Christian Dierker

Fehlertolerante Instrumentenrechner für kompakte Kameras auf Raumsonden

Cuvillier Verlag Göttingen

Fehlertolerante Instrumentenrechner für kompakte Kameras auf Raumsonden

Von der Fakultät Elektrotechnik und Informationstechnik
der Technischen Universität Carolo-Wilhelmina
zu Braunschweig

zur Erlangung der Würde
eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Dissertation

von
Dipl.-Ing. Christian Dierker
aus Löningen

Eingereicht am:	08.01.2007
Mündliche Prüfung am:	13.03.2007
Berichterstatter:	Prof. Dr.-Ing. Harald Michalik
	Prof. Dr. Nicolas Thomas

2007

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.
1. Aufl. - Göttingen : Cuvillier, 2007
Zugl.: (TU) Braunschweig, Univ., Diss., 2007

978-3-86727-288-9

Nonnenstieg 8, 37075 Göttingen
Telefon: 0551-54724-0
Telefax: 0551-54724-21
www.cuvillier.de

1. Auflage, 2007
Gedruckt auf säurefreiem Papier

978-3-86727-288-9

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Datentechnik und Kommunikationsnetze (IDA) der Technischen Universität Carolo-Wilhelmina zu Braunschweig.

Mein herzlicher Dank zum Abschluss meiner Dissertation gilt Prof. Dr.-Ing. Harald Michalik für die Übernahme des Referats. Von ihm habe ich während der gesamten Zeit am Institut umgehende Unterstützung und motivierende Anregungen erhalten.

Ich danke Prof. Dr. Nicolas Thomas für die bereitwillige Übernahme des Koreferats und Prof. Dr.-Ing. Rolf Ernst, der als Vorsitzender der Prüfungskommission eine angenehme Atmosphäre schaffte.

Ich schätze sehr die Zusammenarbeit mit Prof. Dr.-Ing. Fritz Gliem und danke ihm dafür.

Gang Zhou unterstützte mich mit seiner Master-Arbeit auf sehr effektive Weise und Karsten Kamm führte unterschiedliche Messungen durch. Vielen Dank!

Die Arbeit am IDA war sehr abwechslungsreich und hat mir viel Spaß gemacht. Die internationalen Projekte haben mir viele persönliche Erfahrungen gebracht. Vielen Dank an meine Kollegen für die gute Zusammenarbeit.

Ich danke meiner Familie und insbesondere meiner Frau Eva für die große Unterstützung, die ich während der Dissertationsarbeit erhalten habe.

Christian Dierker Weyhe, im Juli 2007

Kurzfassung

Das Ziel bei der Entwicklung eines Rechners für wissenschaftliche Instrumente auf Weltraumsonden ist es, den wissenschaftlichen Nutzen des Instruments durch angepasste Steuerung oder Datenvorverarbeitung zu erhöhen. Bei Rechnern für Kamerainstrumente bedeutet dies, mehr Bilder pro Zeiteinheit zu prozessieren und für die Übertragung zur Erde zur Verfügung zu stellen. Dieser Durchsatz wird im Wesentlichen durch die Kompressionsroutinen bestimmt, die aufgrund der erforderlichen hohen Bildqualität viel Rechenzeit benötigen.

Mit dem Übergang von einem traditionellen strahlungsfesten zu einem strahlungstoleranten Design ergeben sich neue Wege beim Aufbau eines kleinen Instrumentenrechners. Exemplarisch für die Datenverarbeitungsfunktionen eines Kamerarechners wird anhand der qualitativ hochwertigen *Wavelet*-basierten JPEG2000-Kompression gezeigt, wie unter den Randbedingungen einer Weltraummission die Ausbeute, der Ressourcenbedarf und die Fehlertoleranz (Toleranz gegenüber strahlungsbedingten Fehlern) einer DPU für ein kompaktes Kamerainstrument optimiert werden können. Zur Relativierung der Werte des strahlungstoleranten Ansatzes werden Vergleichsimplementierungen basierend auf strahlungsfesten Bausteinen durchgeführt. Beim Übergang zu einem strahlungstoleranten Design zum Aufbau eines kleinen Instrumentenrechners ist die Verfügbarkeit strahlungstoleranter FPGAs mit großen Ressourcen (Systemgatter) von hoher Bedeutung. Hierdurch kann innerhalb eines Bausteins eine Standard-CPU für universelle Aufgaben (Steuerung) zusammen mit dedizierter Logik für eine spezielle Aufgabe (Kompression) integriert werden. Rechenintensive Softwareroutinen können durch schnell und vor allem energieeffizient ausführbare Hardwaremaschinen ersetzt werden, wie es z. B. für die Kompression sinnvoll ist.

Zur Steigerung der Performanz für kompakte Kameras wurde daher ein skalierbarer IDA JPEG2000 Core entworfen, der eine flexible Partitionierung in Hardware und Software erlaubt. Der skalierbare IDA JPEG2000 Core wurde u. a. zusammen mit einer CPU in einem FPGA innerhalb des strahlungstoleranten Instrumentenrechners integriert. Zu Vergleichszwecken erfolgte parallel eine Implementierung des IDA JPEG2000 Core innerhalb einer strahlungsfesten DPU. Durch die energieoptimierte hardwareunterstützte Kompression kann der Durchsatz bei qualitativ hochwertigen Kompressionsergebnissen bis um den Faktor 30 erhöht werden. Es wird für typische Bilderzeugungsraten von einem Bild der Größe 1k × 1k × 14 Bit je 1-2 s durch eine Online-Kompression ein hoher wissenschaftlicher Nutzen des Instruments erzielt. Eine derartige FPGA-basierte Lösung bietet genügend Ressourcen für missionsspezifische Erweiterungen. Während die erzielten Zuverlässigkeiten bei strahlungsfester und strahlungstoleranter DPU gleichwertig sind, ist die strahlungsbedingte Rate reversibler Fehler bei der strahlungstoleranten DPU wesentlich höher. Hier ergibt sich für typische Umgebungsbedingungen rechnerisch eine Nichtverfügbarkeit von 1 min pro Jahr. Da in den meisten Missionen diese moderate Strahlungsempfindlichkeit akzeptiert werden kann, wird mit der strahlungstoleranten Lösung (FPGA und weitere strahlungstolerante Baugruppen) ein kompakter Instrumentenrechner mit hoher Performanz bei energieoptimiertem Betrieb und hoher Flexibilität erzielt.

Inhaltsverzeichnis

1 Einleitung

Auf Raumsonden zur Planetenbeobachtung befinden sich üblicherweise Kamerainstrumente. Sie dienen z. B der Erfassung und Kartographierung von Oberflächen oder der Analyse von Atmosphärenphänomenen. Darüber hinaus ermöglichen sie den Zugang der Allgemeinheit zu den Ergebnissen von Weltraummissionen. Daher gibt es kaum noch Missionen zur Planetenbeobachtung, die nicht mit mindestens einer Kamera ausgestattet sind. Solche Missionen und insbesondere solche mit Landeeinheiten erfordern kompakte, leistungsfähige Kameras, die auf die wissenschaftlichen Erfordernisse der Missionen zugeschnitten sind.

Die Schnittstelle zwischen der Raumsonde und den einzelnen Elektroniken der Instrumentensensoren bilden die Instrumentenrechner (*Data Processing Unit*, DPU). Sie übernehmen sehr spezifische Steuerungs- und Konvertierungsaufgaben sowie zunehmend auch komplexe Aufgaben zur Datenaufbereitung an Bord. Klassische Aufgaben einer DPU sind:

- Akquisition von Sensordaten
- Datenspeicherung
- Ausführung der Kommandos von der Raumsonde
- Senden von Telemetrie-Daten (TM) zur Raumsonde
- Erzeugung von Statusdaten (*Housekeeping,* HK) und Instrumentenüberwachung
- Datenkompression zur Reduzierung der Eingangsdatenmenge
- Datenvorverarbeitung, die aufgrund der Datenmengenreduktion nicht nach der Übertragung ausgeführt werden kann
- Ausführung von Skripten des Anwenders (z. B. Messsequenzen).

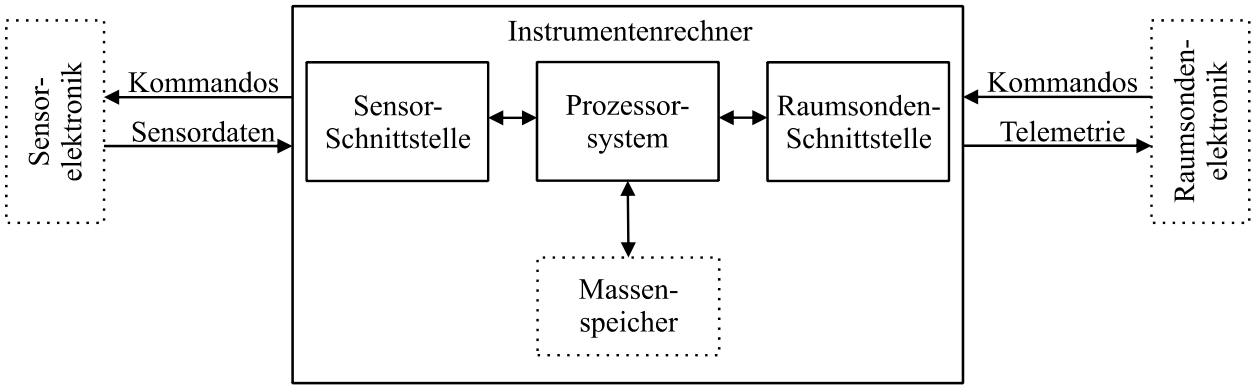

Bild 1.1: Instrumentenrechner

Bild 1.1 zeigt den typischen Aufbau eines Instrumentenrechners, der für unterschiedliche Sensorarten ähnlich ist. Die Instrumente für kompakte Kameras zeichnen sich häufig dadurch aus, dass sie eine hohe Datenmenge produzieren und eine hohe Rechenleistung zur

Bilddatenverarbeitung erfordern. In vielen Fällen gilt dafür das Prinzip, nur die absolut notwendige Bilddatenverarbeitung an Bord der Raumsonde durchzuführen, um Komplexität und Energie zu sparen. Daher reduziert sich die An-Bord-Bilddatenverarbeitung auf die Datenkompression und die dafür notwendigen Vorverarbeitungsschritte. Die benötigte Rechenleistung wird somit i. Allg. durch den Kompressionsalgorithmus bestimmt. Eine Kompression ist wünschenswert, damit die durch Sendeleistung und Entfernung beschränkte Datenrate zur Erde optimal genutzt wird oder um nicht unnötigerweise viel internen Speicher zu beanspruchen. Dafür sind hochwertige und effiziente Kompressionsalgorithmen notwendig. Zur Pufferung der Bilddaten verfügen Instrumentenrechner für Kameras vielfach über eigenen Massenspeicher. Die Bilddatenverarbeitung muss auf diese Weise nicht zwangsweise mit der Erzeugung der Roh-Bilddaten Schritt halten und für die Bedienung der Telemetriestrecke steht mit dem Massenspeicher ein genügend großer Zwischenpuffer zur Verfügung.

Die effiziente Implementierung der Bilddatenverarbeitung in der DPU bekommt damit eine wichtige Rolle bei der Auslegung eines Kamerainstruments: durch die Datenkompression wird die wissenschaftliche Ausbeute erhöht, z. B. gemessen an der Zahl der pro Missionszeiteinheit gewonnenen und übertragenen Bilder. Andererseits ist dafür die Implementierung komplexer Algorithmen in der DPU notwendig, für die Designkomplexität und höherer Energiebedarf auf der Aufwandsseite stehen.

Der Kernansatz in dieser Arbeit ist, den Entwurf einer Kamera-DPU unter den Optimierungsgesichtspunkten Ausbeute, Ressourcenbedarf und Fehlertoleranz unter den Randbedingungen einer Weltraummission zu untersuchen.

Anforderungsprofil für eine Kamera-DPU

Am Institut für Datentechnik und Kommunikationsnetze wurden für eine ganze Reihe von Instrumenten-DPUs erstellt. DPUs neueren Typs wurden für die beiden Instrumente Venus Monitoring Camera (VMC) der ESA-Sonde Venus Express und Dawn Framing Camera (DFC) der NASA-Sonde DAWN entwickelt. Typische Kennwerte dieser und auch anderer Kameras zur Planetenbeobachtung sind:

- Bildgröße: $\geq 1.024 \times 1.024$ Pixel
- Radiometrische Auflösung pro Pixel: ≥ 14 Bit
- Grauwertbild
- Gewünschte Akquisitionsrate: alle 1-2 s ein Bild
- Typische Kompressionsfaktoren: 5-8

Die Bearbeitung der Bilddaten erfolgt im Instrument selbst. Die prinzipielle Möglichkeit, die notwendige Rechenleistung durch einen zentralen Rechner auf der Raumsonde zur Verfügung zu stellen, ist aus Gründen der Heterogenität der Instrumente sowie unter dem Aspekt der üblicherweise stark räumlich verteilten Entwicklung der Instrumente nicht praktikabel. Zudem

lassen sich bei dezentraler Verteilung der Rechenleistung die Schnittstellen zur Raumsonde besser einheitlich definieren.

In Abhängigkeit von der Mission werden unterschiedliche Anforderungen an den Instrumentenrechner gestellt. Allgemeine Anforderungen, die für alle Instrumentenrechner gelten, sind:

- Aufgaben angepasste Rechenleistung
- Adäquate Zuverlässigkeit und Verfügbarkeit
- Einhaltung missionsspezifischer Strahlungstoleranzen
- Geringer Energiebedarf
- Platzsparender Aufbau und geringes Gewicht
- Flexibles Design für verkürzte Entwicklungszeiten
- Modulares Design zur Langlebigkeit eines Entwicklungsansatzes

Entwurfs- und Implementierungsaspekte

Beim Entwurf von Elektronik für Raumsonden müssen die im Weltraum auftretenden Umweltbedingungen beachtet werden. Insbesondere hat die dort auftretende Strahlung einen erheblichen Einfluss auf das Verhalten von Elektronik. Sie kann zu temporärem aber auch zu permanentem Fehlverhalten führen. Der traditionelle Entwurf von Elektronik für die Raumfahrt beinhaltet die Verwendung strahlungsfester (*Radiation Hardened*, RH) Bauteile. Diese sind speziell für den Einsatz im Weltraum entwickelt und erhalten ihre Strahlungstoleranz durch entsprechende Chip-Herstellungs-Technologien. Aufgrund des sehr kleinen Marktes ist die Verfügbarkeit von RH Bauteilen sehr stark begrenzt. Auf strahlungsfesten Bauteilen basierende Rechnersysteme sind als Plattform für Steuerrechner von Raumsonden auf dem Markt verfügbar, allerdings sind sie häufig für den sehr speziellen Einsatz in kleinen Instrumentenrechnern wegen ihres Ressourcenbedarfs nahezu ungeeignet. Bei Instrumentenrechnern werden daher oft maßgeschneiderte Designs eingesetzt.

Da wissenschaftliche Instrumente meist nur für kleine Stückzahlen entworfen werden, ist der Entwurf gleichbedeutend mit der Entwicklung eines Prototyps für ein spezielles System. Dieses wird durch die Verwendung programmierbarer Bausteine (*Field Programmable Gate Array*, FPGA) zur freien Logikimplementierung vereinfacht, denen eine wichtige Rolle beim Entwurf eines Instrumentenrechners zukommt. Mit der Verwendung weltraumtauglicher FPGAs kann man teilweise das Problem umgehen, dass die Anzahl an strahlungstoleranten Bausteinen und damit die verfügbare Funktionalität sehr stark begrenzt ist, indem man die benötigte Funktionalität selbst entwirft. Wie Anwendungsspezifische Integrierte Schaltungen (*Application Specific Integrated Circuit*, ASIC) können FPGAs benutzt werden um Masse, Volumen und Gewicht zu sparen. Bei der geringen Stückzahl sind FPGAs wesentlich kostengünstiger als ASICs.

Zudem ist die Erstellung bis hin zur endgültigen Designverifikation in wesentlich kürzerer Zeit möglich und Änderungen in den Spezifikationen können vergleichsweise schnell eingebaut werden. Bei der Entwicklung von Prototypen sind diese Vorteile von hohem Wert. Bei der Benutzung von FPGAs liegt die Verantwortung für die Entwicklung und die Testzyklen üblicherweise innerhalb eines Hauses und eine kosteneffektive Entwicklung und Produktion von kleinen Serien ist möglich.

Die derzeit zur Verfügung stehenden großen weltraumtauglichen FPGAs mit mehreren Millionen Systemgattern können zusätzlich zu den kombinatorischen und sequenziellen Zellen auch fest verdrahtete Module wie internen Speicher oder Multiplikationseinheiten beinhalten. *System-on-Chip*-Entwürfe für die Implementierung von DPUs sind damit möglich. Insbesondere der in den vergangenen Jahren erheblich angestiegene interne Speicher ermöglicht es, zunehmend spezielle Funktionen direkt im FPGA auszuführen und den Arbeitsdatenstrom außerhalb des FPGA stark zu reduzieren.

Die Integration mehrerer Funktionen in einem FPGA hat folgende Vorteile:

- Höhere Systemintegration
- Höhere Performanz und geringerer Energiebedarf durch verringerte Anzahl von *Chip-zu-Chip*-Verbindungen
- Erhöhte Zuverlässigkeit aufgrund reduzierter Bauteilanzahl
- Vereinfachung des Boardlayouts
- Erhöhte Flexibilität durch Mehrfachprogrammierbarkeit (bei SRAM-basierten FPGAs)
- Wiederverwendung von IP Cores und insgesamt kürzere Entwicklungszeiten

Ein Hauptbestandteil der Entwicklung ist der zeitliche Aufwand für den Entwurf der Schaltungsblöcke innerhalb des FPGA. Die Verwendung großer FPGAs in Instrumentenrechnern führt dazu, dass sich ein Großteil der Logik auf wenige Bausteine konzentriert. Eine modulare Designmethode mit eindeutigen Schnittstellendefinitionen vereinfacht die Wiederverwendung solcher Blöcke in nachfolgenden Projekten, erhöht die Langlebigkeit der Entwurfsarbeit und reduziert die Kosten in nachfolgenden Projekten. Erfolgt das Design der Logikimplementierung durch eine Beschreibungssprache wie z. B. VHDL (*Very High Speed Integrated Circuit Hardware Description Language*), kann eine einfache Implementierung auch in zukünftige FPGA-Familien erzielt werden. Bis auf wenige Ausnahmen haben Änderungen innerhalb des FPGA keine Auswirkung auf das Layout der Platine, so dass das Layout schon in einem frühen Stadium festgelegt werden kann. Die ersten Arbeiten an Logikmodulen können zudem häufig bereits auf vergleichbaren Entwurfsplattformen erfolgen. Die Nachteile der Verwendung großer FPGAs sind im Wesentlichen bei der verringerten Zugänglichkeit interner Funktionen und den daraus resultierenden schlechteren Testmöglichkeiten zu sehen. Ein intensiver Einsatz von Simulationssoftware kann aber diesen Nachteil ausgleichen.

Instrumentenrechner werden meistens als nicht missionskritisch eingestuft. Während die Steuerung einer Raumsonde z. B. zum Zeitpunkt eines Landemanövers auch nicht nur für kurze Zeit ausfallen darf, kann beim Bearbeiten von wissenschaftlichen Daten in einer DPU eine niedrigere Verfügbarkeit akzeptiert werden. Beim Aufbau einer DPU mit FPGAs können andere Ansätze als der klassische Weg mit einer strahlungsfesten CPU zur Bearbeitung von Daten gewählt werden. Während sich FPGAs bisher lediglich am Rande eines Designs befanden, so rücken sie mit zunehmender Größe immer mehr ins Zentrum des Designs. Neben dem klassischen Einsatz eines FPGA als Verbindungselement zweier Einheiten (*Gluelogic*), können nun FPGAs größere Module wie CPUs und Co-Prozessoren enthalten. Dieses ermöglicht es, allgemeine Funktionen (z. B. mit CPU) sowie spezielle Funktionen (z. B. mit Co-Prozessoren) in demselben Baustein zu vereinen und die Performanz stark anzuheben. Aufgrund der ausreichend zur Verfügung stehenden Ressourcen kann eine Aufteilung der Rechenleistung (z. B. zur Bilddatenverarbeitung) in Software und Hardware erfolgen. In kompakten Kameras wurde bisher die Kompression der Bilddaten hauptsächlich durch Softwareroutinen mit entsprechend niedrigem Durchsatz durchgeführt. Eine Hardwareimplementierung war entweder sehr aufwendig (hoher Ressourcenbedarf und keine Anwendungsmöglichkeit für eine kompakte Kamera) oder aber qualitativ nicht ausreichend. Mit größeren FPGAs kann daher die Designoptimierung durch eine qualitativ hochwertige hardwareunterstützte Kompression mit einem kompakten, Ressourcen schonenden Aufbau einer Kamera vereint werden.

Aktuell sind zwei große FPGA-Familien zur höheren Systemintegration in Instrumentenrechnern von Interesse. Zum einen ist es die RTAX-S Familie der Firma Actel, zum anderen die RT-Virtex-II Familie der Firma Xilinx. Bis auf den internen Speicher ist die RTAX-S Familie weitaus robuster gegenüber der im Weltraum auftretenden Strahlung als die SRAM-basierte RT-Virtex-II Familie. Dieses ist insbesondere auf die verwendete *Antifuse*-Technologie und Implementierung dreifach-redundanter Zellen (*Triple Module Redundancy*, TMR) bei Flipflops zurückzuführen. Bei den SRAM-basierten RT-Virtex-II FPGAs können typische strahlungsbedingte Fehler nicht nur im internen Speicher, sondern auch z. B. in der Kombinatorik oder der Konfiguration des Bausteins auftreten. Hierdurch treten unterschiedliche Fehlertypen auf. Die Bausteine der RT-Virtex-II Familie stellen im Vergleich zur RTAX-S Familie jedoch um den Faktor 1,5 mehr Systemgatter und einen um den Faktor 4,8 größeren internen Speicher zur Verfügung.

Bei der Auswahl eines FPGA können die hohe Performanz und die hohe verfügbare Menge an Ressourcen der RT-Virtex-II Bausteine trotz höherer Empfindlichkeit gegenüber Strahlung ausschlaggebend sein. Zudem ist die Entwicklung mit SRAM-basierten Bausteinen praktikabler, da sie mehrfach programmiert werden können. Insgesamt erhält man mit Verwendung eines großen SRAM-basierten FPGA ein flexibles System während der Entwicklung. Das endgültige FPGA-Design kann kurz vor den abschließenden Tests vor Auslieferung des Instrumentenrechners integriert werden. Diese Vorgehensweise führt zu kürzeren Entwicklungszeiten und damit zu niedrigeren Kosten.

Zielstellung der Arbeit

Der mögliche Einsatz der, bezogen auf die Performanz und der zur Verfügung stehenden Ressourcen, überlegenen größeren SRAM-basierten FPGAs mit Weltraumtauglichkeit führt zu der Frage, inwieweit sie trotz ihrer vergleichsweise hohen Empfindlichkeit gegenüber Strahlung und der dadurch ggf. reduzierten Verfügbarkeit in Instrumentenrechnern innerhalb kompakter Kameras vorteilhaft eingesetzt werden können. Den zu erwartenden höheren Fehlerraten stehen unter Betrachtung der Randbedingungen im Weltraum die hohe Performanz, der relativ geringe Leistungsverbrauch und die höheren Integrationsmöglichkeiten für dedizierte Hardware-prozessoren gegenüber.

Das Ziel bei der Entwicklung eines Instruments ist ein hoher wissenschaftlicher Nutzen, der bei Kamerainstrumenten i. Allg. einem hohen Durchsatz an Bildern entspricht. Um bei gleicher Datenrate mehr Bilder zu übertragen, werden Bilder komprimiert. Eine Kompression wird nur akzeptiert, wenn sie auf einem Algorithmus beruht, der qualitativ hochwertige Kompressions-ergebnisse liefert. Derartige Algorithmen benötigen jedoch üblicherweise viel Rechenleistung, so dass der Durchsatz eines Kamerainstruments häufig durch die Kompression begrenzt ist.

Im Folgenden soll gezeigt werden, wie eine hohe Rechenleistung mit akzeptablen Werten für Zuverlässigkeit und Verfügbarkeit und kompakter Bauweise verbunden werden kann. Hiermit entfernt man sich vom traditionellen Ansatz einer strahlungsfesten DPU – strahlungsfeste Bereiche werden in strahlungstolerante Bereiche mit Fehlertoleranz, d. h. Toleranz gegenüber strahlungsbedingten Fehlern, überführt. Analysiert und verglichen werden in dieser Arbeit strahlungsfeste und strahlungstolerante DPU-Implementierungen. Die strahlungstolerante DPU enthält als Kernkomponente einen SRAM-basierten RT-Virtex-II FPGA, der aufgrund des zu erwartenden Vorteils beim Aufbau einer kompakten Kamera mit hoher Performanz gegenüber dem RTAX-S FPGA den Vorrang erhält.

Die Anforderungen an notwendige Rechenleistung werden exemplarisch durch die Bearbeitung des Kompressionsalgorithmus JPEG2000 umgesetzt, der auf der *Wavelet*-Transformation beruht. Dabei ist es nicht das Ziel, ein möglichst hochwertiges Verfahren zu finden. Hingegen soll eine maßgeschneiderte Lösung basierend auf einem hochwertigen *State of the Art* Kompressions-verfahren entwickelt werden, die in einer entsprechenden kompakten DPU-Architektur für Missionen zur Planetenbeobachtung im Weltraum eingesetzt werden kann. Hierbei werden sowohl die Implementierungen des angepassten Algorithmus in Software als auch eine Umsetzung des Verfahrens in dedizierte Hardware als Co-Prozessor-Lösung vorgenommen.

Der Einfluss der Strahlung im All auf die DPUs wird in Anlehnung an die ESA-Missionen Venus Express und ExoMars sowie exemplarisch für einen Planeten mit Magnetfeld für eine Mission im polaren *Low Earth Orbit* (LEO) bestimmt.

In Kapitel 2 werden die Randbedingungen für den Weltraumeinsatz besprochen und strahlungsbedingte Fehlerraten berechnet. Sie verdeutlichen den Unterschied zum Entwurf für erdgebundene Anwendungen und bilden die Grundlage zur Ermittlung der zu erwartenden Fehlerraten für die verschiedenen DPU-Implementierungen.

Kapitel 3 stellt die Besonderheiten des JPEG2000-Standards dar. Es verdeutlicht als Grundlage für die Hardware-Implementierung des Verfahrens, welche Optimierungsmöglichkeiten bei Ausgliederung von Algorithmusteilen aus Software in Hardware vorhanden sind.

In Kapitel 4 werden die Besonderheiten einer Architektur für Instrumentenrechner in Kameras analysiert. Insbesondere ist der Einsatz SRAM-basierter FPGAs im Weltraum als eine Designvariante heraus gestellt.

Kapitel 5 behandelt die Implementierung eines traditionellen Entwurfs einer strahlungsfesten DPU mit einem SPARC-V8-kompatiblen Prozessor in ASIC-Ausführung. Es wird als Variante auch ein Massenspeicher implementiert, der die strahlungsfeste DPU dann in eine überwiegend strahlungsfeste DPU überführt. Die Softwarekompression der Bilddaten bestimmt die Performanz.

In Kapitel 6 wird eine strahlungstolerante DPU mit Verwendung eines SRAM-basierten RT-Virtex-II FPGA und der Implementierung eines Massenspeichers vorgestellt. Aus Vergleichsgründen zu Kapitel 5 beinhaltet der Aufbau ebenfalls einen SPARC-V8-kompatiblen Prozessor, der zusammen mit Logikblöcken wie Schnittstellenlogiken oder Speicherschutz in einem FPGA implementiert ist. Es sind mehrere DPU-Bereiche aus einer strahlungsfesten Version in eine strahlungstolerante Version überführt. Hier bestimmt die Softwarekompression ebenfalls die Performanz.

Mit Kapitel 7 wird eine hardwarebeschleunigte Kompression auf Basis des JPEG2000-Algorithmus vorgestellt. Die Analyse des JPEG2000-Standards in Kapitel 2 zeigt die Möglichkeit, bestimmte Algorithmusanteile zur Geschwindigkeitssteigerung in Hardware ausführen zu lassen. Darauf basierend wurde ein Ressourcen schonender, skalierbarer JPEG2000 Co-Prozessor entwickelt. Dieser ermöglicht eine starke Erhöhung der Performanz und damit des wissenschaftlichen Nutzens des Instruments.

