Eberhard-Karls-Universität Tübingen
Institut für Astronomie und Astrophysik
Abteilung Astronomie

Diplomarbeit

Entwicklung und Test einer Hardware-Elektronik für das IBIS Experiment an Bord des ESA-Satelliten INTEGRAL

Claus Dreischer

November 2001

Inhaltsverzeichnis

1 Einleitung

Diese Arbeit beschreibt eine Hardware-Entwicklung für den ESA Satelliten INTEGRAL. Da die Beschreibung von Hardware sehr unanschaulich werden kann, sind die Aufgaben der einzelnen Baugruppen am Anfang dieser Arbeit, die genauere Beschreibung wie diese Aufgaben realisiert wurden aber erst im Anhang ab Seite 55 zu finden. Dort befinden sich auch die Schaltpläne der wissenschaftlichen Module CSI_AC, MP, SPT und CSI_HIST, die ich entwickelt habe oder an denen ich mitgewirkt habe. Das TC Modul ist in [11] zu finden. Die Schaltpläne erleichtern das Verständnis der Schaltung. In der elektronischen Form dieser Arbeit (als PDF-Datei) ist es so sehr einfach möglich, interessante Aspekte dieser Schaltpläne vergrößert darzustellen.

1.1 Astronomie

Tübingen kann auf eine sehr lange Tradition im Bereich Astronomie zurückblicken. Am 25.9. 1588 erlangte Johannes Kepler in Tübingen das Baccalaureat.

Im Jahre 1604 veröffentlichte er sein erstes astronomisches Werk, „Astronomiae Pars Optica“, in dem erstmals die Funktionsweise der *camera obscura*[1], der Lochkamera, vollständig erklärt wurde. Nach der Erfindung des Fernrohrs und der erstmaligen Verwendung für Himmelsbeobachtungen durch G. Galilei war es Kepler, der bald darauf eine theoretische Erklärung für den sog. Fernrohreffekt liefern konnte.

Durch die Atmosphäre der Erde behindert, konnte in den folgenden Jahrhunderten nur ein kleiner Teil der eletromagnetischen Strahlung der Sterne zur Beobachtung herangezogen werden.

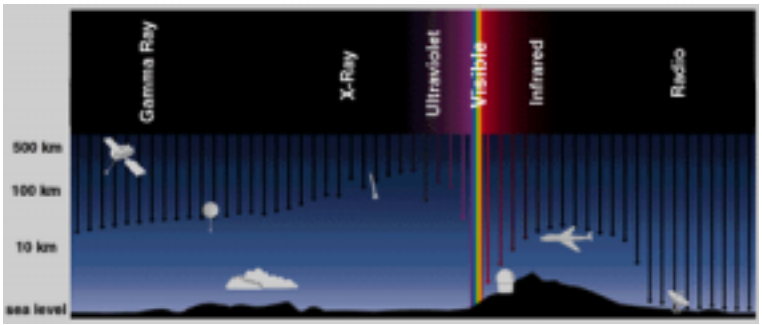


Abbildung 1.1: Das elektromagnetische Spektrum und das Beobachtungsfenster in Abhängigkeit von der Höhe. Vom Boden aus ist nur der optische und der Radio Bereich zu beobachten.

[1] Gerade dieses Prinzip der Lochkamera wird beim INTEGRAL-Satelliten zur bildgebenden Beobachtung benutzt

Erst seit der zweiten Hälfte des vergangenen Jahrhunderts steht mit Ballonexperimenten, Raketen und Satelliten der ganze Bereich der Strahlung der wissenschaftlichen Beobachtung zur Verfügung.

1.2 Röntgen- und Gamma-Astronomie

Photonen aus diesem Bereich sind für das Verständnis hochenergetischer astronomischer Prozesse wichtig. Aus diesem Grund ist INTEGRAL nicht der erste Satellit zur Beobachtung im Gamma-Bereich, sondern kann auf eine ganze Reihe von erfolgreichen Vorgängern blicken:

- SAS-2 1972. Erster Gamma-Satellit, NASA
- COS-B 1975, ESA
- GRANAT 1989, Rußland
- Compton GRO 1991, NASA

Mit der 40-fach besseren Energieauflösung und den besseren abbildenden Instrumenten von INTEGRAL erhofft man sich weitere Einblicke und Erkenntnisse.

Für das Astronomische Institut Tübingen ist die aktive Mitarbeit an Satelliten-Experimenten nicht neu. In den vergangenen Jahren konnte sich das AIT[2] u.a. bei den Satelliten Mir-HEXE, ROSAT, ABRIXAS und XMM einbringen und Know-How in diesem Bereich sammeln.

[2] Astronomisches Institut Tübingen

2 Der INTEGRAL-Satellit

INTEGRAL (**Inte**rnational **G**amma-**R**ay **A**strophysics **L**aboratory) ist das nächste mittelgroße Satelliten-Projekt (M2) der ESA. Wie bei XMM (cornerstone Projekt der ESA) konnte auch bei INTEGRAL unser Institut einen Beitrag zum Bau des Satelliten leisten, um im Gegenzug eigene Beobachtungszeit gewährt zu bekommen.

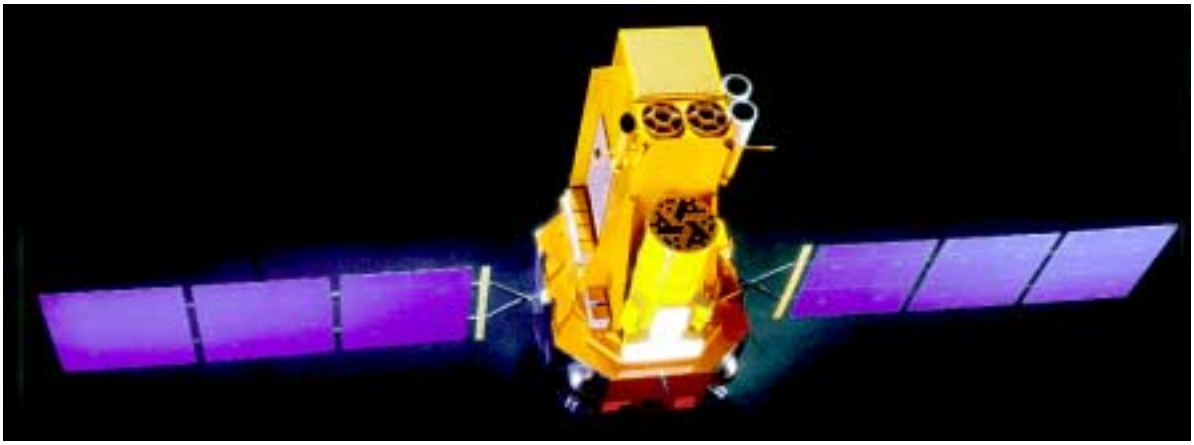


Abbildung 2.1: Der INTEGRAL-Satellit.

Der INTEGRAL-Satellit wurde entwickelt, um genaue Spektroskopie (ΔE: 2 keV FWHM[1] bei 1,3 MeV) und hochauflösende Bilder (Winkelauflösung: 12' FWHM) von Gammaquellen im All zu ermöglichen. Der zu Verfügung stehende Meßbereich reicht von 15 keV bis zu 10 MeV. Dabei stehen durch weitere Instrumente noch der Röntgenbereich von 3 – 35 keV und ein optisches Band (V, 550 nm) zur Verfügung.

Alle Instrumente sind parallel angeordnet, um eine gleichzeitige Beobachtung mit allen Instrumenten zu gewährleisten. Die erreichte Auflösung, die beobachtbaren Spektren, die zeitliche Auflösung der Daten und das große Gesichtsfeld sind einzigartig und lassen interessante Ergebnisse während der Missionsdauer[2] erwarten.

2.1 Aufgaben

Die Beobachtungszeit von INTEGRAL läßt sich in zwei Bereiche unterteilen:

- *Core Programme*
  Neben den beschriebenen Aufgaben im Core Programme dient dieser Teil auch als Be-

[1] Full Width Half Maximum
[2] nominale Betriebsdauer: 2 Jahre, erweiterbar auf 5 Jahre

lohnung für die an der Entwicklung und Betrieb beteiligten Institute. Sie haben hier eine garantierte Beobachtungszeit, welche unabhängig ist vom *General Programme* (s. u.).

  - wiederholte Untersuchung der galaktischen Ebene, *Galactic Plane Scans, GPS*
    * Um die kurzzeitigen Ausbrüche im Gamma-Bereich möglichst früh untersuchen zu können, wird die galaktische Ebene wiederholt beobachtet. Man erhofft sich gerade aus der Anfangsphase eines Ausbruches weitere Erkenntnisse.
    * Erstellung einer zeitlich aufgelösten Karte der galaktischen Ebene mit Kontinuums- wie auch mit Linien-Emissionen wie z. B. ^{26}Al und 511 keV.
  - genaue Untersuchung des galaktischen Zentrums, *Galactic Central Radian Deep Exposure, GCDE*
    * Untersuchung der Kernsynthese durch Linien-Emissionen von ^{26}Al (Proton-Kern Interaktion, 1,809 MeV Linie), ^{44}Ti und 511 keV.
    * Kartierung der Kontinuumsstrahlung außerhalb der galaktischen Zentralregion.
    * Hochauflösende Bilder und Spektren der galaktischen Zentralregion.
    * Erneute Untersuchung von Quellen gefunden durch COMPTEL und CGRO
  - Untersuchung ausgewählter Quellen wie GRS 1915+105, GROJ 1655-40, 1E 1740.7-2942, Cyg X-1, Cyg X-3, GX 339-4, Mrk 501

- *General Programme*
  Der Hauptteil[3] der wissenschaftlichen Beobachtungszeit von INTEGRAL ist für die wissenschaftliche Gemeinschaft vorgesehen. Nach Sammlung aller Beobachtungswünsche wurde ein Beobachtungsplan erstellt, der versucht möglichst allen Wünschen der Astronomen gerecht zu werden.

2.2 Experimente an Bord

Neben dem IBIS Experiment, auf das in 2.3 näher eingegangen wird, wird der INTEGRAL-Satellit durch weitere drei Experimente an Bord vervollständigt.

2.2.1 Kodierte Aperturmaske

Bei allen Instrumenten (bis auf die optische Kamera OMC) an Bord des INTEGRAL-Satelliten wird eine kodierte Aperturmaske verwendet. Im Röntgen- und erst recht im Gamma-Bereich gibt es keine Matrialien, welche wie vergleichbare optische Linsen wirken. Streifender Einfall (wie auf ROSAT und XMM[4]) ist auch nur bis zu einer gewissen Wellenlänge möglich und scheidet für den Gamma-Bereich ganz aus. Um wieder eine abbildende „Optik“ zu bekommen, bedient man sich der kodierten Aperturmaske. Ihre Arbeitsweise gleicht der einer (vielfachen) Lochmaske. Hinter diesem Loch wird das Bild des Objekts auf dem Kopf stehend abgebildet. Durch mathematische Methoden ist es möglich, die vielen Bilder der einzelnen Lochkameras wieder auseinander zu rechnen und zu einem Bild zusammen zu setzen.

[3] 65% – 75%, steigend mit der Missionsdauer
[4] Bei XMM wurde mit großem Aufwand Wolterteleskope (Totalreflexion) verwendet, um Energien von bis zu 12 keV abbilden zu können

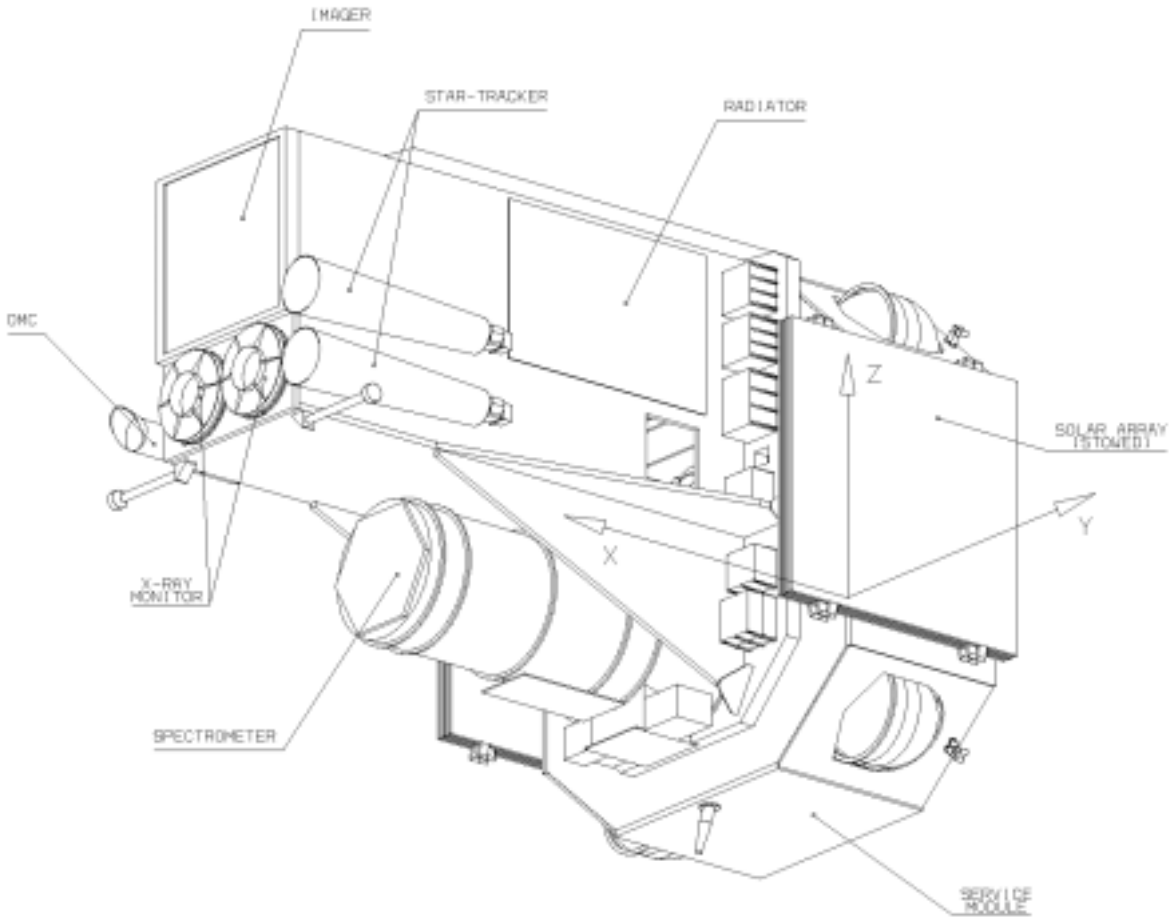


Abbildung 2.2: Der INTEGRAL-Satellit, Schemazeichnung. Rechts von dem Radiator sind eine Reihe von Boxen zu erkennen. Dies sind die Gehäuse für die verschiedenen Elektroniken des Satelliten. Unsere Elektronik (DPE mit der HEPI siehe 2.4 und 2.5) ist eine von ihnen. Das Service-Modul ist das gleiche wie bei XMM, hier konnte Entwicklungszeit gespart werden.

Die wichtgsten Parameter der Instrumente im Überblick:

	SPI	IBIS	JEM-X	OMC
Energiebereich	20 keV – 8 MeV	15 keV – 10 MeV	3 keV – 100 keV	500 – 800 nm
Energieauflösung	0,2 % (1 MeV)	6 % (1 MeV) 9 % (100 keV)	5 % ($>$ 35 keV)	—
Detektorfläche	500 cm^2	2600 cm^2 (CdTe) 3100 cm^2 (CsI)	2 * 500 cm^2	$1024 * 1024$ Pixel
Winkelauflösung	2°	12’	3’	17,6” / Pixel
Gesichtsfeld	16°	$9° * 9°$	$4,8°$	$15° * 15°$
Zeitauflösung	0,1 ms	61 μs	10 μs	s

Die Genauigkeit, mit der IBIS Quellenpositionen bestimmen kann, reicht bis zu etwa 1’, wenn die Stärke der Quelle ausreichend groß ist ($10\,\sigma$).

2.2.2 Spektrometer SPI

Das ***Sp**ectrometer for **I**NTEGRAL* bildet zusammen mit IBIS die beiden Hauptinstrumente zur Beobachtung im Gammabereich. Die hexagonale Detektorfläche besteht aus 19 (hexagonalen) Germanium-Kristallen. Mit Hilfe eines Kryostaten werden die Kristalle auf etwa 85 °K abgekühlt. Zur Reduktion des Hintergrundes wird wie bei IBIS ein aktiver Veto-Schild (siehe 2.3.4) verwendet. Dieser umfaßt fast das gesamte Instrument. 171 cm über den Kristallen befindet sich die kodierte Aperturmaske (siehe 2.2.1).

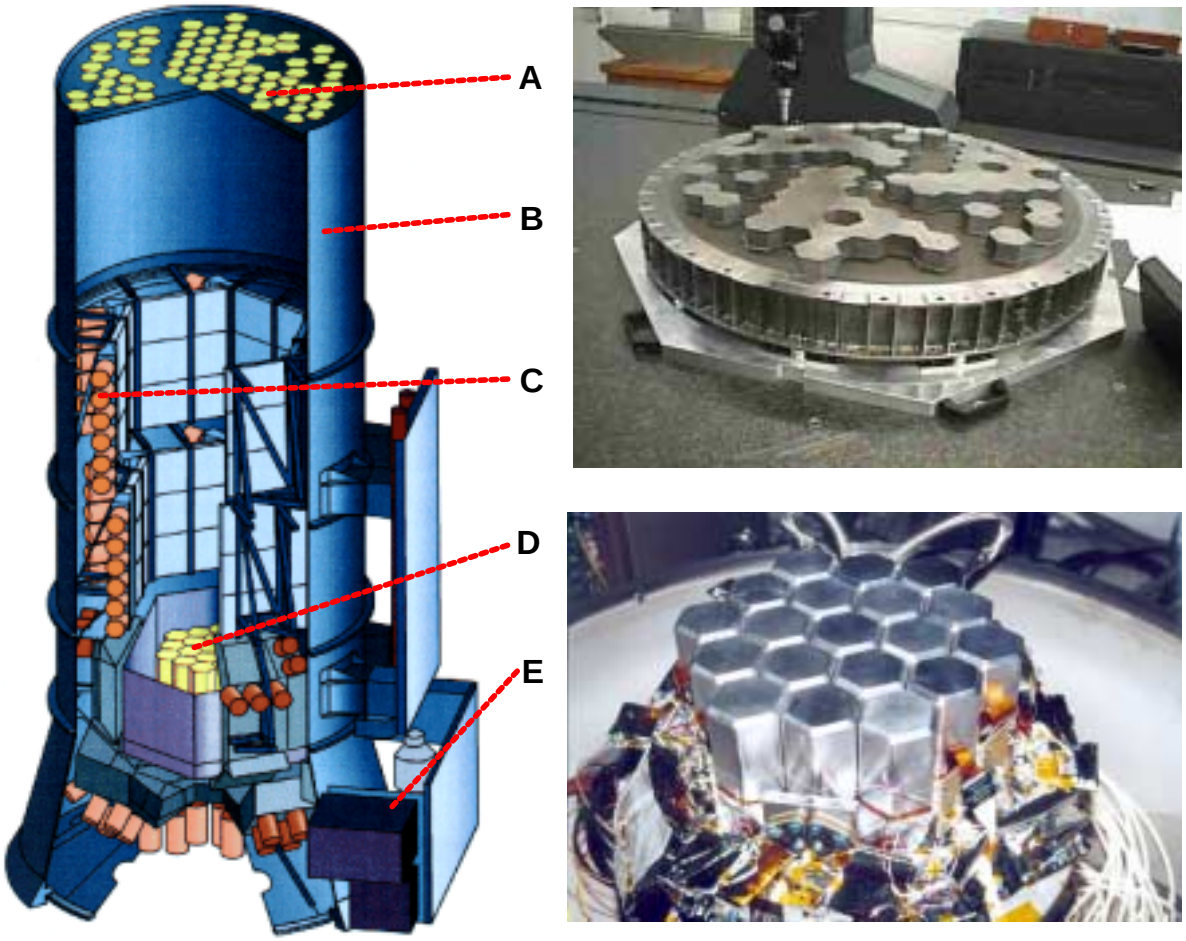


Abbildung 2.3: Das *Spectrometer for INTEGRAL* SPI. **Links:** Schema von SPI. **A:** Kodierte Aperturmaske, **B:** Gehäuse mit passiver Abschirmung, **C:** aktiver Veto-Schild, **D:** Detektor-Kristalle, **E:** Kryostat. **Rechts:** Oben: Die kodierte Aperturmaske während der Entwicklung von SPI. Deutlich ist die Dicke der einzelnen Segmente zu sehen. Unten: Nahaufnahme der 19 Germanium-Kristalle.

2.2.3 Röntgenmonitor JEM-X

Der ***J**oint **E**uropean **X**-Ray **M**onitor* besteht aus zwei identischen Microstrip-Detektoren. Das Innere der Detektoren ist mit Xenon gefüllt (Hochdruck). Als Kollimator zur Sichtfeldbegrenzung dient Molybdän. Die kodierte Aperturmaske befindet sich 3,2 m über den Detektoren.

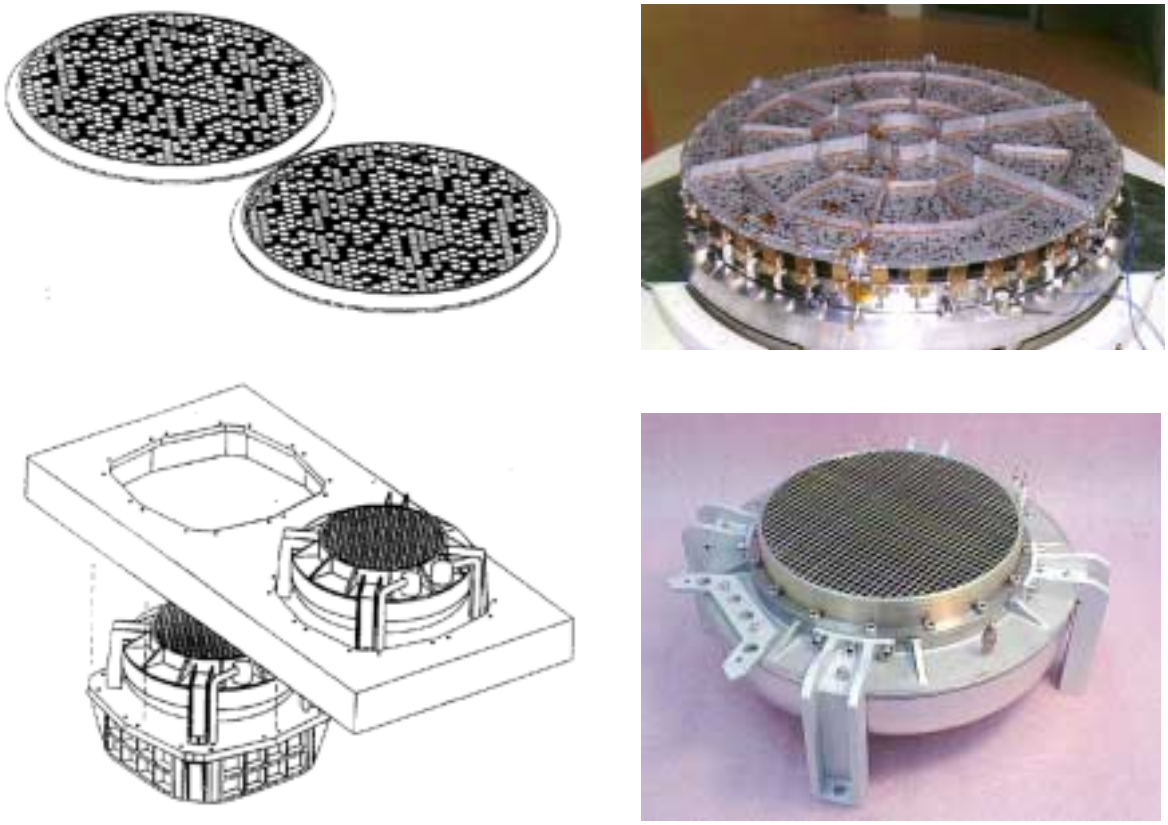


Abbildung 2.4: *Der Joint European X-Ray Monitor* JEM-X. **Links:** Schemazeichnung, **Rechts:** Oben: Die kodierte Aperturmaske von JEM-X fällt deutlich feiner aus als die der anderen Instrumente. Unten: Nahaufnahme eines Microstrip-Detektors.

2.2.4 Optischer Monitor OMC

Die ***O**ptical **M**onitoring **C**amera* ist eine CCD-Kamera für die Beobachtung im sichtbaren Bereich (V-Band, 550 nm). Die Brennweite beträgt 50 mm, die Empfindlichkeit etwa 19,5 mag. Zur Verringerung des Rauschens wird die Kamera auf etwa 193 K abgekühlt. Der CCD-Chip enthält $1024 * 2048$ Pixel, von denen nur eine Hälfte belichtet wird. Die andere Hälfte dient als Zwischenspeicher zum Auslesen.

Abbildung 2.5: Die *Optical Monitoring Camera* OMC; **A:** Blendschutz am Objektiv, **B:** Radiator zu Kühlung der **C:** CCD-Kamera.

2.3 Imager IBIS

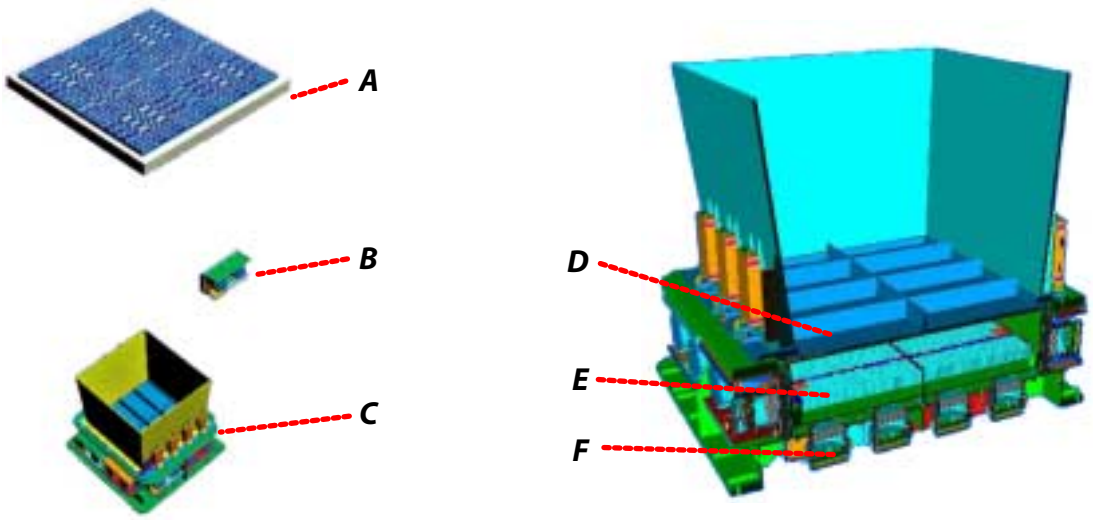


Abbildung 2.6: Der *Imager on Board the INTEGRAL Satellite* IBIS. **Links:** Nicht zu sehen ist die passive Abschirmung die IBIS von den Detektoren bis zur Aperturmaske umschließt. **A:** kodierte Aperturmaske, **B:** Kalibrierungs-Quelle, **C:** Detektorgehäuse (*Hopper*). **Rechts:** Detailansicht der Detektoren. **D:** Ebene des CdTe-Halbleiterdetektors, **E:** Ebene des CsI-Szintillationsdetektors, **F:** Photomultiplier des aktiven Hintergrundschildes Veto.

Der ***I**mager on **B**oard the **I**ntegral **S**atellite* besteht aus zwei unterschiedlichen Detektor-Ebenen zur Vergrößerung des nutzbaren Energiebereich des Instrumentes.

2.3.1 CsI-Detektor PICsIT

Das ***P**ixelated **I**maging **CsI** **T**elescope* besteht aus 4096 (64∗64) thalliumdotierten Cäsiumiodid-Szintillatoren. An jeden dieser Szintillatoren ist eine Photodiode optisch angekoppelt. Je $4 * 4$ von diesen Pixeln sind über ein ASIC[5] zu einer Detektoreinheit zusammen gefaßt. 32 dieser

[5] Application Specific Integrated Circuit

ASICs werden zu einem PICsIT-Modul zusammengefaßt, so daß der ganze Detektor aus 8 dieser Module besteht.

Die Dicke der Pixel ist mit 3 cm auf die 511 keV Elektron-Positron-Linie optimiert. Der nutzbare Energiebereich liegt zwischen 150 keV und 10 MeV.

Es werden zwischen 2000 und 25000 Ereignisse pro Sekunde erwartet.

2.3.2 CdTe-Detektor ISGRI

Der ***I****NTEGRAL* ***S****oft* ***G****amma-****R****ay* ***I****mager* besteht aus 16384 (128∗128) einzelnen Cadmium-Tellur Halbleiterkristallen. Je 4 dieser Pixel sind mit einem ASIC verbunden. 4 dieser ASICs sind zu einer Detektoreinheit zusammengefaßt und 128 dieser Detektoreinheiten bilden ein Detektor-Modul und 8 dieser Module schließlich den ganzen Detektor. Der nutzbare Energiebereich liegt zwischen 15 keV und 250 keV.

Es werden etwa 1220 Ereignisse pro Sekunde erwartet.

2.3.3 Kodierte Aperturmaske

Abbildung 2.7: Die kodierte Aperturmaske von IBIS. Deutlich ist entgegen der Schemazeichnungen die makroskopische Dicke der einzelnen Elemente zu erkennen. Nur so kann bei diesen Energien eine ausreichende Abschirmung erfolgen.

Für IBIS wird eine Maske aus Wolfram verwendet mit $95 * 95$ Elementen. Die Wolfram-Elemente sind jeweils $11,2 * 11,2$ mm groß und 16 mm dick. Damit wird eine Opazität von 70% bei 1,5 MeV erziehlt.

2.3.4 Abschirmung

Um das Gesichtsfeld auf die Aperturmaske einzuschränken, werden die Seitenflächen abgeschirmt.

Passive Abschirmung

Die Seitenflächen von IBIS sind mit Wolfram und Blei abgeschirmt. Bis etwa 200 keV kann so wirksam verhindert werden, daß Photonen an der Aperturmaske vorbei den Detektor erreichen.

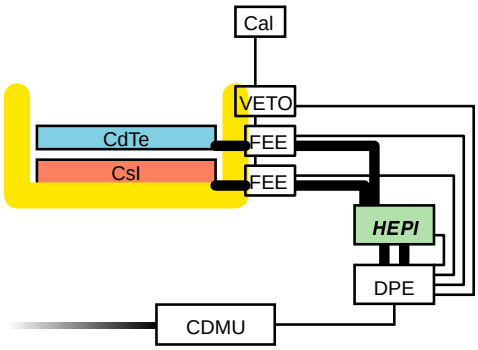
Aktive Abschirmung, Veto-Schild

Die aktive Abschirmung umschließt die Bodenfläche und die Seitenflächen direkt neben den Detektoren. Diese Abschirmung ist in der Wirkungsweise einem Detektor vergleichbar: Er besteht aus BGO-Blöcken (Wismut-Germanium-Oxid) und je zwei Photomultipliern. Registrieren die Photomultiplier ein Ereignis, so geht ein Veto-Signal an die beiden Detektoren von IBIS, ein evtl. gleichzeitiges Ereignis in diesen Detektoren *nicht* zu werten. Es entstammt mit großer Wahrscheinlichkeit einem Photon, welches an der Aperturmaske vorbei durch den aktiven Veto-Schild den IBIS-Detektor erreicht hat.

2.3.5 Kalibrierung

Zur *on board* Kalibrierung wird eine ^{22}Na-Quelle verwendet. Diese ist – bis auf eine konische Apertur in Richtung der Detektoren – vollständig mit einer BGO-Abschirmung umgeben. An dieser Abschirmung sind Photomultiplier angebracht. Ein durch β-Zerfall frei werdendes Positron erzeugt bei seiner Vernichtung mit einem Elektron zwei 511 keV Photonen, die in entgegengesetzter Richtung abgestrahlt werden. Mindestens eines dieser Photonen wird in der BGO-Abschirmung detektiert, das andere Photon kann durch die Öffnung in der BGO-Abschirmung zu den IBIS-Detektoren gelangen. Die Detektoren können nun ein evtl. gleichzeitig gemessenes Ereignis als Kalibrierungs-Ereignis kenntlich machen.

