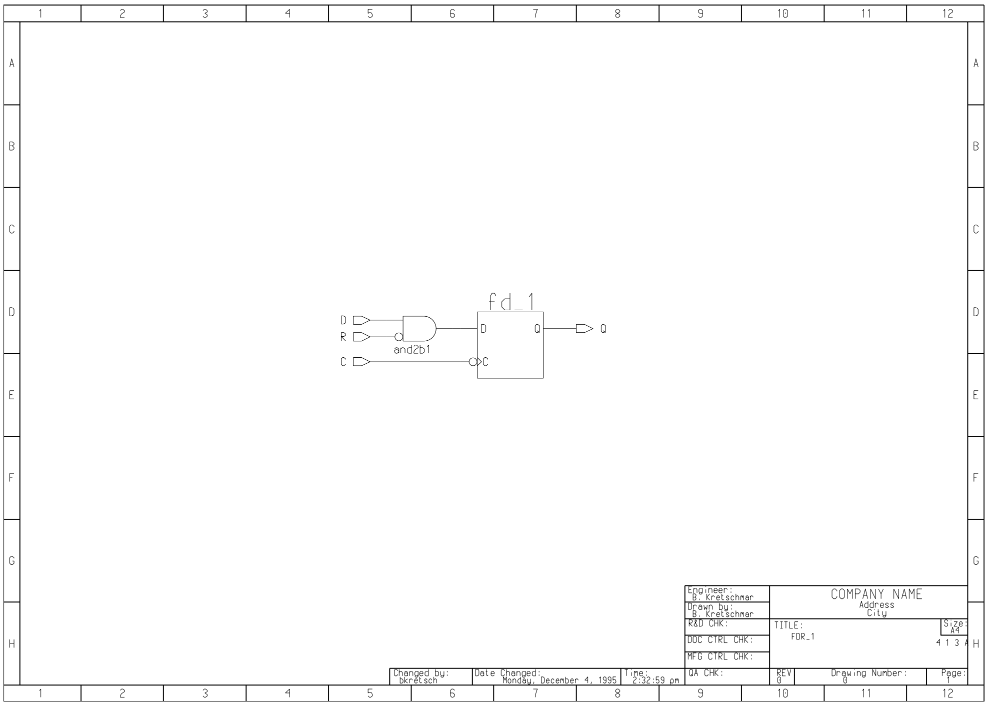


Abbildung D.43: *„FDR_1"*

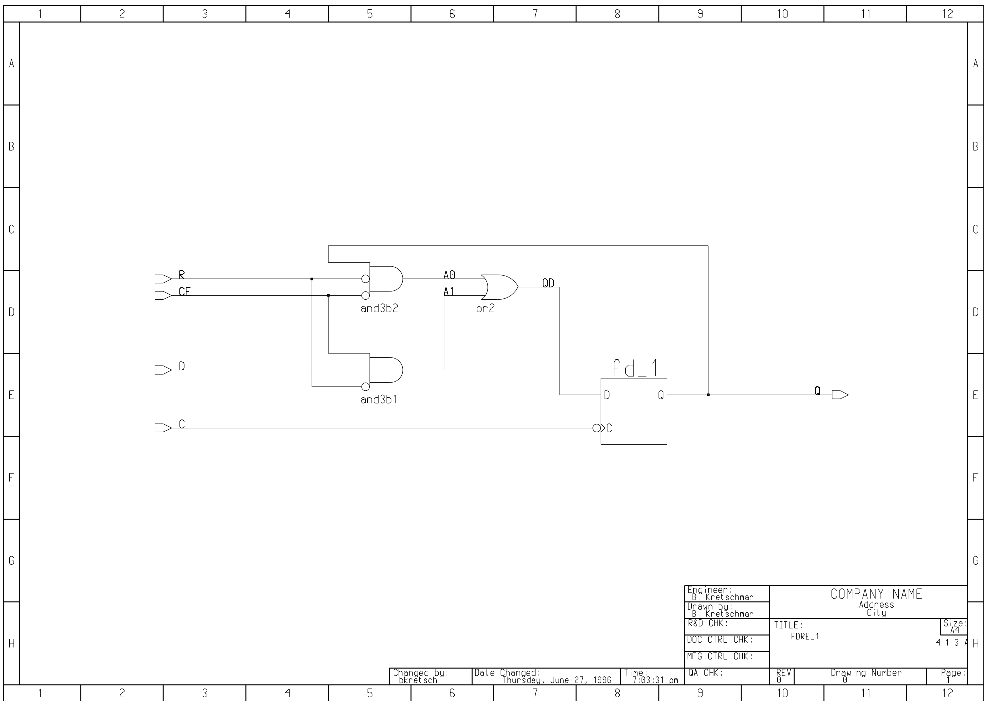


Abbildung D.44: *„FDRE_1"*

Literaturverzeichnis

Appenzeller, I., Mandel, H., Krautter, J., Bowyer, S., Hurwitz, M., Grewing, M., Krämer, G. & Kappelmann, N. (1995). «Medium Resolution Spectroscopy of PKS 2155-304». *ApJ*, **439**, L33–L37.

Bihler, E. (1996). «Analyse spezieller Effekte beim Auslesen der pn-CCD-Detektoren für die Röntgensatelliten XMM und ABRIXAS». *Diplomarbeit*, IAAT.

Bleeker, J. A. M. (1995). «The Significance of the ESA Science Programme». In Guyenne, T.-D. & Battrick, B., *Twenty Years of the ESA Convention*, Vol. ESA SP-387, 57–64.

Brinkman, A. C., Aarts, H. J. M., Boggende, A. J. F. d., Bootsma, T., Dubbeldam, L., Herder, J. W. d., Kaastra, J. S., de Korte, P. A. J., van Leeuwen, B. J., Mewe, R., vn Zwet, E., Decker, T., Hailey, C. J., Kahn, S. M., Paerels, F., Pratuch, S., Rasmussen, A., Branduardi-Raymont, G., Guttridge, P., Bixler, J., Thomsen, K., Zehnder, A. & Erd, C. (1996). «The Reflection Grating Spectrometer on-board of XMM». In Siegmund, G., *EUV, X-Ray and Gamma-Ray Instrumentation for Astronomy VII*, Vol. 2808, 463–480.

Burke, W. R. (1996). *ESA's Report to the 31st COSPAR Meeting*, Vol. ESA SP-1194.