In Kapitel 8 wird die in Kapitel 7 gezeigte hardwarebeschleunigte Kompression in die strahlungsfeste DPU aus Kapitel 5 und in die strahlungstolerante DPU aus Kapitel 6 integriert. Während im strahlungsfesten Design auf eine andere FPGA-Familie ausgewichen und zur Kompression zusätzlicher Zwischenspeicher in Form eines dedizierten Bausteins zur Verfügung gestellt werden muss, kann die Hardwarebeschleunigung bei der strahlungstoleranten DPU zusammen mit dem Prozessor in dem bereits vorhandenen SRAM-basierten RT-Virtex-II FPGA implementiert werden. Abschließend werden in Kapitel 8 Vergleiche bzgl. Performanz, Zuverlässigkeit, Strahlungsfestigkeit, Leistungs- und Energiebedarf, Platzbedarf, Volumen und

Masse sowie Flexibilität zwischen der strahlungsfesten und strahlungstoleranten DPU mit und ohne hardwarebeschleunigter Kompression durchgeführt.

Kapitel 9 gibt eine Zusammenfassung der Ergebnisse dieser Arbeit mit einem Ausblick auf mögliche Erweiterungen.

2 Einsatz im Weltraum

Bei dem Entwurf von Elektronik für den Einsatz in der Raumfahrt sind spezielle Randbedingungen zu berücksichtigen. Nur mit der Kenntnis z. B. der im Weltraum auftretenden Strahlung kann ein gezielter Entwurf erfolgen und können unnötige Überdimensionierungen vermieden werden. Im Folgenden werden daher die Eckpunkte für die richtige Auslegung besprochen.

2.1 Einführung

Ein Instrumentenrechner für die Raumfahrt muss für spezielle Anforderungen entworfen werden:

- Fehlende Wartungsmöglichkeiten
- Hohe Zuverlässigkeit, Datenintegrität und Verfügbarkeit
- Geringe Masse und geringer Energiebedarf
- Besondere Umweltumgebungen im Weltraum wie z. B. Strahlung (siehe Abschnitt 2.2)

Die Anforderungen variieren in Abhängigkeit von der Mission und dem Einsatz des Instruments. Im Allgemeinen sind Instrumente auf Raumsonden für lange Einsätze von einigen Monaten bis zu mehreren Jahren geplant. Bei *Deep Space* Missionen liegen allein die Transferzeiten von der Erde bis zur eigentlichen Zielposition bei mehreren Jahren.

Die **Wartung** eines Instrumentenrechners ist nach dem Start ausschließlich über die installierte Software möglich. Mit dem aktuellen Stand der Technik wird diese Einschränkung durch die Verwendung wiederprogrammierbarer SRAM-basierter FPGAs und damit durch den Hardwareanteil relativiert, der über veränderbare Konfigurationsdaten definiert ist.

Die Zuverlässigkeit eines Systems wird bestimmt durch die Überlebenswahrscheinlichkeit am Ende einer Mission. Sind die Fehlerwahrscheinlichkeiten $P(X_n)$ der N Komponenten unabhängig voneinander, ergibt sich die **Systemzuverlässigkeit** in einer Reihenschaltung bekanntermaßen zu:

$$R = P(X_0) \cdot P(X_1) \cdot \ldots \cdot P(X_{N-1}). \tag{2.1}$$

Die Zuverlässigkeit eines gesamten Systems ist somit kleiner als die Zuverlässigkeit der schlechtesten Komponente. Hoch integrierte Bausteine haben in der Regel eine kleinere Ausfallwahrscheinlichkeit als ihre vergleichbare Ausführung aus mehreren geringer integrierten Bausteinen und es lässt sich hiermit in der Regel ein zuverlässigeres System aufbauen. Damit sind zunehmend integrierte Architekturen auch unter dem Gesichtspunkt einer ansteigenden Zuverlässigkeit zu sehen. Die Zuverlässigkeit kann durch die Auswahl spezieller Bauteile oder durch strukturelle Maßnahmen erhöht werden. Die Bauteile werden nach Möglichkeit nach

höchsten Qualifikationsstandards ausgewählt. Der kleine Markt für solche Bauteile mit den zugehörigen Beschaffungsschwierigkeiten und hohen Kosten je Einheit führen allerdings dazu, dass teilweise auf kommerzielle Bauteile zurückgegriffen werden muss. Durch Maßnahmen wie z. B. die Bestimmung der Strahlungstoleranz können kommerzielle Bauteile für den Einsatz in der Raumfahrt qualifiziert werden. Strukturelle Maßnahmen wie Überstromschutz bei einem *Latchup* (siehe Abschnitt 2.3.2) oder Einbau von Redundanz in der Hardware (siehe Kapitel 4) erhöhen die Systemzuverlässigkeit. Durch einen längeren Betrieb von z. B. mehr als 400 h bei sowohl hohen Temperaturen als auch erhöhten Betriebsspannungen zur künstlich beschleunigten Alterung lassen sich Frühausfälle identifizieren. Dieses *Burn-In* wird in der Regel vor Auslieferung durchgeführt.

Die Häufigkeit von fehlerhaften Messungen und Inkonsistenzen in den Daten wird durch die **Datenintegrität** beschrieben. Die Anforderungen an die Datenintegrität eines Instrumentenrechners sind von der Mission abhängig. Allgemein gilt jedoch, dass die Datenintegrität eines Instruments nicht erheblich größer als die Datenintegrität der Raumsonde sein muss. Diese wird mindestens bestimmt durch die Bitfehlerrate auf der Telemetriestrecke (typisch $< 10^{-6}$ [Fic98]). Durch entsprechende Kanalcodierung wird üblicherweise die für die Nutzdaten geltende Fehlerrate um mehrere Größenordnungen abgesenkt ($10^{-9} – 10^{-11}$). Diese Rate beeinflusst dann die Datenformate, die üblicherweise so ausgelegt sind, dass die Fehlerfortpflanzung (z. B. bei komprimierten Bilddaten) auf Teilblöcke begrenzt ist. Somit sollte die Fehlerrate für den Instrumentenrechner einer Kamera so gering sein, dass nur sehr wenige Bilder einer Messsequenz gestört sind oder ganz ausfallen (z. B. 1 aus 1.000 Bildern). Damit ist für die korrekte Durchführung der Messsequenzen eher die Verfügbarkeit des Instrumentenrechners ein Maß.

Die **Verfügbarkeit** eines Instruments ergibt sich aus der Häufigkeit eines Systemzusammenbruchs und der anschließenden Zeit bis zur Wiederherstellung der vollen Funktionalität. Oftmals ist die Verfügbarkeit aufgrund der beschränkten Ressourcen Masse und Energie jedoch verringert. Für Instrumenten-DPUs ist in [Ger01] eine tolerierbare Nichtverfügbarkeit von einigen Sekunden bis wenige Minuten in der Größenordnung von einmal je Monat bis einmal je Tag angegeben. Während eine Nichtverfügbarkeit bei z. B. Landeprozeduren nicht akzeptiert werden kann, ist sie bei sich teilweise wiederholenden Abläufen, wie z. B. zur Kartographierung, in Grenzen tolerierbar. Ein System mit einer vorgegebenen Verfügbarkeit von 0,99999 darf im Jahr nur maximal 5 Minuten nicht verfügbar sein.

Zur Reduzierung der **Masse** besteht bei der Entwicklung einer DPU ein großer Anreiz, die Bauteile dicht zu packen und, sofern möglich, leichte Bauteile zu verwenden. Leichtere Bauteile haben den Vorteil, dass sie den Vibrationen beim Start der Trägerrakete weniger stark ausgesetzt sind. Durch die Integration mehrerer Funktionen in einem Baustein ist eine zunehmende Miniaturisierung zu verzeichnen. Mit höherer Packungsdichte und somit auch einer höheren spezifischen Masse der DPU besteht die Möglichkeit, deren Gehäuse entsprechend klein und leicht ausfallen zu lassen.

Die an Bord einer Raumsonde zur Verfügung stehende **Energie** ist stark begrenzt. Die typische verfügbare mittlere Leistung bei Instrumentenrechnern liegt bei wenigen Watt. Durch die Verwendung Strom sparender Elektronik und adäquates Power Management innerhalb des Instruments kann die mittlere Leistung reduziert werden. Bauteilgruppen werden bei längeren Betriebspausen abgeschaltet und die zur Verfügung gestellte Rechenleistung für einen optimierten Energiebedarf der jeweiligen Situation angepasst. Der Energiebedarf [Ws] kann als Maß für die Effizienz von Rechenoperationen (z. B. Energiebedarf zur Kompression eines Bildes) angesehen werden. Bei Missionen z. B. zur Venus oder zum Merkur kann verhältnismäßig viel Energie geliefert werden. Es ist jedoch zu beachten, dass die auf der Erde überwiegende Wärmeabfuhr durch Konvektion im Weltraum nicht existiert. Es ist auch hier ein moderater Energiebedarf erforderlich, da durch eine zu hohe Energieaufnahme thermische Probleme auftreten können. Der Energieverbrauch und der Einsatz von Masse sind für Instrumente, die auf Landeeinheiten interplanetarer Missionen eingesetzt werden, wesentlich stärker beschränkt.

2.2 Strahlungsumgebung

Beim Entwurf von Elektronik für den Einsatz auf Raumsonden ist es wichtig, sich mit der im Weltraum auftretenden Strahlung auseinanderzusetzen. Sie stellt die Besonderheit bei der Entwicklung dar und kann unterschiedliche Auswirkungen auf die Elektronik haben. Bei terrestrischen Anwendungen sind vergleichbare Auswirkungen, z. B. durch Neutronen-induzierte Bitwechsel, nur äußerst selten zu beachten. Im Weltraum ist die Elektronik Flüssen von energiereichen Elektronen und Ionen, in der Mehrzahl Protonen, ausgesetzt. Sie haben im Weltraum sehr große freie Weglängen und damit sehr geringe Wechselwirkungswahrscheinlichkeiten.

Zur Bestimmung der Auswirkungen auf die Elektronik ist die Kenntnis über die Wirkung eines einzelnen Teilchens sowie die Häufigkeit ihres Auftretens notwendig. Die Anzahl der Teilchen, die innerhalb einer Zeit durch eine ebene Fläche treten, ist der Partikelfluss Φ [$cm^{-2} \cdot s^{-1}$]. Wird dieser über ein bestimmtes Zeitintervall aufintegriert, erhält man die Fluenz F [cm^{-2}].

Beim Durchdringen von Materie werden Teilchen durch Wechselwirkung mit dem Atomgitter abgebremst. Die Energieabgabe längs der Flugbahn wird durch das **Massenbremsvermögen** (*Linear Energy Transfer,* LET) beschrieben. Sie ist definiert als die Energieabgabe dE eines Teilchens beim Durchqueren einer Materieschicht mit der Flächenmassendichte $\rho \cdot dx$.

$$LET = \frac{dE}{\rho \cdot dx} \qquad \left[MeV \cdot cm^2 \cdot g^{-1} \right]. \tag{2.2}$$

LET entspricht damit der Ionisierungseffizienz eines Teilchens. Das Massenbremsvermögen ist abhängig von Masse und Energie des Teilchens (siehe Bild 2.1). Es ist die Zunahme des Massenbremsvermögens mit steigender Ordnungszahl Z der Teilchen ersichtlich, sowie in weiten Bereichen die Abnahme mit steigender Energie je Nukleon E/n. Abgesehen von der

Abbremsung relativistischer Elektronen mit $E > 10$ MeV ist das Massenbremsvermögen nahezu unabhängig vom Absorbermaterial. Zur Abschirmung von Elektronik führen daher Schirmungen mit gleichen Flächendichten zu nahezu gleicher Wirkung.

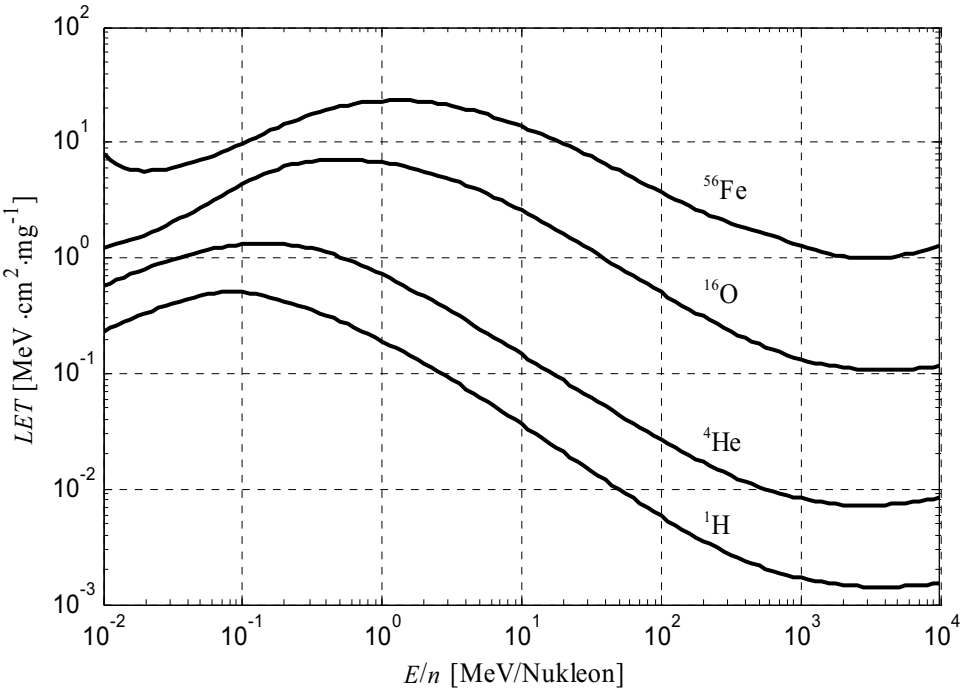

Bild 2.1: LET in Abhängigkeit von Ordnungszahl Z und Energie E [Brä89]

Beim Durchdringen von Materie verteilt sich die Energieabgabe auf drei Mechanismen. Durch die Stoßanregung von Gitterschwingungen werden Phononen und Wärme erzeugt. Bei der Stoßionisation werden Elektron-Loch-Paare erzeugt und Valenzelektronen in das Leitungsband gehoben. Der dritte Anteil an der von einem Teilchen deponierten Energie geht in die Erzeugung von Bremsstrahlungs-Photonen. Oberhalb der Energien von E_g (Bandabstand = 1,1 eV bei Si) wirken diese Photonen selbst wieder ionisierend. Dieser Anteil tritt bei relativistischen Elektronen mit E > 10 MeV auf, ist gegenüber dem Beitrag der direkten Ionisation jedoch für gewöhnlich zu vernachlässigen.

In Silizium (Si) wird pro Energieabgabe von 3,6 eV, in Siliziumdioxid (SiO_2) pro Energieabgabe von 18 eV ein Paar erzeugt. Diese teilen sich folgendermaßen auf [Ger01]:

		Si	SiO_2
Anhebung von Valenzelektronen ins Leitungsband	E_g =	1,1 eV	8,8 eV
Wärme, Bremsstrahlung bei relativistischen Elektronen > 10 MeV		2,5 eV	9,2 eV
Erzeugung Elektron-Loch-Paar	ΔE =	3,6 eV	18,0 eV

Bei Schaltkreisen in MOS-Technologie beruht der wesentliche Schädigungsmechanismus darauf, dass bei Abbremsung der Teilchen in Materie ein Teil der deponierten Ladung in die Ionisation von Gitteratomen und damit die Erzeugung von Elektron-Loch-Paaren entlang der Flugbahn fließt.

Neben dem Massenbremsvermögen ist die **Dosis** beim Entwurf eines Instrumentenrechners wichtig. Sie ist definiert als die in einem Massenelement *dm* deponierte Energie:

$$D = \frac{dE}{dm} = \frac{dE}{\rho \cdot dV} \qquad \left[J \cdot kg^{-1} = Gy \right]. \tag{2.3}$$

Es wird für gewöhnlich die Einheit rad (*Radiation Absorbed Dose*) mit folgender Umrechnung verwendet:

$$rad = 10^{-2} Gy = 6,24 \cdot 10^{7} \frac{MeV}{g}. \tag{2.4}$$

2.2.1 Strahlung im Weltraum

In unserem Sonnensystem wird zwischen drei verschiedenen Strahlungsflüssen unterschieden:

- Galaktische kosmische Strahlung (*Galactic Cosmic Rays,* GCR)
- Sonnenwind
- Solare kosmische Strahlung (*Solar Energetic Particles,* SEP)

Die galaktische kosmische Strahlung (*Galactic Cosmic Rays,* GCR) ist eine permanente, omnidirektionale Hintergrundstrahlung, bestehend aus hochenergetischen Ionen und Elektronen mit Energien bis weit über den GeV-Bereich mit einem Fluss von ca. 4 $cm^{-2} \cdot s^{-1}$. Bild 2.2 zeigt das auf Silizium normierte relative Massenspektrum. Darin sind die Hauptbestandteile mit 85 % Protonen, 14 % Helium und etwa 1 % schwerere Ionen ersichtlich. Das differentielle Energiespektrum ist in Bild 2.3 dargestellt.

Der Sonnenwind ist ein gleichmäßiger Fluss von Elektronen und Ionen mit einer Geschwindigkeit von etwa 400-800 km/s, bestehend aus 96 % Protonen. Wegen der geringen kinetischen Energie wird diese Strahlung bereits von einer relativ dünnen Folie absorbiert und ist für die Elektronik daher ungefährlich.

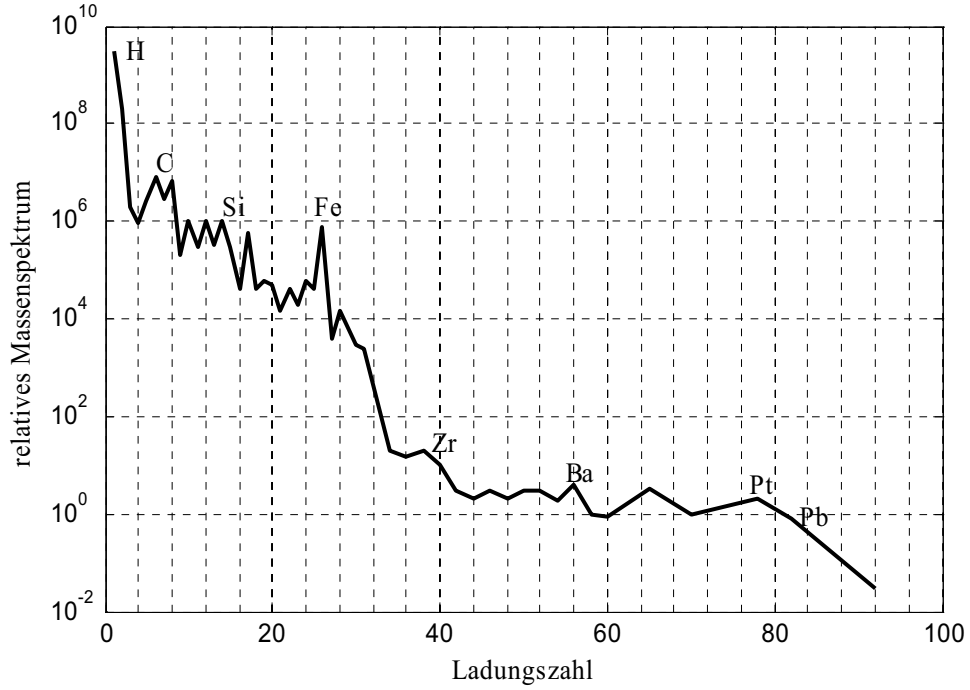

Bild 2.2: Zusammensetzung der galaktischen kosmischen Hintergrundstrahlung, normiert auf Si (= 10^6) [Fey93]

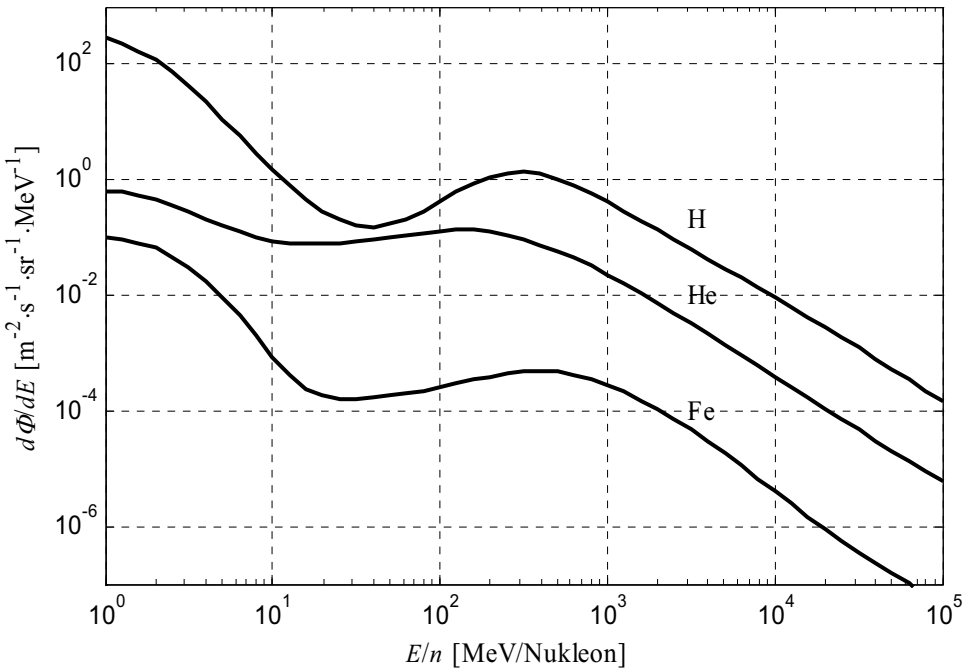

Bild 2.3: Differentielles Energiespektrum der galaktischen kosmischen Hintergrundstrahlung [Tri95]

Solare kosmische Strahlung (*Solar Energetic Particles,* SEP) tritt nur gelegentlich in *Bursts* auf und ist auf eruptive Ausbrüche (*Solar Flares*) großer Massen solaren Plasmas zurückzuführen. Dieser gerichtete Fluss aus hochenergetischen Ionen kann innerhalb weniger Stunden auf einen hohen Wert ansteigen und bis zu fünf Zehnerpotenzen über dem GCR-Fluss liegen. Anschließend bleibt er typischerweise etwa einen Tag auf hohem Niveau um anschließend innerhalb mehrerer Tage wieder nachzulassen. Die Wahrscheinlichkeiten für das Auftreten von *Solar Flares* mit bestimmten Stärken innerhalb des 11-jährigen Zyklus der solaren Aktivität sind in [Ger01] beschrieben. In Bild 2.4 ist die integrierte Fluenz für *Heavy Ions* (HI, mit Atomzahl $Z > 1$) in Abhängigkeit vom LET-Wert für verschiedene *Solar Flare*-Bedingungen, aber auch für die galaktische kosmische Strahlung (GCR) dargestellt. Die Kurve gibt für einen bestimmten LET-Wert die Anzahl an Teilchen mit genau diesem Wert und höher an (*Heinrich Flux*). Die LET-Werte der verschiedenen Ionen-Flüsse, d. h. der verschiedenen Ionenspezies sind dabei zu einem LET-Spektrum zusammengefasst. Gut sichtbar ist ein starker Einbruch bei ca. $LET = 30\ \mathrm{MeV \cdot cm^2 \cdot mg^{-1}}$, der aus dem maximal möglichen LET-Wert von Fe resultiert (siehe Bild 2.1). Für dieselben *Solar Flare*-Bedingungen sowie für GCR ist das differentielle Energiespektrum für Protonen in Bild 2.5 aufgetragen.

Das GCR-Spektrum ist überwiegend bestimmend, lediglich während der kurzen Zeit seltener *Solar Flares* ist das *Solar Flare*-Spektrum maßgebend, sofern keine starke Schirmung gewählt wurde. Das GCR-LET-Spektrum ist aufgrund der hohen Teilchenenergie wesentlich weniger durch Schirmung beeinflussbar als das *Solar Flare*-LET-Spektrum. In [NAS99] ist bei einem Abstand zur Sonne von 1 AU exemplarisch gezeigt, dass das GCR-LET-Spektrum bei starken Schirmdicken von z. B. 12 mm Al um den Faktor 4 reduziert wird, während sich das *Solar Flare*-LET-Spektrum bei gleicher Schirmdicke um mehrere Größenordnungen verringert. Um das *Solar Flare*-LET-Spektrum auf das Niveau des GCR-LET-Spektrums zu reduzieren, ist nach [NAS99] für niedrige LET-Werte ($< 10\ \mathrm{MeV \cdot cm^2 \cdot mg^{-1}}$) eine Schirmdicke von ca. 12 mm Al erforderlich, während für höhere LET-Werte ($\geq 10\ \mathrm{MeV \cdot cm^2 \cdot mg^{-1}}$) etwas weniger, z. B. 7 mm Al ausreichend ist. Gegenüber *Heavy Ions* ist die Abschirmung von Protonen weniger effektiv. Dieses ist in der höheren Protonen-Reichweite begründet und steht im Zusammenhang mit dem geringeren LET-Wert pro Nukleon (siehe Bild 2.1).

Bei der Bestimmung der akkumulierten Dosis ist zu beachten, dass die gelegentlichen SEPs einen großen Beitrag zu der in den Bauteilen deponierten Ladung liefern. Der Beitrag der GCR ist mit < 10 rad/Jahr dagegen vernachlässigbar.

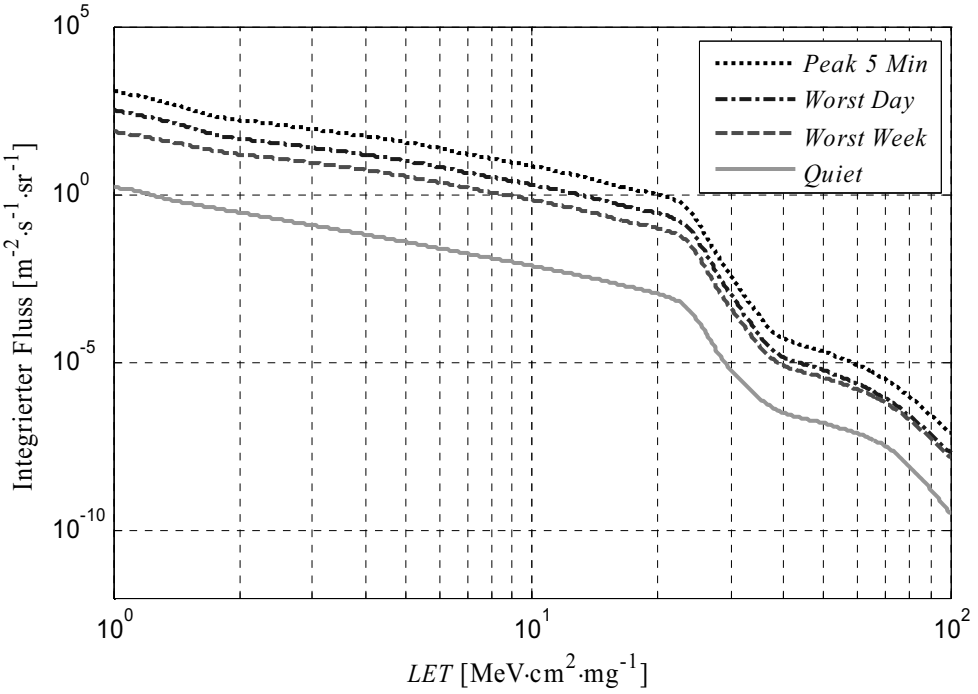

Bild 2.4: Integrierter Fluss bei 1 AU Abstand zur Sonne (1 g·cm^{-2} Schirmung) [Soe05]

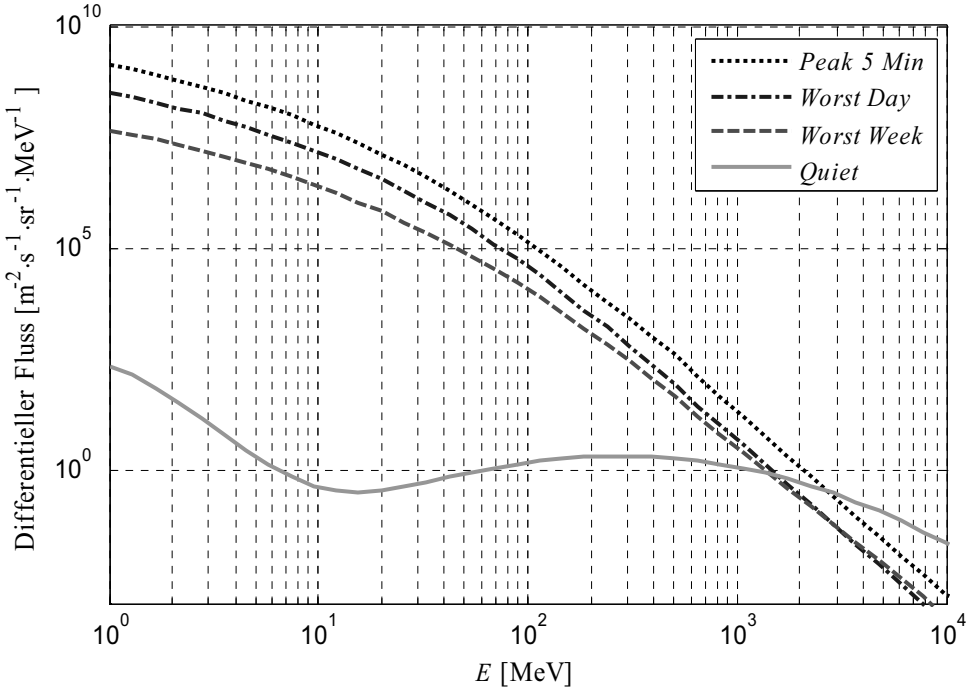

Bild 2.5: Differentieller Protonenfluss bei 1 AU Abstand zur Sonne (ohne Schirmung) [Soe05]

Bei Planeten mit Magnetfeld ergeben sich aufgrund der Wechselwirkung der Strahlung mit dem Magnetfeld besondere Strahlungsbedingungen. Zu erwähnen seien hier z. B. bei der Erde die *Van Allen-Gürtel*, die bestimmte Regionen bezeichnen, in denen große Elektronen- und Protonen-Flüsse existieren. Sie ergeben sich aus der Wechselwirkung des magnetischen Feldes mit der Strahlung. Die Strahlungsgürtel (innerer Strahlungsgürtel mit überwiegend Protonen, äußerer Strahlungsgürtel mit überwiegend Elektronen) liegen rotationssymmetrisch zur erdmagnetischen Achse wie eine Schale um die Erde und erstrecken sich insgesamt von etwa 700 km bis zu 45.000 km Höhe über der Erdoberfläche. Es können von außen eindringende geladene Teilchen vom Magnetfeld eingefangen werden, die dann entlang der magnetischen Feldlinien von Süd nach Nord und wieder zurück pendeln. Für weitere Informationen bzgl. der Wechselwirkung Magnetfeld der Erde mit der Strahlung sei auf [Gli05] verwiesen.