2.3.6 Datenverarbeitung



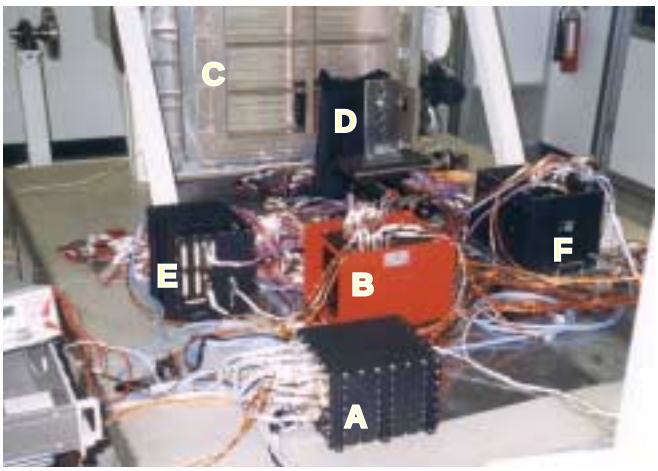


Abbildung 2.8: Schema der Datenerzeugung von IBIS. **Links:** Die Detektoren liefern ihre Ereignisse über die FEE zur HEPI. Nach der Verarbeitung dort kommen sie zur DPE, werden zum Senden vorbereitet und schließlich der CDMU (Common Data Management Unit) übergeben. Hier erfolgt die Übertragung zur Bodenstation. **Rechts:** Testaufbau in einem Reinraum in Mailand bei der Firma LABEN. **A:** Die DPE vom IBIS Experiment mit der HEPI, **B:** Die Elektronik der Veto-Einheit, **C:** Detektorrahmen, **D:** ein bestücktes Detektormodul, **E, F:** FEE von ISGRI und PICsIT.

Einfallende Gamma-Photonen wechselwirken mit den Detektoren. Diese liefern einen Spannungspuls an die Auswerteelektronik der Detektoren (FEE[6]). Dort wird der analoge Span-

[6] Front End Electronic

nungspuls in digitale Daten umgewandelt. Aus diesen Daten wird ein sog. Ereignis zusammengestellt. Dieses Datenpaket (64 Bit) enthält alle Daten, die von den Detektoren geliefert wurden:

- die gemessene Spannung als Amplitudenwert
  CdTe: 11 Bit, Auflösung ca. 0,3 keV
  CsI: 10 Bit, Auflösung ca. 5 keV
- die genaue Zeit, 24 Bit, Auflösung 238,4 ns in 2 sec
- die Pixelkoordinaten
  CsI: 12 Bit, je 6 Bit für die Y- und Z-Richtung,[7]
  CdTe: 14 Bit, je 7 Bit für die Y- und Z-Richtung.
- Identifizierung eines Kalibrierungs-Ereignisses
- vom CdTe-Detektor zusätzlich noch die Anstiegszeit, 8 Bit[8]
- vom CsI-Detektor zusätzlich Identifizierung von Mehrfach-Ereignissen, 2 Bit
  Einzel-, Doppel- und Dreifach-Ereignis.

Kommen zwei oder mehr Ereignisse aus dem CsI-Detektor gleichzeitig aus dem Detektor, so werden sie von der Detektor-Elektronik als sog. Mehrfach-Ereignis gekennzeichnet, d. h. sie stammen vom selben Photon ab, das mit mehreren Pixeln wechselwirken konnte. Liegt eine Gleichzeitigkeit mit den Kalibrierungs-Szintillator vor, so wird das Ereignis als Kalibrierungs-Ereignis gekennzeichnet.

So entstehen die folgenden Datentypen:

- CsI: (Einfach)-Ereignisse
- CsI: Mehrfach-Ereignisse (Doppel-, Dreifach-Ereignisse)
- CsI: Kalibrierungs-Ereignisse
- CdTe: Ereignisse
- CdTe: Kalibrierungs-Ereignisse

Es findet eine erste Verarbeitung dieser Daten statt. Ein Teil der Ereignisse wird hier verworfen und gelangt nicht zur nachfolgenden Elektronik der HEPI (siehe 2.4):

- gleichzeitige Ereignisse mit dem Veto-Signal
- Seltene vier- und mehrfache Ereignisse des CsI-Detektors PICsIT
- Mehrfach-Ereignisse des CdTe-Detektors ISGRI
- Ereignisse, welche innerhalb einer Detektor-Ebene gleichzeitig als Kalibrierungs- und Mehrfach-Ereignisse gekennzeichnet wurden

[7] Das Koordinatensystem des Satelliten spannt mit Y und Z die Detektor-Ebenen auf und zeigt mit X in Beobachtungsrichtung

[8] wird für die Verarbeitung in der HEPI nicht verwendet und unverändert an die DPE weitergereicht

- Mehrfach-Ereignisse, die über die Grenzen eines CsI-Detektor-Moduls gingen

Die Auswerteelektronik der Detektoren sortiert die Ereignisse nach ihrer Entstehungszeit und stellt sie in einem FIFO[9]-Speicher zum Auslesen bereit. Für jedes Ereignis wird ein 64 Bit Datenpaket erzeugt. Dieses wird über zwei schnelle serielle Leitungen (von jedem Detektor eine) auf Anforderung zur HEPI (siehe 2.4) geschickt.

Nach der Bearbeitung der Daten durch die HEPI gelangen die einzelnen Ereignisse oder ganze Histogramme bzw. Spektren über zwei schnelle serielle Leitungen zum Hauptprozessor DPE (siehe 2.5).

Die DPE steuert und verwaltet die Datenverarbeitung von IBIS. Die Daten der HEPI werden zur Übermittlung komprimiert. Es findet eine weitere Datenverarbeitung statt. Diese ist in der Sache komplizierter als die Datenverarbeitung in der HEPI, stellt aber nicht so hohe Anforderungen an die Ausführungsgeschwindigkeit.

2.4 Präprozessor HEPI

Der ***H**ardware **E**vent **P**re-Processor on Board the **I**NTEGRAL Satellite* bildet das Kernstück der Hardware Elektronik die am AIT entwickelt wurde und ist Hauptteil dieser Arbeit.

2.4.1 Datenraten

Betrachtet man die wahrscheinlichen Ereignisse pro Sekunde der Detektoren und die zur Verfügung stehende Telemetrierate zur Erde, dann wird die Hauptaufgabe der HEPI ersichtlich:

Datenreduktion

Aus den Ereignissen pro Sekunde ergeben sich folgende Datenraten:

- ISGRI Detektor: 1200 Ereignisse pro Sekunde $\Longrightarrow$ 78,6 kBaud[10]
  1200 CdTe-Ereignisse pro Sekunde aus der HEPI: 96 kBaud[11]
  Das Interface zwischen dem Detektor und der HEPI läßt maximal 65536 Ereignisse pro Sekunde zu $\Longrightarrow$ 4,194 MBaud.

- PICsIT Detektor: 2000 – 25000 Ereignisse pro Sekunde $\Longrightarrow$ 128 kBaud – 1,6 MBaud
  Aus der HEPI: 160 kBaud – 2 MBaud
  Das Interface zwischen dem Detektor und der HEPI läßt maximal 65536 Ereignisse pro Sekunde zu $\Longrightarrow$ 4,194 MBaud.

- Beide Detektoren zusammen: 238,6 kBaud – 1,6786 MBaud
  Aus der HEPI: 256 kBaud – 2,096 MBaud

Auf der anderen Seite steht für die Übertragung (*downlink*) zur Verfügung:

- Gesamte Telemetrierate aller Instrumente: 85,8 kBaud

- Davon für IBIS vorgesehene Telemetrierate: 59 kBaud

[9] First In First Out
[10] Jedes Ereignis vom Detektor ist 64 Bit groß : $1200 * 64 = 76800$
[11] Jedes Ereignis aus der HEPI ist 80 Bit groß : $1200 * 80 = 96000$

Die Forderung zur Datenreduktion – besonders für die Daten von PICsIT – ist offensichtlich. Die sehr beschränkte Leistung der nachfolgenden DPE[12] (siehe 2.5) macht es unmöglich alle diese Daten zu komprimieren oder zwischen zu speichern, so daß alle Daten zu Erde gesendet werden können. Da dies in der Planungsphase schon abzusehen war, wurde der Präprozessor konzipiert, um aus dem Datenstrom der Detektoren den im Augenblick wichtigen Teil herauszufiltern.

2.4.2 Aufgaben der HEPI

Die HEPI bietet verschiedene Methoden der Datenreduktion für den CsI-Zweig. Bei allen Methoden wurde Wert darauf gelegt, möglichst flexibel zu sein. So sind alle Parameter in der HEPI von der DPE programmierbar.

Die Methoden zur Datenreduktion sind:

- Histogramme
  Mit einem Histogramm erhält man eine Möglichkeit zur drastischen Datenreduktion. Über eine Zeit von typischerweise 1000 Sekunden werden alle eintreffenden CsI-Ereignisse ihrer Position und Energie entsprechend in einem Speicher einsortiert. Die Zeit-Information der Ereignisse geht hierbei verloren. Dafür können selbst extrem hohe Datenraten aufgezeichnet werden.
  Im Prinzip ähnelt dieses Vorgehen einer Farbphotographie: Während der Belichtungszeit werden alle Photonen auf dem Film gesammelt. Die Energie der Photonen entspricht einer Farbe auf dem Film, die Anzahl der Photonen an einer Stelle entspricht der Helligkeit auf dem Film. Nach dem Entwickeln wird das ganze Photo betrachtet.
  Die HEPI bietet verschiedene Arten von Histogrammen die zum Teil gleichzeitig erzeugt werden können:

  - Einfach-Ereignisse
    Alle Einfach-Ereignisse des CsI-Detektors werden in einem Speicherbereich zusammengezählt.
  - Mehrfach-Ereignisse
    Alle Einfach-Ereignisse des CsI-Detektors werden in einem Speicherbereich zusammengezählt. Dieser Speicher ist nicht identisch mit dem Speicher für die Einfach-Ereignisse. Somit kann *gleichzeitig* ein Histogramm der Einfach-Ereignisse und ein Histogramm der Mehrfach-Ereignisse erzeugt werden (siehe Abbildung 2.10)
  - Polarimetrie-Ereignisse
    Alle Doppel-Ereignisse, für die ein Streuwinkel errechnet werden konnte, werden in diesem Speicher abgelegt. Da hier neben der Position und der Energie der Ereignisse auch noch der Streuwinkel gespeichert wird, ist dieser Speicherbereich so groß wie der Speicherbereich für die Einfach-Ereignisse und die Mehrfach-Ereignisse zusammen. Da er auch noch an der selben Adresse im Hauptspeicher der HEPI liegt, kann man nur *entweder* ein Einfach- und Mehrfach-Ereignis Histogramm *oder* ein Polarimetrie-Histogramm erzeugen.
  - Kalibrierungs-Ereignisse
    Zur kontinuierlichen Überwachung der CsI-Szintillatoren mit den angeschlossenen

[12] Data Processing Electronic

Photodioden und ihren benötigten Energie-Korrekturwerten in der HEPI werden in einem von den übrigen Histogrammen unabhängigen Speicher, Kalibrierungs-Ereignisse zusammengezählt. Ein Kalibrierungs-Histogramm kann gleichzeitig mit einem Einfach- und Mehrfach-Histogramm oder mit einem Polarimetrie-Histogramm erzeugt werden.

Der Histogrammspeicher wird von der DPE ausgelesen. Um die Übertragung der Ereignisse nicht zu stören, ist hierfür ein eigenes Interface von der HEPI zur DPE vorgesehen.

- *Spectral Timing*[13] Histogramme
  Ging es bei den obigen Histogrammen darum ein Bild zu erzeugen, wird bei den Spectral Timing Histogrammen Wert auf die genaue zeitliche Auflösung gelegt. Dafür werden die Informationen über die Position eines Ereignisses verworfen. Über eine einstellbare Zeit von 1 ms bis zu 0,5 s werden die Ereignisse in 8 frei einstellbare Energie-Bereiche unterteilt. Nach Ablauf der Integrations-Zeit werden die Daten als Spectal-Timing-Ereignis zur DPE übertragen.

- Energie-Bandpaß
  Werden die Ereignisse einzeln zur DPE geschickt, kann man mit dem Energie-Bandpaß sich auf den interessanten Energiebereich konzentrieren und Ereignisse mit zu wenig Energie (Hintergrund) und Ereignisse mit zu viel Energie aussortieren (*Energy Selection*).

Neben der Aufgabe der Datenreduktion müssen weitere Rechenoperationen mit den CsI-Ereignissen auf der HEPI gemacht werden. Wegen der möglichen Datenraten des CsI-Detektors können diese Operationen nicht in der DPE gemacht werden:

- Energie-Korrektur für jedes CsI-Pixel
- Erkennen und Zusammenführen von zusammengehörenden Mehrfachereignissen
- Berechnung des Streuwinkels von Doppel-Ereignissen
- Erkennen von Compton-Ereignissen[14]
- Datenflußsteuerung mit den Detektoren
- Kommunikation und Datenflußsteuerung mit der DPE

Zur Realisierung dieser Aufgaben in der geforderten Geschwindigkeit wird bei der[15] HEPI auf das Pipeline-Konzept zurückgegriffen. Hier wird das Design in einzelne Aufgaben (Module) unterteilt,[16] die – durch Register getrennt – nacheinander die Daten bearbeiten (siehe 4.1).

[13] zeitlich hochaufgelöstes Spektrogramm
[14] Compton-Effekt: Photon wird im oberen und unteren Detektor registriert (Compton-Streuung)
[15] es heißt zwar korrekt „der Präprozessor“, im allgemeinen Sprachgebrauch hat sich aber „die HEPI“ durchgesetzt
[16] *Top-Down Design*

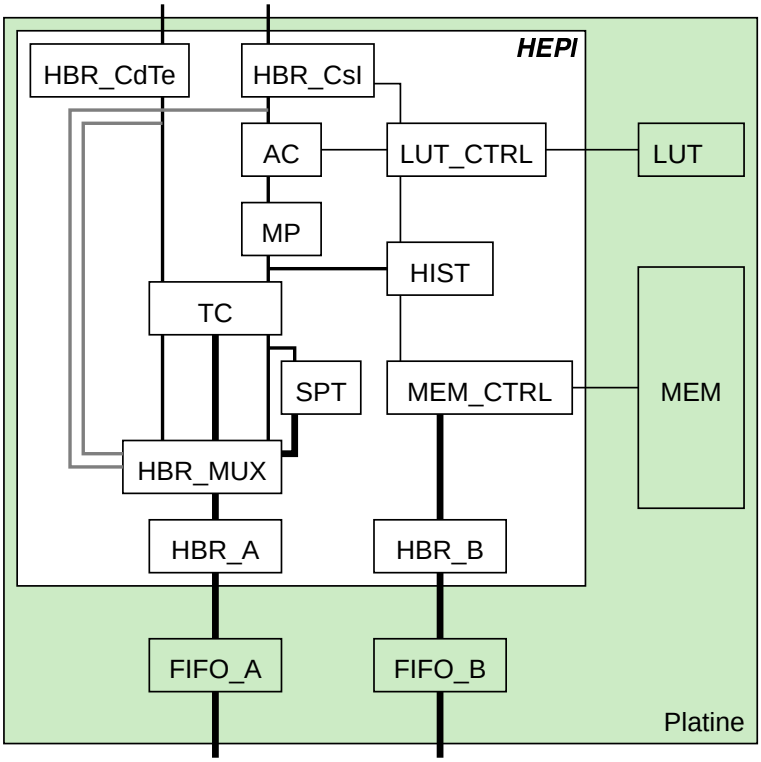


Abbildung 2.9: Datenfluß in der HEPI. Weiß sind die Module/Komponenten welche in dem ASIC der HEPI, Grün sind die Komponenten welche auf den Platinen der HEPI realisiert wurden.

2.4.3 High Bitrate Interface CSI_HBR, CDTE_HBR

Dieses Modul übernimmt die Kommunikation mit den Detektoren und die Datenflußsteuerung. Das verwendete RS-422 Interface arbeitet mit Differenz-Signalen[17].
Die Signale im Einzelnen[18]:

- DAT, Detektor ⟶ HEPI
  Serielle Datenleitung. Hier werden nacheinander die 64 Bits für ein Ereignis zur HEPI gesendet.

- NE, Detektor ⟶ HEPI
  FIFO Not Empty zeigt an, daß mindestens ein Ereignis im FIFO-Speicher des Detektors zum Auslesen bereit ist.

- CLK, HEPI ⟶ Detektor
  Taktsignal der seriellen Übertragung. Dies ist der durchgeschleifte Systemtakt der HEPI (4 MHz[19])

[17] *differential signals*: Das Signal wird in einer positiven und einer negativen (invertierten) Form übertragen. Da Störimpulse auf beide Signale den gleichen Einfluß haben, heben sich die Störimpulse bei der Differenzbildung auf der Empfängerseite auf
[18] Die genaue Bezeichnung der Signale ergibt sich aus unserer Konvention der Namensgebung (siehe 4.2.1)
[19] genauer: 2^{22} Hz= 4 194 304 Hz

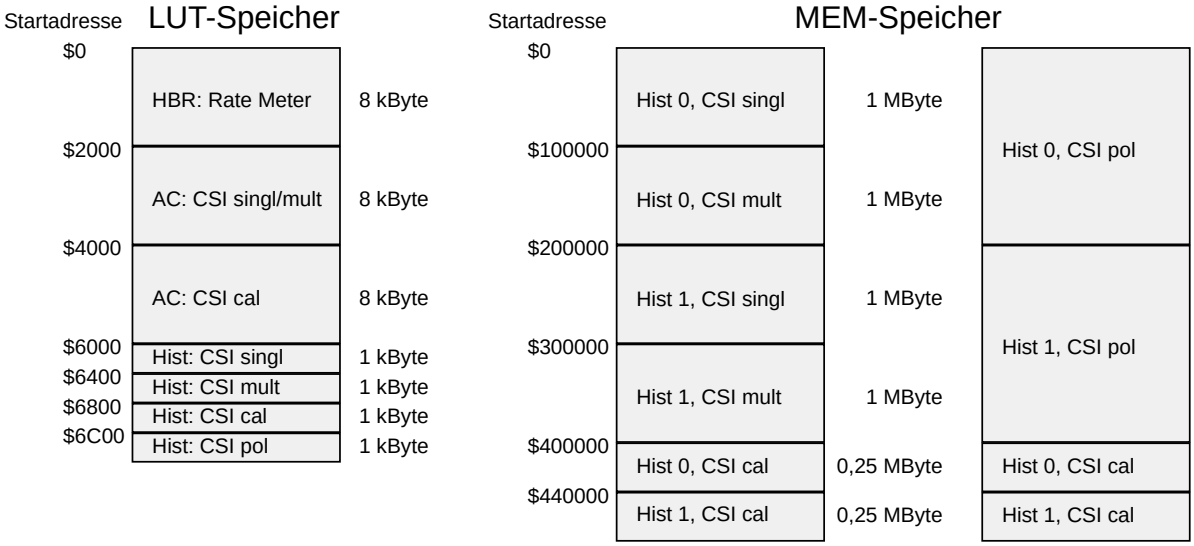


Abbildung 2.10: Der Speicher der HEPI: LUT und MEM. Die Konfiguration des MEM-Speichers ist abhängig von der gewählten Histogramm-Art.

- RNE, HEPI ⟶ Detektor
  Read Next Event ist das Startsignal für die serielle Übertragung, nachdem die HEPI neue Ereignisse im Detektor über die NE Leitung erkannt hat.
- CLR, HEPI ⟶ Detektor
  Clear FIFO löscht das FIFO im Detektor und damit alle darin noch befindlichen Ereignisse.
- SYNC, HEPI ⟶ Detektor
  Synchronisationsimpuls zum genauen zeitlichen Abgleich der Detektoren.

Es wird erkannt, ob neue Daten bei den Detektoren bereitstehen und das Auslesen dieser Daten wird gestartet. Aus dem seriellen Datenstrom für jedes 64 Bit Datenpaket wird das intern verwendete 80 Bit Datenformat erzeugt, welches von jedem folgendem Modul verwendet wird und auch später zur DPE geschickt wird.

Fehler bei der Übertragung werden erkannt und bis zum Löschen via Telekommando gehalten. Ist das Modul inaktiv, werden, auch wenn der Detektor verfügbare Daten signalisiert, keine weiteren Daten eingelesen. Somit kann dieser Zweig der Pipeline in der HEPI „leer laufen“, d. h. Ereignisse, die sich schon in der Pipeline der HEPI befinden, werden weiter bearbeitet und an die folgenden Module weitergereicht. Neue Ereignisse kommen aber nicht nach (Pipeline-Konzept). Ereignisse, welche in dieser Zeit statt finden, werden im FIFO-Speicher des Detektors gehalten. Ist dieser FIFO-Speicher voll, werden weitere Ereignisse im Detektor verworfen und gehen somit der wissenschaftlichen Verarbeitung verloren.

Dieses Modul übernimmt auch die Rolle des *Rate Meters*, d. h. es wird gezählt wieviele Ereignisse von den Detektoren zur HEPI übertragen wurden. Für diese Zahl steht ein 32 Bit Wert in diesem Modul (je eins für CsI und CdTe) zur Verfügung. Ein Überlauf wird erkannt und gehalten bis er via Telekommando gelöscht wird.

Für das CIS_HBR ist zusätzlich im Zusammenspiel mit dem LUT_CTRL das sog. CsI-*Pixel Rate Meter* implementiert. Dieser zählt wie oft jedes einzelne CsI-Pixel getroffen wurde. Im

LUT-Speicher ist für jedes Pixel ein 16 Bit Wert vorhanden, der vom LUT_CTRL selbständig hochgezählt wird. Dieser Speicher kann per Telekommando ausgelesen und gelöscht werden. Ein Fehler kann auftreten, wenn einer der 16 Bit Zähler überläuft (Overflow). In diesem Fall wird der entsprechende Zähler auf seinen Maximalwert gesetzt. Dieser Fehler wird erkannt und gehalten bis er via Telekommando gelöscht wird.

Da die Auswerte-Elektronik der beiden Detektoren unterschiedlich schnell arbeiten, erkennen die Detektoren ein gleichzeitiges Ereignis zu jeweils unterschiedlichen Zeiten. Um diesen Zeitversatz aufzuheben, versorgt dieses Modul die beiden Detektoren mit einen eigenem Synchronisations-Puls. Dieser Puls ist derart verzögert, daß beide Detektoren ein gleichzeitiges Ereignis auch wieder gleichzeitig erkennen. Dazu wird ein 1-Sekunden Puls von der DPE entsprechend verzögert und als 2-Sekunden Puls zu den Detektoren geschickt. Dieser Verzögerungswert ist via Telekommando für jeden Detektor getrennt einstellbar. Er kann in Einheiten des Systemtaktes (238,4 ns) um bis zu 15 μs verzögert werden. Die benötigten Werte werden im Rahmen der Kalibrierung von IBIS ermittelt.

2.4.4 Amplitude Correction CSI_AC

Dieses Modul dient zur Korrektur der Amplitudenwerte der einzelnen CsI-Pixel. Die CsI-Szintillatoren mit optisch angekoppelten Photodioden werden zum Teil von Hand gefertigt. Durch unterschiedliche Empfindlichkeiten der Photodioden und der analogen Vorverstärker kommt es dabei zu Streuungen im Wirkungsgrad der einzelnen Pixel. Die Pixel haben verschiedene Empfindlichkeiten, deshalb erzeugen die Pixel bei gleicher Anregung unterschiedliche Amplituden.

Im CSI_AC Modul wird nun mit einer linearen Korrektur versucht, die unterschiedlichen Empfindlichkeiten der einzelnen Pixel wieder auszugleichen, so daß sich alle Pixel gleich verhalten. Bei dieser Gain- und Offset-Korrektur wird der Amplitudenwert vom Pixel mit einem Korrekturwert multipliziert (Gain) und danach mit einem anderen Korrekturwert addiert bzw. subtrahiert (Offset, im 1er-Komplement). Die Werte für Gain und Offset müssen in einem aufwendigen Kalibrierungsprozeß für jedes einzelne CsI-Pixel gefunden werden. Dies geschieht während der Eichmessung.

Mit den entsprechenden Korrekturwerten aus einem lokalen Speicher auf der Platine der HEPIPlatine (LUT[20]-Speicher, siehe 2.10) werden die Empfindlichkeiten der CsI-Pixel angepaßt. Damit lassen sich im folgendem Modul (siehe 2.4.5) die einzelnen Mehrfach-Ereignisse in Doppel- und Dreifach-Ereignisse zusammenfassen. Dort wird mit den korrigierten Einzelamplituden die Gesamtenergie des Photons berechnet.

Für Kalibrierungs-Ereignisse steht ein eigener Satz von Korrekturwerten bereit.

Fehler werden erkannt und gehalten, bis sie per Telekommando wieder zurückgesetzt werden. Ein Fehler kann auftreten, wenn bei der Multiplikation oder der nachfolgenden Addition der maximal darstellbare Wertebereich von 10 Bit überschritten wird (11. Bit, Überlauf). In diesem Fall erhält das Ereignis den maximal möglichen Amplitudenwert. Bei einer Subtraktion kann das Ergebnis negativ werden (Underflow). Das Ereignis erhält dann die Amplitude 0. Solche Fehler sollten nur vorkommen, wenn die Korrekturwerte für die Pixel nicht richtig angepaßt wurden oder sich das Verhalten der Pixel während der Mission ändert. Da es nur eine Leitung zur Signalisierung von Fehlern gibt, kann von außen nicht zwischen einem Overflow und einem Underflow unterschieden werden.

[20] **L**ook-**U**p **T**able, aus den Eingangsdaten wird in einem möglichst einfachen Prozeß eine Adresse im Speicher gebildet (hashing), wo die zugehörigen Ausgangsdaten zu finden sind

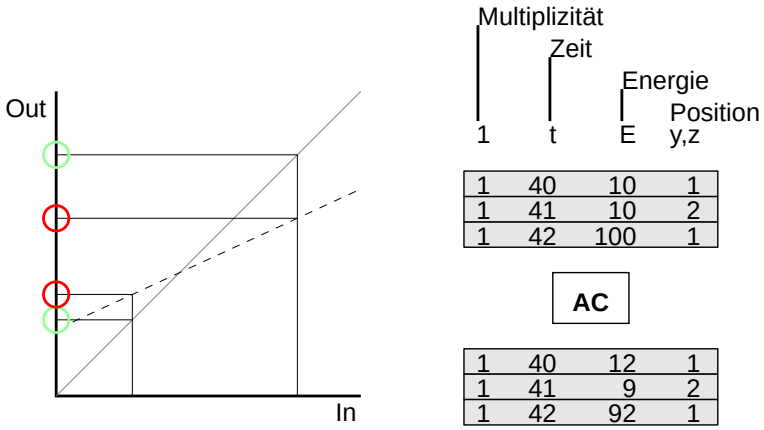


Abbildung 2.11: Annäherung der verschiedenen Empfindlichkeiten einzelner Pixel mit einer Gain & Offset Korrektur. **Links:** Schematische Darstellung der Korrektur für ein Pixel. Die Daten des Detektors (grün) werden gemäß der Korrektur (gestrichelte Linie) ausgegeben (rot). **Rechts:** Änderungen von CsI-Ereignissen durch das AC-Modul.

Durch einen Seiteneffekt bei der Erzeugung des ASICs ist dieses Fehler-Signal nicht immer aussagekräftig: Das Fehler-Signal kann auch von zwei aufeinander folgenden Ereignissen ausgelöst werden, die eigentlich *keinen* Fehler verursachen (siehe A.2.1).

Ist das Modul inaktiv, so passieren die Ereignisse dieses Modul ohne daß die Amplitude verändert wird.

Für die CdTe-Ereignisse wird eine aufwendigere Korrektur in der DPE gemacht.

2.4.5 Multiple Event Reconstruction & Polarimetry CSI_MP

Alle Ereignisse werden im CsI-Detektor als Einfach- oder Mehrfach-Ereignisse gekennzeichnet und an die HEPI weitergegeben. Mehrfach-Ereignisse sind gleichzeitig im Detektor aufgetreten und wurden von einem Photon erzeugt, das seine Energie in zwei oder drei Detektor-Pixel abgegeben hat und damit zwei oder drei Mehrfach-Ereignisse erzeugt hat.

Aufgabe dieses Moduls ist nun:

- Alle diese Mehrfach-Ereignisse wieder zu einem einzigen Ereignis zusammenzufassen (MER[21]), wobei
  - als Einfallspixel das genommen wird, welches wahrscheinlich als erstes getroffen wurde
  - als Energie die Summe der Einzelenergien berechnet wird.
- Handelt es sich bei dem Mehrfach-Ereignis um ein Doppel-Ereignis, so wird zusätzlich ermittelt, ob zwei Nachbarpixel getroffen wurden (POL[22]).
  In diesem Falle wird außer dem Einfallspixel auch das Streupixel über den Streuwinkel mit angegeben. Das Streupixel liegt bevorzugt in Richtung des $\vec{E}$-Vektors des Gammaquants (Compton-Streuung).

[21] Multiple Event Reconstruction
[22] Polarimetrie

Da aus der Reihenfolge der Mehrfach-Ereignisse beim Eintreffen in der HEPI *nicht*[23] auf die Reihenfolge ihres Auftretens im Detektor geschlossen werden kann, muß diese Reihenfolge aus den beteiligten Energien ermittelt werden. Hier helfen die Kernaussagen der Monte-Carlo-Simulation des CsI-Detektors [1].

- Liegt die Gesamtenergie *über* einem gewissen Schwellenwert, so wurde im ersten Pixel die *meiste* Energie deponiert.
- Liegt die Gesamtenergie *unter* einem bestimmten Schwellenwert, so wurde im ersten Pixel am *wenigsten* Energie deponiert.
- Liegt bei Dreifach-Ereignissen die Gesamtenergie *zwischen* zwei Schwellenwerten, so wurde im ersten Pixel eine *mittlere* Energie deponiert.

Diese Schwellenwerte sind frei wählbar und können von der DPE zur HEPI übertragen werden.

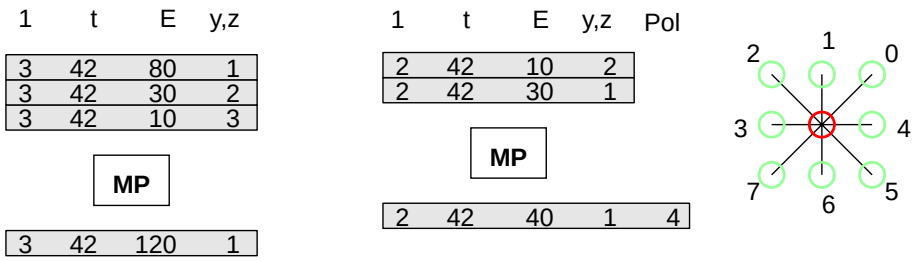


Abbildung 2.12: Links: Das MP-Modul erzeugt aus drei einzelnen Ereignissen (vom Detektor schon als zusammengehörend gekennzeichnet) ein Dreifach-Ereignis. **Rechts:** Bei der Erzeugung eines Doppel-Ereignisses wird zusätzlich versucht (wenn die Ereignisse nebeneinander liegen) den Streuwinkel (Pol) anzugeben.

Die Rekonstruktion kann allerdings nur erfolgen, wenn z. B. zwei Doppel-Ereignisse[24] direkt hintereinander kommen, d. h. dazwischen kein Einfach- oder Dreifach-Ereignis liegt. Die Auswerteelektronik im CsI-Detektor hat durch das zeitliche Sortieren der Ereignisse dafür Sorge zu tragen.

Durch das Aufsummieren der Einzelenergien im Falle von Doppel- und Dreifach-Ereignissen (Mehrfach-Ereignisse) kann der zur Darstellung der Energie in der HEPI verwendete Wertebereich von 10 Bit überschritten werden. Deshalb wird für die Doppel- und Dreifach-Ereignisse die Einheit der Energie in diesem Modul verdoppelt (von 5 keV auf 10 keV). Damit kann mit den zur Verfügung stehenden 10 Bit ein doppelt so großer Energiebereich abgedeckt werden. Es kann auch weiterhin mit den vorhandenen Datenformat gearbeitet werden.

Die Behandlung von Kalibrierungs-Ereignissen ist unabhängig von den Mehrfach-Ereignissen. Nur Einfach-Ereignisse können auch als Kalibrierungs-Ereignisse gekennzeichnet sein, denn Mehrfach-Kalibrierungs-Ereignisse werden bereits im Detektor verworfen.

Ist das Modul inaktiv, so passieren die Ereignisse dieses Modul ohne verändert zu werden, d. h. ein Doppel-Ereignis besteht weiterhin aus zwei Ereignissen.