Casteren, J. v. (1996). «The X-Ray multi mirror spacecraft, a large telescope». In Siegmund, G., *EUV, X-Ray and Gamma-Ray Instrumentation for Astronomy VII*, Vol. 2808, 338–349.

Chambure, D. d., Lainé, R., Katwijk, K. v., Casteren, J. v. & Glaude, P. (1996). «The Status of the X-Ray Mirror Production for the ESA XMM Spacecraft». In Siegmund & Gummin, *EUV, X-Ray and Gamma-Ray Instrumentation for Astronomy VII*, Vol. 2808, 362–375. SPIE.

Englhauser, J. (1996). «Untersuchungen der harten Röntgenstrahlung von Supernova 1987A». *Dissertation*, Technische Universität München.

Kendziorra, E. (1976). «Beobachtungen des CRAB-Nebels und -Pulsars im Harten Röntgenbereich». *Dissertation*, Astronomisches Institut der Universität Tübingen.

Kendziorra, E. (1995). *Requirement Specification for the EPIC PN Eventanalyser (EPEA)*.

Kendziorra, E., Diesch, C. & Mohan, M. (1995). *Unit Design Document for the EPIC PN Eventanalyser (EPEA)*.

Kretschmar, P. (1991). «Kalibration von Hochenergieröntgendetektoren am Spektrum des Crab-Nebels und Untersuchungen am Röntgenpulsar A 0535+26». *Diplomarbeit*, Astronomisches Institut der Universität Tübingen.

Kuster, M. (1997). «Untersuchung spezieller Eigenschaften des pn-CCD Arrays für die Röntgensatelliten XMM und Abrixas». *Diplomarbeit*, Institut für Astronomie und Astrophysik der Universität Tübingen (Abt. Astronomie).