2.3 Auswirkung der Strahlung auf elektronische Bauteile

Strahlung im Weltraum kann in elektronischen Bauteilen zu dauerhaftem oder temporärem Fehlverhalten der Schaltungen führen. Die Effekte sind zu unterscheiden in akkumulierende Effekte, die aus sich anhäufenden Strahlungsauswirkungen vieler Einschläge resultieren (siehe Abschnitt 2.3.1) und Effekte, die durch genau einen energiereichen Einschlag (siehe Abschnitt 2.3.2) ausgelöst werden.

2.3.1 *Total Dose* Effekt

Der *Total Dose* Effekt ist ein akkumulierender, nicht reversibler Effekt mit starken Folgen für die Zuverlässigkeit eines Systems und beruht auf den Auswirkungen der Erzeugung von Elektron-Loch-Paaren. In Bild 2.6 ist ein *Total Dose* Effekt am Beispiel eines n-Kanal MOS Transistors gezeigt. Ein in das *Gate Oxid* eintretendes Teilchen erzeugt entlang der Flugbahn Elektron-Loch-Paare, von denen ein Anteil durch die vorherrschende Feldstärke vor einer sofortigen Rekombination separiert wird. Die Elektronen werden in kurzer Zeit zur *Gate*-Elektrode geschwemmt und rekombinieren dort. Die Löcher dagegen wandern weitaus langsamer in Richtung des Si-Substrats und bleiben teilweise an Haftstellen zum SiO_2 hängen. Sie vergrößern die positive Substrat-Oberflächenladung und verschieben damit die Schwellspannung zu negativen Werten. Dadurch geht der Transistor vom selbst sperrenden Typ in einen selbst leitenden Typ über. Eine Verschiebung der Schwellspannung um einige 100 mV führt in den meisten Fällen bereits zu einem Funktionsausfall der Schaltung.

Die maximal erlaubte Toleranzdosis D_{tol} bis zu einem Funktionsausfall liegt bei typischen Standard-MOS-Bauteilen zwischen einigen krad bis 20 krad. Durch Verringerung der Haftfähigkeit am Si/SiO_2-Übergang kann sie vergrößert werden. Abnehmende Strukturbreiten in der Chip-Technologie führen zudem dazu, dass die je Fläche erzeugte Ladungsmenge kleiner und der *Total Dose* Effekt reduziert wird. Durch diese Maßnahmen und elektrische Anpassungen wie erhöhte Schwellspannungs-Toleranzen kann die Toleranzdosis D_{tol} für strahlungsfeste

Bauteile auf bis zu einige Mrad erhöht werden. Die tatsächlich zulässigen Toleranzdosen D_{tol} für Bauteile werden experimentell durch inkrementelle Zunahme der deponierten Dosis ermittelt.

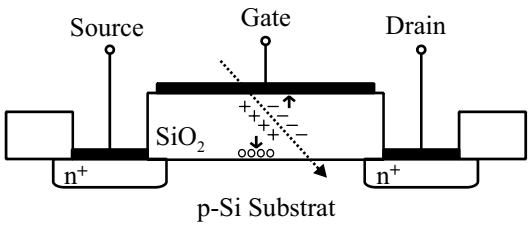

Bild 2.6: *Total Dose* Effekt

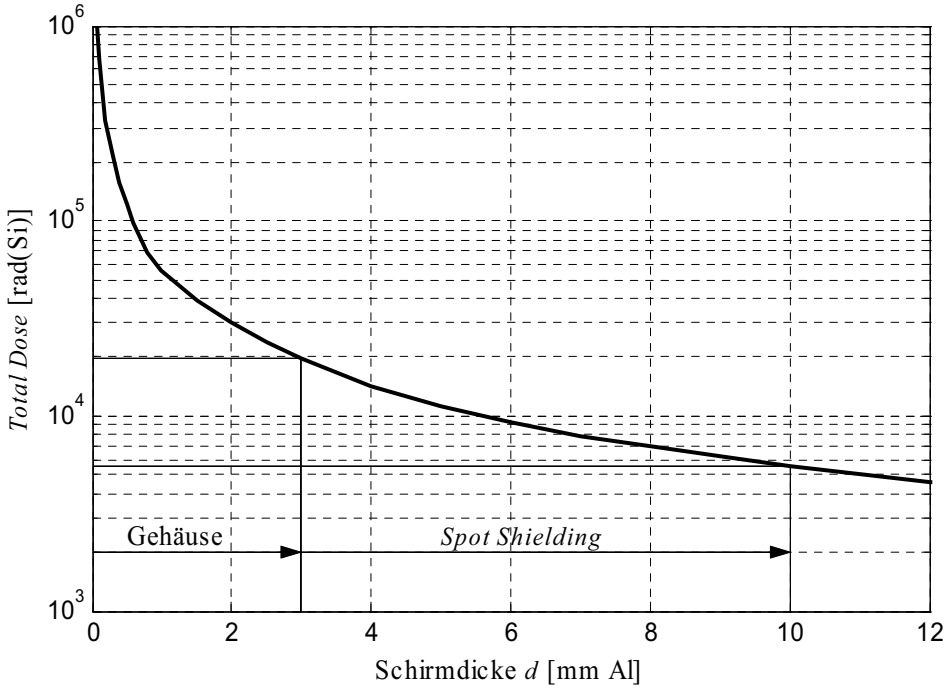

Bild 2.7: *Dose Depth*-Kurve

Die Dosisvorhersage für eine Mission und damit die Bestimmung der im Bauteil insgesamt deponierten Dosis kann mit Hilfe einer *Dose Depth*-Kurve erfolgen. Sie zeigt für die ausgewählte Mission die insgesamt zu erwartende Dosis im Zentrum einer Kugel mit einer Schirmung der Dicke *d* aus Aluminium. Für jedes Orbitintervall wird der Dosisbetrag und durch Summation die zu erwartende Dosis bis zum Ende der Mission bestimmt. Bild 2.7 zeigt ein Beispiel für eine *Dose Depth*-Kurve. Da mit der *Dose Depth*-Kurve die Situation eines Bausteins in der Mitte einer Kugel mit entsprechender Wanddicke und senkrechtem Strahlungsdurchgang durch die Wand beschrieben ist, entspricht die damit berechnete Dosis unnötigerweise dem pessimistischen Fall. Das Schirmungsverhalten der Raumsonde und ein schräger Strahlungsdurchgang durch das Gehäuse und damit die Erhöhung der wirksamen Wandstärke kann

angenommen werden. Zudem führt die gegenseitige Abschirmwirkung der internen Elektronikplatinen zu einer Verringerung der Dosis.

Bei der Berechnung der tatsächlich am Ort eines Bausteins zu erwartenden Dosis werden die Beiträge aller Raumsegmente zur Dosis bestimmt und aufsummiert:

$$D = \int_{2\pi}\int_{\pi} D(\theta,\varphi)\cdot d\theta \cdot d\varphi. \tag{2.5}$$

Praktisch erfolgt diese Berechnung mit Hilfe von *Ray Tracing* Programmen. Unter Verwendung diskreter Schrittweiten der sphärischen Koordinatenwerte φ und θ wird die dazugehörige Menge an durchquerter Masse und damit der jeweilige Dosisbeitrag ermittelt und aufsummiert. Entsprechend des Verlaufs der *Dose Depth*-Kurve tragen Löcher bzw. sehr geringe Wandstärken in der Schirmung überproportional zur Dosis bei und sollten vermieden werden. Die Kenntnis über die Dosis am Ort des Bausteins kann für die gezielte Positionierung eines empfindlichen Bausteins hilfreich sein. Genügt die Toleranzdosis dennoch nicht der berechneten Dosis, kann mit *Spot Shielding* gearbeitet werden (siehe Bild 2.7). Hierbei werden einzelne Bauteile geschützt, z. B. durch Aufkleben von Schirmungsmaterial und möglichst gute Abdeckung des integrierten Schaltkreises (*Die*). Bei der Schirmungswirkung ist die durchquerte Masse entscheidend. Als Material werden hierzu häufig Aluminium und Tantal verwendet. Dabei hat 1 mm Tantal mit $\rho_{Ta} = 16{,}6$ g/cm^3 die gleiche Abschirmwirkung wie 6,15 mm Aluminium mit $\rho_{Al} = 2{,}7$ g/cm^3. Zu beachten ist jedoch, dass harte Schirmungen, z. B. mit Tantal, zur verstärkten Erzeugung von Bremsstrahlung führen können.

Bei der Auslegung einer DPU ist heutzutage der *Total Dose* Effekt, insbesondere aufgrund der abnehmenden Strukturbreiten in der Chip-Technologie, weniger problematisch.

2.3.2 *Single Event* Effekte

Im Gegensatz zum akkumulierenden *Total Dose* Effekt basieren *Single Event* Effekte (SEE) auf einzelnen Ereignissen. Bei einem *Single Event Upset* (SEU) reicht die von einer Ionisationsspur entlang der Flugbahn des Teilchens erzeugte elektrische Ladung aus, um an einem kritischen Schaltungsknoten den Schaltzustand zu verändern. Hierbei handelt es sich um einen temporären Fehler, der i. Allg. nicht zur Schädigung des Bauteils führt. Dieses kann z. B. zu einer Änderung eines Flipflop-Status von „0“ nach „1“ führen oder das Umladen einer DRAM-Zelle hervorrufen.

Anders als beim *Total Dose* Effekt begünstigen verkleinerte Strukturgrößen im Chip das Auftreten von SEUs [Mic05]. Mittels Strahlungstests kann der Schwellen-LET-Wert $LET_{th,\ SEU}$ eines Bausteins ermittelt werden, der mindestens erforderlich ist, damit ein Teilchen einen SEU hervorrufen kann. Hierbei ist zwischen *Heavy Ions* (HI) mit einer Atomzahl > 1 und Protonen zu unterscheiden. Es erzeugen nur diejenigen Teilchen einen SEU, deren Flugbahnen in direkter Nähe empfindlicher Schaltungsknoten verlaufen. Dieser Anteil wird bei *Heavy Ions* durch den

Wirkungsquerschnitt $\sigma_{SEU,\,HI}(LET)$ beschrieben und steigt mit wachsenden LET-Werten der Teilchen bis zu einem Sättigungswert an. In Bild 2.8 ist ein Beispiel für einen Wirkungsquerschnitt mit Annäherung durch die Weibull-Verteilung dargestellt. Der Wirkungsquerschnitt ist definiert als das Verhältnis der Fehlerrate $N_{SEU,\,HI}$ zur Fluenz F_{HI}:

$$\sigma_{SEU,HI}(LET) = \frac{N_{SEU,HI}(LET)}{F_{HI}(LET)} \qquad \left[cm^2\right]. \tag{2.6}$$

Protonen mit Energien im MeV-Bereich können indirekt auch SEUs auslösen. Beim Durchqueren können sie Atomkerne des Halbleitermaterials zerstoßen und ihren Impuls auf die Kernfragmente übertragen. Diese besitzen die mehrfache Protonenmasse, haben entsprechend Bild 2.1 einen höheren LET-Wert und können somit entlang ihrer Spur eine höhere Ladungsträgerdichte erzeugen. Analog zu den HI-induzierten SEUs wird für Protonen der Wirkungsquerschnitt definiert:

$$\sigma_{SEU,Pr}(E) = \frac{N_{SEU,Pr}(E)}{F_{Pr}(E)} \qquad \left[cm^2\right]. \tag{2.7}$$

Zur Auslösung der gleichen SEEs liegen die Sättigungswerte der Protonen-Wirkungsquerschnitte in den meisten Fällen um mehrere, z. B. um sechs Größenordnungen niedriger als die der vergleichbaren *Heavy Ions*-Wirkungsquerschnitte [Swi04]. Meistens beschränkt man sich daher auf die Bestimmung des *Heavy Ions*-Wirkungsquerschnitts. Wenn bei einer Mission mit extrem häufigen intensiven Protonenflüssen gerechnet wird, wie z. B. beim Durchqueren der Südatlantischen Anomalie, oder wenn eine vollständige Charakterisierung des Bausteins gewünscht ist, werden auch Strahlungstests mit Protonen durchgeführt.

Beim Entwurf einer DPU ist die Kenntnis über die zu erwartenden SEU-bedingten Fehlerraten von hoher Bedeutung. Sie stellt die Grundlage für den Aufwand zur Reduzierung der Fehlerauswirkungen und zur Bestimmung der Verfügbarkeit eines Systems dar. Bild 2.9 zeigt eine Übersicht zur Berechnung der SEUs. Für die jeweiligen Missionen werden die zu erwartenden Fluenzen in Abhängigkeit vom LET-Wert für verschiedene solare Umstände vorausberechnet. Ebenso wird die Fluenz solarer Protonen in Abhängigkeit von ihrer Energie bestimmt.

Häufig ist das missionsabhängige LET-Spektrum für eine feste Schirmung (z. B. 1 g/cm^2) bekannt. Ähnlich wie bei der Berechnung der am Baustein zu erwartenden Dosis, kann auch hier mittels *Ray Tracing*-Verfahren eine genauere Bestimmung der Fehlerrate erfolgen. Das Abschirmverhalten ist abhängig von den Spezies. Daher ist eine Aufteilung in die einzelnen Spezies mit ihren zugehörigen Fluenzen zur genauen Berechnung der Fehlerrate notwendig. Für jede dieser Fluenzen $F(E)$ wird separat das LET-Spektrum nach dem Durchgang durch die Schirmung, d. h. nach Verlust einer Teilenergie, berechnet. Kombiniert ergeben diese einzelnen Spektren das Gesamt-LET-Spektrum nach der Schirmung, welches zur Berechnung der

Fehlerrate herangezogen wird. Zu beachten ist jedoch, dass die Energieabgabe eines Ions entsprechend Bild 2.1 dazu führen kann, dass der LET-Wert ansteigt. Analog führt die Berechnung des Energieverlustes von Protonen im Schirmungsmaterial zur Bestimmung des Energiespektrums nach der Schirmung.

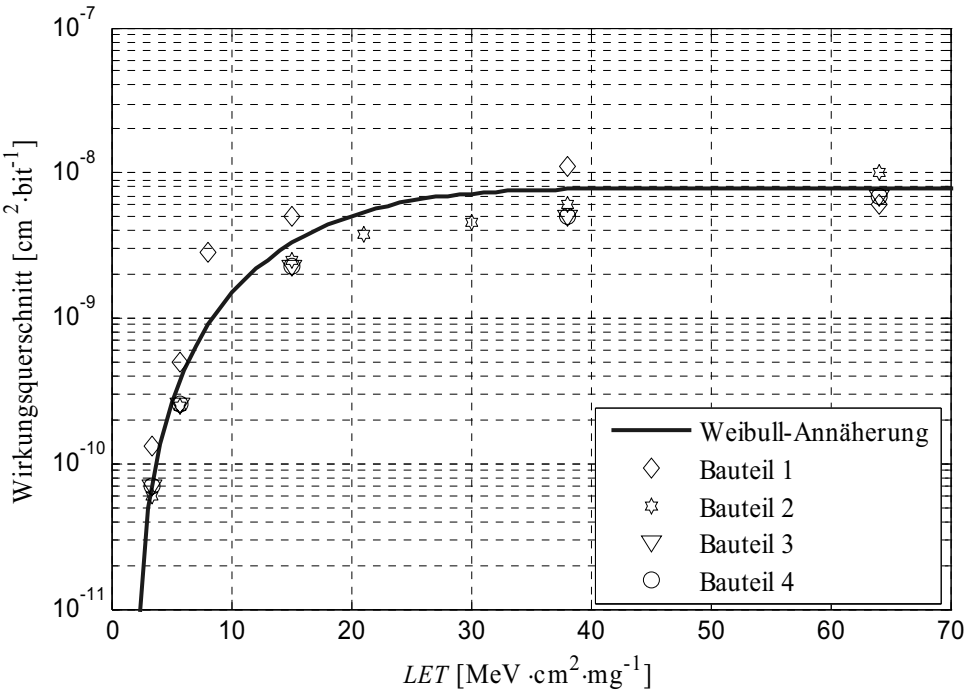

Bild 2.8: Wirkungsquerschnitt eines SRAMs mit vier Testbausteinen [Aer03]

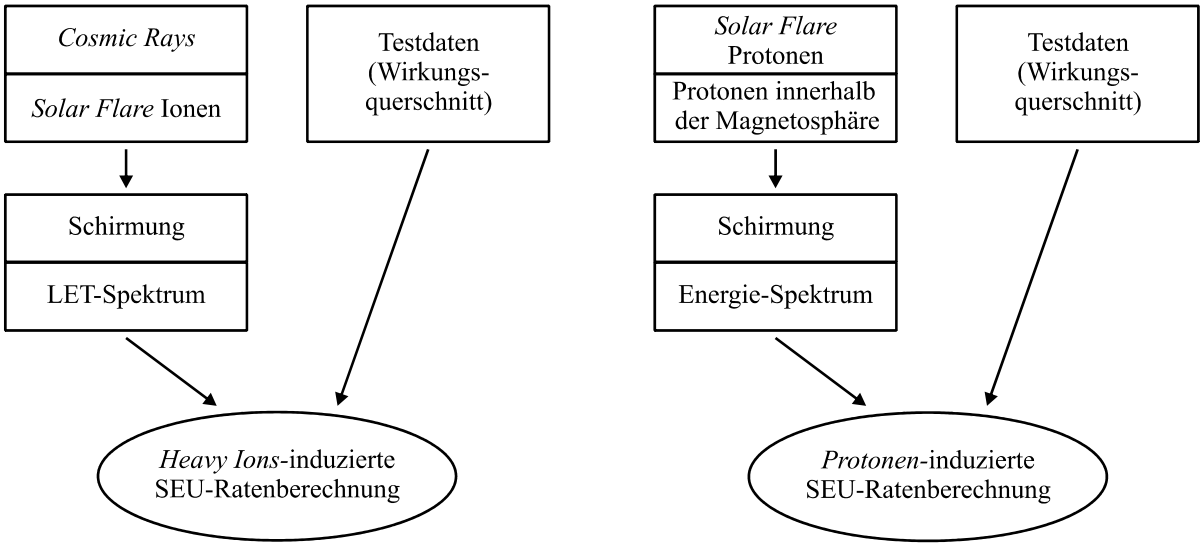

Bild 2.9: Übersicht SEU-Berechnung

Neben den Fluenzen sind die bei Strahlungstests bestimmten Wirkungsquerschnitte für die zu berechnenden Bauteile erforderlich. Bei strahlungsfesten Bauteilen liegen häufig keine umfangreichen Daten der Wirkungsquerschnitte vor. Hier sind LET-Schwellenwerte angegeben, die jedoch sehr hoch liegen und für die Fehlerrate des gesamten Systems von geringerer Bedeutung sind. Im Wesentlichen konzentriert sich die Berechnung der Fehlerraten daher auf die strahlungsempfindlichen Bauteile im Aufbau.

Die Teilchenflüsse erzeugen die HI-induzierten SEU-Raten

$$R_{SEU,HI} = \int \sigma_{SEU,HI}(LET) \cdot \frac{d\Phi}{d(LET)} \cdot d(LET) \tag{2.8}$$

sowie analog die von Protonen erzeugte Fehlerrate

$$R_{SEU,Pr} = \int \sigma_{SEU,Pr}(E) \cdot \frac{d\Phi}{dE} \cdot dE \tag{2.9}$$

und führen zu der gesamten SEU-Fehlerrate

$$R_{SEU} = R_{SEU,HI} + R_{SEU,Pr}\,. \tag{2.10}$$

SEUs können in integrierten Schaltungen unterschiedliche Auswirkungen haben. Diese werden i. Allg. durch die Einführung von Redundanz und damit einhergehend durch die Zunahme der notwendigen Ressourcen gemildert. Häufig benutzt werden auf Mehrheitsentscheid basierende Konstrukte wie *Triple Module Redundancy* (TMR) sowie die Kombination Fehler korrigierender Code mit regelmäßigen Lese- und Schreibzugriffen (Schrubben, *Scrubbing*). Diese Kombination findet insbesondere bei Speicherbausteinen Verwendung. Weitere Informationen hierzu sind in Kapitel 4 zu finden.

DPUs werden mit zunehmender Funktionalität entworfen. Hierdurch werden i. Allg. auch mehr Ressourcen verwendet, die durch SEUs betroffen sein können. Zusammen mit der gestiegenen Empfindlichkeit gegenüber Strahlung aufgrund abnehmender Strukturbreiten führt dieses dazu, dass bei der Auslegung einer DPU heute verstärkt auf SEEs zu achten ist.

Ein weiterer *Single Event* Effekt ist der *Single Event Latchup* (SEL). Dabei wird durch die von einem Teilchen erzeugte Ladung ein parasitärer Thyristor gezündet und dadurch eine andauernde niederohmige Verbindung zwischen beiden Versorgungsschienen hergestellt [Gli05]. Parasitäre Vierschicht-Thyristorstrukturen treten in der *Bulk*-CMOS-Technologie auf (siehe Bild 2.10). Die Folge des gezündeten Thyristors ist die thermische Zerstörung des Bauteils. Analog zur SEU-Empfindlichkeit wird die SEL-Empfindlichkeit durch den Wirkungsquerschnitt $\sigma_{SEL}(LET)$ beschrieben, der von dem Schwellen-LET-Wert $LET_{th,\,SEL}$ bis zu einem Sättigungswert ansteigt. Verringerte SEL-Empfindlichkeit erhält man durch technologische Verbesserungen. Bei der zunehmenden Verwendung epitaktischen Schaltungsaufbaus wird die Struktur in einer

epitaxialen Schicht mit hohem Flächenwiderstand aufgebaut, die ihrerseits auf einem Substrat mit hoher Dotierung und damit niedrigem Flächenwiderstand aufgewachsen ist. Dieses führt dazu, dass der Bahnwiderstand aufgrund des niederohmigen Nebenschlusses deutlich verkleinert wird und die Haltebedingung für den einmal gezündeten Thyristor nicht mehr gegeben ist. Bei strahlungsgehärteten Bauteilen werden SEL-sichere *Silicon On Insulator* (SOI)-Strukturen verwendet.

Die Empfindlichkeit eines Bausteins gegenüber SEL sowie die Häufigkeit des Auftretens können analog zu den SEU bestimmt werden. Das Auftreten eines SEL geht einher mit einem erhöhten Stromverbrauch des betroffenen Bauteils. SEL-empfindliche Bauteile bzw. Baugruppen werden mit *Latchup*-Schutzschaltern, bestehend aus Strommessung, Vergleich, Logik und dem Schalter selbst, gesichert (siehe Bild 2.11). Sobald der gemessene Strom eine definierte Schwelle überschreitet, wird mittels Schalter die betroffene Schaltungsgruppe für eine kurze Zeit (einige 100 ms) von der Versorgungsspannung abgetrennt. Als Folge hiervon erlischt der parasitäre Thyristor und eine thermische Zerstörung des Bauteils wird vermieden. Bei Verwendung eines Schutzschalters ist sicherzustellen, dass die zu schützenden Bauteile nicht über Schutzdioden parasitär weiter gespeist werden. Hierbei würde das Bauteil im abgeschalteten Zustand durch ein Eingangssignal mit *High*-Pegel, das aus einem nicht abgeschalteten Schaltungsbereich kommt, weiter mit Spannung versorgt werden. Dieses hätte zur Konsequenz, dass die Haltebedingung für den gezündeten Thyristor aufrechterhalten würde. Durch angepasste Logikverdrahtung oder zusätzliche Bausteine wird diese parasitäre Speisung unterbunden.

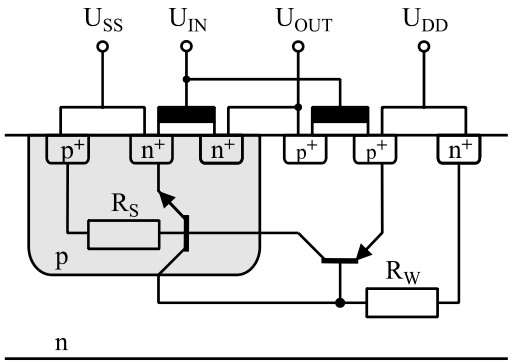

Bild 2.10: Parasitärer Thyristor bei *Bulk*-CMOS-Technologie [Gli05]

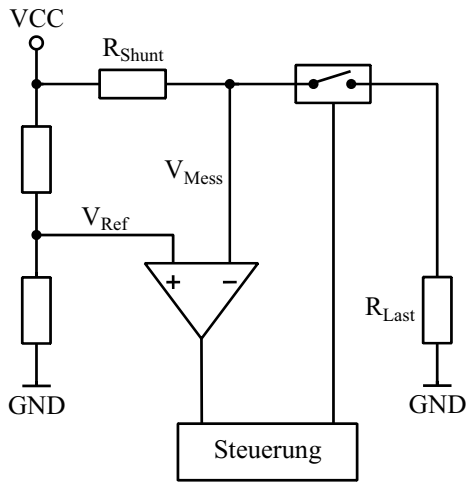

Bild 2.11: *Latchup*-Schutzschalter

3 Bilddatenverarbeitung

Nach einer Analyse der prinzipiellen Möglichkeiten zur Steigerung der Performanz und der Begründung der Auswahl des JPEG2000-Bildkompressionsverfahrens werden in diesem Kapitel die Grundzüge einer Bildkompression mit Schwerpunkt auf dem JPEG2000-Algorithmus dargestellt. Damit soll aufgezeigt werden, inwieweit eine hardwareunterstützte JPEG2000-Kompression zur Steigerung des wissenschaftlichen Nutzens eines kompakten Kamerainstruments beitragen kann. Ursprünglich in Software ausgeführte Funktionen werden gezielt durch Hardwareimplementierungen ersetzt. Ein auf dieser Basis entwickelter und entsprechend angepasster Kompressions-Co-Prozessor wird in Kapitel 7 vorgestellt.