[23] Die Auswerte-Elektronik hat die Ereignisse schon zeitlich sortiert, d. h. gleichzeitige Ereignisse wie die Mehrfach-Ereignisse kommen direkt nacheinander, aber die zeitliche Auflösung reicht nicht aus, das erste Ereignis von dem zweiten/dritten (gestreuten) Ereignis zu trennen

[24] also zwei Datenpakete mit einem gesetzten Typ-Bit „Doppel-Ereignis“. Dies wird von der Auswerteelektronik der Detektoren gesetzt, wenn zwei Ereignisse zeitgleich registriert werden. Es wird hier angenommen, daß ein Photon in zwei Pixeln wechselwirken konnte

2.4.6 Spectral Timing CSI_SPT

Das CSI_SPT Modul ist nach dem CSI_MP das zweite Modul, welches dazu dient die Datenrate des CsI-Detektors durch Zusammenfassen und Weglassen von Information zu verringern. Statt alle CsI-Daten als Einzel-Ereignisse zur Erde zu schicken, werden sie 8 Energie-Intervallen zugeordnet und während einer Integrationszeit unabhängig von ihrer Einfalls-Position einfach nur gezählt. Es wird ein zeitlich hochauflösendes Energie-Histogramm erstellt.

So wird, je nach Datenrate und Integrationszeit, aus vielen Ereignissen mit je 80 Bit, ein einzelnes SPT-Ereignis mit 160 Bit erzeugt.

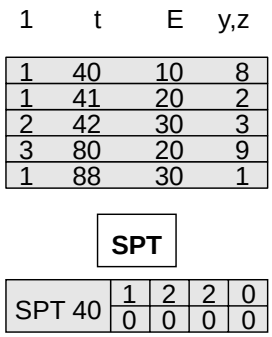


Abbildung 2.13: Schematische Darstellung der Erstellung des Energie-Histogramms. Aus 5 Ereignissen wird (mit z. B. $\Delta t = 50$) ein SPT-Ereignis erzeugt, welches die Anzahl der Ereignisse in den Energiebändern enthält.

Es gibt 9 Schwellenwerte (8 Bit mit 40 keV Auflösung) die 8 Intervalle definieren. Jedem Intervall ist ein 8 Bit-Zähler zugeordnet. Diese 9 Schwellenwerte sind ebenso frei einstellbar wie die Integrationszeit (0,9 – 500 ms) und können von der DPE zur HEPI übertragen werden.

Durch das vorangegangene Modul wird sichergestellt, daß Doppel- und Dreifach-Ereignisse nur einmal gezählt werden, denn sie kommen hier nur noch als ein (Mehrfach-) Ereignis an. Kalibrierungs-Ereignisse werden in diesem Modul nicht gezählt.

Das erste gültige Ereignis nach Einschalten des Moduls setzt mit seiner Zeit den Startwert t_0 für die Zeitintervalle Δt und damit beginnt die Integration. Bei allen nachfolgenden Ereignissen wird überprüft, ob ihre Zeit noch in das aktuelle Zeitintervall zwischen t_0 und $t_0 + \Delta t$ paßt. Mit dem ersten Ereignis, welches nicht mehr in dieses Intervall paßt, wird das aktuelle SPT-Ereignis abgeschlossen und an das nachfolgende Modul gereicht. Das Modul muß nun das richtige Zeitinterfall t_n bis $t_n + \Delta t$ für dieses neue Ereignis finden. Aus implementationstechnischen Gründen gelingt das nur bis zu $n = 31$, d. h. das neue Ereignis darf maximal 32 Zeitintervalle vom aktuellen SPT-Ereignis entfernt sein.
Ist das nicht der Fall wird das nächste SPT-Ereignis mit dem (falschen) Startwert von t_{32} begonnen. Auf diese Weise holt das CSI_SPT Modul die Zeit wieder auf, in dem es SPT-Ereignisse mit nur einem CsI-Ereignis schnell hintereinander und mit einem zeitlichen Abstand von 32∗Integrationszeit auswirft.

Ein Überlaufen eines 8 Bit-Zählers erzeugt einen Overflow-Fehler, der erkannt und gehalten wird, bis er per Telekommando gelöscht wird. Der entsprechende Zähler behält bis zur Erzeugung des nächsten SPT-Ereignisses (und damit Löschen aller Zähler) seinen Maximalwert von 255 bei.

Leere SPT-Ereignisse werden nicht erzeugt.

Ist das Modul inaktiv, so werden keine SPT-Ereignisse erzeugt.

2.4.7 Histogramm CSI_HIST

Das wichtigste Modul zur Datenreduktion ist CSI_HIST. Die einzelnen Ereignisse werden in einem Histogramm im Hauptspeicher der HEPI eingetragen. Energie und Pixel-Position bleiben erhalten, die Zeitinformation der einzelnen Ereignisse dagegen geht vollständig verloren[25]. Es werden 3 Arten von Histogrammen angeboten:

- Einfach- / Mehrfach-Ereignis-Histogramm mit 8 Bit Amplitude
- Polarimetrie-Histogramm mit 6 Bit Amplitude und 3 Bit Streuwinkel
- Kalibrierungs-Histogramm mit 6 Bit Amplitude

Aus diesen Werten und der Pixel-Position wird eine Adresse im Hauptspeicher gebildet (*Hashing*). An dieser Adresse im Speicher wird der Wert dort (8 Bit) um eins erhöht (Zähler).

Wegen der begrenzten Größe des Hauptspeichers muß allerdings bei der Amplitude eine Einschränkung gemacht werden. Es werden nicht alle 10 Bit der Amplitude zur Erzeugung des Histogramms herangezogen, sondern nur 8 Bit (Einfach- und Mehrfach-Ereignisse) bzw. 6 Bit (Polarimetrie und Kalibrierung).

Um die gute Energieauflösung der Detektoren nicht durch ein einfaches Beschneiden der 10 Bit auf 8 bzw. 6 Bit zu verringern, wird für diese Umwandlung das sog. *Binning* verwendet. Jedem einzelnen dieser $2^{10} = 1024$ möglichen Energiewerte wird ein 8 Bit Binning-Wert zugeordnet. Dabei müssen natürlich manche Energiewerte auf die selben Binning-Werte zeigen. Aber da man selber bestimmen kann, wie die Zuordnung Energiewert ↔ Binning-Wert aussieht, kann man hier wichtige Energiebereiche spreizen und linear abbilden (z. B. um bestimmte Linien im Spektrum herum), unwichtige Energiebereiche stauchen und zu einem Binning-Wert zusammenfassen.

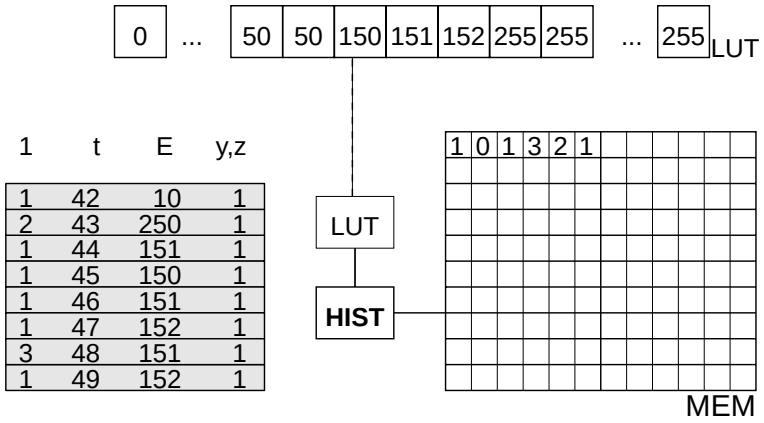


Abbildung 2.14: Schematische Darstellung der Erstellung des Histogramms. 8 Ereignisse werden durch ihre Position und Energie in den Histogramm-Speicher einsortiert. Durch das *Binning* der Energie mit Werten aus dem LUT-Speicher wird in unserem Beispiel die Energie um den Wert 151 genau abgebildet.

Diese Binning-Werte liegen im LUT-Speicher. Jede Histogrammart besitzt einen vollständigen Satz von Binning-Werten. Zusätzlich wird zwischen Mehrfach- und Einfach-Histogrammen unterschieden (Abb. 2.10).

[25] Natürlich weiß man, wann man die Aufnahme des Histogrammes gestartet hat. Die zeitliche Auflösung ist also gleich der Integrationszeit des Histogramms

Zu jedem Ereignis kann jeweils nur ein Zähler erhöht werden. Ist sowohl das Einfach-/Mehrfach-Histogramm als auch das Polarimetrie-Histogramm eingeschaltet, so wird bei einem Polarimetrie-Ereignis (Zweifach-Ereignis mit erkanntem Streuwinkel) nur der entsprechende Polarimetrie-Histogramm Zähler erhöht, nicht aber der entsprechende Mehrfach-Histogramm Zähler.

Da das Auslesen dieses Speichers mit der begrenzten Telemetrierate ein koninuierliches Arbeiten des Histogramms verhindern würde, ist jeder Histogrammspeicher doppelt vorhanden. Wird der eine Speicher ausgelesen, kann gleichzeitig in dem anderen weiter integriert werden ohne die Beobachtung zu unterbrechen. Die Integrationszeit ist beliebig. Sie wird von der DPE durch Umschalten der Histogramm-Speicher und Auslesen des Speichers festgelegt.

Fehler können entstehen, wenn einer der 8 Bit Zähler überläuft (Overflow). Der entsprechende Zähler bleibt bis zum Löschen des Histogrammspeichers auf seinem Maximalwert 255 stehen. Dieser Fehler wird erkannt und gehalten bis er per Telekommando gelöscht wird.

Ist das Modul inaktiv, werden keine Zählimpulse zum Hauptspeicher geschickt, d. h. es wird nicht integriert.

2.4.8 Time Coincidence & Time Stamp C_TC

Um sog. Compton-Ereignisse, d. h. Ereignisse von einem Photon, die sowohl im CdTe-Detektor und durch Compton-Streuung auch im CsI-Detektor Ereignisse ausgelöst haben, zu erkennen, muß die Gleichzeitigkeit der Ereignisse hergestellt werden.

Da durch die unterschiedlichen Datenraten der beiden Detektoren und die FIFO-Speicher der Detektoren es nie gewährleistet werden kann, daß ein CdTe- und ein CsI-Ereignis, welche zusammen in der HEPI verarbeitet werden, wirklich gleichzeitig sind, erfolgt an dieser Stelle im C_TC Modul die genaue zeitliche Synchronisation anhand der Zeitwerte in den einzelnen Ereignissen. Das Modul veranlaßt dann den CMD_CTRL die „zu schnelle“ Pipeline in der HEPI anzuhalten und nur solange von der anderen Pipeline einzulesen, bis sich die Zeiten der Ereignisse in beiden Pipelines wieder gleichen.

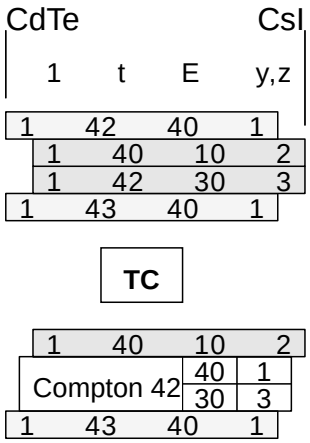


Abbildung 2.15: Schematische Darstellung der Erstellung eines Compton-Ereignisses. 2 CdTe- und 2 CsI-Ereignisse erreichen das TC-Modul. Anhand der Zeit t der Ereignisse wird Gleichzeitigkeit zur Zeit $t = 42$ erkannt und ein Compton-Ereignis erzeugt.

Diese Gleichzeitigkeit ist in einem kleinen Zeitfenster von 238 ns bis 61 μs einstellbar.

Wird eine Gleichzeitigkeit erkannt, so wird ein 160 Bit Compton-Ereignis ausgegeben, welches die Daten der beiden Ereignisse enthält. Die einzelnen Ereignisse werden aber nicht mehr zur DPE weitergegeben (siehe Abb. 2.9), erscheinen aber im Histogramm (falls eingeschaltet).

Unabhängig davon werden die Zeiten von der hohen Auflösung der Detektoren (238,4 ns in 2 s) in eine grobere Einheit (61 μs in 72 h) umgerechnet (*Time Stamp*). Dieses neue Zeitformat wird von der DPE verwendet.

Ist das Modul inaktiv, so werden keine Zeitkoinzidenzen gesucht, aber weiterhin die Zeiten der Ereignisse angepaßt.

2.4.9 Energy Selection CSI_ES

Das letzte Modul zur Datenreduktion wurde im Ausgangs-Multiplexer des C_HBR_A (siehe 2.4.10) realisiert. Wenn für die aktuelle Beobachtung nur ein Energieband interessant ist, so können mit diesem Modul die passierenden Ereignisse auf dieses Energieband beschränkt werden. So kann z. B. ein niederenergetischer Untergrund komplett ausgeblendet werden.

Das Modul enthält zwei frei wählbare Schwellen, welche die obere und untere Grenze des Energiebandes (inklusive) darstellen. Beide Schwellen sind 8 Bit Zahlen. Ihre Einheit ist demnach das 4-fache der Einzel-Ereignisse mit einer Energiedarstellung von 10 Bit. Die Mehrfach-Ereignisse, seit dem CSI_MP Modul mit doppelter Energieeinheit, werden erkannt und entsprechend berücksichtigt.

Ist das Modul inaktiv, passieren alle Ereignisse das Modul.

2.4.10 High Bitrate Interface A und B, C_HBR_A, C_HBR_B

In diesem Modul laufen alle Daten in einem Multiplexer zusammen. Mit diesem Multiplexer wird entschieden, welche Daten zur DPE geschickt werden. Der Multiplexer hat folgende Eingänge:

- zwei 80 Bit Eingänge vom Modul C_TC. Dies sind die CsI- und CdTe-Ereignisse
- 160 Bit für die Compton-Ereignisse vom Modul C_TC
- 160 Bit für die SPT-Ereignisse vom Modul CSI_SPT
- zwei 80 Bit Eingänge direkt von den Detektor Schnittstellen CSI_HBR und CDTE_HBR

Über die direkte Verbindung von den Detektor-Schnittstellen zum Multiplexer dieses Moduls kann man die komplette wissenschaftliche Vorverarbeitung in der HEPI übergehen. Damit hat die DPE Zugriff auf die originalen Daten von den Detektoren mit ihrer genauen Zeitauflösung. Dieser Modus wird *Transparent-Mode* genannt.

Wenn alle Vorverarbeitungen der Daten in der HEPI beendet sind, werden die Ereignisse über das C_HBR_A, die erzeugten Histogramme über das C_HBR_B der DPE zur Verfügung gestellt. Die Daten werden seriell übertragen.

Es werden immer 160 Bit, also zwei 80 Bit Ereignisse oder ein 160 Bit Ereignis mit 5 MHz zur DPE gesendet. Dies ist die nominale Übertragungsfrequenz zur DPE.

2.4.11 Steuerung und Verwaltung

Neben den wissenschaftlichen Modulen sind in der HEPI weitere Module zur Ansteuerung der Speicher und Kommunikation mit der DPE vorhanden. Diese Module wurden von der Firma DD&T, Reutlingen erstellt.

CMD_CTRL

Der CMD_CTRL ist das zentrale Steuerwerk der HEPI. Er empfängt und dekodiert die Befehle von der DPE, antwortet entsprechend und weist die einzelnen Module der HEPI an, wie die Daten verarbeitet werden müssen.

Die Signalisierung von Fehlern aus den Modulen läuft hier zusammen.

LUT_CTRL

Dieses Modul dient der Steuerung des LUT-Speichers, der auf der HEPI-Platine untergebracht ist. Er stellt dem AC-Modul und dem HIST-Modul die benötigten Daten zum richtigen Zeitpunkt (Pipeline) zur Verfügung.

Die Ratemeter-Funktionalität ist hier implementiert: Vom HBR-Modul kommt die Pixel-Adresse, das LUT-Modul liest an der entsprechenden Adresse den Wert aus (16 Bit), addiert eins hinzu und schreibt den Wert wieder an diese Adresse. Ein Überlauf wird erkannt und gehalten.

MEM_CTRL

Das MEM_CTRL Modul dient zum Verwalten des MEM-Speichers. Hier wird auch das eigentliche Zählen der Histogramme vollzogen. Vom Hist-Modul kommt nur die Adresse, das MEM-Modul liest selbständig den Wert (8 Bit) an dieser Adresse aus, addiert end dazu und schreibt den Wert wieder an die Adresse. Ein Überlauf wird erkannt und gehalten.

2.5 Data Processing Electronic DPE

Die ***D**ata **P**rocessing **E**lectronic* stellt die Schnittstelle zwischen der allgemeinen Datenverarbeitung und der IBIS spezifischen Datenverarbeitung her. Jedes Instrument (außer OMC) hat aus Redundanzgründen zwei dieser Rechner, von denen aber immer nur einer benutzt wird. Die DPE ist eine bewährte Technik, die die ESA schon auf anderen Missionen erfolgreich eingesetzt hat.

Kernstück der DPE ist der Prozessor vom Typ MIL-STD-3-1750A. Er basiert auf einer Entwicklung des DoD[26]. Seine 16 Bit CISC Architektur ist vergleichbar der des i8086. Zusätzlich ist eine Fließkomma-Einheit vorhanden. Bei der verwendeten Frequenz von 13 MHz erreicht der 1750A etwa 1 MIPS[27]. Der in der DPE zur Verfügung stehende Speicherbereich von 1 Mword muß durch *Paging* in den 64 kword großen physikalischen Adreßraum eingeblendet werden.

Die Software läßt sich in zwei Teile aufteilen [9][10]:

- IASW, Instrument Application Software. Diese Software übernimmt die Steuerung des Instruments und die Auswertung der Daten. Diese Softwareteil ist für jedes Instrument anders. Für IBIS wurde diese Software am IAAT nach Vorgabe der ESA in ADA[28] entwickelt.

[26] Department of Defense, amerikanisches Verteidigungsministerium
[27] *million instructions per second*
[28] ebenfalls DoD

Abbildung 2.16: Die DPE während der Entwicklung und Tests an unserem Institut. Sichtbar wird der modulare Aufbau der DPE durch Elektronik-Einschübe wie z. B. die HEPI-Platinen. Der dritte Einschub von unten ist leer. Hier wurde gerade die Platine mit den EEPROMs für die DPE entfernt.

- CSSW, Common Service Software stellt das Betriebssystem der DPE aufbauend auf dem darunterliegendem ASTRES 1750 realtime multi-tasking System dar. Die CSSW ist bei jeder DPE gleich.

Die verwendete DPE ist der Grund für die Entwicklung der HEPI. Die beschränkte Rechenleistung der DPE ist nicht in der Lage, die hohen Datenraten der Detektoren zu verarbeiten. Deswegen mußte vor der DPE der Präprozessor HEPI gesetzt werden.

3 Zuverläßigkeit auf Satelliten

In der Vergangenheit ist immer wieder von Rückrufaktionen zu hören gewesen, sei es von Haushaltgeräten, Autos oder Flugzeugen. Dadurch, daß sich der Satellit aber durch seine hohe Umlaufbahn jeder weiteren direkten Einflußnahme entzieht, ist es unmöglich hier Reperaturen oder Verbesserungen durchzuführen.[1] Es muß also unter allen Umständen vor dem Start an jede mögliche Situation gedacht werden.[2]

3.1 Umwelteinflüsse

3.1.1 Mechanische Belastungen

Obwohl im Weltraum Schwerelosigkeit herrscht und nur dadurch die filigranen quadratmetergroßen Sonnensegel-Konstruktionen möglich sind, müssen alle mechanischen Bauteile vorher besonderen Belastungen gewachsen sein:

- Der Start einer Ariane Trägerrakete erzeugt Schwingungen bis zu 10^4 Hz und einer Amplitude von ca. 140 dB. Der Flug selber belastet die Nutzlast mit bis zu 4 g. Das Space Shuttle der NASA belastet die Astronauten mit bis zu 3 g. Früher, zu Zeiten der Saturn- und Apollo-Raketen, betrug die Beschleunigung 8 – 12 g.

- Das Abtrennen der Nutzlast von der Trägerrakete wird mit einer Sprengladung vollzogen. Hier werden Beschleunigungswerte von über 2000 g mit Frequenzen bis 1,5 kHz erzeugt.

Um diesen Beschleunigungen gewachsen zu sein, werden z. B. sämtliche Schraubverbindungen noch mit einem Kunstharz verklebt, um ein Lösen der Schrauben zu verhindern. Ebenso werden die meisten elektronischen Bauteile verklebt, um die Kontakte vor Verbiegungen zu schützen und ein Abreißen zu verhindern [8].

3.1.2 Vakuum

Das Fehlen der Atmosphäre bringt folgende Aspekte mit sich:

- Kein Wärmeaustausch durch Konvektion: Die einfache Möglichkeit überflüssige Wärme (z. B. von ICs) durch einen Lüfter abzuführen fehlt. Der Wärmeaustausch ist nur durch Wärmeleitung (z. B. mit *Heatpipes*) und durch Strahlung innerhalb des Satelliten möglich. Diese Wärme wird dann mit Radiatoren an der Oberfläche des Satelliten abgestrahlt. Neben der aufwendigeren Konstruktion bringt die Maßgabe diese Radiatoren nicht der Sonne auszusetzen auch Restriktionen in der Handhabung des ganzen Satelliten mit sich.

[1] Das Hubble Space Teleskope war eine Ausnahme, weil nicht auf einer hohen Umlaufbahn
[2] Der deutsche Kleinsatellit ABRIXAS zeigte leider, daß das nicht immer klappt.

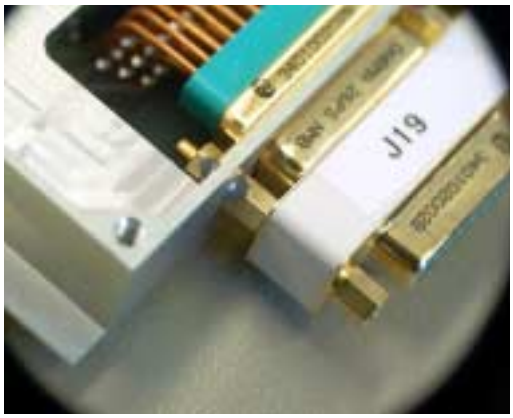


Abbildung 3.1: Links: Auf der Platine der HEPI sieht man den transparenten Kunstharz mit dem die Pins der elektronischen Bauteile fixiert werden. **Rechts:** Gut zu erkennen sind in dieser Makro-Aufnahme die zusätzliche Verklebung aller Schrauben mit einer grauen Kunstharz Masse. Mit „J19“ ist ein sog. *connection saver* gekennzeichnet. Er schützt die Kontakte der Schnittstelle vor Abnutzung während der letzen Tests.

- Ausgasen, Sublimation: Sinkt der Luftdruck unter den Dampfdruck eines Materials, beginnen die Oberflächenmoleküle sich durch spontanes Verdampfen von der Oberfläche zu lösen. Verstärkt wird dieser Vorgang durch Erhöhung der Temperatur. Dieses Material schlägt sich in der Umgebung nieder und verändert dadurch die Eigenschaften des darunterliegenden Materials (optisch, elektrisch, etc.). Eingeschlossene Gasblasen können explosionsartig ausgasen und so weite Teile des Materials zerstören. Ebenso werden eingelagertes Wasser und adsorbierte Gase freigesetzt. Mechanische Teile erfordern die Benutzung von schwerflüchtigen Ölen wie Hochvakuum-Fette bzw. -Öle.

- In niedrigen Umlaufbahnen kommt die Gefahr durch einatomigen Sauerstoff hinzu. Es kommt zur Bildung von Oxiden mit anderen Eigenschaften (mechanisch, elektrisch, etc.) oder leichtflüchtigen Produkten, die zu einem Abtragen von Oberflächenmaterial führt.

Dies stellt hohe Anforderungen an die zu verwendeten Materialien [8].

3.1.3 Temperaturdifferenzen

Für die Komponenten eines Satelliten ergeben sich je nach Lage zur Sonne große Temperaturdifferenzen. Gewöhnliche elektronische Bauteile sind hier ungeeignet. Verwendet werden Bauteile mit erweitertem Temperaturbereich. So ist z. B. der ASIC der HEPI getestet und verifiziert worden für den Betrieb zwischen -55°C und 125°C.

Neben den elektronischen Bauteilen muß auf die unterschiedlichen Ausdehnungskoeffizienten verschiedener Materialien geachtet werden. Als Beispiel seien hier die Pixel des PICsIT-Detektors genannt. Hier wurden die Szintillator-Kristalle mit den Photodioden verklebt.
Während der Entwicklung des Detektors mußte hier der geeignete Kleber gefunden werden.

3.1.4 Strahlung

Ein wichtiger Punkt der bei der Verwendung von Elektronik an Bord eines Satelliten in Betracht gezogen werden muß, ist die ionisierende Strahlung. Stellvertretend seien hier zwei

Beispiele erwähnt:

- Am 9. July 1962 wurde von den USA im Rahmen des Projektes Starfish überirdisch eine Atom-Bombe gezündet. Im Laufe der nächsten 5 Jahre wurden 10 Totalausfälle von Satelliten diesem Ereignis zugeschrieben (einige fielen direkt nach der Explosion aus). Die Explosion führte zu einer Injektion von Elektronen mit bis zu 7 MeV in den van-Allen-Gürtel dessen Fluß um den Faktor 100 anstieg. Diese Elektronen dominierten die ersten 5 Jahre den van-Allen-Gürtel und ließen sich noch bis 8 Jahre nach der Explosion nachweisen.
- 1991 führte ein Latch-up eines 64 kBit CMOS Speichers an Bord des ESA Satelliten ERS-1 zum Abbruch des Experimentes.

Die wichtigsten Strahlungsquellen sind:

- Van-Allen-Gürtel. Geladene Teilchen des Sonnenwinds werden im magnetischen Feld der Erde gefangen. Der Van-Allan-Gürtel besteht aus 2 Teilen:
  - Innerer Gürtel (400 bis 900 km) besteht aus Elektronen und gefangenen Protonen
  - Äußerer Gürtel (bis zu 56000 km) besteht hauptsächlich aus Elektronen

  Durch den hochexzentrischen[3] Orbit von INTEGRAL wird der van-Allen-Gürtel bei jedem Umlauf durchlaufen.
- Ausbrüche auf der Sonne führen zu einem verstärkten Fluß von hochenergetischen Protonen und schweren Ionen. Der Fluß im van-Allan-Gürtel kann dadurch bis um den Faktor 1000 gesteigert werden.
- Kosmische Strahlung bestehend aus Protonen (85%), Alpha-Teilchen (13%) und Kernen von Wasserstoff bis Nickel (2%) und hochenergetische Quanten im Röntgen und Gamma-Bereich. Diese sind das Ziel der Untersuchung von INTEGRAL.

Treffen diese Teilchen auf den Satelliten und dessen Abschirmung, werden duch Bremsstrahlung Quanten im Röntgen- und Gamma-Bereich erzeugt, welche zu einer weiteren Strahlenbelastung der Elektronik führt. Ein wirksamer Schutz ist also nur mit einer dicken (schweren) Abschirmung zu erreichen. Dies scheitert in den meisten Fällen an den Transportkosten.

Strahlungseffekte

Oberflächen-Effekte, Total Dose: Ionisierende Strahlung erzeugt im SiO_2 Elektronen-Loch Paare. Durch die unterschiedliche Beweglichkeit und durch Rekombinationseffekte wird eine positiv geladene Schicht in der Trennfläche Si–SiO_2 erzeugt (Gate, Oxid-Schichten). Dies führt zur Änderung von elementaren Eigenschaften des Halbleiters wie dem Bandabstand und der Beweglichkeit der Ladungsträger. Daraus resultieren folgende Effekte:

- Stand-By Strom vergrößert sich. Als direkte Folge läßt der Wirkungsgrad von Solarzellen im Laufe der Zeit nach
- Der elektronische Baustein wird empfindlicher in Bezug auf seinen Eingangspegel und seine interne Rauschunempfindlichkeit

[3] Perigäum: 10000 km, Apogäum: 152600 km

- Die Schaltgeschwindigkeit verringert sich durch Vergrößerung der *Rise*- und *Fall*-Zeiten

Die dünnen Oxidschichten der N- und P-Transistoren, bzw. deren Bandabstand verändern sich etwa mit -0,6 mV pro Krad(Si) Bestrahlung.

Tiefere Effekte: Hochenergetische Teilchen erzeugen beim Durchgang durch den Halbleiter eine Spur von Elektronen-Loch Paaren. In dieser Spur kann durch die beweglichen Ladungsträger ein Strom fließen. Geht diese Spur nun durch die Ebenen der Stromversorgung des Halbleiters wird ein Kurzschluß erzeugt, der den Halbleiter an dieser Stelle irreversibel zerstört (*Latch-Up*).

Ereicht dieser Strom die Verarmungszone (*depletion zone*) eines Transistors, kann bei ausreichender Ladung der Transistor zu einem Schaltvorgang gebracht werden. Dieser *Soft-Error* ist reversibel. Auswirkungen sind z. B. Bit-Flips in Speicherzellen und Flip-Flops. Je kleiner der Transistor ist (je „moderner“ die verwendete Technologie ist), umso weniger Ladung ist zum Schalten des Transistors nötig. Fehlerkorrekturen können hier helfen [2][3][4][7].

3.2 Strahlungsharte Technologie

Durch die immer weitere Verkleinerung der Strukturen in modernen VLSI-Bausteinen werden diese immer empfindlicher gegenüber ionisierender Strahlung. Dies führt bei Einsatz in Satelliten zu inakzeptabler Fehlerhäufigkeit und Ausfall ganzer Baugruppen.

Der ASIC der HEPI wurde bei der Firma TEMIC Semiconductors produziert. TEMIC verwendet folgende Methoden seine Halbleiterprodukte unempfindlich gegen ionisierende Strahlung zu machen:

- Epitaxial Wafer $4\mu m$, 8 – 15 Ω
- Doppelt hohe Oxidschichten (*well*)
- 4fache Dotierung mit Bor
- 2fache Dotierung mit Phosphor

Mit diesen Methoden können strahlungstolerante Halbleiter mit den gleichen Masken erstellt werden wie die „normalen“ Halbleiterprodukte. So kann auf bewährte Technologie und Bibliotheken zurückgegriffen werden.

3.2.1 Verwendete Technologie

Für den ASIC der HEPI wurde die MG1RT Technologie von TEMIC Semiconductors verwendet. Diese Technologie ist von der ESA zertifiziert und für Satellitenprojekte freigegeben.

Typ	Anzahl Logic Zellen	Verfügbare I/O-Pins
MG1090	88536	212

Die MG1RT ist die strahlungstolerante Version der MG1 Sea of Gates 0,6 μm CMOS Technologie von TEMIC. Zur Verdrahtung werden drei Metallisierungsebenen verwendet. Der Kern besteht aus einer homogenen Matrix (*Sea of Gates*) von Logik-Zellen (*logic cells*), ununterbrochen durch sog. *Routing*- (Verdrahtungs-) Kanäle. Die Größe einer Logik-Zelle beträgt $25 * 13{,}8\,\mu$mn. Jede Logik-Zelle kann die Funktion einer einfachen logischen Verknüpfung wie

z. B. eines NAND-Gatters übernehmen oder einer Speicherstelle (1 Bit RAM oder 4 Bit ROM). Komplexere Funktionen wie z. B. ein Latch werden durch die Verbindung mehrerer Logik-Zellen erzeugt. Für diese Verdrahtung wird hauptsächlich die erste Metallisierungsebene verwendet, so daß genügend Reserven zum Verdrahten dieser komplexen Funktionen untereinander durch die beiden anderen Metallisierungsebenen besteht [6] [5].

Durch die drei Metallisierungsebenen steht eine große Routing-Lapazität zur Verfügung. TEMIC gibt an, bei einem durchschnittlichen (*random logic*) Design immerhin 70% der Logik-Zellen zur Verfügung stellen zu können, nur 30 % gehen durch Verdrahtung verloren. Bei sehr regelmäßigen Speicherstrukturen geht diese *gate utilisation* auf bis zu 200 % hoch. Hier sind dann 2 Bit in einer Logik-Zelle realisiert.