Lumb, D., Eggel, H., Lainé, R. & Peacock, A. (1996). «X-Ray Multi-Mirror Mission - an overview». In Siegmund, G., *EUV, X-Ray and Gamma-Ray Instrumentation for Astronomy VII*, Vol. 2808, 326–337.

Martini, R. (1993). «Aufbau eines Meßplatzes zur Kalibrierung der integrierten Ausleseeinheit CAMEX64 für Röntgen-CCDs». *Diplomarbeit*, Astronomisches Institut der Universität Tübingen.

Mason, K., Cropper, M. & Hunt, R. (1996). «The XMM Optical/UV Monitor Telescope». In Siegmund, G., *EUV, X-Ray and Gamma-Ray Instrumentation for Astronomy VII*, Vol. 2808, 438–447.

Meidinger, N., Bräuninger, H., Hartmann, R., *et al.* (1996). «PN-CCD detector for the European Photon Imaging Camera an XMM». In Siegmund, G., *EUV, X-Ray and Gamma-Ray Instrumentation for Astronomy VII*, Vol. 2808, 492–504.

Schaumberger, J. (1994). «Voll depletierbare pn-CCD's für die Röngenastronomie». *Diplomarbeit*, Astronomisches Institut der Universität Tübingen.

Schuster, P. (1994). «Zeitliche und räumliche Auflösung neuartiger PN-CCDs für den Röntgensatelliten XMM». *Diplomarbeit*, Astronomisches Institut der Universität Tübingen.

Siegmund, O. & Gummin, M. (1996). *EUV, X-Ray and Gamma-Ray Instrumentation for Astronomy VII*, Vol. 2808. SPIE — The International Society for Optical Engineering.

Staubert, R., Kendziorra, E., Trümper, J., Reppin, C., Hoffman, J. A., Pounds, K. A., Giles, A. B. & Morrison, L. V. (1975). «The Crab-Nebula: High energy X-ray observation of a lunar occultation». *ApJ*, **201**, L15–L19.

Trümper, J., Pietsch, W., Reppin, C., Voges, W., Staubert, R. & Kendziorra, E. (1978). «Evidence for strong cyclotron line emission in the hard X-ray spectrum of Hercules X-1». *ApJ*, **219**, L105–L110.

Villa, G., Abbey, A. & Arnaud, M. (1996). «The EPIC system onboard the ESA XMM mission». In Siegmund, G., *EUV, X-Ray and Gamma-Ray Instrumentation for Astronomy VII*, Vol. 2808, 402–413.

Danksagungen

Nicht nur während der Fertigstellung dieser Arbeit, sondern bereits während meines gesamten Studiums haben mich immer wieder Leute unterstützt und mir in vielerlei Hinsicht geholfen. Bei ihnen allen möchte ich mich an dieser Stelle bedanken!

Mein besonderer Dank gilt den Herren Rüdiger Staubert und Eckhard Kendziorra: Sie haben mir nicht nur eine Diplomarbeit anvertraut, wie ich sie mir besser kaum hätte wünschen können, sondern haben mich die ganze Zeit über, die ich hier am Institut verbracht habe, immer nach ihren allerbesten Kräften unterstützt. Außerdem ist es ihnen gelungen, an diesem Institut eine wunderbar kameradschaftliche Atmosphäre zu schaffen, in der es großen Spaß macht zu arbeiten!

Desweiteren möchte ich mich bei Herrn Thierry Courvoisier, Leiter des INTEGRAL Science Data Centers, und seinem Systemadministrator Alain Aubord ganz herzlich bedanken; sie haben mir in selbstloser Weise einen Arbeitsplatz am ISDC ermöglicht, ohne den diese Arbeit kaum hätte fertiggestellt werden können!

Ebenfalls danken möchte ich unseren verschiedenen „Werkstatt-Crews“. Sei es Feinmechanik, Holz oder Elektronik, immer war ein offenes Ohr und aktive Hilfe da, wenn ich mit Problemen oder Fragen ankam! Speziell im Rahmen der Diplomarbeit sind hier die Herren Jürgen Krämer und Heinz Böttcher zu nennen, die den Master-Sequenzer, bzw. den Common-Mode-Filter entworfen haben; sie hatten immer eine Engelsgeduld mit mir, und die Zusammenarbeit war äußerst fruchtbar.