Die Bildverarbeitungsroutinen einer DPU für kompakte Kameras auf Raumsonden bestimmen im Wesentlichen die Performanz der DPU. Um bei vorgegebenen Telemetrieraten möglichst viel Information zur Erde zu übertragen und damit einen hohen wissenschaftlichen Nutzen zu erzielen, ist eine Datenverarbeitung an Bord erforderlich und somit eine hohe Rechenleistung. Das Instrument VIM (Visible-Light Imager and Magnetograph) auf der ESA Raumsonde Solar Orbiter ist ein Beispiel für die zunehmende Datenverarbeitung an Bord [Sol05]. VIM beinhaltet ein CCD der Größe 2k × 2k Pixel, Polarisationsfilter und Interferometer zur Frequenzbegrenzung und zielt u. a. darauf ab, Magnet- und Geschwindigkeitsfelder in der Photosphäre anhand parametrisierter Aufnahmen der FeI-6173Å-Linie zu bestimmen. Für eine Aufnahme werden Einzelaufnahmen mit vier verschiedenen Polarisationsfiltern bei sechs leicht unterschiedlichen Frequenzbändern in der Nähe der FeI-6173Å-Linie verwendet. Aus diesen 24 Einzelaufnahmen der FeI-6173Å-Linie werden anschließend physikalische Größen berechnet. Diese lassen sich mit einer wesentlich geringeren Datenmenge als die Summe der Einzelaufnahmen beschreiben und werden stattdessen zur Erde übertragen. Die Reduzierung erfordert jedoch intensive Berechnungen, u. a. die Inversion der Strahlungsübertragungsgleichung (RTE, *Radiative Transfer Equation*) zur Bestimmung der physikalischen Größen, und damit eine hohe Rechenleistung an Bord.

Die Möglichkeiten zur Steigerung der Performanz unterliegen Randbedingungen wie z. B. Energiebedarf und Platzbedarf. Selbst bei genügend vorhandener Energie sollte zur Vermeidung thermischer Probleme eine moderate Energieaufnahme eingehalten werden. Zur Beurteilung der Steigerung der Performanz bzw. der Beschleunigungsfaktoren wird vielfach die Operationsgeschwindigkeit, gemessen in Millionen Instruktionen je Sekunde (MIPS, *Million Instructions per Second*) verwendet. Sie berücksichtigt lediglich die durchschnittliche Ausführungszeit der Instruktionen und ist daher nicht uneingeschränkt verwendbar. Eine DPU für Kameras hat spezielle, rechenintensive Aufgaben auszuführen, wie z. B. die Kompression eines Bildes oder, wie in dem Beispiel des VIM Instruments, die sehr spezifische Vorverarbeitung vor einer Kompression. Es ist weniger wichtig, wie ein höherer Durchsatz der allgemeinen Aufgaben erzielt wird, sondern wie sich diese sehr speziellen Aufgaben beschleunigen lassen. Beim Entwurf einer DPU für Kameras ist damit die Ausführungszeit dieser speziellen Aufgaben eine wichtige Größe für die Leistungsfähigkeit. Zur Erhöhung der Rechenleistung können

verschiedene Architekturansätze verwendet werden, zwei prinzipiell verschiedene Möglichkeiten werden hier näher erläutert.

Ein simpler Ansatz zur Erhöhung des Beschleunigungsfaktors ist der Parallelbetrieb mehrerer Standard-CPUs. Nach dem Amdahl'schen Gesetz zur Beschreibung des Beschleunigungsfaktors führt dieses zu einer skalierbaren Performanz [Tan06]. Es ist für die Aufteilung der Rechenaufgabe auf mehrere CPUs von hoher Wichtigkeit, den zeitlichen Anteil an sequentiellen Aufgaben zu minimieren und folglich den parallelen Anteil zu maximieren. Im unrealistischen Idealfall entspricht der Beschleunigungsfaktor direkt der Anzahl der CPUs. Eine Erhöhung der Rechenleistung ist lediglich über die Anzahl der CPUs möglich und führt zu einem starken Anstieg des Energie- und Platzbedarfs.

Wie oben dargestellt ist aber die generelle Rechenleistung (z. B. in MIPS) nicht unbedingt maßgebend für das Design einer aufgabenspezifischen DPU. Zur Steigerung der Rechenleistung können daher in einem anderen Ansatz spezielle Funktionen innerhalb dedizierter Logik in Form von Co-Prozessoren ausgeführt werden, die in der Kombination Standard-CPU mit Co-Prozessor zu einem starken Anstieg der aufgabenspezifischen Performanz führen. Dieser Anstieg kann weit über dem der parallel betriebenen Standard-CPUs liegen. Bei zunehmender Spezialisierung wird auf Kosten allgemeiner Funktionalität bei geringerer benötigter Zeit weniger Ladung bewegt und somit weniger Energie verbraucht. Innerhalb eines Co-Prozessors wird z. B. interner Speicher verwendet, der ausschließlich der speziellen Funktion zur Verfügung steht, oder es werden dedizierte Zustandsmaschinen anstelle von Steuerungen basierend auf Programmcode eingesetzt. Die zur Implementierung eines Co-Prozessors zusätzlich notwendigen Ressourcen sind lediglich zur Nutzung für die speziellen Funktionen und nicht für allgemeine Funktionen verwendbar.

Dem Nachteil des höheren Aufwands zur Erstellung der Co-Prozessoren stehen Vorteile wie hohe Performanz und hohe Energieeffizienz gegenüber. Durch die Bereitstellung von Co-Prozessoren ist jedoch mit einem höheren Leistungsbedarf im Ruhezustand zu rechnen, so dass bei energiekritischen Anwendungen ein zusätzliches Energiemanagement notwendig wird. Für eine optimale Implementierung muss zudem untersucht werden, welche Funktionen eines Algorithmus durch einen Co-Prozessor in Hardware und welche sinnvollerweise auf dem Standard-Prozessor in Software ausführt werden sollten.

Da innerhalb einer DPU für kompakte Kameras die Ausführungszeit für spezielle Aufgaben gegenüber der Ausführungszeit für allgemeine Aufgaben ausschlaggebend ist, ist die Verwendung von Co-Prozessoren dem Einsatz mehrerer paralleler Standard-CPUs zur Erhöhung der Performanz vorzuziehen.

Nahezu alle DPUs für Kameras auf Raumsonden beinhalten rechenintensive Kompressionsroutinen als spezielle Funktionen. Die Bilddatenkompression zielt daher darauf ab, die zur Darstellung eines Bildes benötigte Datenmenge sinnvoll zu reduzieren. Bei eingeschränkter Telemetrierate können damit mehr Bilder zur Erde gesendet und der wissenschaftliche Nutzen des Instruments erhöht werden. In dieser Arbeit wird anhand des Bildkompressionsalgorithmus

JPEG2000 als spezielle Funktion gezeigt, inwieweit sich die Performanz bei strahlungsfester und strahlungstoleranter Implementierung einer DPU erhöhen lässt.

Es wird zwischen zwei Arten der Bilddatenkompression unterschieden:

- Verlustlose Kompression und damit gleich bleibende Bildqualität sowie
- verlustbehaftete Kompression unter Inkaufnahme von Qualitätseinbußen.

Für Raumfahrtinstrumente wird sowohl die verlustlose als auch die verlustbehaftete Kompression benötigt. Es werden auf den Raumsonden häufig Kompressionsroutinen verwendet, die auf der Diskreten *Wavelet*-Transformation (DWT) beruhen. Sie erreichen eine hohe Bildqualität, allerdings auf Kosten einer hohen Rechenleistung. Ist diese hohe Rechenleistung verfügbar, z. B. durch hardwareunterstützte Kompression, so kann bei einer hohen Kompressionsgeschwindigkeit gleichzeitig eine hohe Qualität erzielt werden. Bild 3.1 zeigt den Qualitätsvergleich zweier *Wavelet*-basierter Verfahren (JPEG2000 und SPIHT (*Set Partitioning in Hierarchical Trees*)) mit einem Verfahren basierend auf der Diskreten Kosinus Transformation (DCT) (JPEG). Während die Ergebnisse der beiden *Wavelet*-basierten Verfahren sehr ähnlich sind, zeigen sich merkbar geringere Bildqualitäten bei der wesentlich weniger rechenintensiven DCT-basierten JPEG-Kompression. Der Unterschied im Rechenaufwand bei JPEG2000 gegenüber JPEG liegt bei ca. Faktor 7 [Ada00].

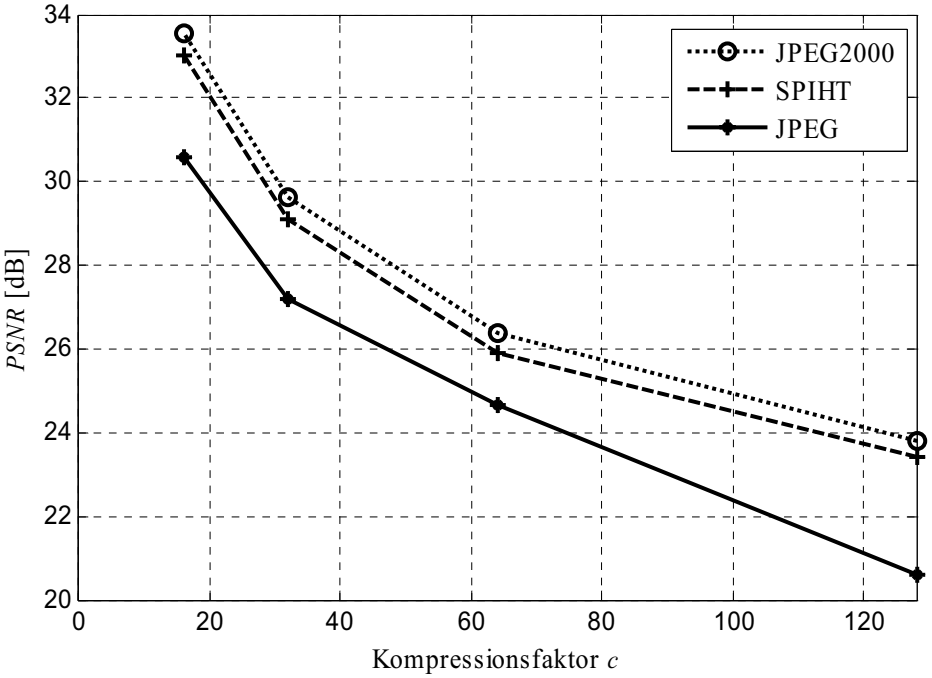

Bild 3.1: Vergleich DWT- mit DCT-basierten Kompressionsverfahren [Ada00]

Bisher wurde von den *Wavelet*-basierten Kompressionsverfahren vielfach der von A. Said und W. Pearlman vorgeschlagene SPIHT-Algorithmus [Sai96] verwendet. Der SPIHT-Algorithmus

zeichnet sich dadurch aus, dass auf das transformierte Bild bandübergreifend hierarchisch zugegriffen wird und die Eltern-Kind-Beziehung, wie sie durch die Transformation zwischen den Bändern besteht, zur Wahrscheinlichkeitsvorhersage genutzt wird [Mit02]. Während der Codierung ist der Zugriff auf alle transformierten Bilddaten erforderlich. Beim SPIHT-Algorithmus werden Listen abgearbeitet, deren Einträge den Zustand und die Koordinaten der transformierten Bildpunkte sowie einen Zeiger zum nächsten Eintrag in der Liste beinhalten. Bis zum Abschluss der Codierung ist ein vollständiger Zugriff auf die Daten dieser Listen notwendig. Sobald die gewünschte Datenrate erreicht ist, kann die Codierung beendet werden. Das Verfahren selbst kann sehr effektiv und mit wenig Programmcode in Software realisiert werden. Nachteilig ist der große Speicher, der zur temporären Ablage der Listen benötigt wird. Bei Softwareimplementierung kann hierzu der Arbeitsspeicher verwendet werden.

Die Steigerung der Performanz durch die direkte Implementierung des SPIHT-Verfahrens in Hardware birgt die Problematik inwieweit die transformierten Bilddaten und die Listen für schnelle Speicherzugriffe lokal mit der Logik zur Abarbeitung des Algorithmus implementiert werden können. Diese lokale Nähe ist für eine hohe Performanz und gute Energieeffektivität erforderlich. In [Fry05] und [Weg06] sind hardwarebeschleunigte SPIHT-Verfahren beschrieben. Sie haben jedoch die Nachteile, dass sie einen hohen Ressourcenbedarf aufweisen und mehrere Bausteine für ihre Implementierung notwendig sind. Die Ursache hierfür liegt insbesondere in der hohen Datenmenge, auf die während der Codierung zugegriffen werden muss, und die zusätzliche Speicherbausteine erforderlich macht. Die lokale Nähe zwischen Logik und temporärem Speicher ist nicht gegeben. Moderate Ressourcenanforderung beinhaltet das SPIHT-basierte Verfahren mit partitionierter *Wavelet*-Transformation [Rit02]. Die Transformation wird in kleinen Bildbereichen durchgeführt, die jeweils codiert werden. Da die Datenmenge, die temporär zur Codierung gehalten werden muss, proportional zur Menge an Pixel in dem Bildbereich ist, kann sie erheblich reduziert und lokal nah zur Logik implementiert werden. Bei dem beschriebenen Verfahren werden allerdings Blockartefakte sichtbar.

Die auf Raumsonden implementierten Algorithmen für DWT-basierte Verfahren lagen, wie das SPIHT-Verfahren, bisher in den meisten Fällen außerhalb existierender Standards. Neben dem Algorithmus zur Kompression musste auch der zugehörige Algorithmus zur Dekompression der Daten auf der Erde entworfen werden. Mit der Einführung des JPEG2000-Standards durch die Joint Photographic Experts Group, Teil der International Organization for Standardization (ISO), ist ein *Wavelet*-basierter Kompressionsstandard mit Kontextmodellierung festgelegt worden. Diese Standardisierung erlaubt die Benutzung verschiedener Anwendungen auf Basis dieses Standards ähnlich zu dem weit verbreiteten auf DCT basierten JPEG-Standard. Der JPEG2000-Standard unterstützt sowohl die verlustlose als auch die verlustbehaftete Kompression und zeigt sehr gute Kompressionseigenschaften sowohl bei niedrigen als auch bei hohen Kompressionsfaktoren.

Im Hinblick auf mögliche Hardwareimplementierungen sind bei der JPEG2000-Kompression im Vergleich zum SPIHT-Algorithmus bessere Integrationsmöglichkeiten zu erwarten. So sind bei

JPEG2000 eine geringere Menge an temporären Daten sowie bessere Parallelisierungsmöglichkeiten anzunehmen. Letzteres ist darin begründet, dass die Codierung im Gegensatz zu SPIHT nicht bandübergreifend erfolgt und somit bei JPEG2000 mehrere Bänder parallel und unabhängig voneinander codiert werden können. Für die Implementierung in einer kompakten Kamera mit sehr beschränkten Ressourcen zeigt die JPEG2000-Kompression daher Vorteile. Die Kompressionsergebnisse von JPEG2000 und SPIHT unterscheiden sich nur minimal. Beim JPEG2000-Algorithmus werden innerhalb des Standards viele optionale Funktionalitäten, wie z. B. die *Region of Interest*-Codierung, unterstützt. Viele dieser optionalen Funktionalitäten sind durch die derzeit verfügbaren JPEG2000-Dekompressions-Programme abgedeckt und die Erstellung angepasster Dekompressionsroutinen ist nicht erforderlich. Bei SPIHT erfordern dagegen erweiterte Funktionalitäten die Abänderung des originalen Algorithmus nach A. Said und W. Pearlman bei Kompression und Dekompression.

Exemplarisch für ein *Wavelet*-basiertes Kompressionsverfahren wird aus den genannten Gründen in dieser Arbeit der *State of the Art* JPEG2000-Algorithmus, der in der Kamera Venus Express / VMC eingesetzt wird, verwendet und in den Abschnitten 3.3 - 3.7 näher erläutert.

3.1 Vorverarbeitende Bildverarbeitungsfunktionen

Werden mit einer CCD-Kamera Aufnahmen erzeugt, weisen diese häufig ein verzerrtes, leicht fehlerhaftes Bild des Aufnahmeobjekts auf. Diese Unzulänglichkeiten können teilweise erkannt und das Bild kann mit dieser Information korrigiert werden. Werden Artefakte z. B. mit scharfen Kanten vorher aus dem Bild entfernt, so entfallen die hierzu notwendigen komprimierten Daten und eine effektivere Kompression kann erzielt werden. Wird die Kompression in der DPU durchgeführt, sind die entsprechenden Korrekturmaßnahmen zur Reduzierung der Nichtlinearitäten ebenfalls in der DPU durchzuführen. Insgesamt sollten dem Kompressionsalgorithmus Daten von guter Qualität, d. h. mit möglichst wenig unnötiger Information zur Verfügung gestellt werden.

3.1.1 *Flat Field*-Korrektur und Entfernung fehlerhafter Bildpunkte

Bei einer Aufnahme entstehen durch verschiedene Einflüsse Artefakte im Bild, die zu einer Qualitätsminderung führen können. Ursachen hierfür können sein:

- Ungleiche Ausleuchtung des Bildes (*Vignettierung*)
- Reflexe durch optische Elemente oder Flächen
- Verunreinigungen

Zudem variiert die Empfindlichkeit der einzelnen Pixel auf dem CCD untereinander. Die Effekte sind nicht nur auf einzelne Pixel beschränkt, sie können auch ganze Spalten oder Teile davon betreffen.

Ein Ausgleich bzw. eine Reduzierung der Effekte kann mit Hilfe der *Flat Field*-Korrektur erfolgen, die einer linearen Korrektur jedes Pixels mit zugehöriger Verstärkung und Offset entspricht:

$$I_{Cal} = \frac{(I_{Raw} - I_{Dark}) \cdot M}{I_{FlatField} - I_{Dark}}. \tag{3.1}$$

Dabei beschreibt I_{Cal} das kalibrierte Bild, I_{Raw} das ursprüngliche und damit nicht kalibrierte Bild, I_{Dark} ein Dunkelbild, $I_{FlatField}$ das *Flat Field*- bzw. Weißbild und M den durchschnittlichen Pixelwert des mit I_{Dark} korrigierten *Flat Field*-Bildes.

Zur Durchführung einer *Flat Field*-Korrektur müssen zwei Kalibrationsbilder erzeugt werden. Das erste Kalibrationsbild ist das Dunkelbild I_{Dark}, bei dem die Akkumulation von Ladung in der Kamera (CCD + Ausleseelektronik) innerhalb einer bestimmten Zeit ohne Beleuchtung des CCD gemessen wird. Bei dem erzeugten Bild haben alle Pixel ungefähr den gleichen Wert, der sich aus dem Offset der Elektronik bzw. aus dem in der entsprechenden Belichtungszeit aufgelaufenen Dunkelstrom ergibt. Mit dieser Bestimmung kann der entsprechende Fehlerwert vom belichteten Bild abgezogen werden, was sich vor allem bei Aufnahmen mit langer Belichtungszeit und niedriger Beleuchtungsstärke bemerkbar macht. Das zweite Kalibrationsbild ist das *Flat Field*- oder Weißbild $I_{FlatField}$, das die Signalstärke jedes CCD-Pixels auf eine bestimmte Beleuchtung beschreibt. Eine ungleichmäßige Ausleuchtung und die unterschiedliche Empfindlichkeit der Pixel können mit dieser Information ausgeglichen werden. Zur Bestimmung des *Flat Field*-Bildes sollte die Beleuchtung stark genug oder die Beleuchtungszeit genügend lang gewählt werden, damit hohe Signalwerte erreicht werden (z. B. 50 % - 75 % der maximalen Signalstärke). Das *Flat Field*-Bild seinerseits wird noch mit dem Dunkelbild korrigiert. Sowohl das Dunkel- als auch das *Flat Field*-Bild sollte unter den gleichen Bedingungen (Belichtungszeit und Betriebstemperatur) aufgenommen werden, wie sie auch bei der Aufnahme des zu kalibrierenden Bildes vorliegen. Mit diesen beiden Kalibrationswerten kann die *Flat Field*-Korrektur entsprechend Gleichung (3.1) durchgeführt werden.

Es gibt des Weiteren überempfindlich reagierende, so genannte *Hot Pixel* mit hohem Dunkelstrom und *Cool Pixel, die* gegenüber der Belichtung weitgehend oder völlig unempfindlich sind. Sowohl *Hot Pixel* als auch *Cool Pixel* resultieren aus Fehlstellen durch den Herstellungsprozess. Der zu erwartende Wert dieser *Bad Pixel* kann i. Allg. durch die Nachbarpixel abgeschätzt werden.

3.1.2 Entfernung von *Cosmic Rays*

Cosmic Rays ist energiereiche, kosmische Strahlung, die vereinzelt auf den CCD-Chip trifft, und eine große Menge an Elektronen freisetzt. Im Bild machen sie sich durch helle scharfe Punkte oder Streifen bemerkbar. *Cosmic Rays* sind zufällig verteilt und sind umso ausgeprägter, je länger die Aufnahmezeit ist.

Eine einfache und sehr zuverlässige Methode zur Erkennung von *Cosmic Rays* ist die zeitliche Aufteilung einer Aufnahme in zwei oder mehrere Einzelaufnahmen (*Cosmic Ray Split*). Mit der erwarteten Verteilung der Unterschiede zwischen zwei Einzelaufnahmen können *Cosmic Rays* aufgrund ihrer zufälligen Verteilung entdeckt werden. Aufgedeckte *Cosmic Ray*-Pixel können anschließend durch die entsprechenden Pixel einer anderen Einzelaufnahme ersetzt werden. Die Einzelaufnahmen werden aufaddiert und liefern, allerdings unter Inkaufnahme eines höheren Rauschens, das Gesamtbild. Die Unterteilung in mehrere Einzelbilder ist nur sinnvoll, wenn sich die Aufnahmesituation während der Gesamtdauer der Belichtung nicht ändert. Für schnell bewegliche Objekte ist diese Methode nicht geeignet, für langsamer bewegliche Objekte kann sie durch Routinen ergänzt werden, um einen Deckungsabgleich zweier Einzelbilder durchzuführen. Andere, weniger zuverlässige Methoden untersuchen mit speziellen Algorithmen Bilder nach vorgegebenen Kriterien, u. a. unter Zuhilfenahme des Histogramms [Far05]. Hierbei bildet das Gesamtbild die Basis, es erfolgt keine Unterteilung in mehrere Einzelaufnahmen. Auffälligkeiten in der Aufnahme, wie z. B. Streifen mit hohen Pixelwerten, können damit zu einem gewissen Grad entdeckt werden. Diese Pixel werden durch Informationen aus den Nachbarpixeln ersetzt. Im Gegensatz zur Mehrfachaufnahme ist die Wiederherstellung des tatsächlichen Pixelwertes jedoch nicht möglich. Ob eine Entfernung der *Cosmic Rays* eingesetzt wird, ist i. Allg. von der durchschnittlichen Häufigkeit der *Cosmic Rays* abhängig.

3.2 Bildkompression

Im Anschluss an die vorverarbeitenden Bildfunktionen erfolgt die Bildkompression. Die meisten Bildkompressionsverfahren lassen sich wie in Bild 3.2 darstellen. Bei der Kompression werden die Bilddaten zunächst durch eine Transformation dekorreliert. Anschließend erfolgt bei verlustbehafteten Verfahren die Quantisierung (optional) und die Codierung der Daten. Ein Modell liefert die zur Entropie-Codierung notwendigen Wahrscheinlichkeiten. Nach Übertragung des komprimierten Datenstroms sowie evtl. vorhandener Nebeninformationen kann die Decodierung erfolgen.

Bei einer Kompression ist der Kompressionsfaktor c als das Verhältnis der Originalgröße $N \times M$ eines Bildes mit B Bit pro Bildpunkt zur Bitmenge der komprimierten Bilddaten d definiert:

$$c = \frac{N \cdot M \cdot B}{d}. \tag{3.2}$$

Alternativ gibt der Bit-pro-Pixel (bpp)-Wert die Anzahl an Bits an, die durchschnittlich zur komprimierten Repräsentation eines Pixels benötigt wird:

$$bpp = \frac{d}{N \cdot M}. \tag{3.3}$$

Die verlustbehaftete Kompression hat zum Ziel, eine möglichst hohe Bildqualität bei möglichst hohem Kompressionsfaktor zu erhalten. Zur objektiven Beurteilung dieser Bildqualität wird häufig der mittlere quadratische Fehler (*Mean Squared Error,* MSE) verwendet:

$$MSE = \frac{1}{N \cdot M} \cdot \sum_{i=0}^{N-1} \sum_{k=0}^{M-1} \left(x_{i,k} - y_{i,k} \right)^2 \tag{3.4}$$

und gibt die Stärke der Verzerrung an. Hierbei entspricht $x_{i,k}$ dem original Pixelwert, $y_{i,k}$ dem aus den komprimierten Daten rekonstruierten Pixelwert. Mittels MSE wird das Signal-Rausch-Verhältnis (*Peak Signal to Noise Ratio,* PSNR) definiert zu

$$PSNR = 10 \cdot \log_{10} \frac{(2^B - 1)^2}{MSE} \tag{3.5}$$

Das Verhältnis zwischen maximal möglicher Signalenergie und der Signalenergie des Pixelrauschens wird dabei logarithmisch repräsentiert. Der absolute Wert spielt eine untergeordnete Rolle, vielmehr ist ein Vergleich der Werte unterschiedlich komprimierter Bilder interessant. Weitere Bewertungskriterien werden hier nicht genannt, da in dieser Arbeit der Entwurf einer Kamera-DPU unter den Optimierungsgesichtspunkten Ausbeute, Ressourcenbedarf und Fehlertoleranz bei den Randbedingungen einer Weltraummission im Vordergrund steht.

In dieser Arbeit wird daher mit Verwendung der PSNR-Metrik ein gebräuchliches Standardverfahren zur Beurteilung der Bild- und schließlich der Kompressionsqualität herangezogen. Dabei ist zu beachten, dass die in dieser Arbeit gezeigten PSNR-Werte sich ausschließlich auf die Bilddatenkompression selbst, unabhängig vom aufgenommenen Objekt, beziehen. Die Entstehungsgeschichte der Daten vor dem Zeitpunkt der Kompression wird nicht beachtet. So bleiben z.B. physikalische Objekteigenschaften, Einflussfaktoren zur Bilddatengenerierung oder die Vorverarbeitung unberücksichtigt.

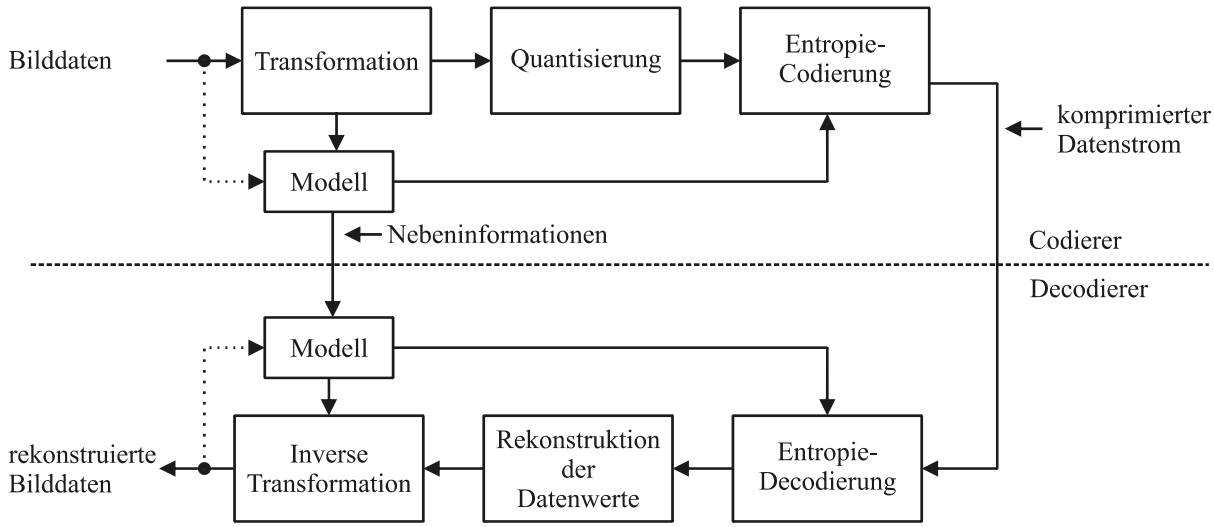

Bild 3.2: Blockschaltbild eines Bildkompressionsverfahrens

3.3 JPEG2000: Übersicht und Eckpunkte

JPEG2000 ist ein neuer Standard für Bildkompression. Aufgrund sehr guter Kompressionsergebnisse wird der JPEG2000-Algorithmus in der Kamera Venus Express / VMC und in dieser Arbeit exemplarisch für einen Bildkompressionsalgorithmus verwendet [Mar04][Kam04].