Ein wichtiges Kriterium für einen Halbleiter sind die Signallaufzeiten. Die Signallaufzeiten sind von den äußeren Faktoren abhängig. Als Verzögerungsfaktor K gibt TEMIC an:

$$K = K_V * K_T * K_P * K_D$$

mit

Spannung	K_V
4,5 V	1,1
5 V	1,0
5,5 V	0,93

Temperatur	K_T
-55°C	0,74
25°C	1,0
145°C	1,38

Prozeß	K_P
best	0,78
typical	1,0
worst	1,28

Strahlung	K_D
0 krad	1
50 krad	1

Multipliziert man alle K-Werte zusammen, erhält man im schlechtesten Fall eine Zunahme der Laufzeiten um 94 %. Dies wird bei der sog. *backannotated* Simulation berücksichtigt um ein wirklichkeitsgetreues Ergebnis zu liefern.

Dem K_P-Wert für die Güte des Prozesses ist man allerdings ohne Einfluß ausgeliefert. Dieser wird bestimmt durch die tatsächlichen Prozeßgenaugigkeiten bei der Erzeugung dieses ASICs. Für die HEPI wurde auf das sog. *silicon risk*[4] Verfahren zurückgegriffen. Hier sind alle ASICs auf einem Wafer und haben somit den selben K_P-Wert.

Als strahlungstolerante Technolgie ist der K_D-Wert natürlich immer 1.

[4] Silicon Risk bedeutet, daß alle Schaltkreise vom ASIC zur gleichen Zeit angefangen werden (EM, QM, FM). Nach der Produktion und Auslieferung des EMs wird die weitere FM-Produktion angehalten. Nach den Tests der EMs werden die FMs wieder freigegeben. Falls die EM-Tests einen Fehler im Design aufzeigen, muß man die angefallenen FM-Kosten (hauptsächlich den Wafer) bezahlen. Diese Methode der ASIC-Herstellung ist preislich günstiger als andere Methoden.

4 Die Entwicklung der HEPI

Bei der Entwicklung der HEPI wurde konsequent *top-down* Entwicklung betrieben. Nachdem klar war, welche Funktionen der Präprozessor HEPI der DPE abnehmen muß (aus Gründen der Verarbeitungsgeschwindigkeit), wurden diese einzelnen Aufgaben weiter spezifiziert und deren Ablauf beschrieben [12][13].

4.1 Pipeline-Konzept

Ein wichtiges Konzept für die hohe Verarbeitungsgeschwindigkeit der HEPI ist das Pipeline-Konzept. Die einzelnen funktionalen Gruppen (Module) können autark ein Ereignis bearbeiten ohne mit den anderen wissenschaftlichen Modulen interagieren zu müssen. Jedes Modul bekommt seine Informationen aus dem internen 80 Bit (160 Bit) Datenpaket welches ein Ereignis beschreibt und legt sein Ergebnis auch wieder in ein 80 Bit Datenpaket ab. Abbildung 4.1 verdeutlicht die Funktionesweise einer Pipeline:

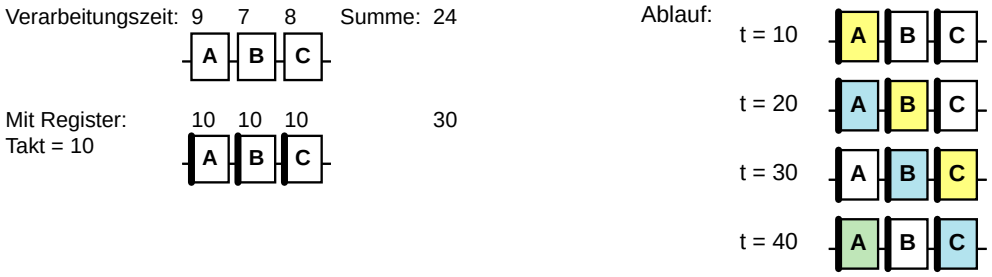


Abbildung 4.1: Pipeline-Konzept

Die drei Module A, B und C mit ihren Verarbeitungszeiten 9, 7 und 8 Zeiteinheiten können nur alle 24 Zeiteinheiten neue Daten aufnehmen, da erst nach dieser Zeit die alten Daten sicher verarbeitet wurden und hinter C stabil zur Verfügung stehen.

Für die Pipeline-Version dieser drei Module wird durch Einbau eines Eingangsregisters vor jedem Modul aus den drei Modulen drei Pipelinestufen gemacht. Dieses Eingangsregister versorgt das entsprechende Modul mit stabilen Eingangsdaten innerhalb eines Pipelinetaktes, obwohl sich die Ausgänge des vorherigen Moduls, während seiner Verarbeitung der nächsten Daten laufend ändern können. Der Pipelinetakt muß größer sein als die maximale Verarbeitungszeit eines der drei Module. In unserem Beispiel wurden 10 Zeiteinheiten gewählt. Zu jedem Pipelinetakt übernimmt ein Modul die Daten von seinem Vorgänger und beginnt unverzüglich mit der Bearbeitung der Daten. Nach einer Latenzzeit von 30 Zeiteinheiten ist das erste Ereignis am Ausgang. Dies ist zwar länger als die 24 Zeiteinheiten ohne die Pipeline, aber die Pipeline liefert ab jetzt mit jedem Pipelinetakt (10 Zeiteinheiten) ein weiteres Ereignis am Ausgang.

4.1.1 Kennzeichnung gültiger Daten, Valid Data

Da zu einem Pipelinetakt auch *keine* neuen Daten von den Detektoren geliefert werden können, müssen „echte“ Daten von den „leeren“ Daten, die dann durch die Pipeline gereicht werden, getrennt werden. Dazu wurde das *Valid Data* Signal (VD) eingeführt. Die Detektor HBRs der HEPI setzen das VD-Signal wenn ein neues Ereignis empfangen wurde, und löschen es in den Übertragungspausen. Das VD-Signal läuft nun parallel zu den 80 Bit Ereignisdaten durch die Pipeline. Die Module können nun anhand des VD-Signals erkennen, ob sie in diesem Pipelinetakte Daten zu verarbeiten haben oder nicht.

4.2 KISS & Konventionen

Das Arbeiten im Team an so einem großen Projekt braucht Regeln. In unserer Arbeitsgruppe wurden Konventionen eingeführt, die das Zusammenspiel der Module der unterschiedlichen Entwickler erleichterten.

Als eines der wichtigsten allgemeinen Regeln sei hier das KISS Prinzip genannt (*keep it sweet and simple*[1]). In Worte gefaßt geht es darum, die Lösung durch einfache Mittel zu erreichen. Darauf aufbauend sind die zwei folgenden Entwicklungsregeln entstanden.

4.2.1 Namensschemata

Bei der Entwicklung werden die einzelnen Module über Signale verbunden. Für diesen doch recht einfachen Vorgang gibt es mehrere Ansätze.

Beispiel:
In Abbildung 4.2 stellt Modul A seine Daten dem Modul B zur Verfügung.

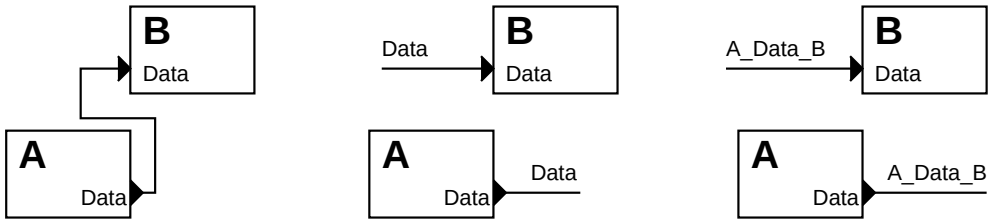


Abbildung 4.2: Konventionen bei der Benamung von Signalen

Der herkömmliche Weg ist in der Abbilung links dargestellt: Jedes Signal wird einzeln vom Ausgang Modul A bis zum Eingang Modul B verbunden (*connection by wire*). Dies ist in sehr einfachen Schaltplänen noch sinnvoll, wird aber sehr schnell sehr unübersichtlich. Es wurde von uns nur noch benutzt, um direkt benachbarte Module auf dem Schaltplan zu verbinden.

Abbildung 4.2 Mitte: Durch *connection by name* können komplexere Schaltpläne entwirrt werden. Das Entwicklungswerkzeug weiß anhand des gleichen Namens eines Signals, welche Bauteile damit zu verbinden sind. Dies ist die bevorzugte Arbeitsweise bei der Entwicklung der HEPI gewesen.

Abbildung 4.2 Rechts: Wird der Schaltplan noch komplexer, bzw. zur Verbindung der wissenschaftlichen Module untereinander, wurde eine weitere Regel eingeführt [11], um z. B. die

[1] auch als *keep it sweet, simple* bekannt.

80 Bit Ereignisdaten die zwischen den wissenschaftlichen Modulen weitergereicht werden, einfach und verwechslungsfrei zu leiten.

Das Schema zur Namensvergabe eines Signals besteht aus drei Teilen:

1. Der Name des Moduls aus dem das Signal kommt, abgekürzt
2. Die Funktion des Signals
3. Der Name des empfangenden Moduls, abgekürzt

In unserem Beispiel also A_Data_B, das Daten-Signal von Modul A nach Modul B.

4.2.2 „Standard"-Bauteile

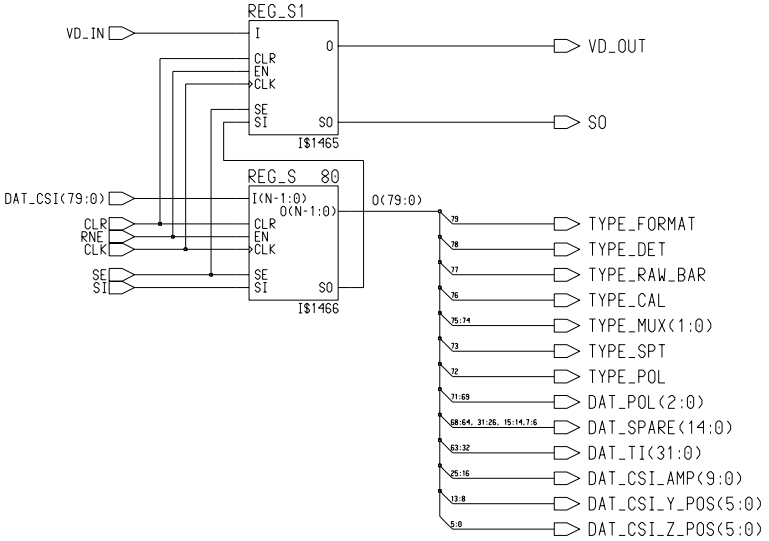


Abbildung 4.3: „Standard"-Bauteil DAT_CSI_IN mit dem *Valid Data* und den 80 Bit Ereignisdaten

Da alle wissenschaftlichen Module ähnlich aufgebaut sind (Eingangsregister mit Pipelinetakt-Steuerung, Aufsplitten der 80 Bit in ihre einzelnen Funktionen und nach der Bearbeitung im Modul wieder Zusammenführen der einzelnen Signale zum standardgemäßen 80 Bit Format), bot es sich an, hier Module zu entwerfen, die diese Aufgaben erledigen und von jedem beteiligten Entwickler verwendet wurden. Abbildung 4.3 zeigt das Modul welches das Eingangsregister implementiert.

Der Vorteil von solchen Standard-Bauteilen liegt in der Zeitersparnis beim Entwickeln und beim Testen. Mit den Standard-Bauteilen muß nur einmal entwickelt und getestet werden, bei individuellen Lösungen muß das jeder Entwickler selber machen. Dort sind Fehler geradezu vorprogrammiert. Änderungen hier müssen nur an einer Stelle gemacht werden und sind sofort im ganzen Design konsistent.

4.3 Vom Zeichnen zum Beschreiben

Die Entwicklung der HEPI ist vergleichbar mit den Veränderungen in der EDA[2]-Industrie. Wurde früher ausschließlich mit Schaltplänen gearbeitet, so wird heutzutage immer mehr mit

[2] *Electronic Design Automation*

VHDL gearbeitet. Die wissenschaftlichen Module wurden als Schaltpläne begonnen, das Zusammenführen aller Module in der HEPI aber mit den VHDL-Quellen vollzogen.

4.3.1 Schaltpläne zeichnen

Das Zeichnen von Schaltplänen (*schematics*) ist der klassische Weg zu einem Design. Hier werden Bauteile auf einem Schaltplan plaziert und deren Ein- und Ausgänge mit Signalen verbunden. Durch Bildung von Hierarchien können komplexe Schaltpläne aufgeräumt werden.

Der Vorteil dieses Ansatzes ist die meist leichtere „Lesbarkeit“ von größeren Designs. Signale können recht einfach auf ihrem Weg durch den Schaltplan verfolgt werden.

Als großer Nachteil ist die Anhängigkeit von der Ziel-Technologie zu nennen. Man verwendet immer Bauteile aus einer Bibliothek einer Technologie. Wechselt man die Technologie, dann muß man alle verwendeten Bauteiele durch ihr Gegenstück aus der Bibliothek der neuen Technologie austauschen.

Die Werkzeuge (*Design Architect*) zum Erstellen der Schaltpläne (*schematic design entry*) stammen von Mentor Graphics Inc. Im Rahmen der EUROPRACTICE Lizenzen ist es für das Institut möglich diese Produkte einzusetzen [17][23].

4.3.2 Programmieren von Hardware-Beschreibungen

VHDL ist eine Beschreibungssprache für digitale Elektronik. Ihren Ursprung fand sie in dem VHSIC-Programm (*Very High Speed Integrated Circuit*) des US Militärs. In diesem Programm wurde klar, daß es sinnvoll ist eine Sprache zu entwickeln, um Strukturen und Funktionen von Integrierten Schaltungen zu beschreiben. Daraus wurde die *VHSIC Hardware Description Language* (VHDL) entwickelt. VHDL ähnelt ADA, ebenfalls vom DoD entwickelt. Nach schrittweisen Verbesserungen wurde VHDL dem Institute of Electrical and Electronical Engineers (IEEE) vorgelegt und schließlich 1987 in Form des IEEE Standard 1076, Standard VHDL Language Reference Manual, verabschiedet [14][15].

Die Vorteile gegenüber der herkömmlichen Schaltplan-Entwicklung sind:

- Beschreibung der Struktur eines Systems, die Aufteilung in Module und Komponenten und die Verbindungen dieser Teile
- Die Spezifikation einer Funktion eines Systems erfolgt mit Ausdrücken ähnlich gängiger Hochsprachen
- Die Simultaion ist möglich noch bevor das System in Hardware umgesetzt wurde. Dies führt zu schnelleren Design-Zyklen und Kostenersparnis
- Die genaue Implementierung ist Aufgabe der Synthese und nicht mehr Aufgabe der Entwickler, die nun durch abstrakte Beschreibung schneller komplexe Systeme entwerfen können

Eine VHDL-Synthese ist vergleichbar mit einem Software-Compiler: Der Quelltext der Hochsprache wird auf die maschinenspezifischen Befehle bzw. logischen Gatter der Zieltechnologie umgesetzt. Als Ergebnis erhält man eine EDIF-Netzliste,[3] in der beschrieben ist wie die Komponenten (Logik-Zellen, Gatter) zu verbinden sind [16].

[3] *Electronic Design Interchange Format*

4.3.3 Übergang vom Zeichnen zum Programmieren

Da die Integration der wissenschaftlichen Module in die HEPI nur auf VHDL-Ebene effizient möglich war, wurde kurz vor dieser Integration ein Schnitt vollzogen: Alle Schaltpläne wurden mit einer Software (*VHDL-Write* von Mentor Graphics Inc.) in VHDL umgewandelt. Nun konnten die wissenschaftlichen Module wie auch die nur in VHDL beschriebenen Steuer-Module in ein Design gepackt werden und zur kompletten HEPI verbunden werden.

Die darauf folgenden Tests im Simulator wie auch in der Hardware (FPGA, siehe 5.3.1), wurden immer auf der VHDL-Datenbasis getätigt. Waren Änderungen nötig (Fehler ausbessern, neue Funktionen implementieren, etc.) wurde dies in den VHDL Dateien gemacht.

Parallel dazu wurde immer versucht diese Änderungen auch im Schaltplan wieder einzuarbeiten, um auch diesen aktuell zu halten. So konnte bei Fragen auf die bessere Lesbarkeit der Schaltpläne zurückgegriffen werden.

Das Problem bei den mit VHDL-Write entstandenen VHDL-Dateien ist die mangelhafte Lesbarkeit. VHDL-Write kann nicht erahnen welche Funktionen der Entwickler mit einem Schaltplan verfolgt hat und dem entsprechend ein „einfaches“ VHDL erzeugen. Im Gegenteil: VHDL-Write erzeugt stur ein Abbild des Schaltplanes in VHDL.

Der Schaltplan für unser Beispiel ist unter D.2 auf Seite 84 zu sehen.

VHDL-WRITE erzeugt folgende Zeilen:

```
   for I_036214 : or2 use entity work.or2(black_box);
   for I_036215 : or2 use entity work.or2(black_box);
begin
   -- CONCURRENT SIGNAL ASSIGNMENTS
      HIST_ADR_LUT <= local_HIST_ADR_LUT;
   -- COMPONENT INSTANTIATIONS
      I_036214 : or2
         port map(
            DIN0 => TYPE_MUX(1),
            DIN1 => TYPE_POL,
            DOUT => local_HIST_ADR_LUT(10)
         );
      I_036215 : or2
         port map(
            DIN0 => TYPE_POL,
            DIN1 => TYPE_CAL,
            DOUT => local_HIST_ADR_LUT(11)
         );
end structure;
```

Von „Hand“ wird ein Entwickler diesen Schaltplan wie folgt übersetzen:

```
begin
      HIST_ADR_LUT(10) <= TYPE_MUX(1)  OR  TYPE_POL;
      HIST_ADR_LUT(11) <= TYPE_CAL     OR  TYPE_POL;
end structure;
```

Mit diesem einfachen Beispiel kann man abschätzen, wie schwierig die Fehlersuche in den großen Modulen wie dem CSI_MP oder CSI_SPT war.

4.4 Ablauf der Entwicklung

Die Entwicklung der HEPI läßt sich grob in folgende Reihenfolge bringen:

1. Erstellung der Schaltpläne
2. Simulation der Schaltpläne. Bis hier hin waren bis zu drei Entwickler beteiligt
3. Erstellung der Testbench um die einzelnen Funktionalitäten der Module zu überprüfen
4. VHDL-Quelltext verändern um Fehler zu beseitigen oder neue Funktionen zu implementieren, ggf. Testbench anpassen
5. Testen der Module in der HEPI mit Hilfe der Testbench
6. Testen der Module der HEPI in der Hardware
7. Systemtests mit Durchspielen realer Beobachtungen
8. Vorbereitung zur ASIC-Migration

Hier wurden viele Iterationen gebraucht. Alleine der Weg von 4 bis 6 wurde mehr als 100 mal durchlaufen.

4.4.1 Design Flow

Der Design Flow im Einzelnen (Abb. 4.4 auf Seite 43):

MENTOR Schematic Die wissenschaftlichen Module wurden zuerst in Form von Schaltplänen entwickelt. Benutzt wurde *Design Architect* von Mentor Graphics Inc.

Modul Test Die Schaltpläne der wissenschaftlichen Module wurden mit *QuickSim* von Mentor Graphics Inc. getestet. Nach diesen Tests war die grobe Funktionaltität sichergestellt.

VHDL Die Steuer-Module und der Rahmen der HEPI wurden von Beginn an in VHDL entwickelt. Dafür reicht ein einfacher Texteditor, Angenehmer sind Editoren mit VHDL *context highlighting* wie z. B. *Emacs*.

VHDL-Write Mit diesem Programm von Mentor wurden unsere Schaltpläne in VHDL übersetzt.

VHDL Design Nach dem Übersetzen der Schaltpläne liegen alle Module der HEPI in VHDL vor. Hier wurden die Module untereinander verbunden und zu einem funktionsfähigem Design zusammengeabut. Dies ist die Datenbasis auf der alle folgenden Änderungen und Fehlerbeseitigungen statt fanden.

CVS Das *Concurrent Versions System* ist eine Datenbank die den kompletten VHDL-Quelltext des HEPI-Projektes (und weitere Quelltexte, Skripte, etc) enthält. Es ist ein Client-Server basierendes System um Versionskontrolle in größeren Projekten zu ermöglichen [22].

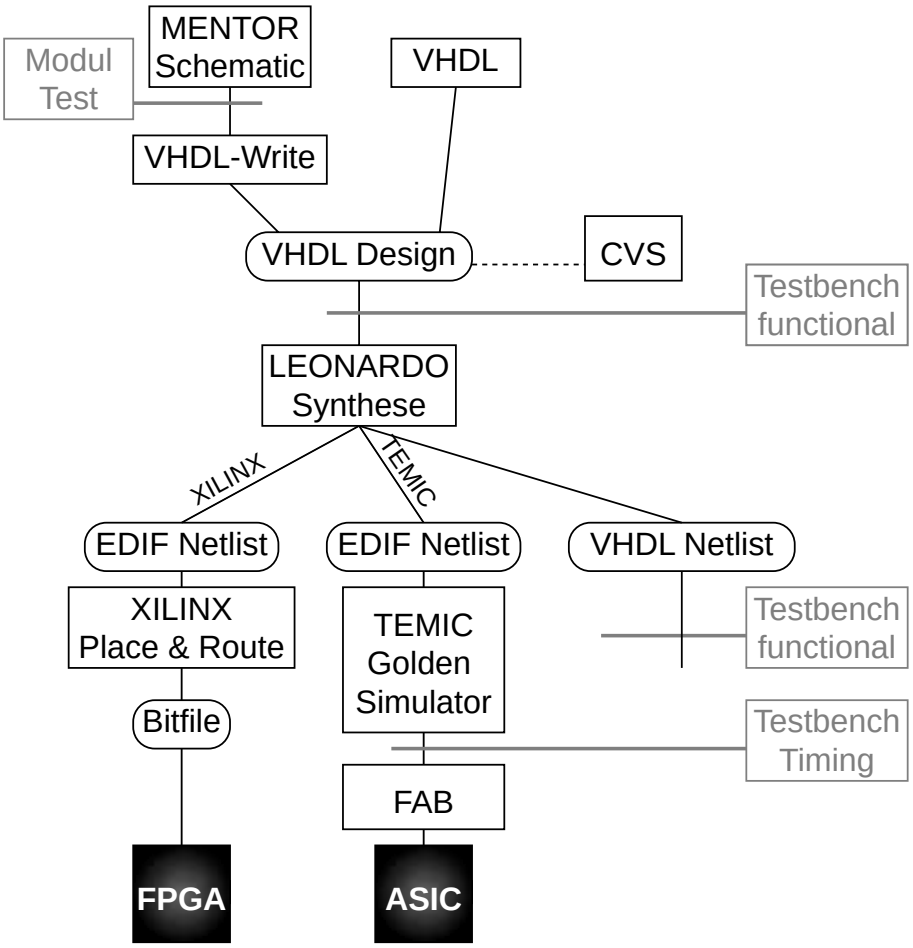


Abbildung 4.4: Der Design Flow

Testbench functional Test der gesamten HEPI. Mit Hilfe der Testbench und der darauf aufbauenden automatischen Tests wird hier die Korrektheit der Funktionen der HEPI überprüft. Als Simulator kam ModelSim von Model Technology (jetzt Mentor) zum Einsatz [18].

Leonardo Synthese von Exemplar Logic (jetzt Mentor) ist das Synthesewerkzeug. Bei der Synthese wird aus dem (technologieunabhängigen) VHDL-Quelltext eine technologieabhängige EDIF-Netzliste erzeugt. Verwendet haben wir das *mappen* auf die XILINX-FPGA Technologie für unsere Hardware-Tests, sowie das endgültige *mappen* auf die ASIC-Technologie von TEMIC [21].

XILINX Place & Route Mit dem *Backend* von XILINX wird aus der EDIF-Netzliste eine Verdrahtung des FPGAs erzeugt. Dieser Vorgang ist weder trivial noch deterministisch und dauert mehrere Stunden.

Bitfile Das *Bitfile* ist das Ergebnis des XILINX Backends. Es ist die Information wie alle SRAM-basierenden Schalter der Verdrahtung des FPGAs zu setzen sind, um die gewünschte Funktionalität zu erzeugen. Dieses Bitfile wird in ein EEPROM gebrannt

und dieses EEPROM in die HEPI-Platine eingesetzt. Beim Einschalten der HEPI (*Power Up*) lädt der FPGA automatisch seine Konfiguration und ist betriebsbereit.

TEMIC Golden Simulator Dieser Simulator ist mit allen Informationen von TEMIC gespickt um ein möglichst wirklichkeitsgetreues Simulieren zu ermöglichen. Dieser Simulator liefert, nach dem Place & Route bei TEMIC, die vorraussichtlichen Verzögerungszeiten zwischen den einzelnen Gattern basierend auf der nun genauen Kenntnis der einzelnen Verdrahtungslängen und -kapazitäten.

Testbench Timing Mit den vom *Golden Simulator* erzeugten *rise*- und *fall*-Zeiten jedes einzelnen Gatters können alle Funktionen der HEPI nochmal simuliert werden. Wichtig hierbei ist die Feststellung, daß kein Signal jetzt „zu lange“ braucht, also das Design mit der vorgesehenen Taktfrequenz arbeitet.

FAB Sind alle Test abgeschlossen, wird der *sign off* zur Produktion gegeben. Jetzt wird der ASIC produziert.

5 Test der HEPI

Während bei der Entwicklung der HEPI konsequent *Top Down* Entwicklung betrieben wurde, wurde beim Test der HEPI nach der *Bottom Up* Methodik vorgegangen:

- Simulation der Schaltpläne mit QuickSim
- Simulation des VHDL-Quelltextes der HEPI und allen (Dummy-) Modulen in der Testbench
- Test der HEPI als FPGA auf der HEPI-Platine mit den Dummy-Modulen
- Simulation einzelner wissenschaftlicher Module in der Testbench
- Test der HEPI als FPGA auf der HEPI-Platine mit einzelnen wissenschaftlichen Modulen
- Simulation der ganzen (alle wissenschaftliche Module) HEPI in der Testbench
- Test der HEPI als FPGA auf der HEPI-Platine mit allen wissenschaftlichen Modulen
- Erstellung der automatischen Tests zur Verifikation von Änderungen im VHDL-Quelltext
- Test der HEPI als Teil des IBIS-Systems. Simulierte Beobachtungen wurden durchgespielt

Abbildung 5.1 auf Seite 46 zeigt die typische Testumgebung in Tübingen.
Linkes Bild: Die beiden Platinen der HEPI im Testrahmen verbunden durch ein Flachbandkabel. Auf der HEPI-Platine ist als größtes Bauteil der FPGA zur Simulation des späteren ASIC zu erkennen. Darunter sind die Bausteine der Ausgangs-FIFOs. Über Testklemmen werde hier gerade einzelne Signal zur Darsterllung im Logicanalyser abgeführt. Der Metallrahmen auf den Schnittstellen ist eine sog. *Breakout-Box*. Hier können die Signale der Schnittstellen ohne Schwierigkeiten abgeführt und die Schnittstellen geschont werden. Die beiden oberen Stecker führen über ein Flachbandkabel zum Detektorsimulator, die beiden unteren zur DPE.
Rechtes Bild: Unter einem anderen Blickwinkel ist auch die DPE zu sehen. Auf der Rückseite der HEPI-Platine führt ein farbiges Flachbandkabel weg. Dies ist der universelle Test-Bus, der direkt an die Pins des FPGA gelötet wurde.

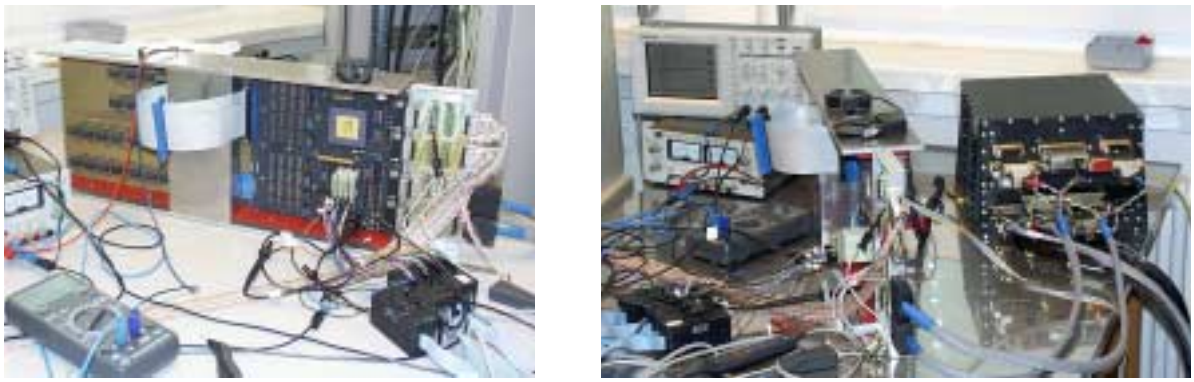


Abbildung 5.1: Typische Testumgebung in Tübingen.

5.1 Simulation

Der Aufwand für die Tests der Hardware ist ungleich größer als Tests durch Simulation. Die DPE muß in einem vorgeschriebenen Ablauf „hoch“ und „runter“ gefahren werden. Alle Arbeiten an der DPE sind zu protokollieren. Hardware-Tests wurden immer von mindestens zwei Entwicklern durchgeführt. Einer steuert die DPE, die anderen untersuchen das Verhalten der HEPI.

Um die Hardware-Tests effizient zu gestalten, wurden alle Änderungen am VHDL-Quelltext erst simuliert. So konnten grobe Fehler und Seiteneffekte erkannt und behoben werden. Dies kann ein Entwickler alleine bewerkstelligen.

5.1.1 Testbench

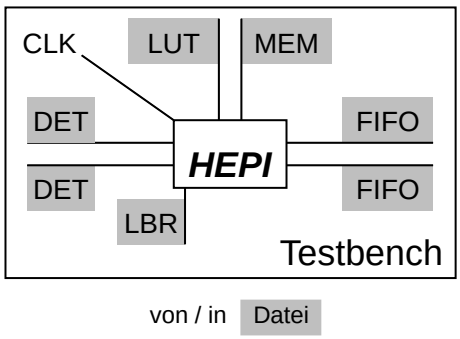


Abbildung 5.2: Die HEPI in der Testbench. Zu jedem Ein- und Ausgang der HEPI gibt es ein Gegenstück in der Testbench, welches die Daten der HEPI aufnimmt, bzw. Signale der HEPI zuführt.

Der VHDL-Quelltext der HEPI besteht aus einer Datei des sog. *Top-Levels*. Alle enthaltenen Module sind in weiteren VHDL-Quelltexten enthalten, auf die sich der Top-Level bezieht. Für die Erzeugung eines Bitfiles für den FPGA wird dieser Top-Level synthetisiert.

Zur Simulation braucht man außer dem Top-Level noch andere Komponenten. Zu jedem Signal welches die HEPI verläßt oder in die HEPI kommt, muß ein Gegenstück in VHDL programmiert werden, welches das Verhalten der echten Hardware möglichst exakt simuliert:

- Die verschiedenen Taktsignale von der DPE
- Die Schnittstelle von den beiden Detektoren. Hier wurde von einer Datei die einzelnen Ereignisse geladen, und zu einem in der Datei angegebenen Zeitpunkt der HEPI zugeführt.
- Der LUT und der MEM-Speicher, beides externe Komponenten auf der HEPI-Platine. Hier kann der Inhalt vor der Simulation geladen werden und nach der Simulation wieder in einer Datei abgespeichert werden.
- Die Kommunikation zwischen der HEPI und der DPE über die LBR-Leitung. Die Befehle werden aus einer Datei gelesen und Antworten der HEPI in eine andere Datei geschrieben.
- Die verarbeiteten Daten der HEPI werden von dem VHDL Gegenstück der FIFO-Bausteine der Platine in eine Datei geschrieben.

All diese zusätzlichen VHDL-Quelltexte werden im Quelltext der Testbench zusammengefaßt und mit dem Quelltext des Top-Levels der HEPI verbunden.

Im Simulator wird die Testbench geladen und simuliert. So kann relativ einfach (editieren von den Einangsdateien (Stimuli)) die HEPI in jede zu testende Situation gebracht werden. Im Simulator können nun Fehlverhalten einzelner Signale zurückverfolgt werden um den Fehler im VHDL-Quelltext zu finden.