Eine ebenfalls lange und intensive Zusammenarbeit verbindet mich mit unserem „ASIC-Team“ von der Firma DD & T: Oliver Renz hat mir mein Wissen bezüglich Mentor® vermittelt; Urs Kanus hat sich mit „meinem“ QSEQ herumgeschlagen und das gesamte Referenz-Design auf einen simulierbaren Stand gebracht; Michael Meißner hat den Common-Mode-Filter von PC auf Workstation übertragen und ebenso wie Frank Lier den ersten Kraftakt mitgemacht, der im August 1996 noch einen ASIC-Auftrag an Thomson ergeben sollte; und unser französischer Praktikant Jean-Marc Nivet hat sich die Mühe gemacht, die von H. Böttcher am Logic-Analyzer aufgenommenen Signalmuster der CPU für den Simulator als Kommando-Dateien aufzubereiten.

Des weiteren möchte ich „den Münchnern“ danken: Herr G. Metzner hat mir einige Prinzipien des CAMEX klargemacht und während unserer Messungen seine Breakout-Box vermißt; Christoph von Zanthier hat alle Fragen bezüglich der CCD-Ansteuerung versucht zu beantworten und mir zu meiner großen Erleichterung das Schaltbild des CAMEX (Abbildung 3.7) zur Verfügung gestellt. Bei den Messungen an der ZETA-Anlage waren vor allem die Herren Elmar Pfeffermann, Norbert Meidinger und ihr Praktikant Felix unermüdlich; ohne sie wäre es mir nicht möglich gewesen, die Erfahrungen mit den 3 CCDs eines Quadranten zu sammeln und in der Schaltung zu berücksichtigen.

Herr Gerhard Krämer hat mir vor allem in Verwaltungsangelegenheiten viel geholfen, mich aber auch mit den Informationen über ORFEUS für die Einleitung versorgt.

Herr Wolfgang Gringel hat mir bei ARIES geholfen und mich Nachmittage lang in seinem Büro am Scanner ertragen.

Meinem Mitdiplomanden Markus Kuster möchte ich danken für die Mühe, die er sich gemacht hat, immer gute Software-Versionen für die Arbeit auf den Workstations bereit zu halten, und dafür, daß er sich in alle Bereiche der EPEA so gut eingearbeitet hat, daß er jedem ein kompetenter Mitarbeiter mit viel Verständnis ist.

Auch dem Rest der „Hundehütten-Crew“ Edgar Bihler, Willy Grubmiller, Roland Brutscher und Bernhard Pflüger sei hier gedankt für die gute Zusammenarbeit und für manches anregende Gespräch fachlicher und nicht fachlicher Natur.

Spezieller Dank gilt auch meinem Schwager Michael Kretschmar, der mir während unseres Studiums ein besonderer Freund geworden ist und mich durch die Prüfungen geschleppt hat, und Dietmar König, dem Dritten in unsererm Bunde.

Ebenfalls gedankt sei Andrea Betsch, die sich die Mühe gemacht hat, den ersten Teil der Diplomarbeit Korrektur zu lesen und mir dabei alle „sowohl-als-auch-Kommata“ gestrichen hat.

Nicht vergessen möchte ich meine Eltern mit ihren neuen Partnern, die dieses Studium geduldig mitgetragen haben und auf die ich mich immer verlassen kann, und Christa Kretschmar, die alle Sprüche über schlechte Schwiegermütter Lügen straft und mir eine echte Freundin geworden ist.

Last but not at all least sei mein Mann Peter genannt, der mich auf Händen trägt und ein schier unerschöpfliches Wissen über Postscript und LaTeX hat; er hat viele Stunden damit zugebracht alle meine diesbezüglichen Wünsche in die Tat umzusetzen, und dieser Arbeit auf diese Weise zu ihrem endgültigen Aussehen verholfen.