Die wichtigsten Merkmale des auf der *Wavelet*-Transformation basierten JPEG2000-Algorithmus sind

- Sehr gute Kompressionsergebnisse
- Verlustlose und verlustbehaftete Kompression in einem Algorithmus
- *Region of Interest* (ROI)-Codierung
- Berücksichtigung von Hardwareimplementierung beim Entwurf des Standards
- Freies standardisiertes Verfahren, beruhend auf der *Wavelet*-Transformation
- Progressive Übertragung in Abhängigkeit von Qualität und Auflösung
- Raum für Metadaten

Im Folgenden werden die Bestandteile des Algorithmus dargestellt, die für kompakte Kamerainstrumente von Bedeutung sind. Mit der Beschreibung dieser Bestandteile kann beurteilt werden, inwiefern Teile des Algorithmus durch eine Hardwareimplementierung beschleunigt werden können.

Bild 3.3 zeigt den Aufbau eines JPEG2000-Kompressionssystems. Im ersten Schritt, der Vorverarbeitung, wird das Gesamtbild in rechteckige, sich nicht überlappende Blöcke, so genannte *Tiles* unterteilt. Sie bilden die Basis für die darauf folgenden Operationen und werden unabhängig voneinander weiterverarbeitet. Nach der *Wavelet*-Transformation eines *Tile* können bei verlustbehafteter Kompression die erzeugten Koeffizienten einer Quantisierung unterzogen werden. Der anschließende Codierer führt den *Embedded Block Coding with Optimized Truncation* (EBCOT)-Algorithmus aus und arbeitet auf Basis relativ kleiner, voneinander unabhängiger *Codeblöcke*, die Daten von einem *Subband* enthalten. Da sowohl die *Codeblöcke* als auch die zuvor genannten *Tiles* unabhängig voneinander sind, kann die Bearbeitung mittels Parallelisierung beschleunigt werden. Die Codierung der *Codeblöcke* ist mit zwei Codierungsstufen, *Tier-1* und *Tier-2,* aufgebaut. *Tier-1* beinhaltet ein Kontext-basiertes Wahrscheinlichkeitsmodell für die sich im *Codeblock* befindlichen Koeffizienten sowie einen Arithmetischen Codierer. Die Codierung ist in Bitebenen organisiert. Es werden die Einträge aller Bitebenen, beginnend mit der höchstwertigen Bitebene, die Einträge ungleich Null enthält, bis zur niederstwertigen Bitebene bearbeitet. *Tier-1* erzeugt einen Bitdatenstrom. Die Aufgabe von *Tier-2* ist es, die einzelnen komprimierten Informationen aus *Tier-1* einer Qualitätsstufe zuzuweisen. Das bedeutet, sie werden im Wesentlichen entsprechend ihrer Wichtigkeit sortiert und bei Bedarf wird die Übertragung der weniger wichtigen Informationen zugunsten eines höheren Kompressionsfaktors unterbunden (*Rate Control*). Bei verlustbehafteter Kompression ist

dieses neben der Quantisierung die zweite Stelle, an der gezielt Information entfernt werden kann. Die *Rate Control* kann bereits parallel zur Quantisierung und zu *Tier-1* wirken. Hierdurch kann sie bei genügender Qualität der codierten Daten weitere unnötige Codierung unterbinden und damit den gesamten Prozess beschleunigen. Die Implementierung der *Rate Control* hat einen bestimmenden Einfluss auf die Qualität des Bildes mit verlustbehafteter Kompression. Bei der Dekompression werden die Schritte in umgekehrter Reihenfolge und jeweils invertiert ausgeführt. Die Zerlegung in *Tiles*, *Codeblöcke* und Bitebenen ist in Bild 3.4 dargestellt. Die wichtigsten Bestandteile des JPEG2000-Kompressionsalgorithmus werden in den Abschnitten 3.4, 3.5 und 3.6 näher beschrieben.

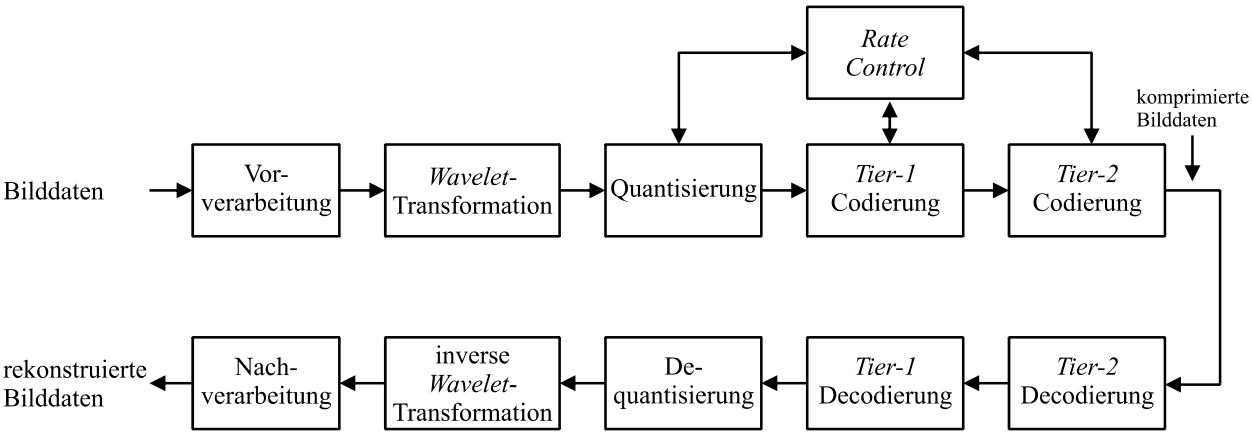

Bild 3.3: JPEG2000-Kompressionssystem

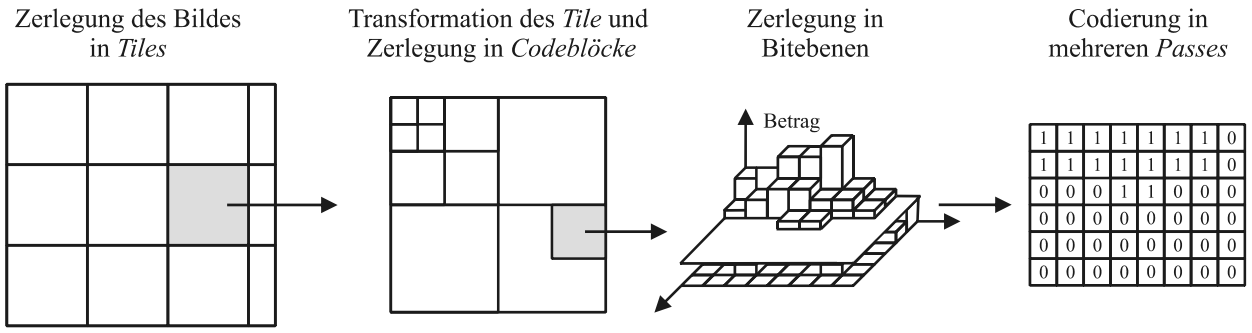

Bild 3.4: Zerlegung und Codierung

3.4 Diskrete *Wavelet*-Transformation

Basis für die JPEG2000-Kompression ist wie bei dem vielfach in der Raumfahrt eingesetzten SPIHT-Verfahren die Diskrete *Wavelet*-Transformation. Gegenüber der häufig zur Analyse im

Frequenzbereich herangezogenen *Fourier*-Transformation, die auf einer Signalzerlegung in lokal nicht begrenzte, trigonometrische Funktionen mit zugehörigen Problemen an den Signalgrenzen beruht, erfolgt bei der *Wavelet*-Transformation die Analyse im Orts- und Frequenzbereich. Die Signale werden in räumlich begrenzte, verschieb- und skalierbare *Wavelets* zerlegt.

Ausgangspunkt einer *Wavelet*-Transformation ist ein Basis- bzw. Mutter-*Wavelet* Ψ. In der Transformation kommen die entsprechend skalierten und örtlich / zeitlich verschobenen Varianten $\Psi_{a,b}$ des Basis-*Wavelets* zum Einsatz.

Mit

$$\Psi_{a,b}(t) = \frac{1}{\sqrt{a}} \cdot \Psi\left(\frac{t-b}{a}\right) \tag{3.6}$$

und a als Skalierungsfaktor (Frequenzdurchtastung), b als Verschiebungswert (Zeit-/ Ortsdurchtastung) kann eine Funktion x durch eine Linearkombination von *Wavelet*-Basisfunktionen mit den zugehörigen *Wavelet*-Koeffizienten $y(a,b)$ dargestellt werden (hier mit $a \in \mathbb{N}^+$, $b \in \mathbb{Z}$):

$$x(t) = \sum_{a=1}^{a=\infty} \sum_{b=-\infty}^{b=\infty} y(a,b) \cdot \Psi_{a,b}(t) \,. \tag{3.7}$$

Bei der kontinuierlichen *Wavelet*-Transformation (CWT) beruht die Analyse darauf, dass die *Wavelets* beim Strecken oder Stauchen ihre Frequenz ändern und sich den verschiedenen Signalkomponenten automatisch anpassen. Bei der Analyse wird ein stark gestrecktes *Wavelet* verwendet, um die tiefen Frequenzen zu bestimmen. Anschließend wird das *Wavelet* schrittweise gestaucht, um immer feinere Frequenzen zu erfassen. Die Rekonstruktion der Ursprungsdaten erfolgt durch die Addition der entsprechenden *Wavelet*-Transformierten. Diese *Wavelet*-Transformierten, auch *Wavelet*-Koeffizienten genannt, ergeben sich aus dem Grad der Übereinstimmung des Signals mit dem entsprechend gestreckten *Wavelet*. Mit $x(t)$ als Wert der Ausgangsfunktion zur Zeit t stellt sich die CWT folgendermaßen dar:

$$y(a,b) = \int_{-\infty}^{+\infty} x(t) * \Psi_{a,b} dt \,. \tag{3.8}$$

Durch die theoretisch mögliche Bestimmung von unendlich vielen *Wavelet*-Koeffizienten ist die CWT sehr rechenintensiv und aufgrund der Überlappung der einzelnen *Wavelets* auch hoch redundant. In der Praxis findet daher häufig die Diskrete *Wavelet*-Transformation (DWT) Verwendung, die sowohl für die Skalierung als auch für die Verschiebung diskrete Werte benutzt. Häufig verwendete DWTs sind dyadische DWTs, bei denen die *Wavelets* um Potenzen von 2 gestreckt werden, also $a = 2^m$ mit $m \in \mathbb{Z}$ als Transformationsstufen.

In der Ausführung als schnelle *Wavelet*-Transformation (*Fast Wavelet Transformation,* FWT) werden als Basisfunktionen die so genannten *Scaling*-Funktionen und die *Wavelets* verwendet. Die Realisierung einer *Scaling*-Funktion erfolgt durch einen Tiefpass-Filter, der nur entsprechend tiefe Frequenzen eines Signals durchlässt, während die *Wavelets* als Hochpass-Filter erfolgen, die nur hohe Frequenzen erfassen. Nach erfolgter Filterung mit Hoch- und Tiefpassfilter wird das Signal einem *Subsampling*-Prozess unterworfen, d. h. unterabgetastet und um den Faktor 2 reduziert. Damit wird jeder zweite Wert des Ausgangssignals entfernt. Diese Reduzierung der Signalwerte entspricht dem Strecken der *Scaling*-Funktion. Rekursiv wird anschließend der niederfrequente Anteil, d. h. der dezimierte Tiefpass-Anteil des Signals einer weiteren Filterung unterzogen. Bei der Analyse werden die Daten somit in Details (durch Hochpass) und Annäherungen (durch Tiefpass) einer bestimmten Auflösung zerlegt. Diese *Subband*-Zerlegung hat die praktische Auswirkung, dass die Funktion in Anteile verschiedener Frequenzbereiche aufgeteilt wird.

Die Filterung entspricht einer Faltung des Eingangssignals mit der Impulsantwort des entsprechenden Filters *h* bei diskreter Ausführung:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k] = x[n] * h[n] \,. \tag{3.9}$$

Bezeichnet man h_1 bzw. h_0 als die Impulsantworten der Hochpass- bzw. Tiefpass-Filter und $y_1^{(m)}$ bzw. $y_0^{(m)}$ als die zugehörigen gefilterten Werte bei der Transformationsstufe *m*, so erhält man folgende eindimensionale Zerlegung:

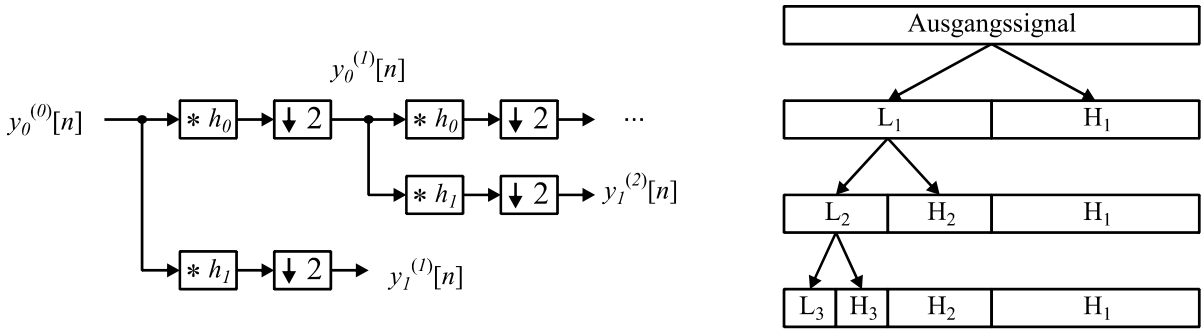

Bild 3.5: Eindimensionale Zerlegung mittels *Wavelet*-Transformation in Tiefpass L und Hochpass H Bereiche [Tau02]

Aus Gleichung (3.9) geht hervor, dass bei der Faltung in Abhängigkeit von der Filterlänge entsprechend viele Signalwerte gewichtet werden müssen. Die Menge der in der Analyse verwendeten Hoch- und Tiefpass-Filter wird als Filterbank bezeichnet. Bei JPEG2000 wird u. a. der *LeGall* 5-3 Filter eingesetzt. Er ist ein FIR-Filter mit Integerkoeffizienten und kann sowohl für verlustlose als auch für verlustbehaftete Kompression eingesetzt werden. Für verlustbehaftete

Kompression kann außerdem der realwertige CDF 9-7 Filter mit entsprechend erhöhter Anzahl an Filterkoeffizienten verwendet werden.

Mit dem 5-3 Integer Filter

$$\text{Hochpass:} \qquad h_1 = \frac{1}{2} \cdot \left(-z^0 + 2 \cdot z^1 - z^2 \right) \tag{3.10}$$

$$\text{Tiefpass:} \qquad h_0 = \frac{1}{8} \cdot \left(-z^{-2} + 2 \cdot z^{-1} + 6 \cdot z^0 + 2 \cdot z^1 - z^2 \right) \tag{3.11}$$

ergeben sich im Zeitbereich folgende Berechnungsvorschriften:

$$\text{Hochpass:} \qquad x'_{2i+1} = \frac{1}{2} \cdot \left(-x_{2i} + 2 \cdot x_{2i+1} - x_{2i+2} \right); \quad i = 0..\frac{N}{2} \tag{3.12}$$

$$\text{Tiefpass:} \qquad x'_{2i} = \frac{1}{8} \cdot \left(-x_{2i-2} + 2 \cdot x_{2i-1} + 6 \cdot x_{2i} + 2 \cdot x_{2i+1} - x_{2i+2} \right); \quad i = 0..\frac{N}{2}. \tag{3.13}$$

Die Aufteilung der Daten in Detail- und Annäherungswerte ergibt die halbierte Anzahl an Ausgangswerten für jeweils Hoch- und Tiefpass und damit die insgesamt gleich bleibende Datenmenge. Bei zweidimensionalen Bilddaten erfolgt die Transformation mit zwei getrennten, aufeinander folgenden eindimensionalen Transformationen. Hierbei werden im ersten Schritt die einzelnen, z. B. horizontalen Bildzeilen als eindimensionale Daten betrachtet und transformiert. Anschließend erfolgt eine vertikale Transformation der bereits horizontal transformierten Bilddaten. Dieser Vorgang wiederholt sich auf den Tiefpassanteil LL, so dass sich das Ursprungsbild hierarchisch in *Subbänder* mit der entsprechenden Transformations- oder Auflösungsstufe (z. B. HL_1, LH_1, HH_1) zerlegt (siehe Bild 3.6).

Das transformierte Bild ist aufgrund dieser Rekursion durch immer kleiner werdende Hochpassanteile und einen Tiefpassanteil gekennzeichnet. In den Hochpassanteilen der ersten Transformationsstufe werden die feinen Bildstrukturen erfasst, in den Hochpassanteilen der folgenden Transformationsstufen zunehmend gröbere Bildstrukturen. Nach der Transformation nehmen die Bildpunkte mit Werten nahe Null zu und es gibt größere Bereiche, in denen die Bildpunkte einen sehr ähnlichen Wert haben. Die Energie konzentriert sich nach der Transformation auf den LL-Anteil.

Die direkte Umsetzung der Filterung nach Gleichung (3.9) erfordert für jeden transformierten Wert Multiplikationen und Additionen in der Anzahl der Filterlänge. In praktischen Anwendungen wird daher häufig die effizientere Implementierung mit *Lifting Steps* verwendet [Tau02]. Mit Hilfe dieses *Lifting*-Schemas können die Hoch- und Tiefpassfilter einer *Wavelet*-

Transformation umgeschrieben werden und es ergeben sich aus Gleichung (3.12) und (3.13) folgende Berechnungsvorschriften:

$$\text{Hochpass:} \qquad x'_{2i+1} = x_{2i+1} - \frac{1}{2} \cdot \left(x_{2i} + x_{2i+2} \right); \qquad i = 0..\frac{N}{2} \tag{3.14}$$

$$\text{Tiefpass:} \qquad x'_{2i} = x_{2i} + \frac{1}{4} \cdot \left(x'_{2i-1} + x'_{2i} + 2 \right); \qquad i = 0..\frac{N}{2}. \tag{3.15}$$

Hierdurch werden beim Tiefpassfilter bereits berechnete Hochpasswerte verwendet und es fallen weniger Rechenoperationen an.

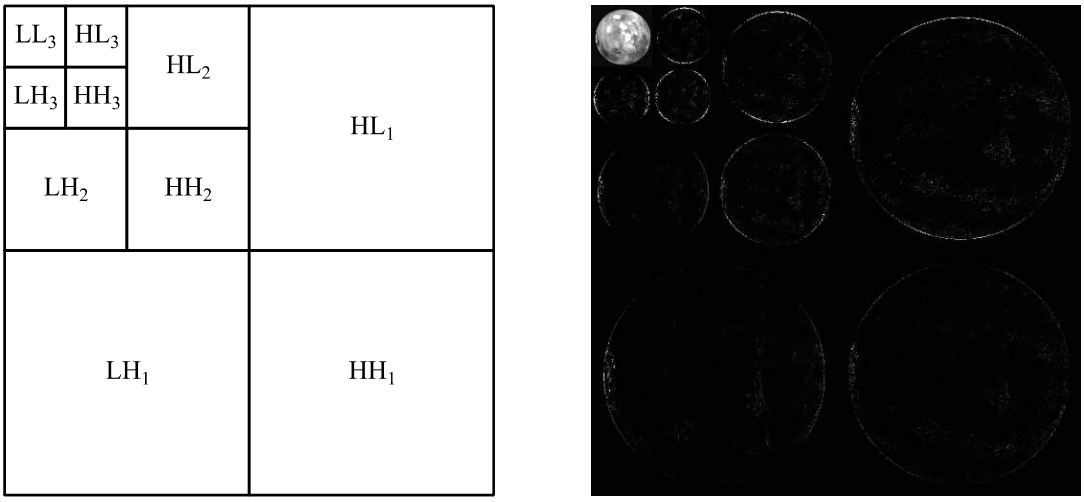

Bild 3.6: Zweidimensionale Zerlegung mittels *Wavelet*-Transformation in Tiefpass L und Hochpass H Bereiche

Eine mögliche Implementierung der diskreten *Wavelet*-Transformation in Hardware erscheint machbar und auch sinnvoll. Es ist ein temporärer Zeilen- / Spalten-Speicher vorstellbar, auf dessen Werte jeweils die Hoch- und Tiefpassfilter für eine eindimensionale Transformation angewendet werden. Werden die Daten einer Auflösungsstufe zeilen- und anschließend spaltenweise unter Verwendung des Zwischenspeichers transferiert, so erhält man nach entsprechender Anzahl von Auflösungsstufen das vollständig transformierte Bild.

3.5 EBCOT

Entsprechend Bild 3.4 wird ein *Tile* nach dessen Transformation weiter in *Codeblöcke* zerlegt, die anschließend unabhängig voneinander mit dem *Embedded Block Coding with Optimized Truncation* (EBCOT)-Algorithmus weiterverarbeitet werden. Ein *Codeblock* erhält die Koeffizienten von genau einem *Subband* und hat eine maximale Größe von 4.096 Werten (z. B. ein Bereich von 64 × 64 Koeffizienten). Durch die hierarchische Zerlegung des Bildes nach der

Transformation liegt die Größe des *Codeblocks* damit bei vergleichsweise kleinen *Tiles* häufig unter 64×64 Koeffizienten. Die voneinander unabhängige Codierung der relativ kleinen *Codeblöcke* ist für Hardwareausführungen von Vorteil. Zum wahlfreien Zugriff auf die Daten eines *Codeblocks* ist nur wenig Speicherplatz erforderlich, der häufig lokal bei den Verarbeitungsfunktionen zur Verfügung gestellt werden kann.

Bild 3.7 zeigt das Blockdiagramm des EBCOT-Algorithmus. *Tier-1* beinhaltet eine Kontextmodellierung sowie eine adaptive Arithmetische Codierung und erzeugt einen *Bitstream*. Dieser wird während *Tier-2* mittels eines *Rate Distortion*-Algorithmus umsortiert. Der *Rate Distortion*-Algorithmus kann nach vollständiger Codierung eines *Tile* in *Tier-1*, aber auch schon während der Codierung aktiv sein.

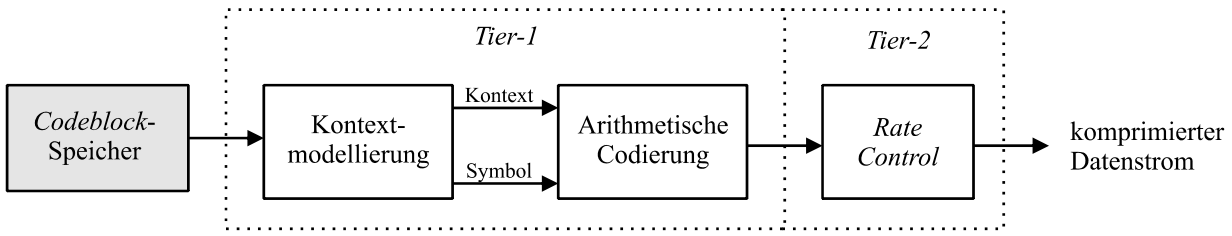

Bild 3.7: EBCOT-Verfahren

3.5.1 Kontextmodellierung

Bei der Kontextmodellierung wird ein *Bitplane Coder* (BPC) verwendet. Dieser ist Bitebenen-orientiert (siehe Bild 3.4) und arbeitet beginnend mit der höchstwertigen Ebene, die mindestens einen Wert ungleich Null enthält, bis hin zur niederstwertigen Ebene. Ein wichtiger Begriff ist hierbei die Signifikanz eines Koeffizienten, die während der Modellierung eines jeden Koeffizienten gültig ist. Zu Beginn der Codierung eines *Codeblocks* sind alle Koeffizienten nicht signifikant und werden in genau der Bitebene signifikant, in der der Koeffizient die erste 1 enthält, d. h. an der Stelle des höchstwertigen Bits des Koeffizienten. Abgesehen von der obersten Bitebene mit Werten ungleich null wird jede Bitebene dreimal vollständig mittels so genannter *Passes* nach einem festen streifenförmigen Muster hintereinander bearbeitet. In Bild 3.8 ist die Bearbeitung der Streifen, die eine maximale Höhe von vier Koeffizienten und die Breite des *Codeblocks* haben, gezeigt. Im ersten Durchgang, dem *Significance Pass* (SP) werden die Bits einer Bitebene codiert, deren Koeffizienten in den vorhergehenden Bitebenen noch nicht signifikant waren. Während des zweiten Durchgangs, dem *Magnitude Refinement Pass* (MRP), wird das Bit der aktuellen Bitebene für diejenigen Koeffizienten codiert, die bereits in vorhergehenden Bitebenen signifikant wurden. Im dritten und letzten Durchgang, dem *Cleanup Pass* (CP), werden die restlichen, noch nicht codierten Bits der Bitebene codiert. Dieses betrifft die Koeffizienten, die insignifikant bleiben. Jedes Koeffizienten-Bit innerhalb einer Bitebene wird nur in einem der drei *Passes* codiert und liefert bei der Codierung ein Symbol-Kontext-Paar an den Arithmetischen Codierer. Zur Erzeugung des Symbols und des Kontexts greifen die

Passes auf Grundoperationen zu (*Zero Coding* (ZC), *Sign Coding* (SC), *Magnitude Refinement Coding* (MRC)*, Run Length Coding* (RLC)) [Bol00]. Der Kontext-Vektor eines Koeffizienten besteht aus den Signifikanz-Zuständen der acht nächsten Nachbarn, entsprechend Bild 3.9, so dass dessen Bestimmung häufige Zugriffe auf die Daten erforderlich macht. Liegen die nächsten Nachbarn außerhalb des aktuellen *Codeblocks*, so werden sie als nicht signifikant betrachtet. Die möglichen Kontext-Vektoren werden auf lediglich 18 Kontexten abgebildet, d. h. nicht jede Kombination aus Signifikanz-Zuständen der nächsten Nachbarn erzeugt unabhängige Kontexte. Die Abbildung auf Kontexte erfolgt zudem in Abhängigkeit der einzelnen Codierungs-*Passes*.

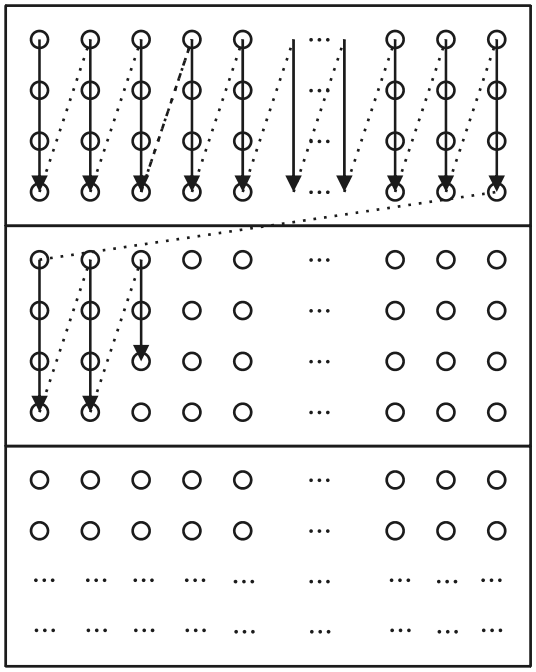

Bild 3.8: Streifenbearbeitung innerhalb eines *Codeblocks*

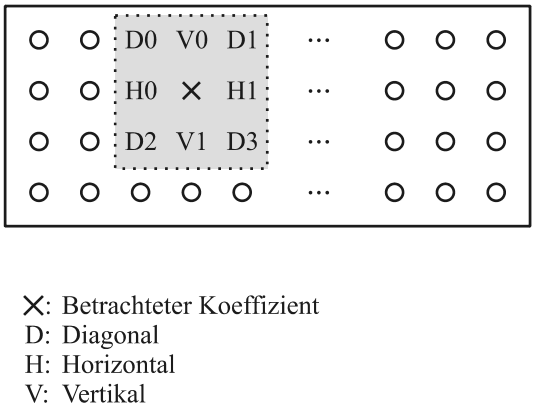

Bild 3.9: Kontextfenster mit diagonalen, horizontalen und vertikalen Beiträgen

Die Arbeitsweise des *Bitplane Coder* am *Codeblock* zeigt, dass es sich bei der Kontextmodellierung auf Basis von Bitebenen im Wesentlichen um Bitoperationen mit sehr häufigen Zugriffen auf die Daten handelt. Zusammen mit dem relativ geringen Speicherbereich, der zur Abarbeitung eines *Codeblocks* notwendig ist, sind Hardwareimplementierungen des *Bitplane Coder* gut möglich und können zu einem sehr starken Performanzgewinn gegenüber einer Softwarelösung führen. Verglichen zur DWT sind weit mehr Zugriffe und eine längere Bearbeitungszeit der Daten erforderlich.