5.1.2 Automatische Tests

Für jedes wissenschaftliche Modul der HEPI wurde ein Test erstellt der möglichst die ganzen Funktionen der Module ausnützt. Dazu wurden der Testbench spezielle Stimuli-Dateien und Kommando-Dateien vorgesetzt. Die Ausgabe wurde wiederum in eine Datei gespeichert. Zu Beginn wurden diese Ausgabe-Dateien „von Hand“ nachgerechnet und überprüft. Dies ist leider sehr aufwendig, muß aber nur einmal pro Modul gemacht werden. Waren die Ausgaben der HEPI korrekt, so wurden sie als Soll-Dateien gespeichert. So konnte durch geeignete Unix Skripte der ganze Test automatisch ablaufen. Zum Schluß wurde ein Dateivergleich (diff) der Ausgabe-Datei und der Soll-Datei gemacht. Durch das Ausführen aller einzelner Tests nacheinander (zusammengefaßt durch ein weiteres UNIX Skript), konnte so ein kompletter funktioneller Test der HEPI gemacht werden. Wichtig waren diese Simultaionen in der Entwicklungsphase, denn so konnte ohne viel Aufwand die Richtigkeit von durchgeführten Änderungen bestätigt werden. Auch gab es keine Fehler durch nicht erkannte Seiteneffekte mehr, da immer alle Module zum abschließenden Test einer Änderung getestet wurden.

5.2 Design for Testability

Unter *Design for Testability* versteht man Methoden auf Designebene um das Bauteil möglichst einfach testen zu können. Für die HEPI waben wir verschiedene Ansätze implementiert.

5.2.1 Dummy-Module

Um die einzelnen wissenschaftlichen Module der HEPI einzeln, d. h. ohne daß andere wissenschaftliche Module die Daten verändern, testen zu können, wurde zu Begin der Hardware-Tests von jedem wissenschaftlichen Modul eine sog. Dummy-Version erstellt. Mit solch einem Bitfile

konnte dann z. B. die Funktion der Amplituden-Korrektur des CSI_AC-Moduls und die Speicherzugriffe während der Amplitudenkorrektur überprüft werden, ohne daß andere Module auf den Speicher zugreifen.

Um die Pipeline-Struktur in der HEPI nicht zu verändern, besteht ein Dummy-Modul aus dem Eingangsregister zur Übernahme der Daten zum Pipelinetakt. Damit werden dem nachfolgendem Modul die Daten unverändert übergeben. Eventuelle andere Ausgänge wurden mit einfachen, meist konstanten Werten versehen.

VHDL unterstützt die Implementation von verschiedenen Designs. In einem VHDL-Quelltext können mehrere *architecture*-Strukturen enthalten sein, in unserem Falle also die Implementation des wissenschaftlichen Moduls in einer architecture und die Dummy-Version in einer anderen. Da beide architectures nach außen gleich aussehen (sie benutzen die selbe *port*-Definition), konnte hier Zeit für Fehlersuche gespart werden. Die Auswahl der richtigen architecture erfolgt im darüberliegenden Modul (*Top-Level*) mit einer *configuration* Anweisung.

Der Wechsel zwischen dem Modul und seiner Dummy-Version konnte so ohne großen Aufwand vollzogen werden.

5.2.2 Transparent Mode

Der Datenfluß in der HEPI sieht die Möglichkeit vor, die ganze wissenschaftliche Bearbeitung der Ereignisse in den Modulen zu umgehen, und die Daten des Ereignisses unverändert der DPE zur Verfügung zu stellen.

Die Idee hinter diesem Modus war, bei einer extrem niedrigen Datenrate der Detektoren (wenn also die HEPI zur Datenreduktion nicht gebraucht wird), die Detektordaten mit ihrer genauen Zeitauflösung der Beobachtung bereit zu stellen.

Während der Entwicklung der HEPI haben wir von diesem Modus ausgiebig Gebrauch gemacht, um sicher zustellen, daß unsere Eingangsdaten korrekt sind. Immer wenn eine Version der HEPI ein fehlerhaftes Verhalten zeigte, wurde erst überprüft, ob die Eingangsdaten wie vorgegeben übertragen wurden. Aus diesen Erkenntnissen mußte der Detektorsimulator angepaßt werden.

5.2.3 Universeller Test-Bus

Da man in einen elektronischen Baustein nicht hineinschauen kann wie es z. B. bei Software mit einem Debugger möglich ist oder bei einer Platine mit einem Oszilloskop, wurden über freie Pins des FPGA bei Bedarf interne Werte der HEPI durch einen universellen Test-Bus (*probes*) nach außen sichtbar gemacht.

Dazu mußte der VHDL-Quelltext so angepaßt werden, daß die Signale von Interesse aus ihrer Verschachtelung in der HEPI-Hierarchie nach „oben“ auf das *Top-Level* des Designs geführt wurden und dort mit den *Ports* des Test-Buses nach draußen geführt wurden. Mit dem Logic-Analyser konnten so die Änderungen dieser Werte zusammen mit anderen äußeren Signalen genau beobachtet werden. Besonders die genauen zeitlichen Abläufe bei den seriellen Schnittstellen konnten so relativ komfortabel überprüft werden.

5.2.4 Heart Beat

Zur Überprüfung der groben Funktionalität der HEPI wurde an einer freien Leitung – welche durch die DPE abgefragt werden kann – ein einfaches 1 Hz Signal angeschlossen. Dieses Signal

wird aus dem SYNC Puls von der DPE erzeugt. Dafür wird es über Register gesichert und ändert sich nur dann periodisch, wenn

- die Versorgungsspannung korrekt ist, so daß die HEPI funktionieren kann
- der 4 MHz Systemtakt anliegt (mit ihm wird das Register gesteuert)
- der SYNC Puls von der DPE bei der HEPI ankommt.

So kann auch im Weltraum eine erste Diagnose der HEPI durch die DPE stattfinden.

5.2.5 Scan Chain

Ein wichtiges Instrument um den ASIC zu testen, sind die sog. *Scan Chains*. Dies sind Register, die neben ihrem eigentlichen Eingang einen Test-Eingang (Scan In, SI besitzen. Über ein weiteres Signal (Scan Enable, SE) wird auf den Test-Eingang umgeschaltet. Der Ausgang des Registers wird mit dem Test-Eingang des nächsten Registers verbunden. So wird eine Registerkette erzeugt, die seriell über den SI Eingang geladen werden kann. Siehe Abb. 4.3 auf Seite 39.

Wichtig sind diese Register bei den ASIC-Tests, siehe unten.

5.3 Hardware Test

In diesem Kapitel wird ein Überblick gegeben über die Tests die mit der Hardware durchgeführt wurden. Einige der Tests wurden mit unserer Mithilfe in Mailand und Madrid ausgeführt.

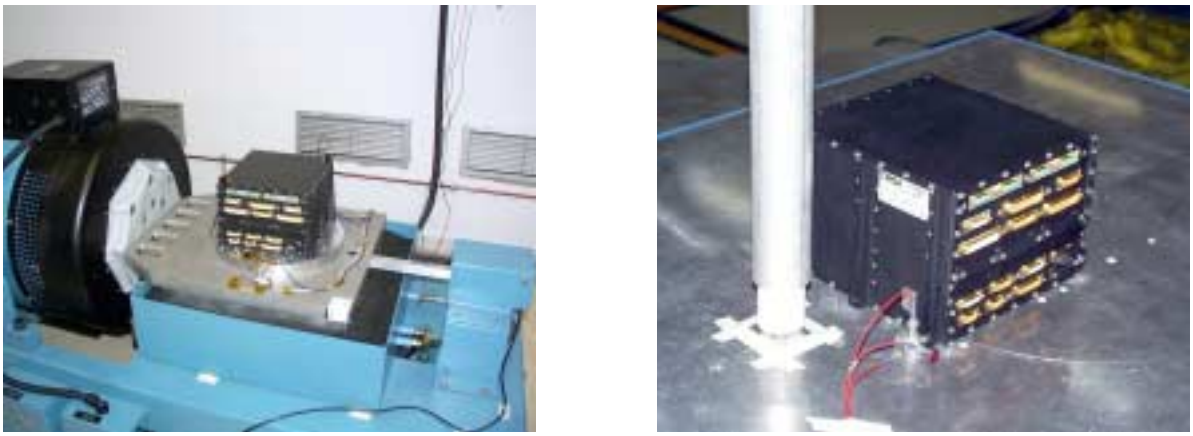


Abbildung 5.3: Die Schock- und Vibrationstests der DPE mit der HEPI wurden in Madrid durchgeführt. **Links:** Die Apparatur zur Erzeugung der Vibrationen ist im Prinzip ein riesiger Lautsprecher. Hier wurde die DPE mit bis zu 20 g geschüttelt. **Rechts:** Durch einen definierten Schlag auf die Montagefläche der DPE wird die Elektronik bis zu 1500 g ausgesetzt.

5.3.1 FPGA

Ein wichtiges Werkzeug für die Hardware-Tests war der FPGA der Firma Xilinx. Ein FPGA (*Field Programmable Gate Array*) ist ähnlich der verwendeten TEMIC Technologie aufgebaut.

Als kleinste Einheit sind sog. *Logic Blocks* vorhanden, die in einer Matrix angeordnet sind. Die gewünschten logischen Funktionen werden erreicht, indem die Ein- und Ausgänge dieser Logic Blocks geeignet miteinander verbunden werden. Auch innerhalb der Logic Blocks können durch Umschalten der Verdrahtung Veränderungen vorgenommen werden [19].

Während die Verdrahtung bei der TEMIC Technologie durch die Metallisierung festgeschrieben wird, gibt es bei einem FPGA ein Netz von Verdrahtungen über den logischen Bauteilen. An den Knotenstellen sind Schalter eingebaut, die wahlweise eine Verbindung herstellen. Diese Schalter sind vergleichbar mit der SRAM Technologie und können jederzeit geändert werden. Die Beschreibung aller Zustände aller Knotenpunkte liefert das sog. *Bitfile*. Nach Laden dieser Daten ist der FPGA in der gewünschten Form konfiguriert [20].

Mit Hilfe eines FPGAs kann so ein Prototyp eines ASIC im System zu einem Bruchteil der Kosten getestet werden. Wir haben davon ausgiebig Gebrauch gemacht und im Laufe der Entwicklung mehr als 100 Versionen der HEPI im FPGA getestet.

Der FPGA liest beim Einschalten automatisch das Bitfile aus einem EEPROM aus, und ist danach einsatzklar (siehe Abb. 5.4).

Abbildung 5.4: EM und FM der HEPI. **Links:** EM Platine der HEPI. Über eine Adapterplatine ist ein EM des HEPI ASIC eingesetzt. Dadurch kann der ASIC getestet werden. Die leere Fassung rechts über dem ASIC enthält normalerweise das EEPROM für den FPGA. **Rechts:** Eine FM Platine der HEPI. Alle Bauteile sind durch qualifizierte Typen ersetzt worden.

5.3.2 Performance- und Dauertests

Die Tests im Simulator und mit der Testbench können immer nur einen kleinen Zeitabschnitt abdecken. Um zu überprüfen, ob die Hardware auch einer Dauerbelastung gewachsen ist, mußte wieder auf die Hardware-Tests zurückgegriffen werden.

Der Detektor-Simulator wurde so geändert, daß er einstellbare Datenraten und Pausen erzeugt. Da die Eingangsdaten definiert waren, mußten auch die Ausgangsdaten der HEPI immer gleich sein. Testläufe über mehrere Tage wurden unternommen um dies zu bestätigen.

Als Ergebnis kann festgehalten werden, daß die HEPI ohne Probleme die maximale Datenrate der Detektoren (Detektoren schicken ununterbrochen Ereignisse) bewältigen kann. Der Engpaß

der Verarbeitung liegt meist an der Verbindung DPE – CDMU/Downlink, die hier an ihre Grenzen stoßen.

Abbildung 5.5: Das Entwicklungslabor in Tübingen. **Links:** Im frühen Stadium der Entwicklung stand noch keine DPE zur Verfügung. Diese wurde durch Simulatoren ersetzt. Die beiden Schaltschränke in der Mitte simulieren den Satelliten nach der HEPI. **Rechts:** Typische Arbeitsumgebung zum Test der HEPI. Von Links: PC mit Detektorsimulator (Hard- und Software), PC zum Brennen und Löschen der EEPROMs, die Platinen der HEPI im Testrahmen, die DPE, Logicanalyser, CDMU Simulator.

5.3.3 Integrations-Tests

Die Tests am Astronomischen Institut Tübingen konnten immer nur mit Simulatoren arbeiten, da die anderen Komponenten – wie Detektoren oder DPE – zu Beginn noch gar nicht existierten, bzw. noch nicht lauffähig waren.

Ein wichtiger Schritt war das Zusammenfügen der „echten“ Hardware. An mehreren Terminen wurden im Reinraum der Firma LABEN in Mailand/Italien die verschiedenen Komponenten von IBIS zusammengetragen und miteinander verbunden (Abb. 2.8 auf Seite 16). Jetzt konnte das ganze IBIS-System als Einheit getestet werden.

5.3.4 ASIC-Test

Ging es bei den bisherigen Hardware-Tests darum die Funktionsweise zu überprüfen, haben die ASIC-Tests einen anderen Anspruch. Hier geht es darum die Fertigung des ASIC zu überprüfen, da diese nicht immer völlig fehlerfreie Bauteile liefert.

Als Faustregel für die Erkennung fehlerhafte Bausteine gilt:

- Es kostet 1 $ ein defekten IC auszutauschen
- Es kostet 10 $ ein defektes IC auf einer Platine zu finden und auzutauschen
- Es kostet 100 $ so eine Platine in einem System zu finden
- Es kostet 1000 $ eine defekte Komponente beim Kunden zu finden

Da der INTEGRAL-Satellit ein Einzelstück ist, *muß* jedes Bauteil vor dem Start auf völlige Funktionalität überprüft werden. Ein Austauschen wie in obiger Faustregel ist nicht möglich.

Das Ziel der Tests ist es, mit einem Minimum an Test-Vektoren jede Fehlermöglichkeit zu überprüfen. Bei der ASIC Produktion geht man von folgenden Fehlermöglichkeiten aus:

- *stuck-at* Fehler machen etwa 90 % aus. Ein Signal kann nicht geschaltet werden, es bleibt an einem Pegel „kleben“
- *stuck-open* Fehler sind in einer Art Memory-Effekt zu erkennen. Zum Erkennen muß der Übergang 0→1 und 1→0 getestet werden
- Kurzschluß, zwei Leitungen haben elektrischen Kontakt. Dieser Fehler macht bis zu 30 % aller Fehler aus
- Lauzeitfehler. Die Verzögerungen auf kritischen Pfaden ist länger als berechnet

Um alle Fehlermöglichkeiten abdecken zu können, gibt es drei Ansatzmöglichkeiten:

- Ausführlicher Test. Alle Kombinationen von Test-Vektoren werden ausprobiert
- Funktionaler Test. Test-Vektoren testen alle Funktionen des Bauteils
- Testen aller möglichen Fehlerquellen. Für jeden möglichen Fehler muß ein Test-Vektor gefunden werden

Als Beispiel dient uns ein 74181, eine Kombination von 4 Bit Addierer, Subtrahierer und ALU mit 16 arithmetischen Funktionen und 16 logischen Funktionen. Der Baustein hat 14 Eingänge. Das Ziel ist eine Fehler-Abdeckung von 100 %:

- Ausführlicher Test braucht 2^{14} Test-Vektoren.
- Funktionaler Test braucht 448 Test-Vektoren. Pro logischer Funktion 8 Test-Vektoren, pro arithmetischer Funktion etwa 20.
- Testen aller erkennbarer Stuck-at Fehlerquellen erfordert 47 Test-Vektoren.

Das Design der HEPI ist ungleich komplexer und die ASIC-Tester von TEMIC haben nur eine Kapazität von 250000 Test-Vektoren. Ausgehend von den funktionalen Simulationen wurde mit Hilfe der Scan-Chain Register versucht die Fehlerabdeckung (*fault-coverage*) zu maximieren.

Die erreichte Fehlerabdeckung bei der HEPI liegt bei 89%.

6 Zusammenfassung, Ausblick

Das Astronomische Institut Tübingen ist das einzige universitäre Institut in Deutschland, welches experimentelle Weltraumastronomie betreibt. So sind in den vergangenen Jahren etliche Diplomarbeiten direkt oder indirekt im All gewesen. Die HEPI ist jetzt mit knapp 10 Mannjahren Entwicklungszeit das umfangreichste Design welches am AIT entwickelt und realisiert wurde.[1]

Als ich dem Projekt beitrat, stand das Konzept der HEPI und die Aufteilung in die verschiedenen Module. Zusammen mit den beiden anderen Hardware-Entwicklern, Bärbel Kretschmar und Nikolai von Krusenstiern, haben wir die Schaltpläne der wissenschaftlichen Module erstellt. Das Histogramm-Modul CSI_HIST war mein Beitrag dazu. Als der Schritt zum VHDL-Design vollzogen wurde, habe ich die HEPI übernommen. Jetzt galt es mit Unterstützung der Firma DD&T den Widrigkeiten und Unzulänglichkeiten der Software zu trotzen und einen reproduzierbaren Weg vom VHDL-Design zum FPGA EEPROM zu finden.

Die anfängliche Abschätzung über die Größe des Xilinx FPGAs erwies sich als zu klein. Auch die neu beschafften FPGAs wurden mit der Zeit (Einbau aller wissenschatlicher Module) zu klein. So mußten wir mit Hilfe der ESA auf ein Vorserien-Modell des größten FPGAs dieser Serie zurückgreifen. Auch hier erwies sich die *leading edge of technology* häufig als *bleeding edge*. Der Mangel des Vorserien-Modells konnte mit einer modifizierten Stromversorgung behoben werden. Mit der Erstellung einer Hardware-Testumgebung konnte hier mit Oszilloskopen und programmierten Logic-Analysern sehr effizient jede neue Version der HEPI getestet werden. Gleichzeitig habe ich die Testbench immer weiter verfeinert und mit den automatischen Tests eine schnelle Überprüfung und Qualifizierung von Änderungen im VHDL Quelltext bereitgestellt. Im Laufe der Zeit habe ich an den meisten Modulen Anpassungen vornehmen müssen um geänderten Anforderungen gerecht zu werden. Durch die vielen Testläufe im Simulator, Testbench und der Hardware konnten so alle[2] Fehler der HEPI gefunden werden.

Bei den verschiedenen Tests in Mailand und Madrid konnten wir Vergleiche ziehen zwischen den Arbeiten der am INTEGRAL-Satelliten beteiligten Industrie und der Arbeit an der HEPI von unserem vergleichsweise kleinen Institut.

Dieser Vergleich läßt mich stolz auf die Arbeit an der HEPI zurückblicken.

[1] Von der Anzahl der Gatter etwa vergleichbar mit einem MC68000 Prozessor

[2] alle bekannten Fehler ...

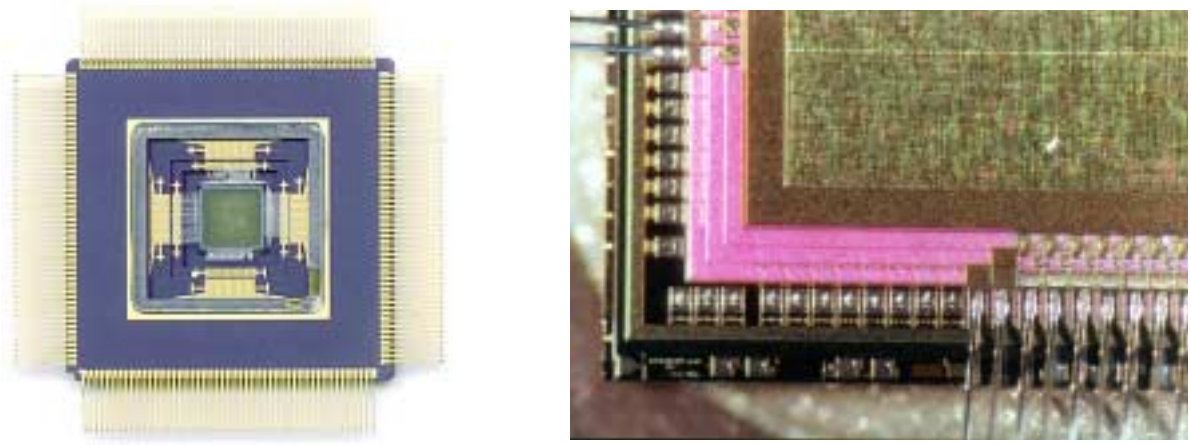


Abbildung 6.1: Der ASIC der HEPI. **Links:** Ein EM der HEPI mit entferntem Deckel. **Rechts:** Eine Nahaufnahme des *die* des ASIC. Deutlich sind die drei Metallisierungsebenen dieser Technologie zu erkennen.

In Zukunft aber werden immer leistungsfähigere strahlungsharte Prozessoren wie z. B. Alpha, Sparc oder PowerPC CPUs die Entwicklung solcher ASICs überflüßig machen. Dann können auch so umfangreiche Aufgaben wie die der HEPI direkt im Prozessor durchgeführt werden.

Die Entwicklung eines vergleichbaren Systems kommt dann mit etwas mehr Software-Entwicklungs-, aber mit deutlich weniger Hardware-Entwicklungsaufwand aus.

Auf der anderen Seite werden die für unsere Tests verwendeten FPGAs immer strahlungshärter. Diese könnten in Zukunft die Rolle eines strahlungsharten ASIC zu einem Bruchteil der Kosten übernehmen. So muß für eine zukünftige Satelliten-Mission wieder genau abgewogen werden, welche Kombination aus Hard- und Software zum Einsatz kommt.

A CSI_AC

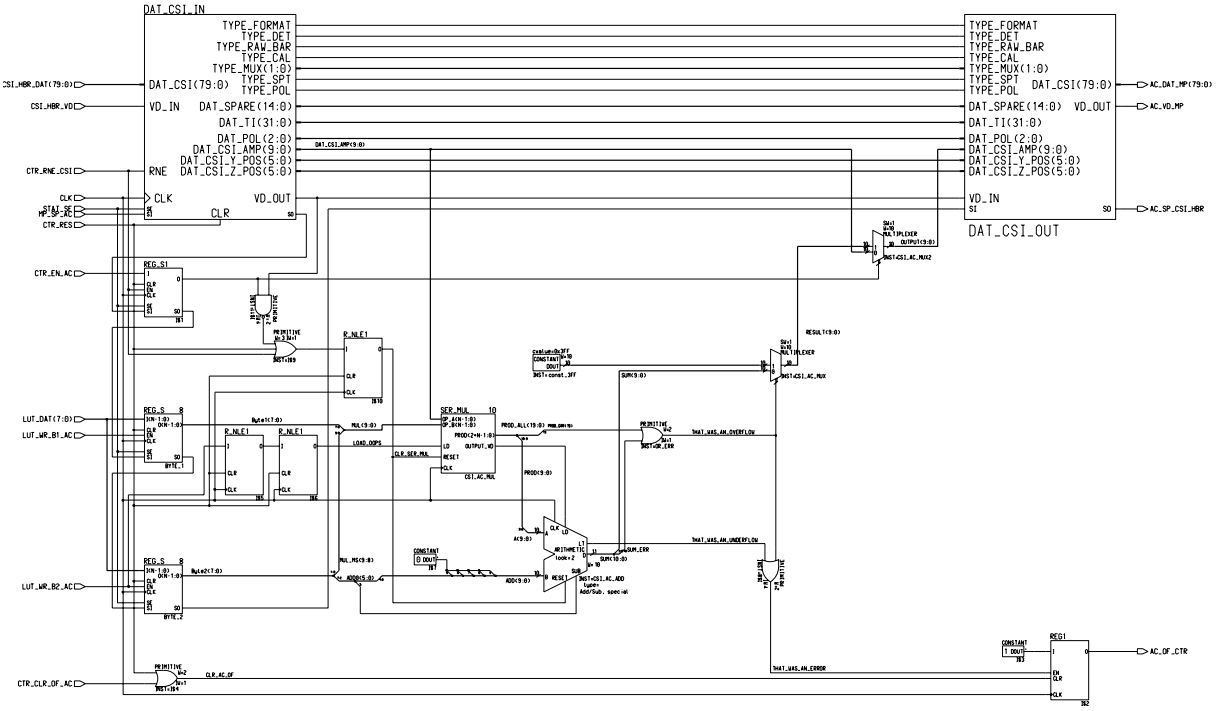


Abbildung A.1: Schaltplan vom CSI_AC Modul

Eingänge:

CLK	4 MHz Systemtakt
CTR_RES	Reset-Signal vom CMD_CTRL
CTR_EN_AC	Enable-Signal vom CMD_CTRL
CTR_RNE_CSI	CsI-Pipeline Signal, fehlt bei Pipelinestop
CSI_HBR_DAT(79:0)	80 Bit CsI-Ereignis
CSI_HBR_VD	dazugehöriges ValidData
CTR_CLR_OF_AC	Lösch-Signal für das ErrorRegister (Over-/Underflow)
LUT_DAT	Tristate-Bus vom LUT
LUT_WR_B1_AC	Schreib-Signal für BYTE_1 vom LUT Bus
LUT_WR_B2_AC	Schreib-Signal für BYTE_2 vom LUT Bus

Ausgänge:

AC_DAT_MP(79:0)	80 Bit CsI-Ereignis
AC_VD_MP	dazugehöriges ValidData
AC_OF_CTR	Fehler-Signal (Over-/Underflow) zum CMD_CTRL

A.1 Ablauf einer Amplitudenkorrektur

Damit in der HEPI mit den korrekten Pixelamplituden gerechnet werden kann, ist der AC das erste Modul nach dem CsI-HBR. Das CsI-HBR liefert mit gültigen Daten (VD) an den AC gleichzeitig die Pixel-Adresse an den LUT_CTRL. Am Anfang dieses RNE (Read Next Event) holt der LUT_CTRL nacheinander die beiden Bytes aus dem LUT und übergibt sie dem AC. Mit einer Verzögerung von 2 Takten nach dem Übergeben des zweiten Bytes beginnt der serielle Multiplizierer[1] zu arbeiten und ist nach 11 Takten fertig (OUTPUT_VD). Damit wird der Add/Sub gestartet der nach weiteren 4 Takten fertig ist. Mit dem 4. Takt erscheinen auch die Over-/Underflow Flags, die, verODERt, in einem Register als AC_OF_CTRL gehalten werden. Im Falle eines Overflows wird statt der fehlerhaften/unvollständigen Amplitude über einen Multiplexer der theoretische Maximalwert $3FF ausgegeben. Im Falle eines Underflows wird vom Add/Sub ein $000 ausgegeben.

A.2 Details

Mit jedem RNE (CTR_RNE_CSI) werden die Eingangsdaten in Registern übernommen. Das betrifft das Einschalten des ACs (CTR_EN_AC), sowie auch die 80 Bit Csi Daten (CSI_HBR_DAT(79:0)). Dieses CTR_RNE_CSI dient gleichzeitig dazu eventuelle bestehende (alte) Over-/Underflow-Bedingungen zu löschen. Dazu werden die RESET-Eingänge des Multiplizierers (SER_MUL) und des ADD/SUBs (CSI_AC_ADD) über das Signal CLR_SER_MUL miteinander gelöscht. Es ist:

CLR_SER_MUL <= CTR_RES or not (local_VD_OUT and local_CTR_EN_AC) or CTR_RNE_CSI;

Die beiden local_* Signale sind die Eingangssignale hinter den Registern. Über den LUT_DAT(7:0) Bus kommen die beiden Bytes vom LUT zum AC. Diese enthalten den Multiplikant (Gain) und den Addant (Offset). Das Signal zur Übernahme ist LUT_WR_B1_AC für das erste Byte und LUT_WR_B2_AC für das zweite Byte. Die Write Signale kommen zum Takt 17 und 19.

Die Bytes enthalten:

Byte 1: (PixelAdd. * 2):

Bit	7	6	5	4	3	2	1	0
	G7	G6	G5	G4	G3	G2	G1	G0

[1] Als Kapazitätsprobleme im FPGA auftraten wurde dieser Multiplizierer von einer Parallelen in eine (kleine) serielle Variante umgebaut. Alle Tests wurden mit dem seriellen Multiplizierer gemacht. Bei der Umsetzung in den ASIC wurde aber wieder auf einen parallelen Multiplizierer zurückgegriffen. Dies führt zu unerwünschten „Seiteneffekten“

Byte 2: (PixelAdd. * 2 + 1):

Bit	7	6	5	4	3	2	1	0
	O5	O4	O3	O2	O1	O0	G9	G8

mit G = Gain, O = Offset, O5 = SignBit

Die Wertigkeit ist:
G9 = 2^0, G8 = 2^{-1}, G7 = 2^{-2}, ...
O = Unsigned: 0 = -0: %000000 = %100000

Um sicher zu gehen, daß die Daten stabil anliegen, wird zum Starten des seriellen Multiplizierers das mit 2 Registern um 2 Takte verzögerte LUT_WR_B2_AC Signal verwendet.

A.2.1 SER_MUL

Eingänge:

OP_A(9:0)	Operant A
OP_B(9:0)	Operant B
LD	Übernahme-/Startsignal
RESET	Rücksetzsignal
CLK	Systemtakt

Ausgänge:

PROD(10:0)	das fertige Produkt
OUTPUT_VD	Gültigkeitssignal

Der Multipliziervorgang wird durch einen Puls an LD gestartet. Dort ist das um 2 Takte verzögerte Schreibsignal des 2. Bytes. Allerdings läuft der Multiplizierer auch nach einer fallenden Flanke auf dem RESET Eingang an. Aber ein OUTPUT_VD wird nur erzeugt, wenn vorher ein LD kam. Nach 11 Takten steht das Ergebnis am Ausgang. Nach 12 Takten steht das OUTPUT_VD am Ausgang. Der nachfolgende Addierer hat also 1 Takt Vorsprung bei der Berechnung der Daten.

Von den 20 Ausgangsbits werden nur 11 weiterverwendet: Das oberste (19) geht als PROD_ERR verODERt mit dem Overflow des Addierers (SUM_ERR) als THAT_WAS_AN_OVERFLOW sowohl auf den SEL Eingang des Multiplexers als auch verODERt mit dem THAT_WAS_AN_UNDERFLOW als THAT_WAS_AN_ERROR auf den Eingang des Error-Registers. Der Multiplexer schaltet um zwischen dem korrekten Ergebnis aus dem CSI_AC_ADD und der festen Konstanten $3FF (max. darstellbare Amplitude).

Errata

Nomenklatur:
„Error“: Signal welches im AC erzeugt wird.
„Fehler“: fehlerhaftes Verhalten des verwendeten Multiplizierers.

Fehlerbeschreibung:
CSI_AC zeigt einen Fehler an CSI_AC_OF_CTRL, obwohl mit den verwendeten Amplituden und den zu den Pixel gehörenden LUT-Werten kein Overflow im Multiplizierer und kein Overflow/Underflow im Addierer erzeugt werden kann.
Fehleranalyse:
Der Fehler ist ein Overflow des Multiplizierers. Bei der Entwicklung der HEPI wurde (aus *Area* Gründen im FPGA) auf einen „seriellen“ Multiplizierer zurückgegriffen. Dieser ist relativ klein (Area), benötigt aber für jedes Ergebnis-Bit einen Takt. Zusätzlich muß dieser serielle Multiplizierer über ein Signal gestartet werden, um von da mit den Eingangswerten die Multiplikation vorzunehmen. Das Endergebnis ist dann nach n Takten stabil und fluktuiert nicht. Deshalb konnte für das Erkennen eines Overflows direkt auf das oberste Bit dieses Ergebnisses zurückgegriffen werden, ohne besondere Maßnahmen bezüglich der Gültigkeit zu treffen.
Da der eine Operand des Multiplizierers ein 10 Bit Wert aus der LUT ist, dieser aber über die beiden Bytes aus der LUT verteilt ist, wird im CSI_AC der Multiplizierer erst dann gestartet wenn beide Bytes aus der LUT eingelesen wurden und somit stabil in zwei internen Registern der CSI_AC liegen.
Für den ASIC wurde dann der serielle Multiplizierer durch einen parallelen (schnell aber groß) ersetzt. Dieser erzeugt (von internen Durchlaufzeiten (wenige Gatter) einmal abgesehen) direkt aus seinen beiden Operanten ein Ergebnis. Das immer noch vorhandene (aus Kompatibilitätsgründen) Start-Signal des (seriellen) Multiplizierers wird nur über 2 Register verzögert als Valid-Signal weitergegeben.

In der Zeit zwischen dem Einlesen des ersten Wertes der LUT und des zweiten Wertes der LUT kann jetzt eine Kombination anliegen, welche eine Multiplikation mit einem Wert deutlich über 1,000 erzeugt, die dann auch sofort durchgeführt wird und als Ergebnis das Overflow-Signal erzeugt, welches dann von außen sichtbar im Error-Register gespeichert wird.

Beispiel:
Ein Ereignis mit Adresse A hat in der LUT die Werte $00 $02 stehen (Addition mit 0,0000 und Multiplikation mit 1,0000). Das nächste Ereignis mit der Adresse B hat in der LUT die Werte $55 $55 (Addition mit 21 und Multiplikation mit 0,50505) stehen. In der kritischen Zeit zwischen den beiden LUT-Werten sind die Werte im CSI_AC aber $55 $02 (Addition mit 0, Multiplikation mit 1,0505). Ist der Amplituden-Wert des Ereignis B entsprechend groß, wird durch diese Multiplikation ein Overflow erzeugt! Dieser verschwindet vom Ausgang des Multiplizierers, wenn das zweite Byte aus der LUT übernommen wird. Zu diesem Zeitpunkt ist ein Error schon erkannt worden (Error Register).

A.2.2 CSI_AC_ADD

Eingänge:

A(9:0)	Operant A
B(9:0)	Operant B
SUB	Moduswechsel: Addierer/Subtrahierer
LD	Übernahme-/Startsignal
RESET	Rücksetzsignal
CLK	Systemtakt

Ausgänge:

D(10:0)	fertige Summe
LT	Negativsignal, „less then“

Zu Beginn der Entwicklung gab es an dieser Stelle nur einen Addierer, weil der Offset nur positive Werte annehmen konnte. Später hat man entschieden, daß hier auch subtrahiert werden soll. So ist aus einem einfachen Addierer ein etwas komplexeres Bauteil geworden. Das Vorzeichen-Bit ist das MSBit vom zweiten Byte. Damit bei der durchlaufenden Berechnung, bzw. das Anliegen von Eingang B (aus Byte 2) und Warten auf Eingang A (aus SER_MUL) keine noch-nicht-gültigen Over-/UnderFlows entstehen können und vom Error Register als „echte“ Fehler erkannt werden, werden die Ausgänge LT und D(10) erst 5 Takte nach dem LD freigeschaltet. Das eigentliche Addieren ist nicht geclocked (wohl aber der Output), und geht somit in einem Bruchteil dieser Zeit (ripple-through). Zurückgesetzt wird mit dem obigen CLR_SER_MUL. So wird verhindert, daß unbeabsichtigte Over-/UnderFlow Bedingungen zur falschen Zeit auftreten können. Falls es beim Subtrahieren zu einem Underflow (negatives Ergebnis) kommen sollte, wird der Ausgang auf $000 gesetzt und das LT (Less Than) Flag gesetzt Das LT geht als THAT_WAS_AN_UNDERFLOW verODERt mit dem THAT_WAS_AN_OVERFLOW in das Error-Register.

Der Ausgang (AC_OF_CTR) dieses Error-Registers ist direkt beim Auslesen der *essenial HK* zu sehen, er wird nicht im CMD_CTRL (oder sonstwo) zwischengespeichert. Gelöscht wird dieses Register über den modulweiten CTR_RES vom CMD_CTRL ODER direkt vom CMD_CTRL über das CTR_CLR_OF_AC (zusammen ergeben sie das CLR_AC_OF Signal).

Sollten die Daten es bis hier her geschafft haben, gehen sie noch durch einen Multiplexer, bis sie das rettende DAT_CSI_OUT erreichen. Dieser Multiplexer wird durch das (registerte) local_CTR_EN_AC geschaltet. Ist der AC nicht an, so gehen die Amplituden unverändert durch.

B CSI_MP

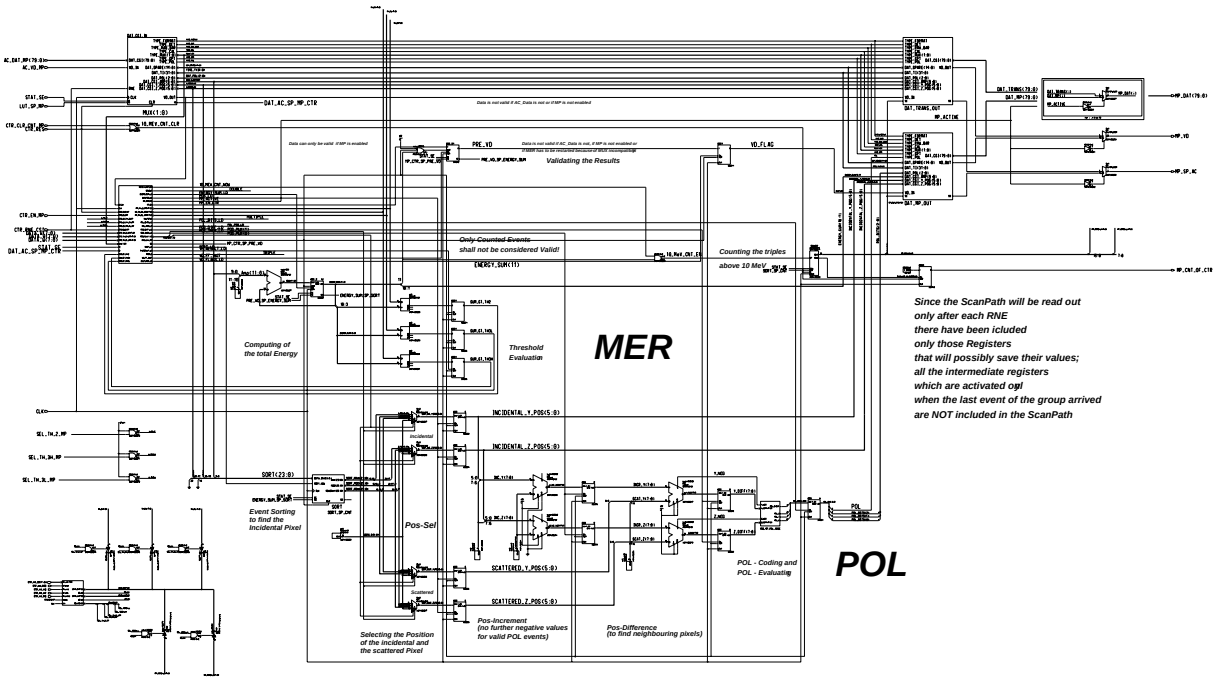


Abbildung B.1: Schema von MP, Multiple Event Reconstruction & Polarimetry

Eingänge:

CLK	4 MHz Systemtakt
CTR_RES_MP	Reset-Signal vom CMD_CTRL
CTR_EN_MP	Enble-Signal vom CMD_CTRL
CTR_RNE_CSI	Pipeline-Takt, fehlt bei Pipeline-Stop
AC_DAT_MP(79:0)	80 Bit Detektordaten von Modul CSI_AC
AC_VD_MP	das dazugehörige Gültigkeitsbit (Valid Data)
CTR_CLR_CNT_MP	Lösch-Signal für den 10_MEV-Zähler
CTR_TH_2_MP(7:0)	8 Bit Schwellenwert für Zweifach-Ereignis
CTR_LD_TH_2_MP	das dazugehörige Gültigkeitsbit (Valid Data)
CTR_TH_3L_MP(7:0)	8 Bit unterer Schwellenwert für Dreifach-Ereignis
CTR_LD_TH_3L_MP	das dazugehörige Gültigkeitsbit (Valid Data)
CTR_TH_3H_MP(7:0)	8 Bit oberer Schwellenwert für Dreifach-Ereignis
CTR_LD_TH_3H_MP	das dazugehörige Gültigkeitsbit (Valid Data)

Ausgänge:

MP_DAT(79:0)	80 Bit Detektordaten
MP_VD	das dazugehörige Gültigkeitsbit (Valid Data)
MP_EN_CTR	Enable-Rückmeldung an den CMD_CTRL
MP_CNT_CTR(15:0)	16 Bit 10 MeV Zähler
MP_CNT_OF_CTR	Fehler-Signal dazu (Overflow) zum CMD_CTRL
MP_TH_2_CTR(7:0)	8 Bit geladener Schwellenwert für Zweifach-Ereignis
MP_TH_3L_CTR(7:0)	8 Bit unterer Schwellenwert für Dreifach-Ereignis
MP_TH_3H_CTR(7:0)	8 Bit oberer Schwellenwert für Dreifach-Ereignis

Innerhalb der HEPI wird die Multiplizität in den 2 Bit MUX(1:0) des Typ-Feldes in folgender Codierung angegeben:

MUX(1)	MUX(0)	Bedeutung
0	0	verboten
0	1	Einfach-Ereignis
1	0	Zweifach-Ereignis
1	1	Dreifach-Ereignis

Dies hat den Vorteil, daß ein Bit (MUX(1)) sofort angibt, ob es sich um ein Mehrfach-Ereignis handelt oder nicht.

B.1 Die eigentliche Rekonstruktion (MER)

B.1.1 Einfach-Ereignisse, die sogenannten Singles

Hier ist keinerlei Rekonstruktion notwendig. Die Eingangswerte werden daher unverändert als Ausgangswerte übernommen (MP_ACTIVE <= ´0´).

B.1.2 Mehrfach-Ereignisse, die sogenannten Multiples

Bei allen Mehrfach-Ereignissen werden im Prinzip die oben erwähnten weiteren Stellen benötigt. Um einerseits einen möglichst kleinen Verlust zu haben, andererseits aber alle Multiples gleich zu behandeln, wurde folgende Lösung angewendet: Die Energie-Auflösung wird um einen Faktor 2 verschlechtert, indem als Gesamtenergie ENERGY_SUM(10:1) weitergereicht wird – natürlich zusammen mit der Multiplizität; die Ereignisse, deren Gesamtenergie tatsächlich eine weitere Stelle benötigen (ENERGY_SUM(11), nur bei Dreifachen möglich), werden in einem getrennten Zähler gezählt: 10_MEV_CNT, da diese Ereignisse eine Gesamtenergie haben, die mehr als 10 MeV entspricht.

Für die Ermittlung der Einfalls-Position werden Amplitude und Position jedes Teilereignisses in einen Sortierer geladen, der die Ereignisse nach der Amplituden-Höhe sortiert. Auf diese Weise kann nach dem Einlesen des letzten Teil-Ereignisses sehr leicht auf die Position dessen zugegriffen werden, das die minimale, mittlere oder maximale Amplitude enthalten hat.

- Dreifach-Ereignisse, die sogenannten Triples:
  Die Monte-Carlo-Erkenntnisse können direkt umgesetzt werden:
  Die Einfallsposition steht in

SORT_MIN	falls	Summe $\leq$ TH_3L
SORT_MID	falls	TH_3L $<$ Summe $\leq$ TH_3H
SORT_MAX	falls	Summe $>$ TH_3H

Das Ergebnis ist allerdings nicht gültig (VD = ´0´), sollte ENERGY_SUM(11) anzeigen, daß das Ereignis nur gezählt werden soll, weil seine Gesamtenergie den Wertebereich der 10 Bit übersteigt.

- Zweifach-Ereignisse, die sogenannten Doubles:
  Die Rekonstruktion der Doppel-Ereignisse ist am einfachsten auf der Basis derjenigen der Triples zu verstehen, wenn man sich vorstellt, die beiden TH_3-Schwellen rücken zu einer TH_2-Schwelle zusammen. Berücksichtigt man nun noch, daß der Sortierer seine Plätze von Minimal nach Maximal auffüllt, daß also bei nur zwei Ereignissen dasjenige mit der höchsten Amplitude am Platz von SORT_MID steht und SORT_MAX leer ist, so befindet sich die Position des Einfallspixels in

SORT_MIN	falls	TH_2 $<$ Summe
SORT_MID	falls	Summe $>$ TH_2

B.2 Die Polarimetrie (POL)

B.2.1 Sind zwei beteiligte Pixel benachbart?

Nachdem sowohl Einfalls- als auch Streu-Pixel feststehen, wird überprüft, ob es sich dabei um Nachbarpixel handelt, da nur bei diesen wirklich von Streuung ausgegangen werden kann. Damit man bei der Codierung echter Pol-Ereignisse später nicht mit negativen Werten arbeiten muß, wird sowohl der Y-Wert als auch der Z-Wert der Einfallsposition um 1 erhöht (INCR = Increment; INC = Incidental):

INCR_Y(7:0) <= INC_Y(7:0) + 1
INCR_Z(7:0) <= INC_Z(7:0) + 1

Nun wird die Position des gestreuten Pixels abgezogen
(DIFF = Difference; SCAT = Scattered):

DIFF_Y(7:0) <= INCR_Y(7:0) - SCAT_Y(7:0)
DIFF_Z(7:0) <= INCR_Y(7:0) - SCAT_Z(7:0)

Jetzt sind die Voraussetzungen geschaffen, zu überprüfen, ob es sich um zwei benachbarte Pixel handelt oder nicht. Sind die beteiligten Pixel tatsächlich benachbart, so muß das Ergebnis zwischen 0 und 2 liegen; ist es größer oder negativ, so sind die Pixel zu weit auseinander
(NEG = Negativ):

Y_DIFF_OK falls DIFF_Y(7:0) $<$ 3
Z_DIFF_OK falls DIFF_Z(7:0) $<$ 3
NEIGHBOUR <= Y_DIFF_OK and Z_DIFF_OK and not Y_NEG and not Z_NEG

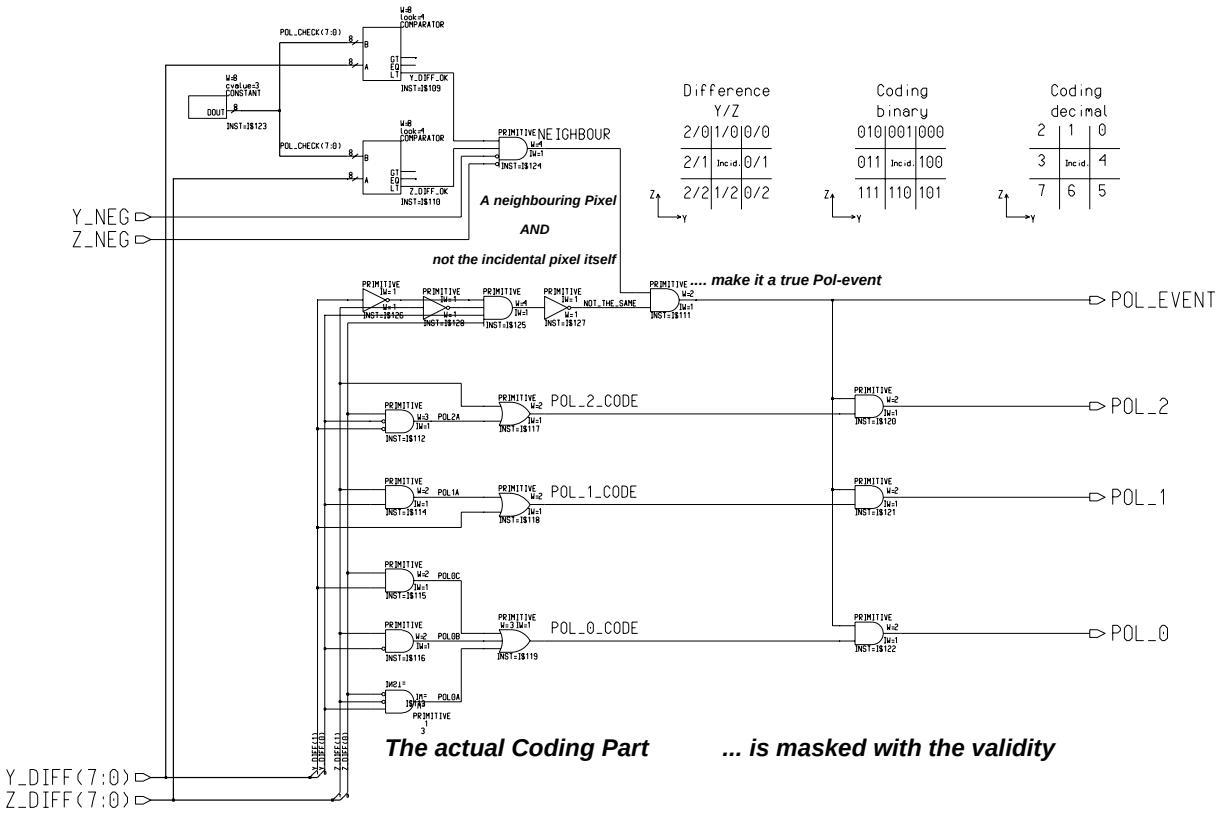


Abbildung B.2: Schaltplan von CSI_MP_POL_CODE

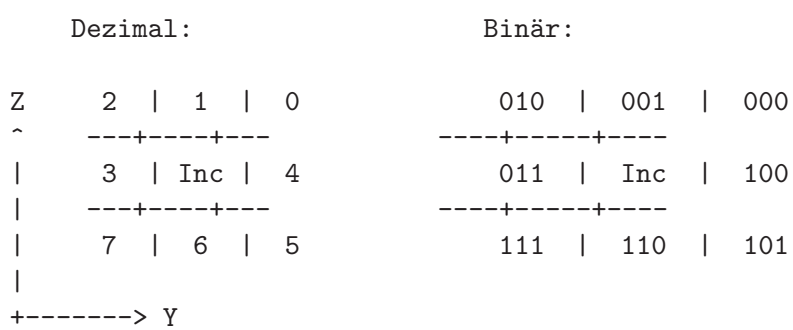
Die Codierung der Pol-Ereignisse

Zur Veranschaulichung der beiden DIFF-Ergebnisse, sollen sie zunächst grafisch auf die 9 Pixel übertragen werden, die an einem Pol-Ereignis beteiligt sein können. Die Beschriftung der einzelnen Pixel soll dabei [Y_DIFF/Z_DIFF] sein:

```
     Dezimal:                  Binär:

Z    2/0 | 1/0 | 0/0           10/00 | 01/00 | 00/00
^    ---+----+---           ----+-----+----
|    2/1 | 1/1 | 0/1           10/01 | 01/01 | 00/01
|    ---+----+---           ----+-----+----
|    2/2 | 1/2 | 0/2           10/10 | 01/10 | 00/10
|
+-------> Y
```

Als nächstes muß eine möglichst einfache Codierung dieser Grafik gefunden werden. Nachdem die Position des zentralen Einfallspixels vollständig in den Positionsbits INCIDENTAL_Y_POS(5:0) und INCIDENTAL_Z_POS(5:0) übertragen wird, können die verbleibenden 8 möglichen Streu-Pixel in 3 Bit codiert werden. Um möglichst nahe an der binären Darstellung der Differenzen zu bleiben, wurde folgende Codierung gewählt:



Außer diesen POL_BITS(2:0), die das Streupixel codieren, gibt es noch das Signal POL (oder auch TYPE_POL), welches das Zweifach-Ereignis als echtes Pol-Ereignis kennzeichnet:

POL <= NEIGHBOUR and not (Einfall-Pixel = Streu-Pixel), also
POL <= NEIGHBOUR and not (Z(1) and Y(1) and not (Z(0)) and not (Y(0)))

Dieses Signal maskiert alle POL_BITS, deren Gleichungen sich mittels KV-Diagrammen wie folgt ergeben haben:

POL_BITS(0) <= POL and (not (Y(0)) Z(1) or Y(1)Z(0) or Y(0) not (Z(1)) not (Z(0)))
POL_BITS(1) <= POL and (Y(1) or Y(0)Z(1))
POL_BITS(2) <= POL and (Z(1) or not (Y(1)) not (Y(0)) Z(0))

Einfach- und Mehrfach-Ereignisse: MP_ACTIVE

In Abschnitt B.1.1 wurde bereits erwähnt, daß Singles einfach unverändert von AC_DAT_MP(79:0) nach MP_DAT(79:0) durchgereicht werden sollen. Ähnliches gilt auch für den Fall, daß MP als Modul nicht eingeschaltet wurde – hier wird nur sichergestellt, daß die Daten als ungültig weitergegeben werden (siehe nächsten Abschnitt B.2.1), aber ansonsten ebenfalls unangetastet bleiben.

Dies legt die Lösung nahe, einen Multiplexer zu verwenden, der entweder die im Großen und Ganzen unveränderten Daten als MP_DAT(79:0) weiterreicht (MP ausgeschaltet oder Single), oder die in MP neu ermittelten Daten (Multiples bei eingeschaltetem MP) auf den Ausgang schaltet. Das Signal MP_ACTIVE schaltet diesen Multiplexer, und berechnet sich daher als

MP_ACTIVE <= MP_EN_CTR and MUX(1)

MP_ACTIVE hat als weitere Aufgabe im „inaktiven“ Fall für eine Initialisierung mit Null zu sorgen, sowohl des gesamten Operationswerks, in dem die in den Abschnitten B.1 und B.2.1 erklärte Funktionalität in Hardware umgesetzt ist, als auch des in Abschnitt B.2.2 beschriebenen Herzstückes des Steuerwerks.

Gültige Daten: Das Valid-Flag VD

So wie mit jedem Pipeline-Zyklus nicht nur Ereignisdaten, sondern auch die Information über ihre Gültigkeit nach MP geladen werden (AC_DAT_MP(79:0) und AC_VD_MP), so wird auch beides mit jedem RNE_CSI von MP an die nachfolgenden Module weitergegeben (MP_DAT(79:0) und MP_VD). An gültige MP-Daten (MP_VD = ´1´) werden zwei Anforderungen gestellt:

1. Das Modul als ganzes muß vom zentralen Steuerwerk (CMD_CTRL) auch eingeschaltet worden sein (CTR_EN_MP = ´1´), und

2. Keines der zur Rekonstruktion verwendeten Teilereignisse war ungültig.

Wie in Abschnitt B.2.1 bereits ausgeführt wurde, werden Singles und Multiples unterschiedlich behandelt. Dies gilt auch hier wieder:

1. Einfach-Ereignisse (Singles):
   Beide Punkte werden dadurch erfüllt, daß das Eingangs-VD-Signal mit dem Enable-Signal maskiert wird:

   $$\mathsf{MP\_VD} <= \mathsf{AC\_VD\_MP} \text{ and } \mathsf{MP\_EN\_CTR}$$

2. Mehrfach-Ereignisse (Multiples):
   In diesen Fällen wird mit 2 Flip-Flops gearbeitet: Das Erste wird am Anfang jeder Rekonstruktion mit MP_EN_CTR initialisiert und wird von jedem ungültigen Ereignis auf ´0´ gesetzt; Das Zweite wird am Anfang der Rekonstruktion auf ´0´ gesetzt und übernimmt am Ende jeder Rekonstruktion den Wert des ersten Flip-Flops und damit die dort entstandene Gesamtgültigkeit.

B.2.2 Das Steuerwerk CSI_MP_CTR

Das für MP entwickelte Steuerwerk hat vor allem die Aufgabe, dafür zu sorgen, daß nur solche Teilereignisse zusammegefaßt werden, die auch tatsächlich – oder zumindest mit sehr hoher Wahrscheinlichkeit – zusammengehören. Daneben steuert es den Ablauf der in Abschnitt B.1 beschriebenen Funktionalität und speichert die vom Zentral-Steuerwerk gelieferten Parameter (Enable und Schwellenwerte).

Das Herzstück des Steuerwerks

Die größte Schwierigkeit bei der Entwicklung von MP stellte die Tatsache dar, daß dies das einzige Modul ist, das seine Ergebnisse (per se)

1. nicht innerhalb eines einzigen Pipeline-Takt (RNE_CSI-Periode) ermitteln kann und

2. die über mehrere Pipeline-Takte verteilten Daten auf ihre Gleichartigkeit bezüglich der Multiplizität (MUX(1:0)) überprüfen muß.

Es mußte also eine Steuerung entwickelt werden, die

1. die zeitlich geordneten Eingangsdaten zu Gruppen aus einem (Singles), zwei (Doubles) oder drei (Triples) gleichartigen (Teil-) Ereignissen unterteilt;

2. den Anfang einer solchen Gruppe erkennt und das gesamte Operationswerk initialisiert;

3. das letzte Teilereignis einer solchen Gruppe erkennt, da die oben vorgestellten Rekonstruktions- und Polarimetrie-Algorithmen erst dann korrekt arbeiten, wenn auch alle Teilereignisse zur Verfügung stehen;

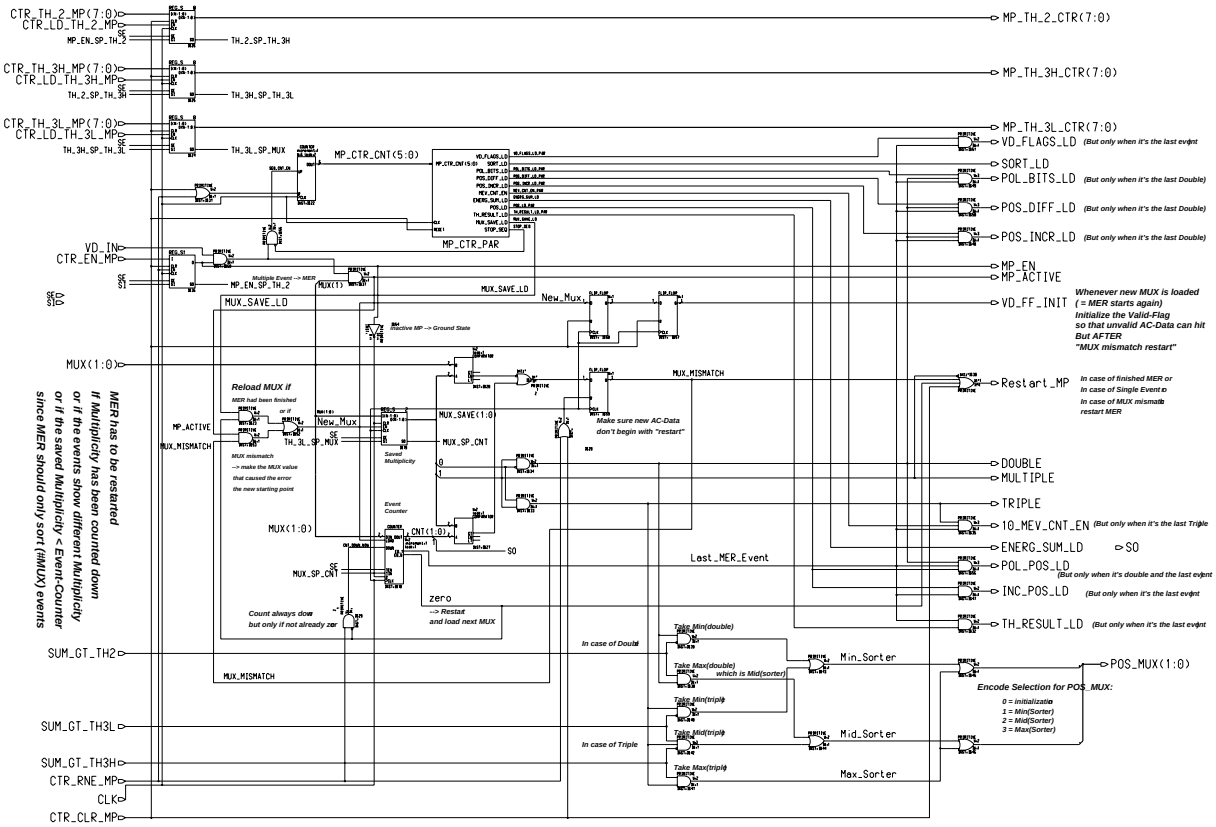


Abbildung B.3: Schaltplan von CSI_MP_CTR

4. eine unvollständige Gruppe erkennt – sprich bemerkt, wenn die neu geladenen Eingangsdaten in ihrer Multiplizität nicht zu der aktuellen, noch unvollständigen Gruppe gehören – und entsprechende Maßnahmen ergreift:

   a) Alle zu dieser unvollständigen Gruppe gehörenden Daten werden gelöscht

   b) Das Ereignis, das die Unvollständigkeit offensichtlich gemacht hat, wird als Anfang einer neue Gruppe genommen, da sich – bei Fehlen nur eines Teilereignisses! – sonst Ketten unvollständiger Gruppen bilden könnten.

Diese Anforderungen konnten alle durch die Verwendung im Wesentlichen eines 2 Bit Registers, eines 2 Bit Zählers und Komparatoren erfüllt werden: Zähler und Register laden beide am Anfang eines Rekonstruktions-Zyklus die Multiplizität MUX(1:0). Während der Zähler seinen Wert jedoch mit jedem neuen Ereignis (RNE) um 1 verringert, behält das Register seine Daten, um sie mit der Multiplizität der neuen Daten zu vergleichen. Auf diese Weise legt das Register die Gruppencharakteristik fest (Double oder Triple – bei Singles ist MP ja „inaktiv“ (Abschnitt B.2.1: MP_ACTIVE)), während der Zähler Anfang und Ende dieser Gruppe definiert: Ende, wenn der Zähler auf 1 steht, und Anfang, wenn er mit dem Einlesen des nächsten Ereignisses (oder auch durch MP_ACTIVE = ´0´) auf Null gesetzt wurde. Treten Unstimmigkeiten (MUX_MISMATCH) zwischen diesen 3 Werten CNT(1:0) des Zählerstandes, MUX_SAVE(1:0) des Resgisters und MUX(1:0) der neuen Eingangsdaten auf, so ist das eine der Bedingungen, mittels RESTART_MP alle Daten des Operationswerks zu löschen.

Weitere Situationen, die einen RESTART_MP erfordern, sind

- Das Zentral-Steuerwerk verlangt es mittels CTR_CLR_MP, das übrigens auch alle sonst unberührten Daten wie Schwellenwerte oder MP_Enable löscht;
- Eine Rekonstruktion ist nicht erforderlich (MP_ACTIVE = ´0´) – das Modul ist also nicht eingeschaltet (MP_EN = ´0´) oder das eingelesene Ereignis ist kein Mehrfachereignis (MUX(1) = ´0´);
- Der Zähler hat mit dem Einlesen des ersten Ereignisses einer neuen Gruppe auf Null herruntergezählt (ZERO), und zeigt damit an, daß jetzt, direkt im Anschluss an das Einlesen der Daten nach MP, ein neuer Rekonstruktions-Zyklus gestartet werden muß.

Insgesamt gilt:

MUX_MISMATCH	<= (MUX(1:0) = not(MUX_SAVE(1:0)) or (CNT(1:0) > MUX_SAVE(1:0)))
RESTART_MP	<= MUX_MISMATCH or ZERO or not MP_ACTIVE or CTR_CLR_MP
CNT_DOWN_NOW	<= RNE and not ZERO
Last_MER_Event	<= CNT[1], (CNT(1:0) = ´01´)

Die Ablaufsteuerung

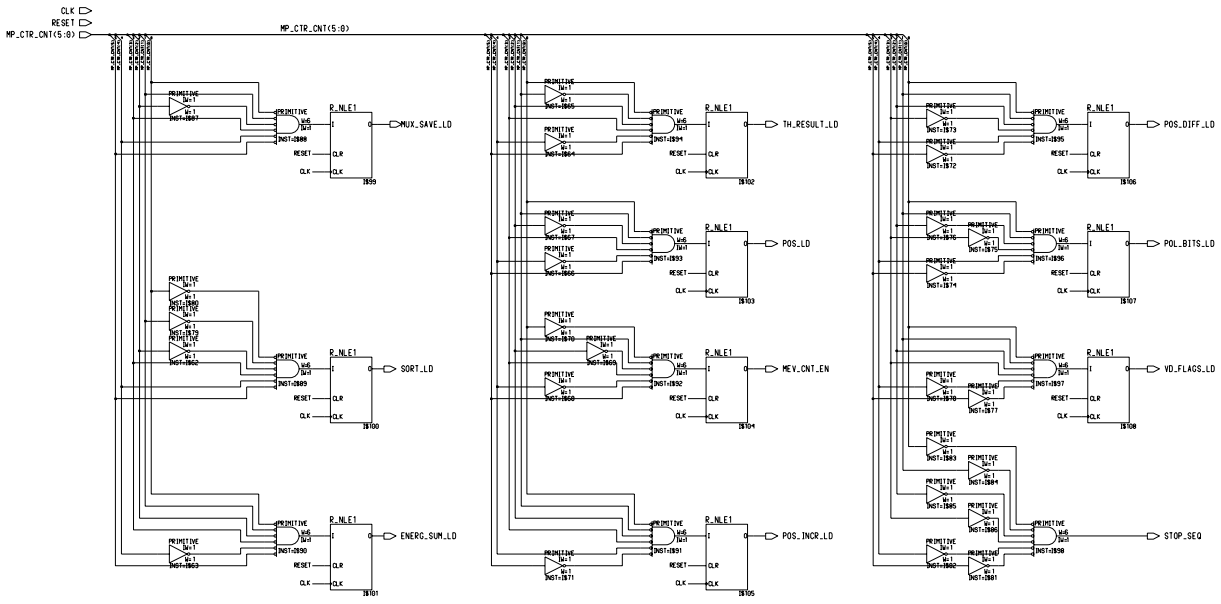


Abbildung B.4: Schaltplan von MP_CTR_PAR

Zum Einlesen eines Detektor-Ereignisses in die HEPI werden 64 Takte benötigt. Damit die Daten kontinuierlich von einer Pipelinestufe an die nächste weitergegeben werden können, ist der normale Pipeline-Zyklus (von einem einlesenden RNE zum nächsten) daher diese 64 Takte lang. Allerdings werden im Modul TC (Time Coincidence and Time Stamp) die Daten beider Detektor-Ebenen (CSI oder auch PICsIT und CDTE oder auch ISGRI) zu einem einzigen

Datenstrom zusammengefügt. Dieses „Verzahnen“ der verschiedenen Ereignisse führt dazu, daß sowohl die CDTE-, als auch die CSI-Pipeline immer wieder angehalten werden müssen.