3.5.2 Arithmetische Codierung durch *MQ Coder*

Die prinzipielle Arbeitsweise eines Arithmetischen Codierers besteht darin, das Intervall [0; 1[auf Basis von Wahrscheinlichkeiten für die Eingangssymbole rekursiv zu unterteilen. Eine direkte Umsetzung des Arithmetischen Codierers hat zur Folge, dass das Intervall $[C_i; C_i + A_i[$ mit der Basis C_i und der Breite A_i immer kleiner wird und daher eine unendlich hohe Genauigkeit erforderlich wäre. Da dieses technisch nicht zu realisieren ist, sind Verbesserungen am Algorithmus entwickelt worden, die auf eine neue Skalierung bzw. Normierung während der Codierung beruhen. Die Arithmetische Codierung wird im JPEG2000-Standard durch den *MQ Coder* durchgeführt und setzt Bit-Symbole mit ihren Kontexten durch adaptive binäre Codierung in ein Entropie-codiertes Codewort um. Der *MQ Coder* ist eine Variante des von IBM Research entwickelten *Q Coder* [Pen88] mit veränderter Wahrscheinlichkeitsabschätzung (*MQ Coder = Modified Q Coder*). Der *Q Coder* ist durch eine multiplikationsfreie, renormalisierungsgesteuerte Wahrscheinlichkeitsabschätzung und *Bit Stuffing*, d. h. automatisches Hinzufügen von Bits zum codierten Datenstrom, gekennzeichnet.

Bild 3.10 zeigt das prinzipielle Funktionsbild des *Q Coder*. Das Intervall mit der Breite A und der Basis C, welches das Codewort enthält, ist in ein *Least Probable Symbol* (LPS) und *More Probable Symbol* (MPS) Intervall unterteilt (siehe Bild 3.11). Die Größe der Subintervalle ist durch die erwartete LPS-Wahrscheinlichkeit Q_e und dementsprechend die erwartete MPS-Wahrscheinlichkeit $1\text{-}Q_e$ bestimmt. Während zur Codierung eines LPS keine Änderungen am Codewort erfolgen, wird bei der Codierung eines MPS die Intervallbreite des LPS zum Codewort addiert. Die Codierung erfolgt mit Festkomma Integer Arithmetik. Sobald der Integer-Wert der Intervallbreite A unter 0,75 fällt, wird A verdoppelt, so dass die Intervallbreite immer in den Bereich [0,75; 1,5[fällt und damit um den Wert 1 gehalten wird. Dieser Vorgang wird als Renormalisierung bezeichnet. Sobald A verdoppelt wird, wird auch die Basis C verdoppelt. Der in der Kontextmodellierung erzeugte Kontext bildet den Index für eine *Lookup*-Tabelle, die zum einen das MPS, d. h. das wahrscheinlichste nächste Symbol für diesen Kontext, liefert, und zum anderen indirekt die geschätzte Wahrscheinlichkeit angibt, mit der das MPS erwartet wird. Diese beiden Werte bilden den Index auf eine weitere *Lookup*-Tabelle, die innerhalb einer Wahrscheinlichkeitstabelle die Q_e-Werte liefert, wie sie vom Standard gegeben sind [Bol00].

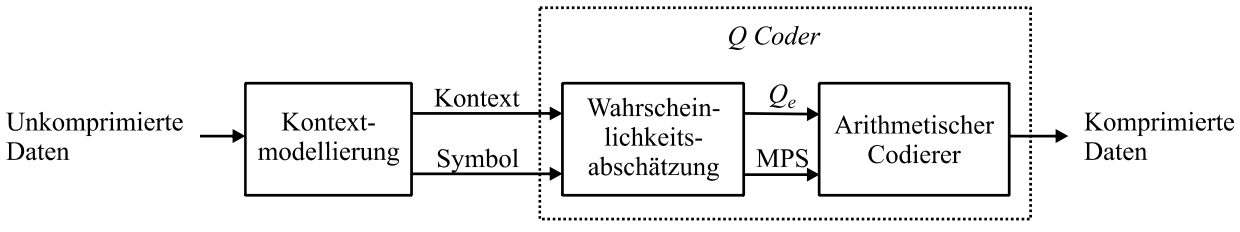

Bild 3.10: *Q Coder*

Die Intervallbreite und der Startpunkt werden folgendermaßen verändert:

$$A = A \cdot (1 - Q_e) \qquad C = C + A \cdot Q_e \qquad \text{(bei MPS Codierung)} \qquad (3.16) \qquad (3.17)$$

$$A = A \cdot Q_e \qquad C = C \qquad \text{(bei LPS Codierung).} \qquad (3.18) \qquad (3.19)$$

Da sich A im Bereich [0,75; 1,5[befindet, kann eine einfache Annäherung bei der Intervallaufteilung vorgenommen werden [Bol00]. Hierdurch ergibt sich

$$A = A - Q_e \qquad C = C + Q_e \qquad \text{(bei MPS Codierung)} \qquad (3.20) \qquad (3.21)$$

$$A = Q_e \qquad C = C \qquad \text{(bei LPS Codierung).} \qquad (3.22) \qquad (3.23)$$

Die vorgenommene Annäherung kann in manchen Fällen dazu führen, dass das LPS-Subintervall größer als das MPS-Subintervall ist. Als Beispiel zu nennen ist hier der Fall mit $Q_e = 0{,}5$ und $A = 0{,}75$. Bei der Annäherung werden $^1/_3$ des Intervalls dem MPS und $^2/_3$ dem LPS zugeordnet. Sobald dieser Fall auftritt, werden die MPS- und LPS-Subintervalle getauscht. Dieser Zustand kann jedoch nur auftreten, wenn eine Renormalisierung erforderlich ist.

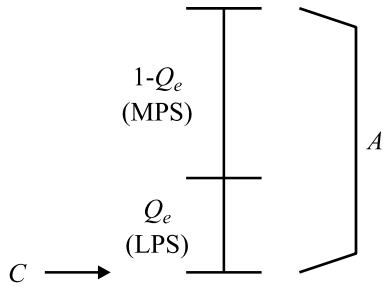

Bild 3.11: Aufbau des Intervalls

Der beschriebene Algorithmus kann folgendermaßen zusammengefasst werden:

1: Lese neues Symbol-Kontext-Paar
2: **if** MPS wird codiert **then**
3: $C = C + Q_e$
4: $A = A - Q_e$
5: **else**
6: $A = A \cdot Q_e$
7: **end if**
8: **if** ($A < 0{,}75$) **then**
9: Renormalisierung von A und C
10: Aktualisiere Q_e
11: **end if**

Der Aufbau eines *MQ Coder* kann als eine Zustandsmaschine angesehen werden, die eine Reihe von Symbolen $x \in \{0;1\}$ und die zugehörigen Kontexte $cx \in \{0;1 \ldots 18\}$ auf ein Codewort abbildet. Bild 3.12 zeigt den Aufbau des *MQ Coder*, dessen Funktionsweise sich durch die drei Hauptbestandteile beschreiben lässt:

1. **Interne Zustände:** Ein Satz von Registern (A, C, $\tilde{t}$, $\tilde{T}$, L). Das A Register beinhaltet eine normalisierte Version der Intervalllänge und das C Register die untere Intervallgrenze. $\tilde{t}$ ist ein Rückwärtszähler, der dem System am Punkt $\tilde{t} = 0$ signalisiert, dass teilweise codierte Bits aus dem C Register in den temporären Byte-Speicher $\tilde{T}$ übertragen werden sollen. L beinhaltet die Länge der bereits erzeugten Bytes.
2. **Kontextzustände:** Ein Kontextzustand (I_{cx}, S_{cx}) für jeden möglichen Kontext cx. Das einzelne Bit $S_{cx} \in \{0, 1\}$ beschreibt das MPS für den Kontext cx. $I_{cx} \in \{0 \ldots 46\}$ ist ein 6-Bit-Wert und beinhaltet den Index zu der Tabelle mit MPS-Wahrscheinlichkeiten in Abhängigkeit vom Kontext. Nach Codierung wird I_{cx} entsprechend einer *Lookup*-Tabelle aktualisiert. Die Anfangswerte von I_{cx} in Abhängigkeit vom Kontext sind fest definiert.
3. **Wahrscheinlichkeitsabschätzung:** Eine *Lookup*-Tabelle mit Wahrscheinlichkeitsregeln zum Interpretieren und Ändern des aktuellen Kontextzustands (I_{cx}, S_{cx}). Die Tabelle gibt mit $Q_e(I_{cx})$ die geschätzte LPS-Wahrscheinlichkeit für den aktuellen Kontext an. $NLPS(I_{cx})$ und $NMPS(I_{cx})$ beschreiben den neuen Wert für I_{cx} in Abhängigkeit davon, ob das codierte Symbol ein MPS oder LPS ist. Mit $SW(I_{cx})$ wird bestimmt, ob die Symbole für LPS und MPS getauscht werden sollen.

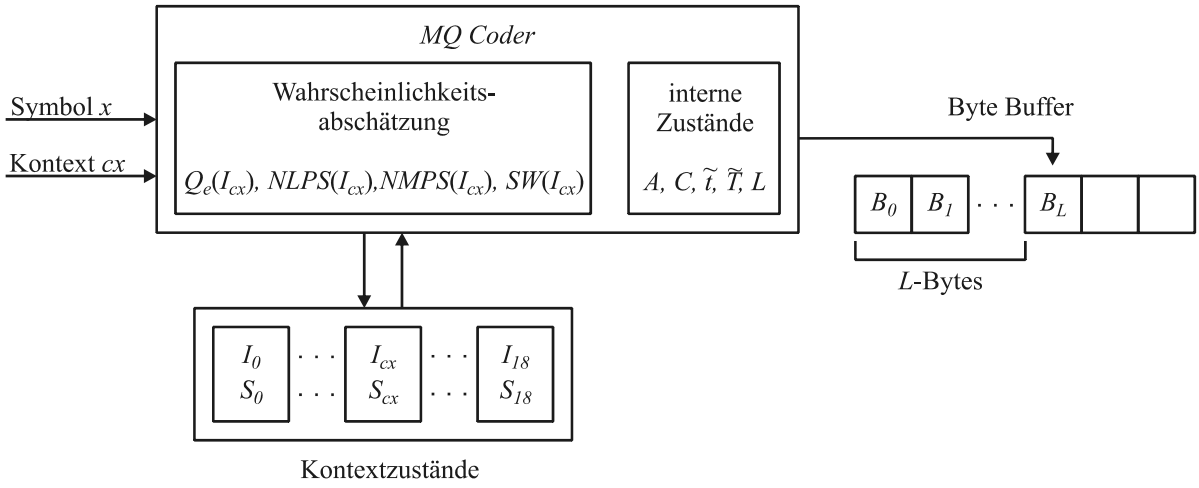

Bild 3.12: Aufbau des *MQ Coder*

Die Verwendung von *Lookup*-Tabellen und Zustandsmaschine erlaubt eine effiziente Implementierung des *MQ Coder* in Hardware. Der Speicher für die *Lookup*-Tabelle kann aufgrund der geringen Größe räumlich dicht an der Logik selbst angebracht werden. Sobald vom *Bitplane Coder* (BPC) ein Symbol-Kontext-Paar erzeugt wurde, kann dieses durch den *MQ Coder* weiterverarbeitet werden. In möglichen Hardwareimplementierungen können deshalb

BPC und *MQ Coder* weitgehend parallel arbeiten. Zur zeitlichen Entkopplung zwischen dem BPC Ausgang und dem *MQ Coder* Ausgang kann ggf. ein kleiner FIFO-Speicher als temporärer Zwischenspeicher für die Symbol-Kontext-Paare dienen. Ebenso wie beim BPC wird verglichen zur DWT eine hohe Anzahl an Takten zur Bearbeitung der Daten erwartet, so dass bei einer Hardwareimplementierung ein hoher Performanzgewinn gegenüber einer Softwarelösung erzielt werden kann.

3.5.3 *Tier-2*, *Rate Control*, Formatierung

Die Informationen der *Passes* aus der *Tier-1* Codierung werden mit Zusatzinformationen innerhalb von *Tier-2* in Paketen zusammengefasst und in einem vollständigen JPEG2000 *Codestream* ausgegeben. Die Pakete können nach verschiedenen Kriterien ausgewählt und sortiert im JPEG2000 *Codestream* derart angeordnet sein, dass Skalierbarkeit und progressives Decodieren eines unvollständigen *Codestream* möglich sind. Bei verlustbehafteter Kompression wird über ein Auswahlverfahren (*Rate Control*) bestimmt, welche *Passes* verwendet werden.

Die *Rate Control* entscheidet darüber, welche Daten bis zur Beendigung der *Tier-2* Codierung wegfallen, um eine Zielbitrate zu erreichen. Zwei Methoden werden hierbei vom JPEG2000-Standard unterstützt. Bei der ersten Methode werden die Quantisierungsintervalle verändert, um durch Codierung einer geringeren Anzahl von Symbolen bei erhöhter Intervallbreite den gewünschten Kompressionsfaktor zu erzielen. Das allerdings hat u. U. eine mehrfache Ausführung von *Tier-1* zur Folge und ist damit sehr rechenintensiv. Daher wird häufig die zweite Methode verwendet, die gezielt Informationen aus der *Tier-1* Codierung wegfallen lässt. Um zu bestimmen, welche Daten weggelassen werden dürfen, wird innerhalb der *Codeblöcke i* für jeden *Pass k* der Beitrag ΔD_i^k zur Reduzierung der Verzerrung (*Distortion*) in Abhängigkeit von der Anzahl der hierzu benötigten Bytes ΔR_i^k berechnet. Der erzeugte *Bitstream* kann somit an den Stellen, die durch das Ende von codierten *Passes* bestimmt sind, abgebrochen werden. Zur Berechnung der Verzerrung können unterschiedliche Metriken verwendet werden (z. B. MSE). Die Bestimmung der MSE erfolgt u. a. in Abhängigkeit vom verwendeten *Pass* und wird bereits während der Codierung im *MQ Coder* durchgeführt. Hierbei werden z. B. signifikant codierte Bits höher bewertet, als Bits, die im *Refinement Pass* codiert wurden. Der Beitrag ΔD_i^k ist zudem von der aktuellen Bitebene und dem *Subband* abhängig.

Bild 3.13 zeigt die Verteilung der einzelnen Beiträge eines *Codeblocks* im *Rate Distortion*-Diagramm. Gezeigt ist die Gesamtverzerrung des *Codeblocks*, die mit jedem *Pass* der komprimierten Länge ΔR_i^k um den Wert ΔD_i^k reduziert wird. Die Steigung (*Slope*) S_i^k zwischen zwei benachbarten Abbruchstellen *k* kann als Wichtigkeit des *Pass* interpretiert werden. Ein *Pass* mit größerer (negativer) Steigung bedeutet, dass dieser die Verzerrung mit weniger Bytes reduzieren kann. Aufgrund der Codierungsreihenfolge werden die erwarteten wichtigsten Werte zuerst, die weniger wichtigen später codiert, so dass sich i. Allg. ein monoton abfallender Kurvenverlauf ergibt. Ergibt sich durch die Daten eines *Pass* kein monoton abfallender Kurvenverlauf, dann sind diese Daten einzusortieren, damit Steigungen mit $S_i^k > S_i^{k-1}$ für $k\text{-}1 > 0$

unterbunden werden. Nach Beendigung eines *Tile* ist es das Ziel, für alle *Codeblöcke* mit ihrer zugehörigen *Rate Distortion*-Kurve die richtige Abbruchstelle zu finden, um für das *Tile* einen optimalen *Bitstream* zu erzeugen. Es werden die Steigungen aller *Codeblöcke* eines *Tile* betrachtet.

Unter der Voraussetzung eines additiven Verhaltens der einzelnen Verzerrungen ergibt sich die Gesamtverzerrung zu

$$D = \sum_{i,k} \Delta D_i^{\,k} \,. \tag{3.24}$$

Sie ist unter der Nebenbedingung

$$R = \sum_{i,k} \Delta R_i^{\,k} \le R_{MAX} \tag{3.25}$$

mit der insgesamt zulässigen Rate R_{MAX} zu minimieren. D. h. mit Hilfe dieser *Rate-Distortion*-Berechnung wird der *Codestream* solange mit Paketen aufgefüllt, bis die maximal erlaubte Datenlänge R_{MAX} nach Gleichung (3.25) und damit der eingestellte Kompressionsfaktor erreicht ist.

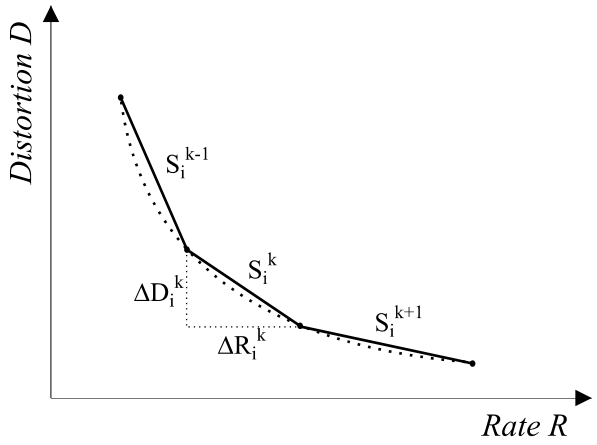

Bild 3.13: *Rate Distortion*-Kurve eines *Codeblocks i*

Der ausschließlich während *Tier-2* arbeitende *Post-Compression Rate-Distortion* (PCRD)-Algorithmus beginnt mit der Auswahl der *Passes,* wenn alle bestimmenden Informationen einer abgeschlossenen Menge (z. B. die *Passes* eines *Tile*) vorhanden sind [Bol00]. Damit hat er Zugriff auf alle Daten dieser Menge und kann sie derart reduzieren, dass die bestmögliche Qualität der Bilddaten bei vorgegebenem Kompressionsfaktor erreicht wird. Dieses führt jedoch dazu, dass bei höheren Kompressionsfaktoren die meisten der von *Tier-1* zuvor aufwendig erzeugten *Pass*-Informationen wegfallen. Die Zeit zur Erzeugung der *Pass*-Informationen ist damit proportional zur Menge der Ursprungsdaten.

Alternativ kann die *Rate Control* schon während der *Tier-1* Codierung eingreifen. Bei Erreichen des geforderten Kompressionsfaktors wird eine weitere Codierung der Restdaten unterbunden und dementsprechend ein höherer Durchsatz erzielt. Als Beispiel für eine Implementierung ist die Vorgabe der geschätzten Ziellänge für jeden *Codeblock* zu nennen. Die zur Erzeugung der *Tier-1*-Daten notwendige Zeit ist vom Kompressionsfaktor abhängig. Da hierbei jedoch die Entscheidung darüber, was wegfällt, auf einer noch nicht vollständig abgeschlossenen Menge beruht, kann nicht, wie bei PCRD, die bestmögliche Qualität der Daten erreicht werden. Mit der vorzeitigen Beendigung der Codierung können darauf folgende *Passes* mit einer überdurchschnittlich hohen Steigung nicht mehr in die Bearbeitung mit einbezogen werden.

Ebenso werden *Rate Control*-Verfahren eingesetzt, die eine Kombination der vorher genannten Methoden beinhalten [Cha03]. Während *Tier-1* werden dabei nicht alle *Passes*, jedoch mehr *Passes* als für den komprimierten *Codestream* erforderlich sind, codiert. Anschließend erfolgt eine weitere Auswahl auf diese beschnittene Menge während *Tier-2*. Hierdurch lassen sich gute Ergebnisse mit ratenabhängiger Kompressionszeit kombinieren.

Nach Beendigung der *Tier-2* Codierung werden die Daten in eines von zwei vom JPEG2000-Standard vorgesehenen Dateiformaten gebracht. Dieses ist zum einen das JPEG2000 *Codestream*-Dateiformat mit minimaler Funktionalität, zum anderen das JP2-Dateiformat, das eine hohe Zahl von Zusatzinformationen zulässt und im Kern das JPEG2000 *Codestream*-Dateiformat beinhaltet. Diese Formate bestehen aus einzelnen Segmenten, die durch Segment-Marker gekennzeichnet sind und Informationen bzgl. des codierten Bildes sowie die komprimierten Daten selbst enthalten. Hierdurch erfolgt u. a. die eindeutige Zuordnung der codierten Daten zum jeweiligen Teil des Originalbildes (z. B. Ursprungs-*Tile*), damit eine korrekte Decodierung der Daten ermöglicht wird.

Die vollständige Implementierung von *Tier-2, Rate Control* und Formatierung in Hardware ist weniger geeignet, da ein Zugriff auf eine große Menge bereitzustellender Daten (alle *Passes*) vorhanden sein muss. Während *Tier-2* und der Formatierung wird die meiste Zeit für den Datentransfer benötigt. Bei der *Rate Control* sind in Abhängigkeit vom implementierten Algorithmus verschieden aufwendige Berechnungen erforderlich, die sich auf die Informationen größerer Einheiten (*Passes*) beziehen. Sinnvoll kann es sein, diejenigen Teile der *Rate Control*, die schon zeitnah während der Codierung ermittelt werden können und für die nur eine beschränkte Datenmenge genügt, in Hardware zu implementieren (z. B. Teile der Steigungs-Berechnung), und den Rest in Software durchzuführen. Verglichen zum BPC und *MQ Coder* ist mit einer weitaus geringeren Bearbeitungszeit zu rechnen.

3.6 *Region of Interest*-Codierung

Innerhalb des JPEG2000-Standards ist die Funktionalität der *Region of Interest* (ROI)-Codierung von hoher Bedeutung. Sie bietet die Möglichkeit, interessante Regionen eines Bildes gegenüber

weniger interessanten Regionen mit höherer Qualität zu komprimieren (siehe Bild 3.14). Hierdurch kann die zur Verfügung stehende Übertragungsrate gezielt zugeordnet werden.

Bild 3.14: Beispielbild mit ROI-Codierung, Gesamtkompressionsfaktor 20

Durch Skalierung werden bei der ROI-Codierung die zu einem ROI gehörenden *Wavelet*-Koeffizienten zu höheren Bitebenen verschoben. Die ROI-Koeffizienten heben sich damit von den Koeffizienten außerhalb des ROI, den Hintergrundkoeffizienten, ab und werden bevorzugt bearbeitet. Hierzu ist es erforderlich, dass eine im Zeitbereich vorgegebene ROI-Maskierung auf den Frequenzbereich umgerechnet wird (siehe Bild 3.15). Die Abhängigkeit von Pixel im Zeitbereich zu den Koeffizienten im Frequenzbereich ist aus der Transformation bekannt (siehe Abschnitt 3.4).

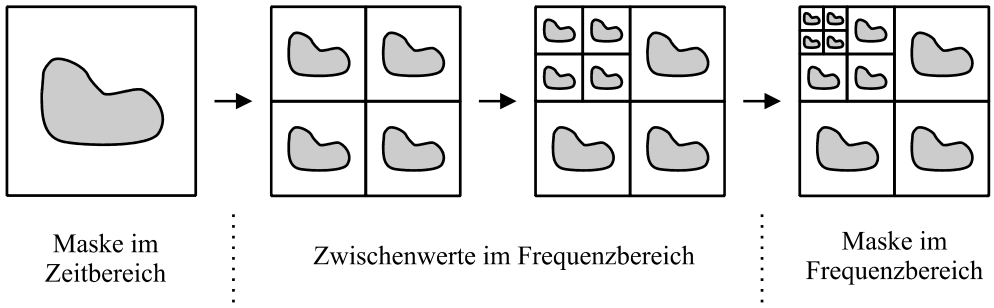

Bild 3.15: Zuordnung der ROI-Pixel im Zeit- und Frequenzbereich bei dreistufiger Transformation

Zur ROI-Codierung können unterschiedliche Methoden angewendet werden. Bei der *General Scaling Based*-Methode werden nach der Transformation die ROI-Koeffizienten gegenüber den Hintergrundkoeffizienten um einen Skalierungswert s angehoben (siehe Bild 3.16). Sie befinden sich damit in höheren Bitebenen und werden beim anschließenden Codierungsprozess mit höherer Priorität zuerst behandelt. Bei verlustbehafteter Codierung werden so die ROI-

Koeffizienten mit einer höheren Qualität übertragen wie der Hintergrund. Im Decoder werden die ROI-Koeffizienten vor der inversen Transformation wieder herunter skaliert. Dazu erfordert die *General Scaling Based*-Methode die Unterbringung des Skalierungswertes *s* und der ROI-Maskierung im *Codestream*. Neben diesem Offset im Datenstrom erhöht die Unterbringung der ROI-Maskierung die Komplexität des Coders. ROI-Bereiche wie Rechtecke oder Ellipsen können zwar schon mit wenigen Bits beschrieben werden, allerdings erfordern ROI, die auf reale Objekte bezogen sind, i. Allg. frei wählbare Bereichsmuster.

Eine andere Methode zur ROI-Codierung ist die *Maxshift*-Methode, die eine Erweiterung der *General Scaling Based*-Methode ist [Bol00]. Der Skalierungswert *s* wird in Abhängigkeit von der höchsten Bitebene des Hintergrunds derart gewählt, dass nach der Skalierung alle Bits der ROI in höheren Bitebenen liegen als alle Bits außerhalb der ROI (siehe Bild 3.16). Anhand der Zugehörigkeit zu einer Bitebene kann der Decoder die ROI-Koeffizienten identifizieren und eine Übertragung der ROI-Maskierung ist nicht erforderlich. Die *Maxshift*-Methode hält die Bitebenen für ROI und Hintergrund vollständig getrennt. Hierdurch können unterschiedliche Kompressionsfaktoren für ROI und Hintergrund gewählt werden. Bei der *General Scaling Based*-Methode besteht diese Möglichkeit nicht.

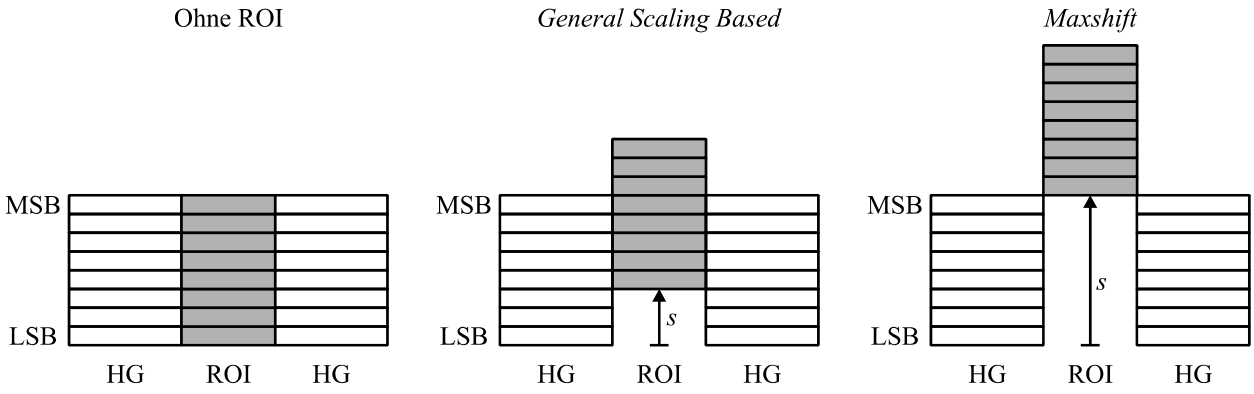

Bild 3.16: *General Scaling Based*- und *Maxshift*-Methode zur ROI-Codierung [Chr00], HG: Hintergrund

Eine mögliche Hardwareimplementierung der ROI-Codierung scheint unkompliziert zu sein. Es muss hierzu in der Hardware die zu einem *Tile* gehörige ROI-Maske im Frequenzbereich (siehe Bild 3.15) gespeichert sein, mit der anschließend bestimmt wird, welche Koeffizienten nach der Transformation und vor der Codierung angehoben werden. Das Anheben selbst kann durch einfache Schiebe-Operationen erfolgen, der zeitliche Aufwand hierzu ist sehr gering. Aufgrund einer höheren Anzahl von Bitebenen ist mit einer erhöhten Codierzeit zu rechnen.

3.7 Software Referenz-Implementierung JasPer

Die unterschiedlichen Softwareimplementierungen des JPEG2000-Algorithmus unterscheiden sich im Wesentlichen in der Performanz sowie im Umfang der möglichen Zusatzfunktionen. In manchen Implementierungen kann zudem die *Rate Control* schon während *Tier-1* eingreifen. Die JasPer-Implementierung [Ada05] ist eine freie Software, die eine offizielle Referenz-Implementierung des im ersten Teil des JPEG2000-Standards beschriebenen Kompressions-algorithmus enthält. Sie ist in der Programmiersprache C geschrieben und steht als Quellcode unter der JasPer-Lizenz öffentlich zur Verfügung. Bei der Ausführungszeit sowohl für verlustlose als auch für verlustbehaftete JPEG2000-Codierung ist kennzeichnend, dass aufgrund der mehrfachen Durchgänge und der bitseriellen Operationen der größte Teil der Rechenleistung von *Tier-1* benötigt wird, gefolgt von der *Wavelet*-Transformation.