Beides zusammen hat dazu geführt, daß für die Ablauf-Steuerung ein Zähler verwendet wurde (MP_CTR_CNT(5:0)), der von RNE (oder CTR_CLR_MP) auf Null gesetzt wird, und dann bei MP_ACTIVE mit jedem Takt bis 63 zählt. Hat er dies erreicht, kommt entweder gleich wieder ein RNE oder MP wird in seinem Ablauf angehalten bis diese Pipeline weiterarbeitet. Es gilt also:

$$\text{SEQ\_CNT\_EN} <= \text{MP\_ACTIVE and not STOP\_SEQ}$$

Bei der Entwicklung des Moduls wurde vor allem auf zwei Dinge Wert gelegt:

1. Alle Teilergebnisse, die sich bei der Abarbeitung ergeben, sollen in Registern zwischengespeichert werden;

2. Diese Register sollen entweder gültige Zwischenergebnisse oder Null enthalten (und bis auf den Ausgang durchreichen).

Als Folge dieser beiden Anforderungen wird zur Ablaufsteuerung eine Kombination der beiden Zähler verwendet: MP_CTR_CNT(5:0) gibt den vollständigen Rekonstruktions- und Polarimetrie-Ablauf vor, wie er de facto erst nach Einlesen des letzten Ereignisses der Rekonstruktions-Gruppe stattfinden kann, und CNT(1:0) filtert diese Steuerung, je nach dem, ob die durch MP_CTR_CNT(5:0) „angesprochene“ Aktion beim Einlesen jedes Teil-Ereignisses (Sortieren, Energie aufsummieren) oder eben erst bei Vollständigkeit der Gruppe stattfinden darf. Im Folgenden soll der Ablauf Schritt für Schritt durchgegangen werden; der besseren Übersicht zuliebe, sollen dabei die eingeführten Signalnamen anstelle der Zählerstände verwendet werden. Eine Zuordnung der Namen zu den Takten wird in zeitlicher Reihenfolge vorangestellt, wobei die Zählerstände als Dezimalzahlen angegeben werden (z. B. MP_CTR_CNT[45] für MP_CTR_CNT(5:0) = ´101101´):

MP_CTR_CNT[4]	MUX_SAVE_LD
MP_CTR_CNT[7]	SORT_LD
MP_CTR_CNT[16]	ENERGY_SUM_LD
MP_CTR_CNT[18]	TH_RESULTS_LD_PAR
MP_CTR_CNT[20]	POS_LD_PAR
MP_CTR_CNT[21]	MEV_CNT_EN_PAR
MP_CTR_CNT[32]	POS_INCR_LD_PAR
MP_CTR_CNT[40]	POS_DIFF_LD_PAR
MP_CTR_CNT[44]	POL_BITS_LD_PAR
MP_CTR_CNT[56]	VD_FLAGS_LD_PAR
MP_CTR_CNT[63]	STOP_SEQ

Nun zum eigentlichen Ablauf:

1. Der Rekonstruktions-Teil
   Der erste Takt wird durch RNE definiert: CTR_RNE_CSI $\longrightarrow$ MP_CTR_CNT[0] und (sofern die Bedingungen erfüllt sind) CNT_DOWN_NOW

Der nächste Schritt ist wichtig, schließlich muß das „Herzstück“ erst ermitteln, ob die Rekonstruktion neu gestartet werden muß oder nicht. Ist ein RESTART_MP notwendig, so kommt es auf seine Ursache an, wann und wie lange es aktiv ist
(t = eine Gatterlaufzeit; T = Takt):

- Auslöser MP_(not_)ACTIVE $\longrightarrow$ RESTART_MP
  von t nach RNE bis t nach RNE, wenn MULTIPLE

- Auslöser ZERO $\longrightarrow$ RESTART_MP
  von t nach CNT[0] bis 2 T + t nach MUX_SAVE_LD

- Auslöser MUX_MISMATCH $\longrightarrow$ RESTART_MP
  von t nach MP_CTR_CNT[1] bis 2 T + t nach MUX_SAVE_LD

Spätestens nach 2 Takten sollte die Situation allerdings geklärt sein, und eine neue Gruppe kann im Zweifelsfall angefangen werden:

NEW_MUX <= (MUX_SAVE_LD and ZERO) or (MP_ACTIVE and MUX_MISMATCH)

Und damit auch am Ende die richtige Gültigkeit dasteht, muß das erste Gültigkeits-Flip-Flop initialisiert werden:

VD_FF_INIT	<=	2 T nach NEW_MUX
		$\longrightarrow$ PRE_VD <= MP_EN_CTR (außer AC_VD_MP = ´0´)
PRE_VD	<=	MP_EN_CTR AND aller AC_VD_MP

Die nächsten Schritte sind recht einfach:
Zunächst soll der Sortierer das neue Ereignis einsortieren, und die bis zu diesem Teilereignis entstandene Gesamtenergie gespeichert werden:

SORT_LD und ENERGY_SUM_LD

Dann kommt es auf den Gruppenstatus an; die folgenden Schritte werden nur ausgeführt, wenn die Gruppe vollständig ist (Last_MER_Event = ´1´):

Die Ergebnisse der Schwellenwert-Vergleiche SUM_GT_THxx werden mit TH_RESULT_LD geladen und an MP_CTR zur Bestimmung der MUX-Ansteuerung übergeben, die die Pixel-Position des Einfalls- und des Streu-Ereignisses bestimmt:

TH_RESULT_LD	<=	Last_MER_Event and TH_RESULT_LD_PAR
MIN_SORTER	<=	(DOUBLE and not SUM_GT_TH2) or
		(TRIPLE and not SUM_GT_TH_3L)
MID_SORTER	<=	(DOUBLE and SUM_GT_TH2) or
		(TRIPLE and SUM_GT_TH_3L and not SUM_GT_TH_3H)
MAX_SORTER	<=	TRIPLE and SUM_GT_3H

Daraus folgt:

POS_MUX(0) <= MIN_SORTER or MAX_SORTER
POS_MUX(1) <= MID_SORTER or MAX_SORTER

Es wurden 4 Multiplexer verwendet (Y- und Z-Komponente, Einfalls- und Streu-Position), die als Eingangs-Signale die Y- und Z-Komponenten der Sortierer-Ausgänge bekommen:

Y_MIN(5:0) <= SORT_MIN(15:10) Z_MIN(5:0) <= SORT_MIN(22:17)
Y_MID(5:0) <= SORT_MID(15:10) Z_MID(5:0) <= SORT_MID(22:17)
Y_MAX(5:0) <= SORT_MAX(15:10) Z_MAX(5:0) <= SORT_MAX(22:17)

Eingang:	Einfalls-MUX	Streu-MUX
0	ZERO_6(5:0)	ZERO_6(5:0)
1	MIN(5:0)	MID(5:0)
2	MID(5:0)	MIN(5:0)
3	MAX(5:0)	ZERO_6(5:0)

Ein paar Takte später sollten die richtigen Werte sowohl an den Registern der Einfalls-Position wie auch an denen der Streu-Position anliegen und können übernommen werden:

POS_LD <= Last_MER_Event and POS_LD_PAR

Außerdem steht nun auch fest, ob das Ereignis nur gezählt werden soll, weil es den 10 Bit Wertebereich der Multiple-Energie überschreitet:

10_MEV_CNT_NOW <= TRIPLE and Last_MER_Event and MEV_CNT_EN_PAR

Im Falle von Dreifach-Ereignissen wäre man nun fertig; bei Doppelten-Ereignissen schliesst sich die Polarimetrie an:

2. Der Polarimetrie-Teil
   Zunächst werden alle 4 Angaben (Y, Z von Einfall- und Streupixel) mit zero2 auf 8 Bit erweitert, um auf jeden Fall weder beim Inkrementieren, noch bei der Differenz-Bildung Schwierigkeiten mit dem Wertebereich zu bekommen. Das potentielle Streupixel kann ja so weit weg sein, daß eine Differenz zum Einfallspixel trotzdem alle 7 Bit der inkrementierten Einfallsposition braucht, oder sogar negativ wird. Für den weiteren Ablauf sollten beide Fälle unterscheidbar sein.

   Nun werden Y- und Z-Komponente der Einfallsposition jeweils um 1 erhöht und in Register übernommen:

   POS_INCR_LD <= Last_MER_Event and DOUBLE and POS_INCR_LD_PAR

Danach werden die jeweils 8 Bit der Y- und Z-Komponenten der Streuposition davon abgezogen. Bei der Gelegenheit wird festgestellt, ob das Ergebnis negativ ist, und auch die Differenz zwischengespeichert:

POS_DIFF_LD <= Last_MER_Event and DOUBLE and POS_DIFF_LD_PAR

Nun kann das Polarimetrie-Ergebnis gemäß der Gleichungen aus Abschnitt B.2.1 codiert und in das letzte Register übernommen werden:

POL_BITS_LD <= Last_MER_Event and DOUBLE and POL_BITS_LD_PAR

Zu guter Letzt muß noch die ermittelte Gültigkeit dem Ausgang zur Verfügung gestellt werden:

VD_FLAGS_LD <= Last_MER_Event and VD_FLAGS_LD_PAR

und der Ablauf kann von vorne beginnen, sofern er nicht durch STOP_SEQ und ein fehlendes RNE angehalten wird (was durchaus auch zwischen den Teilereignissen geschehen kann!).

C CSI_SPT

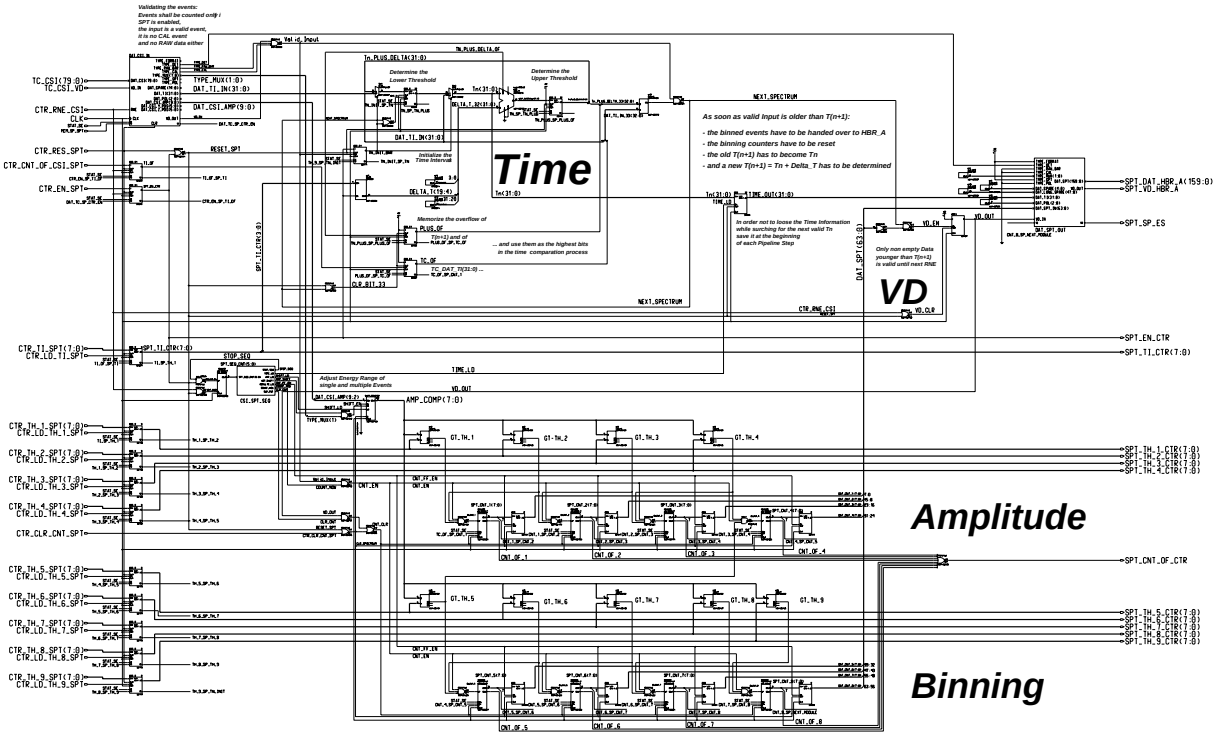


Abbildung C.1: Schema von SPT, Spectral Timing

Eingänge:

CLK	4 MHz Systemtakt
CTR_RES_SPT	Reset-Signal, löscht alle SPT-Register und Zähler
CTR_EN_SPT	Enable-Signal
CTR_RNE_CSI	Pipeline-Takt und Übernahme-Signal für neue Daten
TC_CSI(79:0)	80 Bit CSI-Daten von Modul TC
TC_CSI_VD	die dazugehörige Gültigkeit
CTR_CNT_OF_CSI_SPT	Aktuelles Ereignis bräuchte ein Bit mehr Zeit
CTR_CLR_CNT_SPT	löscht alle Energie-Intervall-Zähler
CTR_TH_1_SPT(7:0)	8 Bit Schwellenwerte, Schwelle 1
CTR_LD_TH_1_SPT	und ihre Übernahmesignale
CTR_TH_2_SPT(7:0)	Schwelle 2
CTR_LD_TH_2_SPT	

CTR_TH_3_SPT(7:0)	Schwelle 3
CTR_LD_TH_3_SPT	
CTR_TH_4_SPT(7:0)	Schwelle 4
CTR_LD_TH_4_SPT	
CTR_TH_5_SPT(7:0)	Schwelle 5
CTR_LD_TH_5_SPT	
CTR_TH_6_SPT(7:0)	Schwelle 6
CTR_LD_TH_6_SPT	
CTR_TH_7_SPT(7:0)	Schwelle 7
CTR_LD_TH_7_SPT	
CTR_TH_8_SPT(7:0)	Schwelle 8
CTR_LD_TH_8_SPT	
CTR_TH_9_SPT(7:0)	Schwelle 9
CTR_LD_TH_9_SPT	
CTR_TI_SPT(7:0)	8 Bit Integrationszeit
CTR_LD_TI_SPT	und ihr Übernahmesignal

Ausgänge:

SPT_DAT_HBR_A(159:0)	160 Bit SPT-Ereignis
SPT_VD_HBR_A	die zugehörige Gültigkeit
SPT_EN_CTR	Enable-Rückmeldung
SPT_CNT_OF_CTR	Fehler-Signal (Overflow eines Intervall-Zählers
SPT_TH_1_CTR(7:0)	8 Bit geladene Schwellenwerte, Schwelle 1
SPT_TH_2_CTR(7:0)	Schwelle 2
SPT_TH_3_CTR(7:0)	Schwelle 3
SPT_TH_4_CTR(7:0)	Schwelle 4
SPT_TH_5_CTR(7:0)	Schwelle 5
SPT_TH_6_CTR(7:0)	Schwelle 6
SPT_TH_7_CTR(7:0)	Schwelle 7
SPT_TH_8_CTR(7:0)	Schwelle 8
SPT_TH_9_CTR(7:0)	Schwelle 9
SPT_TI_CTR(7:0)	8 Bit geladene Integrationszeit

C.1 Algorithmische Umsetzung

Das Modul erzeugt zeitlich hochaufgelöste Energie-Histogramme. Ein SPT-Ereignis ist 160 Bit groß.

C.1.1 Die Eingangsgrößen

Als wichtigste Eingangsgrössen werden benötigt:
Die bereits zeitkorrigierten Daten TC_CSI(79:0),
das Gültigkeitsbit TC_CSI_VD,
die 9 Schwellenwerte CTR_TH_1_SPT(7:0) bis CTR_TH_9_SPT(7:0),
die (kodierte) Integrationszeit CTR_TI_SPT(7:0).

Außerdem wird für die korrekte Bestimmung der Integrationszeit-Intervalle noch eine weitere Information benötigt: CTR_CNT_OF_CSI_SPT.

C.1.2 Gültige Eingangs-Daten

Nicht alle Eingangsdaten sollen bei der Zusammenfassung in Zählern berücksichtigt werden: Sowohl wenn es sich um Rohdaten (TYPE_RAW_BAR = ´0´) handelt als auch bei Kalibrierungs-Ereignissen (TYPE_CAL = ´1´), aber auch im Falle daß das Vorgängermodul TC die Daten als ungültig bezeichnet (VD_IN = ´0´) muß sichergestellt sein, daß SPT diese Daten ignoriert. Es wurde daher ein Signal geschaffen, das anzeigt, ob dieses Ereignis berücksichtigt werden soll oder nicht:

$$\text{Valid\_Input} <= \text{VD\_IN and TYPE\_RAW\_BAR and not}\text{TYPE\_CAL}$$

C.1.3 Die Bestimmung der Zeitintervalle

Die gültigen Daten sollen innerhalb gleich langer Zeitintervalle zusammengefaßt werden. Es ist daher wichtig, zunächst einen zeitlichen Anfang für den Betrieb zu finden, und danach aufgrund der eingelesenen Zeitinformation festzustellen, ob die aktuellen Daten noch in die aktuelle Integrationszeit gehören, oder ob für sie ein späteres Zeitintervall gefunden werden muß. Dabei ist es von Vorteil, daß die Eingangs-Daten in zeitlicher Reihenfolge geliefert werden.

Die Interpretation des Integrationszeit-Parameters (CTR_TI_SPT(7:0))

Von wissenschaftlicher Seite wurde die Forderung aufgestellt, die Integrations- zeit solle zwischen 1 ms (2^{12} Systemtaktzyklen) und 500 ms (2^{21} Takte) einstellbar sein. Nachdem festgestellt wurde, daß für die direkte Angabe dieser Integrationszeit 9 Bit benötigt werden, aber nur 8 Bit zur Verfügung stehen, wird nun mit einem 4:16-Decoder gearbeitet, der die unteren (LSB) 4 Bit von CTR_TI_SPT der Hälfte der Integrationszeit DELTA_T_32(31:0) zuordnet. Im vorherigen Modul TC ist die Zeitauflösung der Daten von 238,4 ns (Systemtakt) auf 61,04 μs (2^8 Systemtakte) umgesetzt worden. Daher werden diese 16 Parameter-Bits nicht den Bits 27:12 zugeordnet, sondern DELTA_T_32(19:4); alle anderen Delta-Bits werden auf ´0´ gesetzt.

Die Bestimmung gültiger Integrationsintervalle

Um die eingelesenen gültigen Daten anhand ihrer mitgelieferten Zeitinformation „ihrem“ Integrationsintervall zuordnen zu können, muß zum einen vor allem das Ende des Integrationsintervalls der aktuellen Zählung (Messung) bekannt sein, zum anderen muß es jederzeit möglich sein, auch das nächste Intervall korrekt zu bestimmen. Zu diesem Zwecke – und weil den Ausgangsdaten immer der Anfang des zugehörigen Zeitintervalls als Zeitinformation mitgegeben werden soll – werden in SPT grundsätzlich sowohl die Anfangszeit Tn(31:0) als auch die Endzeit Tn_PLUS_DELTA(31:0) des aktuellen Integrationszyklus in Registern gespeichert. Die verschiedenen Integrationszyklen einer Beobachtung sollen immer gleich lang sein (einstellbar über CTR_TI_SPT(7:0)). Es gilt daher:

$$\text{Tn\_PLUS\_DELTA(31:0)} <= \text{Tn(31:0)} + \text{DELTA\_T\_32(31:0)}$$

Da die Daten in zeitlicher Reihenfolge eingelesen werden, kann – nach der Bestimmung der Anfangszeit T0 – davon ausgegangen werden, daß die aktuellen Daten älter als Tn sind.

Die Datenzeit DAT_TI_IN(31:0) muß zur Intervallzuordnung daher nur mit Tn_PLUS_DELTA verglichen werden:

?? DAT_TI_IN $\geq$ Tn_PLUS_DELTA ??

Dieser Vergleich ist allerdings nicht ganz trivial, denn 32 Bit Zeitinformation werden aller Voraussicht nach nicht unbedingt für jeden Betriebszyklus ausreichen. Es müssen daher zwei Fälle berücksichtigt werden:

- Die Berechnung von Tn_PLUS_DELTA hat einen Überlauf erzeugt, ohne daß die im vorangehenden Modul TC bestimmte Zeitinformation die 32 Bit bereits ausgereizt hätte. Das kann dazu führen, daß Tn_PLUS_DELTA(31:0) << DAT_TI_IN(31:0), aber Tn_PLUS_DELTA > DAT_TI_IN ist!

- Umgekehrt kann in der Zeitbestimmung der CSI-Ereignisse im TC-Modul ein Überlauf stattgefunden haben, was den Fall zur Folge haben kann, daß Tn_PLUS_DELTA(31:0) > DAT_TI_IN(31:0) berechnet wird, aber Tn_PLUS_DELTA < DAT_TI_IN richtig ist!

Beide Fälle können abgefangen werden, wenn nicht die 32 Bit Zeitinformation direkt zum Vergleich herangezogen werden, sondern ein zusätzliches 33. Bit berücksichtigt wird:

Tn_PLUS_DELTA(32)	<= PLUS_OF
DAT_TI_IN(32)	<= TC_OF

Wichtig dabei sind folgende Punkte:

- Das Zentralsteuerwerk liefert CTR_TI_OF_SPT zusammen mit den Daten, für die TC_OF das erste Mal berücksichtigt werden muß, also mit dem selben Ereignis, bei dem im TC-Modul der Überlauf auftritt. Auf diese Weise kann sichergestellt werden, daß die Daten und das „oberste Zeitbit“ gleichzeitig nach SPT übernommen werden, und als Zeitinformation immer die korrekten 33 Bit verwendet werden.

- PLUS_OF wird immer zusammen mit Tn_PLUS_DELTA gespeichert.

- Beide Signale TC_OF und PLUS_OF werden – sind sie einmal auf ´1´ gesetzt – erst dann gemeinsam gelöscht, wenn das nächste Integrationsintervall beginnt:

NEXT_SPECTRUM if DAT_TI_IN_33(32:0) $\geq$ Tn_PLUS_DELTA_33(32:0)

Ein weiterer Spezialfall tritt auf, wenn die Daten eine derartige zeitliche Lücke aufweisen, daß mehrere DELTA_Ts seit dem letzten Valid_Input verstrichen sind. In diesem Falle muß SPT diese Zeit so schnell wie möglich aufholen, ohne daß dabei die Bedingung der gleich langen Zeitintervalle verletzt würde. Dies wird schnell einsichtig, wenn man sich folgenden Sachverhalt klar macht:

Um ein Ereignis im korrekten Zeitintervall zählen zu können stehen (inklusive Einlesetakt) 64 Systemtakte zur Verfügung. Während dieser Zeit muß das Zeitintervall gefunden worden sein, für das DAT_TI_IN_33 <= Tn_PLUS_DELTA_33 ist (s. o.), und das Ereignis in dem entsprechenden Zähler gezählt worden sein. Ist dies nicht der Fall, so kann dem Ereignis nur das als letztes gefundene (falsche) Intervall zugeordnet werden, und die Suche nach den richtigen Intervallgrenzen geht mit dem nächsten Valid_Input weiter.

Aus diesem Grund kann die Schleife Tn ⟶ Tn_PLUS_DELTA ⟶ NEXT_SPECTRUM (wobei NEXT_SPECTRUM gleichbedeutend ist mit einem neuen Tn) in 2 Takten (3. Takt = neues Tn) abgearbeitet werden und ist nicht an den allgemeinen Ablaufzähler gekoppelt.

Eine (möglicherweise) recht massive Folgerung aus diesen 2 Takten pro Intervallfindung und 64 Takten pro Ereignisverarbeitung ist die Tatsache, daß zwei gültige Ereignisse maximal 32 Deltas Zeitabstand haben dürfen, sollen sie korrekt gezählt werden.

Die Bestimmung der Anfangszeit

Es wurde bereits erwähnt, daß im SPT-Modul die gültigen Daten während einstellbarer Integrationsintervalle zusammengefaßt werden sollen. Um diese immer gleich langen Zeitintervalle (von Tn bis Tn_PLUS_DELTA = T(n+1)) berechnen zu können, muß nach der Inbetriebnahme des Moduls (CTR_EN_SPT = ´1´ vom Zentralsteuerwerk, bzw. die Rückmeldung an dasselbe SPT_EN_CTR = ´1´) auf sinnvolle Weise die Anfangszeit T0 der Messung bestimmt werden. Da nur gültige Daten innerhalb von SPT berücksichtigt werden sollen, wurden die 32 Bit Zeitinformation des ersten Valid_Inputs nach Inbetriebnahme als T0 definiert und in das Register für Tn geladen.

Im vorherigen Abschnitt wurde darauf hingewiesen, daß vor allem das Ende der Integrationsintervalle Tn_PLUS_DELTA benötigt wird, das (normalerweise) einen Takt nach der Übernahme von Tn in seinem Register gespeichert wird. Damit der (ungültige!) Tn_PLUS_DELTA-Wert, der mit dem ungültigen Tn berechnet wird nicht zur Ermittlung von NEXT_SPECTRUM herangezogen wird, muß gleichzeitig mit dem ersten gültigen Tn auch der Wert für Tn_PLUS_DELTA ermittelt und in das entsprechende Register geladen werden.

Erreicht wird beides, indem mit zwei Multiplexern und einem Flip-Flop als Steuerung der Multiplexer gearbeitet wird: Nach Inbetriebnahme ist das Flip-Flop auf ´0´ gesetzt (TN_INIT_BAR = ´0´); die Register für Tn und Tn_PLUS_DELTA übernehmen mit jedem Takt die Zeitinformation der Eingangsdaten bzw. die Summe dieser Zeitinformation und DELTA_T. Sobald Valid_Input = ´1´ wird, übernimmt dieses Steuer-Flip-Flop den Wert von SPT_EN_CTR – ist das Modul eingeschaltet, kann jetzt nur noch Tn_PLUS_DELTA in das Tn-Register bzw. Tn in den Addierer übernommen werden; bei ausgeschaltetem Modul wird weiterhin DAT_TI_IN übernommen:

```
case SPT_EN_CTR = ´1´ (Modul eingeschaltet)
   if TN_INIT_BAR = ´0´ (n=0)
      Tn(31:0)            <= DAT_TI_IN(31:0)
      Tn_PLUS_DELTA(31:0) <= DAT_TI_IN(31:0) + DELTA_T(31:0)
   else
      Tn(31:0)            <= Tn_PLUS_DELTA(31:0)
      Tn_PLUS_DELTA(31:0) <= Tn(31:0) + DELTA_T(31:0)
   endif
else (Modul ausgeschaltet)
```

```
  Tn(31:0)            <= 0
  Tn_PLUS_DELTA(31:0) <= 0
endcase
```

C.1.4 Die Zuordnung der Ereignisse in die verschiedenen Intervalle

Die Energie-Intervalle: Das sogenannte Binning

Wie bereits erwähnt wurde, können von der Erde 9 8-Bit-Schwellenwerte TH_1 bis TH_9 kommandiert werden, die es erlauben, das gesamte Energiespektrum in 10 (beliebige) Bereiche einzuteilen, auch wenn sich diese Unterteilungen natürlich nur im Wertebereich der CSI-Amplitude, genauer gesagt von 20 keV bis 10 MeV auswirken. Numeriert man diese Bereiche (mit aufsteigender Energie) von 0 bis 9 durch, so ist den Bereichen 1 bis 8, die im Gegensatz zu den Bereichen 0 und 9 jeweils durch eine obere und eine untere Schwelle definiert sind, jeweils ein 8-Bit-Zähler zugeordnet: Bereich n und Zähler CNT_n haben als zugehörige Schwellen TH_n und TH_(n+1). Die Zuordnung der Ereignisse zu einem dieser Zähler geschieht nun anhand ihrer Amplitude:

$$\text{CNT\_n} <= \text{CNT\_n} + 1 \text{ if } \text{TH\_n} \leq \text{Amplitude} < \text{TH\_(n+1)}$$

Bisher wurde nicht darauf eingegangen, wie denn die 10 Bit der Amplitude den 8 Bit der Schwellenwerte zum Vergleich zugeordnet werden sollen. Dies ist etwas trickreich, da in einem vorherigen Modul (MP, in dem die Mehrfach-Ereignisse zu einzelnen Ereignissen zusammengefaßt werden) der Wertebereich der ursprünglichen Mehrfach-Ereigneisse („Multiples“) im Vergleich zu demjenigen der Einfachereignisse („Singles“) um 1 Bit verschoben wurde. Damit die einzelnen Schwellenwerte sich nicht unterschiedlich auf Singles und Multiples auswirken, müssen alle Singles um ein Bit nach rechts geschoben werden. Danach werden die 8 obersten Bit (AMP_COMP(7:0)) mit den Schwellenwerten verglichen (MUX(1) = ´1´ gilt dann, wenn es sich um ein Multiple handelt):

```
Case MUX(1) = ´1´ (Multiple)
     AMP_COMP(7:0) <= DAT_CSI_AMP(9:2)
else (Single, MUX(1) = ´0´)
     AMP_COMP(7:0) <= Shift_Right{DAT_CSI_AMP(9:2)}
endcase

For n = 1 bis 8
     GT_TH_n <= 1          if {TH_n(7:0) > AMP_COMP(7:0)}
     CNT_n <= {CNT_n + 1}  if {GT_TH_(n+1) and not {GT_TH_n}}
endfor
```

Die Integrationsintervalle

Es wurde bereits erklärt, wie Anfang und Ende der Integrationszeit ermittelt werden, allerdings ohne deutlich zu definieren, in welches Zeitintervall das eine Ereignis gehört, das durch seine Zeitinformation DAT_TI_IN einen Wechsel (NEXT_SPECTRUM) verursacht. Einzig zwei Hinweise wurden gegeben, daß dieses Ereignis bereits dem neuen Zeitintervall zugeordnet werden soll:

- Bei der Definition von NEXT_SPECTRUM heißt es DAT_TI_IN $\geq$ Tn_PLUS_DELTA.

- Als Zeitiniformation wird den Zählerdaten die Anfangszeit Tn – also das vorherige Tn_PLUS_DELTA – mitgegeben.

Diese Zuordnung hat allerdings schwerwiegende Folgen im Schaltungsentwurf: Alle Daten, die zum Versenden an die schnelle (HEPI-) Schnittstelle HBR_A weitergegeben werden sollen (Tn und die 8 Zählerstände) müssen in zusätzlichen Registern zwischengespeichert werden! Dies ist sofort einsichtig, wenn man sich folgende Punkte vergegenwärtigt:

- Mit einem neuen Zeitintervall wird eine neue Anfangszeit Tn definiert

- Alle Zähler CNT_n müssen gelöscht werden, bevor das erste Ereignis gezählt werden kann

- Der Zeitpunkt, all dies zu tun, ist erst bekannt, wenn das erste Ereignis des neuen Integrationszyklus bereits eingelesen wurde

- Die (erst dann gültigen) Ausgangsdaten müssen solange am Ausgang anliegen, bis sie vom HBR_A übernommen werden können

- Das HBR_A übernimmt die Daten erst zum selben Zeitpunkt, wenn bereits das nächste (also potentiell zweite!) Ereignis nach SPT eingelesen wird.

C.2 Gültige Ausgangs-Daten: Das Valid-Flag VD

So wie mit jedem Pipeline-Zyklus nicht nur Ereignisdaten (Amplitude, Zeit, „Verwaltungs-Information“ TYPE, ...), sondern auch die Information über ihre Gültigkeit nach SPT geladen werden (TC_CSI(79:0) und TC_VD), so kann auch beides mit jedem RNE_CSI von SPT an das nachfolgende Modul HBR_A weitergegeben werden (SPT_DAT_HBR_A(159:0) und SPT_VD_HBR_A). An gültige SPT-Daten (SPT_VD_HBR_A = ´1´) werden drei Anforderungen gestellt:

- Das Modul als ganzes muß vom zentralen Steuerwerk(CTR) auch eingeschaltet worden sein (CTR_EN_SPT = ´1´), und
- der Zählvorgang für einen Integrationszyklus muß beendet sein.
- Es müssen tatsächlich Ereignisse gezählt worden sein – die Telemetrie ist zu kostbar (Rate sehr gering im Vergleich zur Datenmenge), als daß man gerne leere Information versenden würde.