Dieses lässt darauf schließen, dass der JPEG2000-Algorithmus aufgrund der mehrfachen Durchgänge und der vielen bitseriellen Operationen innerhalb einer Hardwareimplementierung zu einem starken Gewinn an Performanz führen kann.

4 Architektur einer DPU für kompakte Kamerainstrumente

In der Entwicklung kompakter Kamerainstrumente werden zunehmend strahlungstolerante Bausteine eingesetzt. Sie werden mit Fehlerschutzmechanismen für den Einsatz in der Raumfahrt versehen. Nachdem in diesem Kapitel ein typischer Aufbau eines Instrumentenrechners für kompakte Kameras mit den zugrunde liegenden Bauteilen analysiert wird, werden die notwendigen Fehlerschutzmechanismen mit Schwerpunkt auf strahlungstolerante SRAM-basierte FPGA dargestellt.

In Kapitel 2 wurden die Hintergründe der wesentlichen Fehlermechanismen für Elektronik in der Raumfahrt besprochen. Es hat sich gezeigt, dass insbesondere die *Single Event Upsets* (SEU) aufgrund ihrer unterschiedlichen Auswirkungen schwer zu beherrschen sind. Während *Single Event Latchup* (SEL) und *Total Dose* in strahlungstoleranten Bausteinvarianten unkritisch sind, können SEUs in diesen Varianten auftreten und müssen besonders berücksichtigt werden. Bei strukturellen Maßnahmen zur Verminderung der Fehlerrate kann eine einhundertprozentige Eliminierung der Fehlerwirkung nicht garantiert werden. Vielmehr ist das Ziel solcher Maßnahmen, die Fehlerwirkung drastisch zu reduzieren, um mit Toleranz gegenüber strahlungsbedingter Fehler insgesamt ein sicheres System aufzubauen. Nach Vorstellung des Aufbaus einer Muster-DPU für ein Kamerainstrument werden in diesem Kapitel verschiedene strukturelle Maßnahmen zur Minderung der Fehlerwirkung besprochen.

Beim Entwurf von Elektronik für Raumsonden sind verschiedene Randbedingungen zu beachten. Diese sind abhängig von der Ebene, in der sich die Elektronik innerhalb des Elektroniksystems der Raumsonde befindet. Instrumente, die als nicht missionskritisch eingestuft sind, dürfen häufig nur eine geringe Masse und einen niedrigen Leistungsverbrauch aufweisen. Es ist aus wissenschaftlicher Sicht jedoch eine hohe Performanz des Instruments und damit der DPU erwünscht. Dieses führt dazu, dass z. B. unter Nutzung der Redundanz auf niedriger Ebene (Bauteilebene und niedriger) versucht wird, mit Schutzmechanismen effektiv Fehler zu umgehen. Der Entwurf impliziert somit auch das Ziel, mit adäquaten Mitteln ein für die jeweilige Mission bzw. Anwendung genügend zuverlässiges und verfügbares System bei geringer Komplexität zu entwerfen.

4.1 Hardwareaufbau der Musterkonfiguration

Bild 4.1 zeigt den Aufbau einer Muster-DPU, wie sie in vielen Fällen Verwendung findet (z. B. Venus Express / VMC; DAWN / DFC). Die Komponenten werden auf aktuell verfügbare, für den Weltraum einsetzbare Bauteile abgebildet:

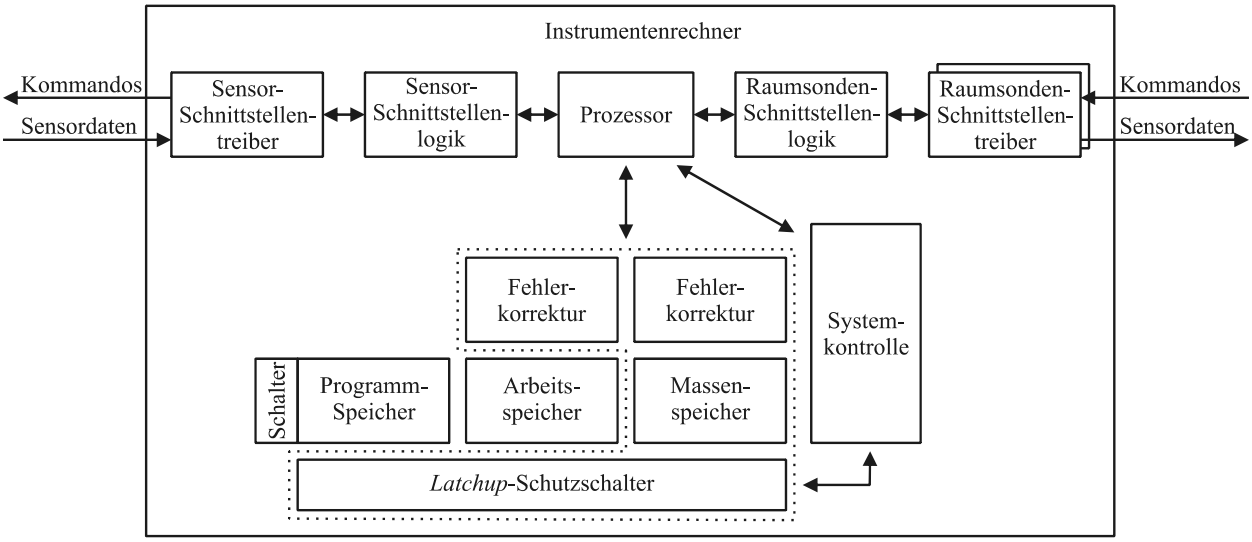

Bild 4.1: Aufbau der Muster-DPU;
gestrichelt eingerahmt: Komponenten für einen strahlungstoleranten Entwurf

- **Prozessor**: Die Auswahl an strahlungsfesten Prozessoren ist sehr stark eingeschränkt. Derzeit wird von Seiten der ESA der 32-Bit-Prozessor AT697E propagiert, der als strahlungsfester Baustein von ATMEL gefertigt wird [Atm06]. Die ersten Prototypen sind bereits vorhanden. Der Prozessor beinhaltet den fehlertoleranten LEON2-FT Kern der ESA, ist in 0,18 µm Technologie gefertigt und liefert 86 MIPS Dhrystone 2.1 bei 100 MHz. Er beinhaltet eine SPARC-V8 Integer Einheit, Fließkommaprozessor (FPU), separaten Befehls- und Datencache (32 kByte bzw. 16 kByte) und eine Speichersteuerung mit SDRAM-Funktionalität. Der LEON2-Prozessor verwendet den offenen *Advanced Microcontroller Bus Architecture* (AMBA)-Standard von ARM in Form des *Advanced High Performance Bus* (AHB) und des *Advanced Peripheral Bus* (APB). Die Strahlungsfestigkeit innerhalb des ASIC wird u. a. durch Fehlerkorrektur für die Registerbänke, Fehlererkennung beim Cache-Speicher sowie TMR-Design für Flipflops und bei der Taktverteilung erzielt. Das Design ist voll statisch und kann mit einem der Anwendung entsprechend reguliertem Takt versehen werden. Ein besonderes Merkmal dieses strahlungsfesten Bauteils ist es, dass parallel dazu der LEON2-Prozessor als *Intellectual Property* (IP) Core frei verfügbar ist und sich somit für Vergleichszwecke sehr gut eignet [Gai05]. Dieses erlaubt es, den gleichen Prozessortyp entweder als diskreten strahlungsfesten Baustein oder aber als IP Core in einem FPGA zu verwenden, um

dadurch verschiedene Ansätze miteinander vergleichen und mit derselben Entwicklungsumgebung entwickeln zu können. Der IP Core beinhaltet standardmäßig keine FPU. Diese bräuchte viele Ressourcen und müsste ggf. zusätzlich eingebunden werden. Der zunehmende Einsatz auf europäischen Raumsonden und die Verfügbarkeit des LEON2-Prozessor sowohl als strahlungsfester Baustein als auch als IP Core führen u. a. zu der Entscheidung, ihn in dieser Arbeit zu verwenden.

- **Speicher:** Beim klassischen Aufbau einer DPU werden strahlungsfeste Speicherbausteine verwendet, die verglichen zu ihren kommerziellen Gegenstücken, neben einer geringen Verfügbarkeit und hohen Kosten, eine höhere Masse und einen erhöhten Leistungsverbrauch aufweisen. Durch Verwendung einer Fehlerkorrektur kann häufig auf strahlungsfeste Speicherbauteile verzichtet werden. Für den schnellen, wahlfreien Zugriff auf den **Arbeitsspeicher** werden vielfach statische asynchrone Speicher (SRAM) verwendet, z. B. in Form von 8-Bit breiten 4 MBit-Bausteinen. Bei strahlungstoleranter Verwendung sind für eine 32-Bit-Anbindung an den Prozessor und eine *Reed Solomon*-Fehlerkorrektur (siehe Abschnitt 4.3) insgesamt sechs parallele Bausteine notwendig. Zur Implementierung eines **Massenspeichers** bieten sich dynamische Speicher an, die über hohe Speicherdichten verfügen (z. B. SDRAM oder DDR SDRAM). Diese dynamischen Speicher sind nicht in strahlungsfester Ausführung erhältlich. Hier stehen 8-Bit breite kommerzielle Bausteine mit Kapazitäten von 256 MBit bis 512 MBit zur Verfügung. In Abhängigkeit von der notwendigen Speichergröße lassen sich zudem Speicher-Stacks verwenden, die aus übereinander angebrachten Einzelbauteilen bestehen, deren Anschlüsse bis auf den Anschluss zur Auswahl des aktiven Einzelchips parallel geschaltet sind. Mit einer erhöhten Bauweise lässt sich eine höhere Speicherdichte erzielen, ohne die Platine selbst zu vergrößern. SDRAM- und auch DDR SDRAM-Speicher benötigen eine zusätzliche Steuerungseinheit, ihre direkte Ansteuerung wird nicht, wie beispielsweise bei SRAM, von jedem Prozessor direkt unterstützt. Der AT697E (LEON2-FT) Prozessor beinhaltet bereits eine Steuerungseinheit mit Unterstützung für SDRAM, so dass in der Muster-DPU SDRAM Verwendung findet. Beim **Programmspeicher** und Parameterspeicher wird traditionell auf strahlungsfeste PROMs und strahlungsfeste EEPROMs zurückgegriffen. Das PROM beinhaltet lediglich einen kleinen *Bootloader*, der bei Verwendung SRAM-basierter FPGAs zusammen mit der FPGA-Konfiguration abgelegt werden kann. Im EEPROM sind das Hauptprogramm, Skripte, Kalibrationswerte und Konfigurationen gespeichert. Durch die geringe Speicherdichte dieser nichtflüchtigen Speicher (NVRAM) lassen sich bei den sehr beschränkten Ressourcen eines Instrumentenrechners nur relativ wenige Daten ablegen (z. B. 1 MBit pro EEPROM-Baustein). Sollen mehrere Versionen des Hauptprogramms (Größe z. B. 450 kByte) abgelegt werden, um eine erhöhte Sicherheit bei der Implementierung neuer Versionen zu erzielen, so ist dieses in vertretbarem Maße bei kompakten Kameras kaum ausschließlich mit EEPROM-Bausteinen möglich. Für eine strahlungstolerante Implementierung kann daher die Kombination aus EEPROM- und NAND Flash-Bausteinen eingesetzt werden, wie sie in der Kamera Venus Express / VMC implementiert ist. Die Speichergröße dieser

Flash-Bausteine kann z. B. 128 MBit oder größer betragen. Für eine sichere Funktionsweise ist es notwendig, auf Ergebnisse von Strahlungstests des verwendeten Flash-Bauteils zugreifen zu können. Ist das Verhalten eines Flash-Bausteins unter Strahlungseinfluss bekannt, so kann durch entsprechend intensive Ausnutzung von Redundanz innerhalb eines Bausteins, aber auch vollständige Bauteilredundanz sowie Abschalten bei Nichtbenutzung gezielt ein gesicherter Einsatz im Weltraum erfolgen. Hierzu werden nichtflüchtige Parameter zur Nutzung des Flash-Bausteins, z. B. dessen *Bad Block*-Tabelle, in einem strahlungsfesten EEPROM gespeichert, die Daten selbst im Flash-Baustein. Die Fehlerbehandlung erfolgt in Software, da die Zugriffsgeschwindigkeit auf das NVRAM im Gegensatz zum Arbeits- oder Bildspeicher eine untergeordnete Rolle spielt. Dieses erlaubt die gezielte Anwendung von unterschiedlichen, auch komplexeren Korrekturmaßnahmen, die bei einer Hardwareimplementierung unverhältnismäßig aufwendig wäre. Zur Reduzierung des Energiebedarfs sowie strahlungsbedingter Fehler ist der nichtflüchtige Speicher mit einem Schalter versehen, der die Bausteine bei Nichtbenutzung von der Versorgungsspannung abtrennt. Bei einer typischen aktiven Zeit von insgesamt 1 min pro Tag entsprechend einem aktiven Zeitanteil von $< 10^{-3}$ tragen diese Bereiche praktisch nicht zum durchschnittlichen Leistungsverbrauch des Instrumentenrechners bei.

- **Schnittstellen:** Die **Sensor-Schnittstelle** ist 8-Bit parallel ausgeführt und überträgt 1k × 1k × 16 Bit Daten (14 Bit Bilddaten + 2 Bit Synchronisierungsflags) entsprechend einer Übertragungsgeschwindigkeit von 32 MBit/s in 0,5 s (in Anlehnung an Venus Express / VMC). Eine zusätzliche asynchrone serielle Verbindung wird zur Steuerung des Sensors durch die DPU benötigt, eine weitere zur Übertragung von Informationsdaten vom Sensor an die DPU. Während bei kurzen Verbindungsleitungen zwischen der DPU und der Sensor-Elektronik die einfache Übertragung auf TTL-Pegel ausreichend ist, sollte bei längeren Verbindungsleitungen auf eine differentielle Übertragung, z. B. mittels *Low Voltage Differential Signal* (LVDS), übergegangen werden. Die **Raumsonden-Schnittstellen** sind je nach Mission vorgegeben. Bei Missionen der ESA werden vielfach die *Remote Terminal Unit* (RTU)-Schnittstelle sowie die *SpaceWire*-Schnittstelle eingesetzt [Mat00, Sib02, ECS02]. Bei der RTU-Schnittstelle handelt es sich um eine serielle, in 16-Bit-Wörtern organisierte Schnittstelle mit niedrigen Übertragungsraten (< 32 kBit/s). Sie wird bidirektional sowohl zur Kommandierung (*Telecommand*, TC) als auch zur Übertragung von Messdaten (*Telemetrie*, TM) eingesetzt. Die Übertragung erfolgt differentiell auf RS-422-Pegel und wird von der Raumsonde initiiert. Die von Kamerainstrumenten erzeugten Datenmengen sind i. Allg. jedoch sehr hoch, so dass diese Schnittstelle nicht ausreicht, die Datenmenge in adäquater Zeit zu übertragen. Sofern die Raumsonde es unterstützt, erhalten Instrumente mit einer hohen Datenmenge zusätzlich eine Schnittstelle auf Basis des *SpaceWire*-Standards. Diese serielle Schnittstelle erlaubt Transferraten von bis zu 400 MBit/s und arbeitet mit differentieller Übertragung auf LVDS-Pegel. Die Transferraten können der Anwendung entsprechend angepasst werden (z. B. 10 MBit/s). Logisch erfolgt die Übermittlung der Daten durch den Austausch von

Token und es besteht eine ständige bidirektionale Verbindung. Der Datenfluss erfolgt i. Allg. vom Instrument zur Hauptelektronik der Raumsonde (z. B. deren Massenspeicher) und kann, sofern die Anwendung dieses erfordert, ebenso in entgegen gesetzter Richtung von der Raumsonde zum Instrument erfolgen. Letzteres kann z. B. dazu verwendet werden, um eine Aktualisierung der Software des Instruments durchzuführen oder dem Instrument größere Mengen an Kalibrationsdaten zur Verfügung zu stellen. Aufgrund der Anforderung an ein Instrument müssen die Raumsonden-Schnittstellen für gewöhnlich doppelt ausgeführt werden, d. h. als nominelle und redundante Schnittstellen. Bei Missionen der NASA wird häufig die MIL-STD-1553 Schnittstelle mit einer maximalen Übertragungsrate von 1 MBit/s verwendet. Diese Schnittstelle hat jedoch einen erheblich höheren Verbrauch der Ressourcen Masse und Leistungsverbrauch als die RTU- und die *SpaceWire*-Schnittstelle. Sie wird bei Missionen der ESA seltener verwendet und in dieser Arbeit nicht weiter betrachtet. Der Aufbau der Schnittstellen lässt sich aufteilen in die diskreten Anteile Steckverbindung und Konvertierung der Signalpegel über Treiberbausteine sowie einen Logikteil, der die Anpassung zur Anbindung an das Prozessorsystem übernimmt. Eine Schnittstellenlogik beinhaltet häufig FIFO-Speicher zur zeitlichen Entkopplung von Schnittstelle und Prozessorsystem und wird auf unterschiedliche Weise implementiert. Ist eine doppelte Ausführung der Schnittstelle, wie z. B. bei der Raumsonden-Schnittstelle, erforderlich, so bezieht sich dieses nicht zwangsweise auch auf eine doppelt vorhandene Logik. Die redundante Ausführung von Stecker und Signalpegelanpassung, die mit einer hohen Anzahl von Bauteilen eine insgesamt reduzierte Zuverlässigkeit aufweisen, ist genügend.

- **Systemkontrolle:** Allgemein erforderliche Komponenten wie die Taktgenerierung sowie übergeordnete Funktionen zur Herstellung und Wiederherstellung des Systems sind notwendig. Zur Systemüberwachung gehört neben dem *Watchdog*-Zähler und *Latchup*-Schutzschalter das gezielte Starten des Systems. Bei Verwendung SRAM-basierter FPGAs muss der Baustein mit der entsprechenden Konfiguration geladen und im Fehlerfall neu konfiguriert werden. Es ist sicherzustellen, dass die Systemüberwachung eine sehr geringe Strahlungsempfindlichkeit aufweist.

4.2 Software

Die Software einer DPU ist in verschiedene Ebenen eingeteilt und muss folgendes ermöglichen:

- Sicheres Starten des Instruments inkl. Wiederherstellung nach einem unerwarteten Systemzusammenbruch
- Ausführung der Grundfunktionen sowohl autonom als auch auf Initiierung durch ein Kommando von der Raumsonde
- Akquisition der Bilddaten
- Ausführung von Bibliotheksfunktionen (z. B. Kompression)

- Ausführung von Skripten auf virtuellen Maschinen und damit Nutzung aller implementierten Funktionen und Bibliotheksfunktionen
- Möglichkeit zur Aktualisierung des Hauptprogramms oder einzelner Programmteile.

Ein kleines Programm, lokalisiert im PROM, beinhaltet den *Bootloader* und wird nach einem System Reset aufgerufen. Seine Aufgabe ist es, das Hauptprogramm unter Beachtung Fehler korrigierender Maßnahmen aus dem NVRAM in den Arbeitsspeicher zu laden und dort auszuführen. Sowohl der *Bootloader* als auch das Hauptprogramm müssen die Möglichkeit vorsehen, neue Softwareversionen auf das Instrument zu laden. Einerseits ist es für die geplanten Aktualisierungen der Software notwendig, andererseits für den Fall, dass das Hauptprogramm nicht fehlerfrei aus dem NVRAM geladen werden konnte. In diesem Fall ist garantiert, dass die DPU zumindest mit dem *Bootloader*, der in einem sicheren PROM gespeichert ist, arbeiten kann, um weitere Fehlerdiagnose und Fehlerbehebung betreiben zu können. Während der *Bootloader* als kleines Programm mit minimaler Funktion anzusehen ist, werden an das Hauptprogramm Anforderungen bzgl. Multitaskingfähigkeit gestellt. Daher basiert das Hauptprogramm auf einem Echtzeit-Betriebsystem (*Real-Time Operating System,* RTOS), welches Multitaskingfunktionen zur Verfügung stellt, um umfangreiche Software strukturiert und in unterschiedlichen, parallel arbeitenden Tasks ablaufen zu lassen. Für den SPARC-V8-kompatiblen LEON2-Prozessor sind unterschiedliche RTOS verfügbar (z. B. eCos oder RTEMS, beide als *Open Source* [eCo06][Oar06]). Das Hauptprogramm beinhaltet verschiedene Bibliotheken mit Funktionen z. B. zur Bildbearbeitung oder zum Zugriff auf den Massenspeicher. Zur Steuerung des Instruments im operationellen Betrieb werden Softwaresysteme mit virtuellen Maschinen eingesetzt (z. B. *On-board Command Language,* OCL [Wit03]). Solche Systeme ermöglichen die überwachte Ausführung unterschiedlicher, parallel ablaufender Skripte. Sie greifen im Wesentlichen auf die bereitgestellten Bibliotheksfunktionen zu. Anpassungen an den operationellen Betrieb erfolgen somit durch Anpassungen an den Skripten ohne Änderungen am Hauptprogramm. Diese Entkopplung ist wichtig, da die Skripte und das Hauptprogramm i. Allg. in unterschiedlichen Arbeitsgruppen erstellt werden. Neue Skripte können unabhängig vom Hauptprogramm auf das Instrument geladen und abgesichert, d. h. durch Ausführung auf einem virtuellen Prozessor mit Überprüfungen und ohne direkten Zugriff auf die Hardware, ausgeführt werden. Die Verwendung solcher Softwaresysteme bietet dem Anwender ein hohes Maß an Flexibilität.

4.3 Schutz der Speicherelemente

Unter Ausnutzung von Redundanz werden in strahlungstoleranten Speicherbausteinen Daten zusammen mit einem Fehler korrigierenden Code abgespeichert. Tritt ein Fehlerfall, z. B. durch einen SEU, in entweder den Daten oder dem gespeicherten Code auf, dann können die Daten bis zu einem gewissen Fehlergrad vollständig restauriert werden. Die redundante Information wird also genutzt, um die ursprünglichen Daten zurück zu gewinnen. Verwendete Codes sind z. B. der *Reed Solomon* Blockcode oder der *Hamming* Code. Der *Reed Solomon* Blockcode ist Symbol-

orientiert und sieht vor, ein Symbol vollständig wiederherstellen zu können. Dieses ist vor allem interessant, da Fehler häufig als *Burst*-Fehler mit mehr als einem Bitfehler auftreten. Bei einer gewählten Symbolbreite von 8 Bit wird bei einer Datenbreite von 32 Bit (entsprechend 4 Symbole) eine Codebreite von 16 Bit (2 Symbole) benötigt, d. h das nutzbare Datenvolumen sinkt auf $^{32}/_{48} = {}^{2}/_{3}$ ab. Werden 8-Bit breite Bausteine verwendet, so lassen sich alle Symbole innerhalb eines Datenwortes in unterschiedlichen Bausteinen ablegen, und sogar der vollständige Ausfall eines Bausteins (entsprechend einem Symbol) wird abgefangen. Dieses hat einen großen Einfluss auf die Zuverlässigkeit einer Speichergruppe. Da Fehler in mehr als einem Symbol i. Allg. nicht durch diesen Code korrigiert werden können, wird durch *Memory Scrubbing* versucht, diesen Zustand nicht auftreten zu lassen. In der Kombination *Memory Scrubbing* und Fehlerkorrektur werden periodisch alle Speicherzellen unter Verwendung der Fehlerkorrektur gelesen und sofort wieder zurück geschrieben. Es ist das Ziel, in einem Datenwort einen Ein-Symbol-Fehler zu korrigieren, bevor sich dieser mit einem neuen Ereignis zu einem nicht korrigierbaren Zwei-Symbol-Fehler akkumuliert. Die Häufigkeit des Auftretens von Ein-Symbol-Fehlern zusammen mit der Wahrscheinlichkeit des Auftretens eines Fehlers in einem schon beschädigten Datenwort führen zur Bestimmung der maximal erlaubten Zeit zwischen zwei vollständig abgeschlossenen Schrubbintervallen [Ger01]. Im Gegensatz zum *Reed Solomon* Blockcode arbeitet der *Hamming* Code nicht auf Basis von Symbolen. Er kann lediglich ein fehlerhaftes Bit korrigieren und ist daher für Einzelbitfehler, nicht für *Burst*-Fehler geeignet [Mic05].

Da diese Fehlerbehebung auf Redundanz beruht, ist mit einer höheren Masse und einem erhöhten Leistungsverbrauch zu kalkulieren. Für schnellen Zugriff wird in jedem Fall eine Fehlerkorrektur in Hardware benötigt. Die Einführung einer Fehlerkorrektur bringt Verzögerungen auf dem Datenpfad mit sich und setzt deshalb bei schnellem Arbeits- oder Bildspeicher die Leistungsfähigkeit herab. Dieses tritt verstärkt auf, wenn die Fehlerkorrektur abhängig von der Implementierung der DPU in einem zusätzlichen Baustein durchgeführt wird und daher die nicht zu vernachlässigenden Laufzeiten der Ein- und Ausgangspuffer noch hinzukommen. Es werden in der Fehlerkorrektur Strukturen eingesetzt, die im Fall eines korrekten, nicht zu korrigierenden Datenwortes, nur minimale Verzögerungen einfügen. In den allermeisten Zugriffen tritt daher nur diese minimale Verzögerung in Kraft. Hingegen wirkt in den sehr seltenen Fällen, in denen ein Fehler im Datenwort detektiert wird, eine größere Verzögerung zur Korrektur des Datenwortes. Bei Verwendung eines externen Bausteins zur Fehlerkorrektur kann der Unterschied Fehlerfall – kein Fehlerfall bezogen auf die Gesamtverzögerung inkl. Ein- und Ausgangspuffer jedoch klein sein.

Sind die Zugriffe auf den jeweiligen Speicher selten und ihre Zugriffsgeschwindigkeit von geringerem Interesse, so kann die Fehlerkorrektur auch in Software durchgeführt werden. Dieses ist häufig bei nichtflüchtigen Programm- oder Parameterspeichern z. B. bei PROM, EEPROM oder Flash-Speichern der Fall. Auch hier basiert die Fehlerkorrektur auf der Ausnutzung der gespeicherten Redundanz. Insbesondere beim kritischen Flash ist eine intensive Behandlung notwendig und in der recht umfassenden Kontrolllogik des Flash-Bausteins begründet, die bei

einem SEU unterschiedlich starke Auswirkungen auf den gesamten Baustein haben kann [Brü06]. Die Daten können aufgrund der sehr hohen Kapazität mit einer umfassenden Redundanz gespeichert werden (z. B. für TMR), so dass unter Kenntnis ihrer Wirkungsweise unter Strahlung eine sichere Funktionsweise erfolgen kann. Es ist üblich, selten verwendeten Speicher, wie den Programmspeicher, mit einem Schalter zu versehen und bei Nichtbenutzung von der Versorgungsspannung abzutrennen. Neben der Reduzierung des *Total Dose* Effektes wird das Auftreten SEU-bedingter Fehler sehr stark vermindert.

4.4 Strahlungsbedingte Fehler im RT-Virtex-II FPGA

Mit der Verfügbarkeit großer strahlungstoleranter FPGAs, hier vorzugsweise SRAM-basierte RT-Virtex-II FPGAs, ergeben sich neue Wege zur Implementierung eines Instrumentenrechners. FPGAs der Virtex-II Familie sind als strahlungstolerante Versionen mit unterschiedlichen Ressourcenmengen erhältlich (siehe Tabelle 4.1). Sie weisen eine hohe Toleranz bei der *Total Dose* auf, sind gegen SELs immun und bieten eine hohe Performanz und eine so hohe Menge an Ressourcen, wie sie derzeit von keinem anderen weltraumtauglichen FPGA geboten werden. Allerdings sind diese FPGAs sehr empfindlich gegenüber *Heavy Ions* und empfindlich gegenüber Protonen-induzierten SEUs. Soll ein RT-Virtex-II FPGA in einem Instrumentenrechner Verwendung finden, so sind Kenntnisse über das Verhalten der strahlungsbedingten Fehler erforderlich. Für die Auslegung eines Instrumentenrechners sind die Vorhersage der SEU-Wahrscheinlichkeit und ihre Auswirkungen wichtig. Sind diese bekannt, so können durch geeignete Maßnahmen die RT-Virtex-II FPGAs trotz ihrer niedrigen LET-Schwellenwerte im Weltraum eingesetzt werden.