Alle Punkte können mittels der Verwendung eines Flip_Flops erfüllt werden, das mit RNE_CSI oder RESET_SPT (darin verbirgt sich not CTR_EN_SPT) gelöscht und bei VD_EN gesetzt wird.

VD_OUT <= 1 if aktive Taktflanke bei VD_EN = ´1´
VD_EN <= NEXT_SPECTRUM and (CNT_1 or CNT_2 or ... or CNT_8)
VD_CLR <= CTR_RES_SPT or not SPT_EN_CTR or CTR_RNE_CSI

C.3 Die unterschiedlichen Rücksetz-Stufen und -Bedingungen

Innerhalb SPT müssen verschiedene Teile mit unterschiedlichen Bedingungen gelöscht werden können. So macht es z. B. keinerlei Sinn, wenn die Leitung, die die Zähler löscht, ebenfalls die Zeitregister oder gar die Schwellenwerte und die Integrationszeit löschen würde. Es gibt daher unterschiedliche „Rücksetz-Stufen“:

- CTR_RES_SPT
  löscht alle Register innerhalb SPT, seien es (ohne Anspruch auf Vollständigkeit) die per Kommando von der Erde aus ladbaren 8-Bit-Parameter (Integrationszeit, Schwellenwerte), die Zählerstände, Zeitintervallgrenzen, Gültigkeitsregister oder auch das Register, in dem das Einschalten des Moduls gespeichert wird.

- RESET_SPT <= CTR_RES_SPT or not SPT_EN_CTR
  läßt nur die Parameter und die „Eingangsregister“, also die Register für die Daten samt ihrer Eingangsgültigkeit (DAT_CSI_IN) für CTR_EN_SPT und CTR_CNT_OF_CSI_SPT in Ruhe; alle anderen werden gelöscht.

- NEXT_EVENT <= VD_CLR = CTR_RNE_CSI or RESET_SPT
  löscht das Gültigkeitsregister und den Ablaufzähler, so daß ein neues Ereignis bearbeitet werden kann.

- CTR_CLR_CNT_SPT
  löscht alle (Binning-) Zähler samt der dazugehörenden Register und das Amplituden-Schieberegister über einen 2-stufigen Prozeß:

  1. CLR_SPECTRUM <= CTR_CLR_CNT_SPT or RESET_SPT
     löscht Schieberegister und Zähler-Register
  2. CNT_CLR <= CLR_SPECTRUM or (VD_OUT and CLR_CNT)
     (CLR_CNT kommt von der Ablaufsteuerung) löscht die eigentlichen Zähler

- CLR_BIT_33 <= RESET_SPT or NEXT_SPECTRUM
  löscht die Register, in denen die Überlauf-Bits PLUS_OF und TC_OF gespeichert werden.

C.4 Die Ablaufsteuerung

Zum Einlesen eines Detektor-Ereignisses in die HEPI werden 64 Takte benötigt. Damit die Daten kontinuierlich von einer Pipelinestufe an die nächste weitergegeben werden können, ist der normale Pipeline-Zyklus (von einem einlesenden RNE zum nächsten) daher diese 64 Takte lang. Allerdings werden im Modul TC (Time Coincidence and Time Stamp) die Daten beider Detektor-Ebenen (CSI oder auch PICsIT und CDTE oder auch ISGRI) zu einem einzigen Datenstrom zusammengefügt. Dieses ‘Verzahnen’ der verschiedenen Ereignisse führt dazu, daß

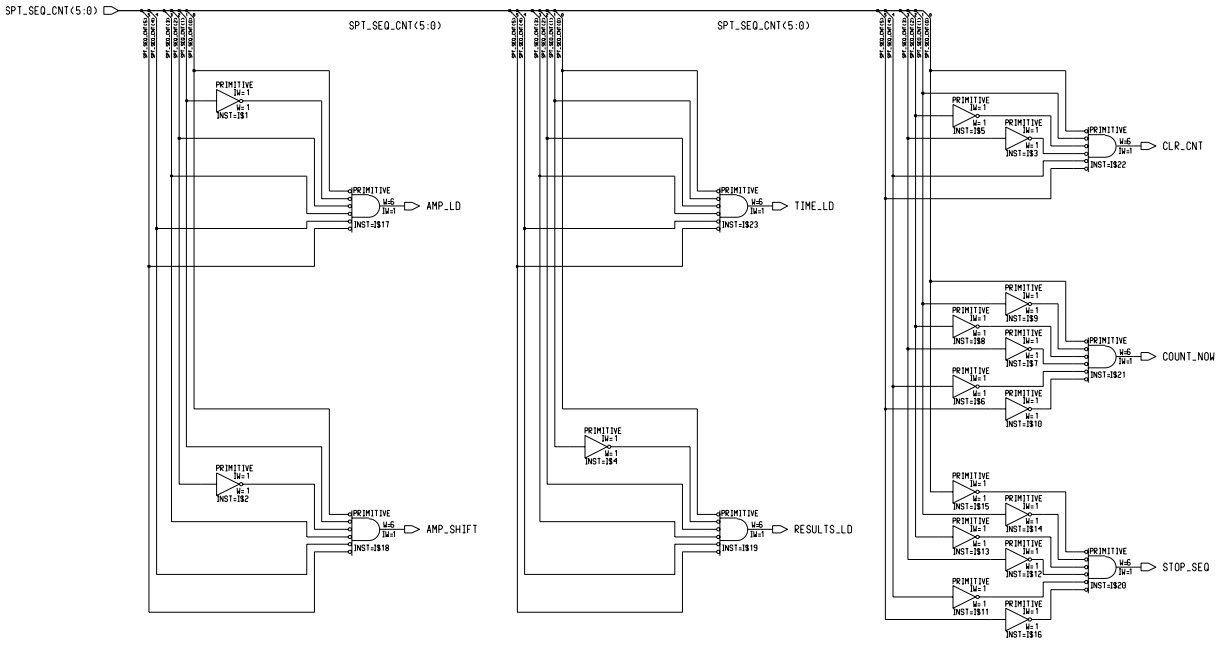


Abbildung C.2: Schema von CSI_SPT_SEQ_CNT, Spectral Timing

sowohl die CDTE, als auch die CSI-Pipeline immer wieder angehalten werden müssen. Beides zusammen hat dazu geführt, daß für die Ablauf-Steuerung ein Zähler verwendet wurde (SPT_SEQ_CNT(5:0)), der von RNE (oder CTR_CLR_SPT) auf Null gesetzt wird, und dann bei SPT_EN_CTR mit jedem Takt bis 63 zählt. Hat er dies erreicht, kommt entweder gleich wieder ein RNE, oder SPT wird in seinem Ablauf angehalten bis diese Pipeline weiterarbeitet. Es gilt also:

SEQ_CNT_EN <= SPT_EN_CTR and not STOP_SEQ

Im Folgenden soll der Ablauf Schritt für Schritt durchgegangen werden; der besseren Übersicht zuliebe sollen dabei die eingeführten Signalnamen anstelle der Zählerstände verwendet werden. Eine Zuordnung der Namen zu den Takten wird in zeitlicher Reihenfolge vorangestellt, wobei die Zählerstände als Dezimalzahlen angegeben werden (SPT_SEQ_CNT[45] z. B.):

SPT_SEQ_CNT[0] TIME_LD
SPT_SEQ_CNT[2] RESULTS_LD
SPT_SEQ_CNT[2] AMP_LD
SPT_SEQ_CNT[4] AMP_SHIFT
SPT_SEQ_CNT[12] CLR_CNT
SPT_SEQ_CNT[62] COUNT_NOW
SPT_SEQ_CNT[63] STOP_SEQ

Nun zum eigentlichen Ablauf: Nachdem sowohl die Bestimmung des aktuell richtigen Zeitintervalls als auch die Gültigkeit der Ausgangsdaten unabhängig vom Ablaufzähler sind, bleibt für die Ablaufsteuerung nur die Übernahme der Ausgangsdaten und das eigentliche Zählen übrig:

Wie bei der Erklärung der Zuordnug der Ereignisse zu den einzelnen Integrationsintervallen klar geworden ist, werden die Daten, die an das HBR_A weitergegeben werden sollen, bis auf

die Gültigkeit jeweils in dem vorhergehenden Pipelinezyklus erzeugt. Da sich Tn aber schon mit dem ersten Takt nach dem einlesenden RNE ändern kann, ist es notwendig TIME_OUT spätestens mit diesem Takt zum Ausgang zu übernehmen. Handelt es sich bei SPT_SEQ_CNT um einen synchron rückgesetzten Zähler, so wird sich das löschende RNE auch erst einen Takt nach dem Einlesen bemerkbar machen, also just in dem Takt, mit dem sich auch Tn ändern kann. Dies hat bewirkt, daß für die Übernahme von Tn nach TIME_OUT SPT_SEQ_CNT[0] zuständig ist: TIME_LD geht direkt an den Übernahme-Eingang (EN) des entsprechenden Registers.

Bei der Übernahme der Zählerstände in die Ausgangsregister kann man sich diese "Hast" sparen, da auch der Zählvorgang selber an den Ablaufzähler gekoppelt ist; wichtig ist nur, daß die Daten vor dem Zählen oder dem Löschen der Zähler übernommen werden. Auch hier ist keine weitere Logik notwendig:

CNT_FF_EN <= RESULTS_LD

Ähnliches gilt für die Übernahme der neuen Amplitude in das Schieberegister: Es muß nur so früh geschehen, daß genügend Zeit (Takte) bleibt, damit der Abgleich der Wertebereiche zwischen Singles und Multiples und der Vergleich mit den Schwellenwerten noch vor dem Zählen durchgeführt werden können:

SHIFT_LD <= AMP_LD

Es ist recht einsichtig, daß eine Amplitude nur dann rechtsverschoben werden kann, nachdem sie auch geladen wurde; AMP_SHIFT muß daher nach AMP_LD und vor dem Zählimpuls kommen. Der eigentliche Schiebevorgang darf außerdem nur dann geschehen, wenn es sich bei dem eingelesenen Ereignis um ein ursprüngliches Einzelereignis handelt, dessen Wertebereich demjenigen der ursprünglichen Mehrfachereignisse angepaßt werden muß:

SHIFT_EN <= AMP_SHIFT and not TYPE_MUX(1)

Bevor allerdings tatsächlich gezählt werden kann, muß festgestellt worden sein, ob das aktuelle Ereignis zu den bisher entstandenen Zählerständen dazugezählt werden darf, ob es sich also immer noch um das selbe Integrations-Intervall handelt, oder ob damit ein neuer Zählzyklus beginnt, die Zähler also vorher gelöscht werden müssen. NEXT_SPECTRUM kann hierzu nur indirekt herangezogen werden, da es unter Umständen nur sehr kurz am Anfang dieses Pipelinezyklus aktiv ist. Es wurde daher auf VD_OUT zurückgegriffen, ein Signal, das auf jeden Fall vom Auftreten von NEXT_SPECTRUM bis zum nächsten RNE stabil ist (CLR_SPECTRUM <= CTR_CLR_CNT_SPT or RESET_SPT VD_OUT):

CNT_CLR <= CLR_SPECTRUM or (VD_OUT and CLR_CNT)

Der letzte Schritt, das eigentliche Zählen, wird am besten möglichst spät während des Pipelinezyklus eingeplant, da die korrekte Ausführung davon abhängt, daß das richtige Zeitintervall vorher gefunden wurde. Um jedes Ereignis auch wirklich nur einmal zu zählen, ist es wichtig, daß der Zählimpuls nur während eines Taktes aktiv ist. Da der Ablaufzähler mit SPT_SEQ_CNT[63] angehalten werden kann, steht dieser letzte Takt nicht zur Verfügung; der letzte Zählerstand von SPT_SEQ_CNT, der sicher nur während eines Taktes pro Pipelinezyklus aktiv ist, ist daher SPT_SEQ_CNT[62], und als allgemeine Zählbedingung ergibt sich:

CNT_EN <= Valid_Input and COUNT_NOW

Insgesamt gilt daher für den n-ten Zähler:

EN_CNT_n <= CNT_EN and GT_TH_(n+1) and not GT_TH_n

D CSI_HIST

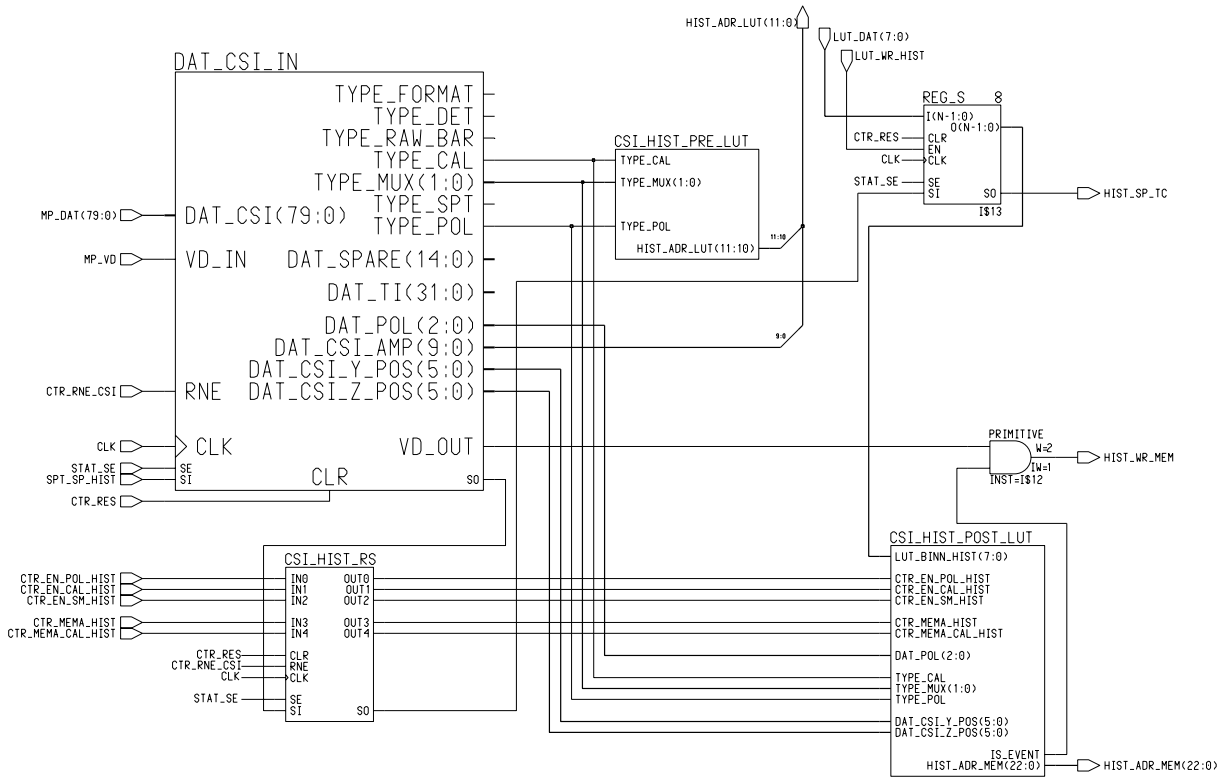


Abbildung D.1: Schema von CSI_HIST, Histogramm

Eingänge:

CLK	4 MHz Systemtakt
CTR_RES_SPT	Reset-Signal, löscht alle CSI_HIST-Register
CTR_EN_SPT	Enable-Signal
CTR_RNE_CSI	Pipeline-Takt und Übernahme-Signal für neue Daten
MP_DAT(79:0)	80 Bit CSI-Daten von Modul TC
MP_VD	die dazugehörige Gültigkeit
CTR_EN_SM_HIST	Anforderung Singl/Multi-Histogramming
CTR_EN_CAL_HIST	Anforderung Calibration-Histogramming
CTR_EN_POL_HIST	Anforderung Polarimetry-Histogramming
CTR_MEMA_HIST	Selektion des Singl/Multi-Histogrammspeichers
CTR_MEMA_CAL_HIST	Selektion des Calibration-Histogrammspeichers

LUT_DAT(7:0)	8 Bit Binning Daten
LUT_WR_HIST	Schreib-Signal zum Übernehmen der Binning Daten.

Ausgänge:

HIST_ADR_LUT(11:0)	12 Bit Adresse an die LUT
HIST_ADR_MEM(22:0)	23 Bit Adresse an MEM
HIST_WR_MEM	Schreib-Signal für MEM zum Übernehmen der Adresse.
HIST_EN_SM_CTR	geladene Anforderung Singl/Multi-Histogramming
HIST_EN_POL_CTR	geladene Anforderung Calibration-Histogramming
HIST_EN_CAL_CTR	geladene Anforderung Polarimetry-Histogramming
HIST_MEMA_CTR	geladene Selektion des Singl/Multi-Histogrammspeichers
HIST_MEMA_CAL_CTR	geladene Selektion des Calibration-Histogrammspeichers

D.1 Funktionsweise

Das Modul nimmt die Amplitudenwerte eines CsI-Ereignisses und leitet sie an den LoopUpTable (LUT) Controller weiter. Dieser ordnet dem Amplitudenwert einem Amplituden- (Energie) Fenster zu (sog. *Binning*). Dieses Energie-Fenster wird noch im gleichen RNE-Takt (aber nicht im gleichen CLK-Takt) wieder an das Histogramm-Modul übergeben. Mit den Steuer-Bits (CTR_EN_XXX_HIST) wird daraus eine Adresse erzeugt, welche dem nachfolgenden MEM-Controller sagt, an welcher Adresse der Inhalt um eins erhöht werden muß (*histogramming*).

D.1.1 CSI_HIST_PRE_LUT

Dieses Modul erzeugt die Adressen für die LUT. Die LUT hat die Binning-Werte (8 Bit breit (7:0)) in einem 14 Bit Adreßraum. Die 4 Bereiche mit den Binning-Werten für Singl-, Multi-, Calibration- und Polarimetry-Event beginnen bei $6000. Jeder Bereich ist 1024 Worte groß (Die Amplituden-Werte haben eine Breite von 10 Bit (9:0)):

$6000	Singl-Event
$6400	Multi-Event
$6800	Calibration-Event
$6C00	Polarimetry-Event

Binär ergibt sich für die Erzeugung der Adressen folgendes:

Bit 13, Bit 12	sind immer 1. Sie erzeugen den Basiswert $6000. (Wird in der LUT gemacht)
Bit 11, Bit 10	werden abhängig vom TYPE_XXX gesetzt.
Bit 11	= TYPE_MUX(1) + TYPE_POL
Bit 10	= TYPE_CAL + TYPE_POL

Diese beiden Bits (11:10) der 12-Bit-Adresse werden im Modul CSI_HIST_PRE_LUT erzeugt.

Von der LUT zurück bekommt man für Singl- und Multi-Event Typen 8 Bit, für Calibration- und Polarimetry-Events 6 Bit. Diese Daten werden im RS_8 Register gehalten. Ein WriteEnable liefert die LUT, wenn sie die Daten liefern kann (LUT_WR_HIST).

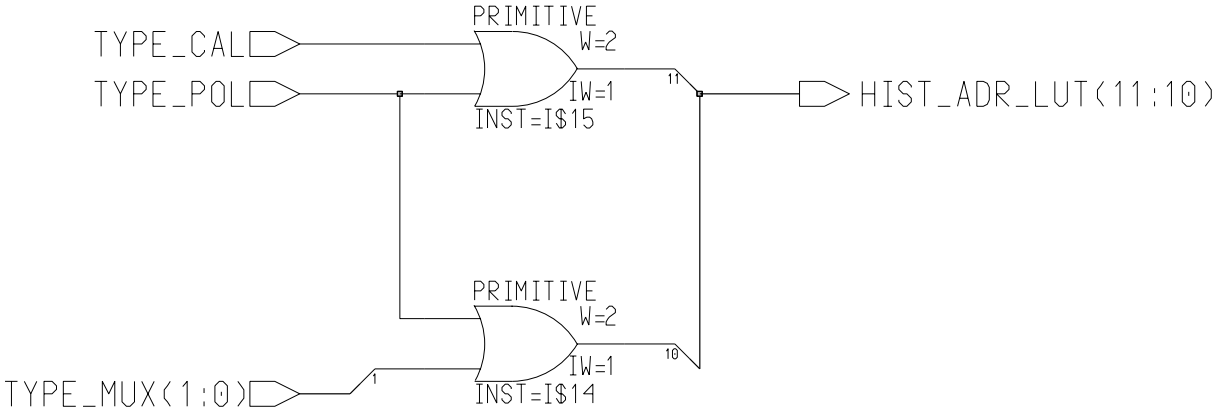


Abbildung D.2: Schaltplan vom CSI_HIST_PRE_LUT Modul

D.1.2 CSI_HIST_POST_LUT

Dieses Modul erzeugt die Adressen für den MEM. Der MEM zählt an dieser Adresse zum bisherigen Wert noch eins dazu. Die Events werden aufgespalten in Y- und Z-Koordinaten, (gebinnte) Energien und (wenn TYPE_POL) Polarimetry-Daten. Erst wenn sich zwei Events in allen diesen Daten gleichen werden sie auf die gleiche Adresse gezählt.

Da 0 bis 4 verschiedene CTR_EN_XXX_HIST anliegen können und jeder Event aber nur einem Typ zugeordnet ist, werden intern 4 Steuerleitungen mit den entsprechenden IST-Zuständen gebildet.

IS_CAL und IS_POL ist eine einfache verUNDung zwischen den entsprechenden Typen aus dem Event (TYPE_XXX) und dem Enable des CTR (CTR_EN_XXX_HIST). Für die Singl- und Multi-Events müssen ähnliche Zustände ausgeschlossen werden:

IS_SINGL	<=	CTR_EN_SM_HIST and TYPE_MUX(0) and not TYPE_MUX(1) keine Triple-Events
IS_MULTI	<=	CTR_EN_SM_HIST and TYPE_MUX(1) Polarimetry-Events sind Doubles, haben aber die höhere Priorität
IS_POL	<=	CTR_EN_POL and TYPE_POL
IS_CAL	<=	CTR_EN_CAL and TYPE_CAL

Diese Zustände werden verODERt als IS_EVENT ausgegeben und gibt an: Event ist vom verlangtem Typ. Mit diesen Zuständen wird am Multiplexer das richtige Adreß-Schema ausgewählt.

Die Speicherbereiche für die Histogramme der einzelnen Events lauten:

	MEMA = ´0´	MEMA = ´1´
Calibration	$400.000 – $440.000-1	$440.000 – $480.000-1
Polarimetry	$000.000 – $200.000-1	$200.000 – $400.000-1
Singl	$000.000 – $100.000-1	$200.000 – $300.000-1
Multi	$100.000 – $200.000-1	$300.000 – $400.000-1

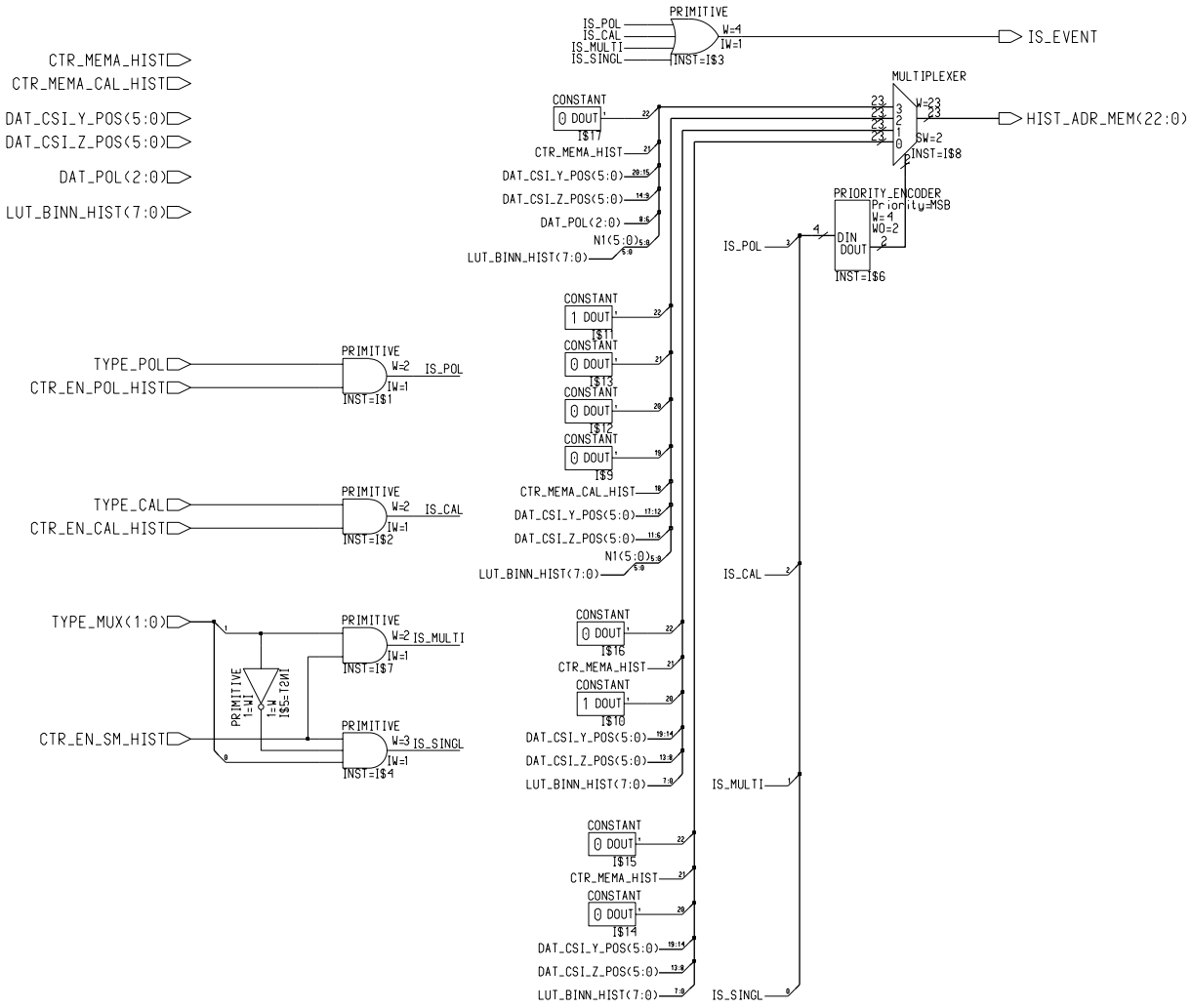


Abbildung D.3: Schaltplan vom CSI_HIST_POST_LUT Modul

In der Binär-Darstellung bedeutet das für die Bits der Adresse HIST_ADR_MEM(22:0):

	22211111111110000000000 21098765432109876543210
Polarimetry	%0SYYYYYYZZZZZZPPPBBBBBB
Calibration	%1000CYYYYYYZZZZZZBBBBBB
Multi	%0S1YYYYYYZZZZZZBBBBBBBB
Singl	%0S0YYYYYYZZZZZZBBBBBBBB

mit:

S =	CTR_MEMA_HIST
C =	CTR_MEMA_CAL_HIST
Y =	DAT_CSI_Y_POS(5:0)
Z =	DAT_CSI_Z_POS(5:0)
P =	DAT_POL(2:0)
B =	LUT_DAT(7:0)

Die Binning-Werte müssen LSB sein, um bei der Übertragung (downlink) bei evtl. auftretenden Fehlern besser korrigierbar zu sein. Es ist besser für ein Pixel mehrere Energiewerte zu verlieren (da besser rekonstruierbar), als für mehrere Pixel einen Energiewert zu verlieren. In der Realisation werden 4 Adreßbusse mit der jeweils richtigen „Daten-Konfektionierung“ erzeugt und an einen Multiplexer angeschlossen. Dieser Multiplexer wählt den richtigen Adreß-Bus über seinen Select-Bus aus:

	Eingang	binär
Polarimetry	3	%11
Calibration	2	%10
Multi	1	%01
Singl	0	%00

Die Daten für den Select-Bus werden von einem Encoder aus den vier IS_XXX Zuständen erzeugt. Dieser Encoder hat als Priorität das MSB, sollten zwei Zustände gleichzeitig eintreffen, so wird immer der mit der höheren Wertigkeit verwendet. Vorkommen kann das nur falls der CMD_CTRL anordnet Singl/Multi- und Polarimetrie[1]-Ereignisse zu beachten. In diesem Fall wird also nur die Adresse für den Polarimetrie-Ereignisse erzeugt (und nicht für den Multi(Zweifach)-Ereignis). Liegt kein Ereignis an, also keines der IS_XXX Zustände gesetzt, wird am Decoder die %00 am Select-Bus generiert, dies entspricht einem IS_SINGL Zustand.

[1] von Natur aus Zweifach-Ereignisse

Abbildungsverzeichnis

Literaturverzeichnis

[1] G. Malagerti, E. Caroli, G.Di. Locco. PICsIT Multiple Event Reconstruction. IN-IM-TES-TN-0013, Is. 10.0, 4.7.96

[2] TEMIC Semiconductors, Radiation, TEMIC Radiation Policy, MATRA MHS, 10 Mar. 97

[3] T. Corbiere, J.L. Venturin. Dose Rate Effects, Investigation of Dose Rate Effects on CMOS Submicronic Technologies. TEMIC Semiconductors MATRA MHS, Spet. 95

[4] T. Corbiere, V. Lassere, B. Thomas, S. Hachad, R. Ecoffet, S. Duzellier. Hardening CMOS, Hardening a CMOS Technology Against Total Dose and Heavy Ion Induced Latch-up, TEMIC Semiconductors MATRA MHS, Spet. 95

[5] TEMIC Semiconductors, MG1RT Sea of Gates Series 0.6 Micron CMOS, MATRA MHS Rev. A, 03/09/96

[6] TEMIC Semiconductors, ASIC Design Manual MG1/MG1RT, ATD-TS-GR-R0162, 2.1, September 1996

[7] National Semiconductor, Radiation Owner's Manual, lit. #650221-001, 2.5K, 1/99

[8] P. Fortescue, J. Stark, Spacecraft Systems Engeneering, Seond Edition, Wiley, 1995

[9] M. Stuhlinger, Die Echtzeit-Datenverarbeitung des IBIS Experiments auf INTEGRAL, Diplomarbeit, Tübingen, IAAT, 1999

[10] E. Göhler, Entwicklung einer Steuerungsoftware des IBIS.Experiments auf INTEGRAL, Diplomarbeit, Tübingen, IAAT, 1999

[11] N. v. Krusenstiern, Entwicklung und Realisierung des Hardware Event Processors für den Imager IBIS an Board des Satelliten INTEGRAL, Diplomarbeit, Tübingen, IAAT, 1998

[12] R. Volkmer. HEPI Interface Description. Technical Report, IN-IM-TUE-TN/EL-018, IAAT, 1998-2000

[13] R. Volkmer. HEPI Design Report. Technical Report, IN-IM-TUE-DES-001, IAAT, 1998-2000

[14] P. J. Ashenden, The Designer's Guid To VHDL. Seond Edition, Morgan Kaufmann Publishers, 2001

[15] http://www.vhdl.org/

[16] http://www.edif.org/

[17] http://www.mentor.com/

[18] http://www.model.com/

[19] http://www.xilinx.com/

[20] The Programmable Logic, Data Book, Xilinx, 1996

[21] http://www.exemplar.com/

[22] http://www.cvshome.org/

[23] http://www.europractice.com/

Danksagung

Diese Diplomarbeit wäre ohne Unterstützung nicht möglich gewesen.
Stellvertretend für alle, welche mir in dieser Zeit beistanden, möchte ich folgende erwähnen:

Prof.Dr. Rüdiger Staubert, dem Leiter der Gamma-Gruppe an unserem Institut danke ich für das Thema dieser Arbeit und das mir entgegengebrachte Vertrauen diese Aufgabe zu meistern.

Dr. Eckard Kendziorra, als Ansprechpartner für allerlei organisatorische/technische Fragen.

Dr. Reiner Volkmer, dem Projektleiter für den Beitrag (Hard- und Software) des AIT am IBIS-Satellit. Ihm gilt mein besonderer Dank für die sprichwörtliche „offene Tür“ weit über die HEPI hinaus.

Bärbel Kretschmar und Nikolai von Krusenstiern, den beiden *Hard*werkern, von denen ich das HEPI Projekt übernommen habe und deren fruchtbare Zusammenarbeit.

Martin Stuhlinger und Eckart Göhler, den beiden *Soft*werkern der DPE. Sie haben für die Hardware-Tests so manche Nachtschicht mit mir einlegen müssen.

DD&T, Reutlingen für die kollegiale Zusammenarbeit und Hilfe beim Erstellen und „Entwanzen“ von VHDL-Modulen.

Meine Familie!