RT-Virtex-II FPGA	**XQR2V1000**	**XQR2V3000**	**XQR2V6000**
System Gates	1.000.000	3.000.000	6.000.000
Konfigurierbare Logik-Blöcke (CLBs)	40 × 32	64 × 56	96 × 88
Slices	5.120	14.336	33.792
Benutzer-Flipflops (FF)	10.240	28.672	67.584
18 kBit *Block RAM*	40	96	144
Block RAM Bits	720	1.728	2.592
18 Bit × 18 Bit Multiplikationseinheiten	40	96	144
DCMs	8	12	12
IOBs (Maximum)	432	720	1.104

Tabelle 4.1: Strahlungstolerante Bausteine der RT-Virtex-II Familie [Xil04a]

In Bild 4.2 ist der Aufbau eines RT-Virtex-II FPGA gezeigt. Neben den Ein- und Ausgangsblöcken (IOBs) sind unterschiedliche interne Blöcke die Hauptbestandteile des FPGA.

Die wichtigsten internen Blöcke sind:

- Konfigurierbare Logik-Blöcke (*Configurable Logic Blocks*, CLB). Sie beinhalten Elemente für kombinatorische und synchrone Logik inkl. Speicherelementen (siehe Bild 4.3). Ein CLB ist weiter unterteilt in vier *Slices*, die ihrerseits mittels zwei *Lookup*-Tabellen (LUT) Funktionen abbilden und zwei Flipflops enthalten.
- *Block RAMs* mit 18 kBit großem *Dual Port* Speicher
- Multiplikationseinheiten für 18 Bit × 18 Bit Multiplikation
- Digitaler Taktmanager (*Digital Clock Manager*, DCM-Blöcke) zur Taktgenerierung

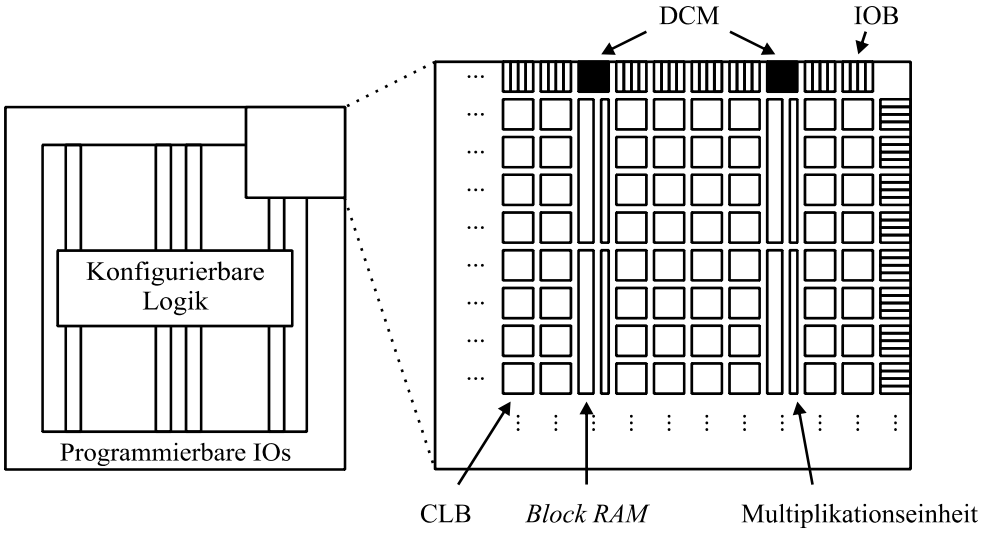

Bild 4.2: Architektur eines RT-Virtex-II FPGA [Xil04a]

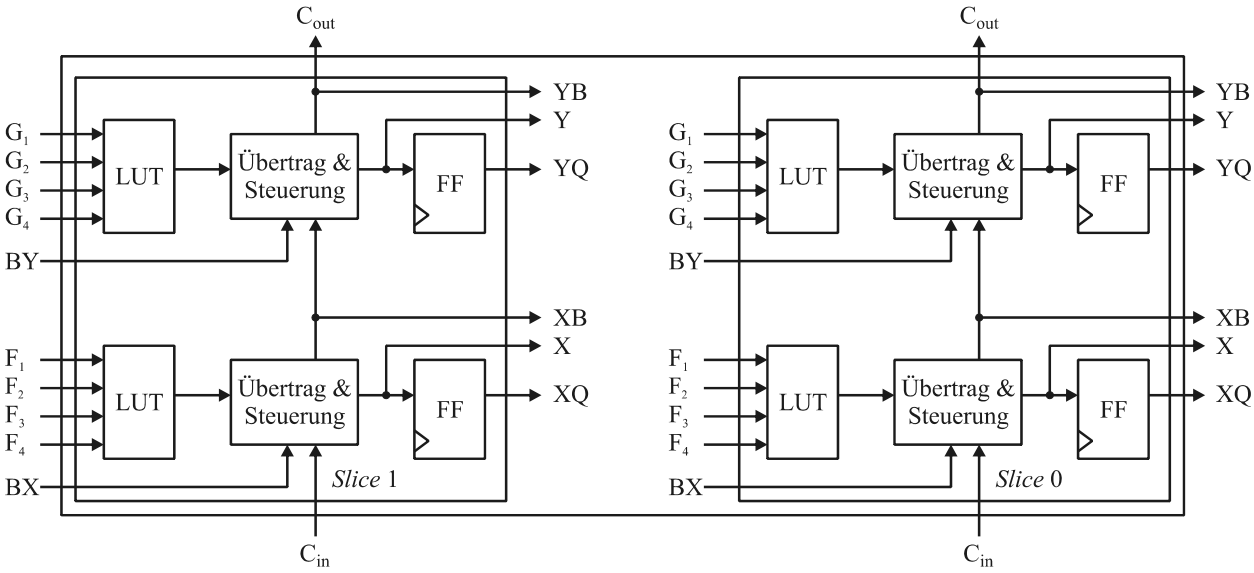

Bild 4.3: Vereinfachte Darstellung der Hälfte eines *Configurable Logic Block* (CLB) mit zwei *Slices* [Xil01]

Zur effizienten Nutzung der FPGA-Ressourcen ist ein umfangreiches Verdrahtungssystem notwendig. Um eine konfigurierbare Verbindung zwischen den Komponenten des FPGA zu ermöglichen, existieren programmierbare, auf SRAM-Zellen basierte Verbindungspunkte und Verbindungsmatrizen. Die Verbindungen selbst erfolgen durch Matrizen mit programmierbaren Punkt-zu-Punkt-Verbindungen (Verbindung von Leitungssegmenten), sowie durch Multiplexer und Puffer [Xil04a]. Zur Konfiguration des FPGA werden nach Aktivierung der Versorgungsspannung die flüchtigen SRAM-Konfigurationszellen mit Daten aus einem nichtflüchtigen Speicher beschrieben. Tabelle 4.2 zeigt am Beispiel des mittelgroßen Bausteins der RT-Virtex-II Familie die zur Konfiguration notwendige Datenmenge und den Anteil der einzelnen Gruppen an der Gesamtkonfigurationsmenge. Es ist ersichtlich, dass der Hauptteil der Konfigurationsbits zur Bestimmung der Verdrahtung und der Konfiguration selbst, gefolgt von den *Block RAM* Bits, benötigt wird.

	XQR2V3000	
Allg. Konfiguration / Verdrahtung Bits	7.347.524	76%
Block RAM Bits	1.769.472	18%
LUT Bits	458.752	5%
Benutzer-Flipflops	28.672	< 1%
DCM Bits	ca. 160 × 12	< 1%
Gesamt Konfiguration Bits	9.595.304	100 %

Tabelle 4.2: Aufteilung der Konfigurationsbits am Beispiel des XQR2V3000 FPGA

Alle in Tabelle 4.2 gezeigten Bits sind in SEU-empfindlichen SRAM-Zellen abgelegt. Verglichen zu den direkten Strahlungsauswirkungen auf diese SRAM-Zellen sind die Auswirkungen auf die Kombinatorik sehr klein bzw. vernachlässigbar [Shi02]. Daher werden bei der Bestimmung der zu erwartenden Fehlerraten nur diejenigen Fehler betrachtet, die mit einem Zustandswechsel einer Speicherzelle einhergehen.

Strahlungsbedingte Fehler können sehr unterschiedliche Auswirkungen haben. Die Änderung eines Konfigurationsbits kann z. B. dazu führen, dass eine LUT nicht mehr als einfache LUT arbeitet, eine Verbindung nicht mehr zwischen den beiden ausgewählten Endpunkten besteht oder ein Eingang aus anderer Quelle gespeist wird. Die SEU-bedingten Bitwechsel lassen sich im RT-Virtex-II FPGA in vier Gruppen einteilen:

1. Bitwechsel in Konfigurations-Logik
    a. Logische Funktionen durch *Lookup*-Tabelle (LUT) mit der Folge der Änderung der Funktion selbst (siehe Bild 4.4 oben)
    b. Verdrahtung (z. B. Multiplexer, siehe Bild 4.4 unten)
    c. Benutzereinstellungen (I/O-Typ, *Block RAM*-Typ, *Slice*-Einstellungen, Takt-Einstellungen)

2. Bitwechsel in *Block RAM*-Bits
3. Bitwechsel in den Benutzer-Flipflops
4. Bitwechsel in architektonischen Sondereinheiten (*Single Event Functional Interrupts*, SEFI)
   a. Konfigurationslogik
   b. JTAG- und *SelectMAP*-Konfigurationsschnittstelle
      (Verlust des Konfigurations-Zugriffes)
   c. *Power-On* Reset (POR):
      Globales Zurücksetzen der internen Logik, Verlust der Konfiguration
   d. Reset-Logik

Das Auftreten SEU-bedingter Fehler der Gruppen 1-3 ist proportional zur verwendeten Menge der jeweiligen Ressource. Die strahlungsbedingten Bitwechsel in architektonischen Sondereinheiten (SEFI) dagegen betreffen einzelne, für sich abgeschlossene Schaltungsbereiche zum grundsätzlichen Betrieb des FPGA (z. B. Reset-Logik). Sie treten selten auf, haben dafür aber größere Auswirkungen. Ein SEFI erzeugt sehr viele Fehler und kann dementsprechend detektiert werden. Alle SEFI können durch gezielte Behandlung der Konfigurationsschnittstelle behoben werden [Yui02].

Wird die Auswirkung von Strahlung auf einen RT-Virtex-II FPGA untersucht, so ist ersichtlich, dass die Fehlerrate eines Moduls, abgesehen von den SEFI, quantitativ proportional zu ihrem Ressourcenbedarf ist. Je mehr Konfigurationsdaten für ein Design relevant sind, umso größer sind die Auswirkungen von Strahlung darauf. Das bedeutet, dass die Wahrscheinlichkeit, dass genau diese Ressourcen durch Strahlung verändert werden, die Fehlerrate für dieses Modul bestimmt. Zur Berechnung der Fehlerrate eines FPGA ist daher zuerst der Ressourcenbedarf eines Moduls, z. B. der Sensor-Schnittstelle, zu ermitteln. Anschließend erfolgt für jedes Modul die Berechnung der SEU-Rate für die unterschiedlichen Ressourcentypen. Die ressourcen-abhängigen Wirkungsquerschnitte sowie die Strahlungsumgebung für die jeweilige Mission sind hierzu notwendig. Die Auswirkungen eines SEU können auf einzelne Module unterschiedliche Auswirkungen haben. Ein Modul kann z. B. ständig in Betrieb sein und damit sehr empfindlich reagieren, ein anderes dagegen jedoch nur selten aktiv sein und zudem noch vor jeder Verwendung in den Ursprungszustand zurückgesetzt werden. Nach Bestimmung der Fehlerrate eines Moduls wird die Fehlerrate für den gesamten FPGA bestimmt. Bei der Berechnung ist zu beachten, dass selbst in einem dichten Design der Ressourcenbedarf für die Verdrahtung weniger als 12 % beträgt und damit nicht jeder SEU direkt Fehler erzeugend wirkt [Car06]. Bild 4.5 zeigt die Anzahl an notwendigen Konfigurationsfehlern zum Verursachen eines wirkungsvollen Fehlers in der Verdrahtung.

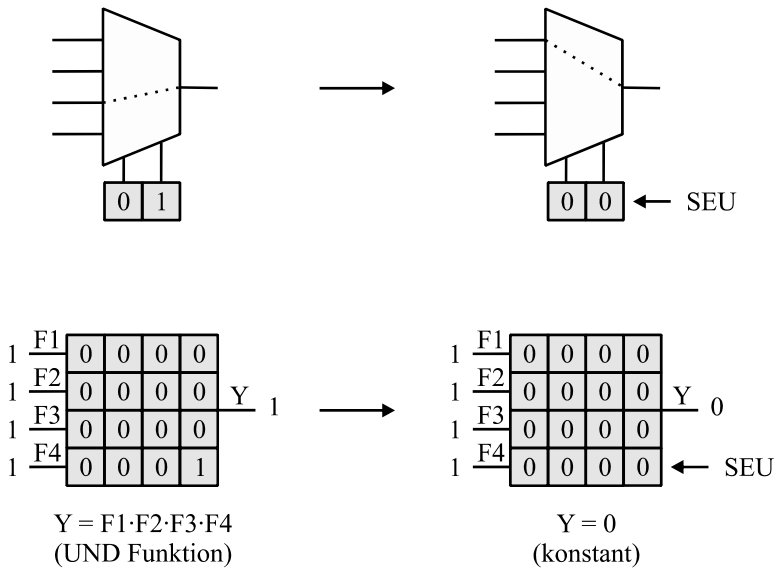

Bild 4.4:
Oben: Falsche Auswahl durch SEU in Multiplexer-Konfiguration zur Verdrahtung;
Unten: Funktionsänderung durch SEU in LUT Konfiguration [Gra03]

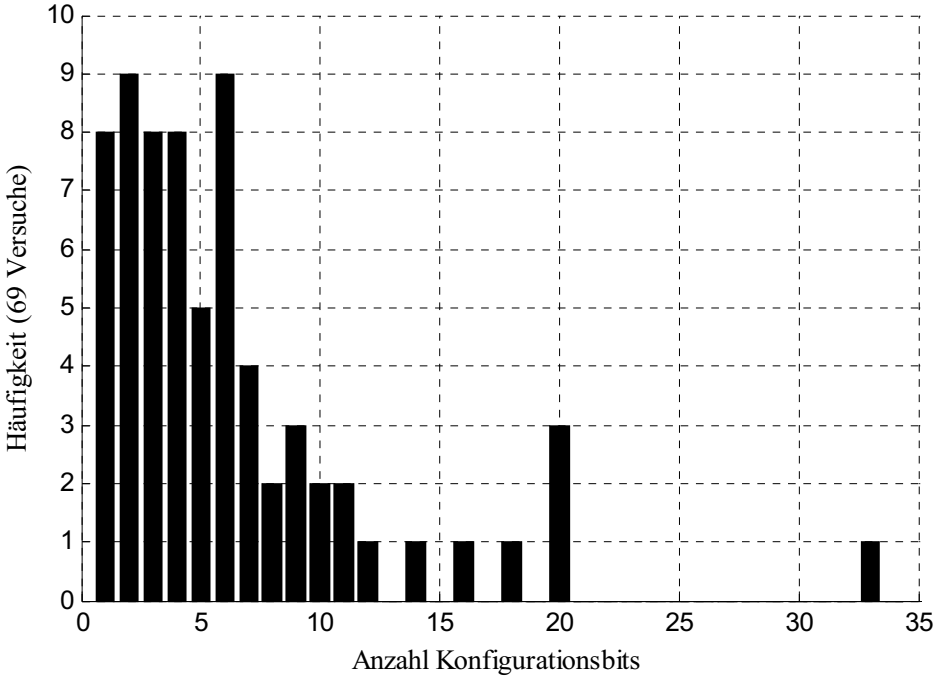

Bild 4.5: Histogramm Verdrahtungsfehler [Car06]

4.5 Maßnahmen gegen strahlungsbedingte Fehler beim RT-Virtex-II FPGA

Von der Herstellerfirma Xilinx werden in Abhängigkeit vom zu schützenden Element (Konfigurationsspeicher, LUT, *Block RAM*, IOB, …) verschiedene Wege zur Minimierung strahlungsbedingter Fehler empfohlen [Car99]. Die dort angegebenen Maßnahmen zur Reduzierung der Strahlungsempfindlichkeit basieren bei Logik im Wesentlichen auf TMR-Konzepten (siehe Bild 4.6 bis Bild 4.9). Um Fehler in der Konfiguration abzufangen, wird ein Modul dreifach parallel ausgeführt und dessen Ausgänge mit einem Entscheider (*Voter*) verbunden. Dieser Entscheider kann selbst auch dreifach aufgebaut sein (z. B. als *Minority Voter* unter Verwendung von IOs, siehe Bild 4.9) und liefert das richtige Ergebnis, solange nur einer der drei Teilmodule von einem Fehler betroffen ist. TMR-Design zusammen mit einem periodischen Auffrischen der Konfiguration (Schrubben), d. h. ohne vorheriges Löschen, gewährleisten, dass ein sicheres System aufgebaut werden kann. Ein Fehlerfall tritt erst auf, sobald zwei der drei Teilmodule oder aber der Entscheider einen Fehler aufweisen. Die Schrubbrate ist derart zu wählen, dass die Wahrscheinlichkeit für das Auftreten eines Fehlers in mehr als einem Teilmodul innerhalb eines Schrubbzyklus sehr gering ist. Für die Ein- und Ausgangsblöcke, deren Funktionsweise auch durch den Zustand von SRAM-Zellen definiert ist, wird angeregt, für ein Signal drei Anschlüsse zu verwenden (siehe Bild 4.8 und Bild 4.9). Mit Hilfe von *Tri-State* Puffern im FPGA erfolgt die Durchführung der Entscheidung (mit *Minority Voter*) außerhalb des FPGA. Die Anzahl der Ein- und Ausgänge wird verdreifacht. Bei *Block RAMs* wird eine dreifache Ausführung mit Schrubben oder eine Fehlerkorrektur empfohlen, letztere hilft bei Konfigurationsfehlern jedoch nur bedingt. Mit Hilfe des Softwaretools XTMRTool [Xil04b, Saa04] der Firma Xilinx können entsprechende Maßnahmen auf den gesamten Baustein oder auf einzelne Module begrenzt automatisch angewendet werden.

Dem Sicherheitsaspekt dieser Maßnahmen stehen wesentliche Nachteile gegenüber:

- Möglicher Anstieg des Ressourcenverbrauchs auf mehr als das dreifache
- Starker Anstieg des Leistungsverbrauchs
  (Parallelisierung ohne sichtbaren Informationsgewinn)
- Geringere Performanz aufgrund längerer Signallaufzeiten

Instrumentenrechner, die häufig als nicht missionskritisch eingestuft sind, unterliegen einer sehr starken Ressourcenbeschränkung. Daher sind die oben genannten Maßnahmen in den meisten Fällen nicht vollständig für alle Module des FPGA umzusetzen. Ohne alternative Maßnahmen wäre ein Design jedoch überhaupt nicht geschützt. Es hätte eine sehr geringe Verfügbarkeit, da nach einem SEU keine Wiederherstellung erfolgt. Eine sinnvolle Implementierung der Fehlerbehandlung entspricht der Suche nach dem richtigen Verhältnis der Ressourcen Performanz, Energiebedarf, Größe, Masse und Maßnahmen gegen strahlungsbedingte Fehler. Je nach Grad des zu implementierenden Schutzes ergibt sich bei den RT-Virtex-II FPGAs ein breites Spektrum von Fehlerbehandlungsmethoden.

Eine ressourcenintensive Methode wie TMR-Design erlaubt es in den meisten Fällen die unmittelbare Auswirkung eines SEU zu korrigieren, bevor eine Fehlfunktion des Systems auftritt. Dadurch ist das System ohne einen Wiederherstellungsmechanismus weiter funktionsfähig. Auf diese Eigenschaft muss bei Ressourcen schonenden Wegen in den meisten Fällen verzichtet werden. Hierbei handelt es sich im Wesentlichen um eine Kombination aus:

1. Erkennung eines Fehlers
2. Durchführung des Wiederherstellungsprozesses.

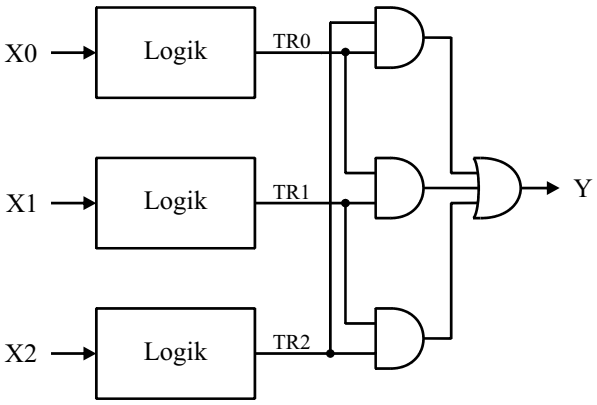

Bild 4.6: TMR mit einfachem Entscheider [Car99]

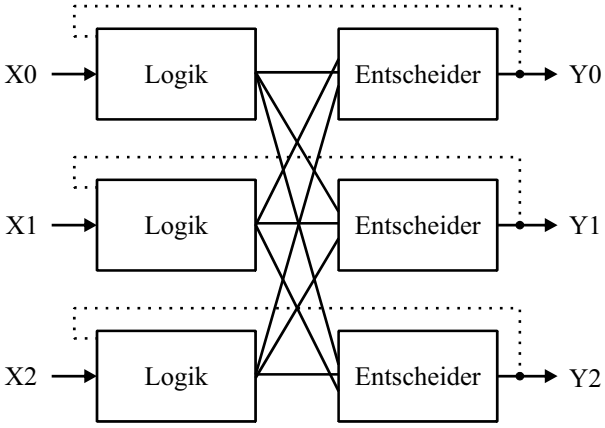

Bild 4.7: TMR mit dreifachem Entscheider und Rückkopplung [Ste04]

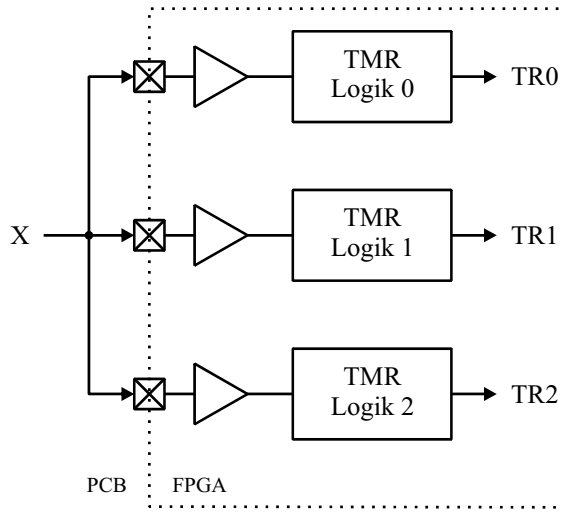

Bild 4.8: TMR mit dreifacher FPGA-externer Eingangsbeschaltung [Car06]

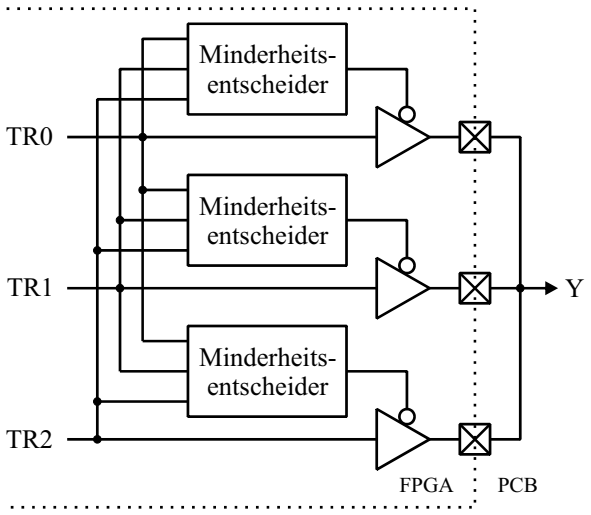

Bild 4.9: TMR-Entscheider mit dreifacher FPGA-externer Ausgangsbeschaltung unter Verwendung von *Tri-State*-Puffern [Car06]

Analog zur Konfiguration eines RT-Virtex-II FPGA, d. h. zur Beschreibung des SRAM-basierten Konfigurationsspeichers mit externen Daten, können Konfigurationsdaten wieder aus dem Baustein ausgelesen werden. Das Zurücklesen innerhalb eines *Readback*-Verfahrens liefert damit eine erhebliche, wenn auch nicht vollständige Menge an Informationen über den Zustand des Bausteins. Die Daten sind über den Baustein verteilt für verschiedene Konfigurationsblöcke vorhanden. Es sind z. B. die Daten für die Verdrahtung oder die Ein- und Ausgangsblöcke blockweise mit Daten über den aktuellen *Block RAM*-Inhalt verteilt. Ein Ansatz für die Bestimmung von Konfigurationsfehlern besteht darin, alle Daten, die Aussagen über nicht absichtlich veränderte Bausteinzustände erlauben, auszulesen und sie für einen Vergleich zu verwenden. Hierdurch kann ein fehlerhafter Zustand einer Speicherzelle aufgespürt werden. Die auszulesenden Daten können z. B. Ein- und Ausgangsblöcke oder die Verdrahtung sein, während Änderungen im *Block RAM* zu erwarten sind und insbesondere beim *Block RAM* auch nicht ausgelesen werden dürfen. Durch gezieltes Auslesen können bestimmte Bereiche der Konfiguration ausgeklammert werden. Der Vergleich wird derart ausgeführt, dass die ausgelesenen Daten jeweils Byte für Byte mit ihrem erwarteten Wert verglichen werden. Dieses erfordert die Speicherung von *Readback*-Daten als Referenz innerhalb einer *Bit*-Datei. Der Inhalt einer *Mask*-Datei dient dazu, die für einen Vergleich relevanten und nicht relevanten Konfigurationsbits voneinander zu unterscheiden [Xil05a].

Bild 4.10: *Readback*-Verfahren zur Überprüfung der Konfiguration

Bild 4.10 zeigt die Ausführung des *Readback*-Verfahrens. Zur Durchführung werden die Konfigurationsdaten von einer Kontrolllogik parallel aus dem FPGA ausgelesen und mit den Informationen der *Bit-* und *Mask-Datei* verglichen. Bei Detektierung eines Fehlers wird unmittelbar die Wiederherstellung eingeleitet und die Nichtverfügbarkeit des Systems dadurch sehr kurz gehalten. Um einen Fehler schnell erkennen zu können, sollte das Auslesen der Konfiguration periodisch in kurzen Abständen erfolgen. Bei Verwendung genügend schneller Bausteine für die *Bit-* und *Mask*-Dateien kann das *Readback* bei einem XQR2V3000 FPGA z. B. alle 100 ms erfolgen. Im Falle einer Wiederherstellung sind 100 ms zum Laden der Konfiguration, 300 ms zum Laden des Programms und ca. 400 ms zur Ausführung der Wiederherstellungsmaßnahmen durch die Software notwendig. Nach Auftreten und Detektierung eines Fehlers kann das System somit innerhalb einer Sekunde wieder einsatzbereit sein.

Im Vergleich zur Konfiguration des FPGA ist für das *Readback*-Verfahren neben der *Readback*-Kontrolllogik noch die Speicherung der *Mask-Datei*, i. Allg. in einem zusätzlichen Baustein, erforderlich. Dieses hat Einfluss auf die Zuverlässigkeit, die Größe der DPU und erhöht den Leistungsbedarf der DPU.

Fehler in der Benutzerlogik können nicht mit dem *Readback*-Verfahren entdeckt werden, da Änderungen in den Benutzer-Flipflops und *Block RAMs* erlaubt sind. Zur Detektierung von strahlungsbedingten Fehlern in diesen Bereichen kann der *Watchdog*-Zähler verwendet werden, wie er auch für Programmierfehler innerhalb der DPU eingesetzt wird. Typische Zeiten bis zum Auslesen eines *Watchdog* Resets sind einige 100 ms bis zu einigen Sekunden. Hiermit können Fehler detektiert werden, die zu einem Systemzusammenbruch oder zum Ausfall eines Software Tasks führen. Fehler in Logikmodulen, die nicht direkt im Programmablauf eingebunden sind, führen selten zur Auslösung eines *Watchdog* Resets. Hier können Module in regelmäßigen Abständen mit Testvektoren stimuliert werden, um anschließend die korrekte Bearbeitung dieser Vektoren mit dem erwarteten Ergebnis zu vergleichen. Bei Ungleichheit kann dieses Modul lokal zurückgesetzt oder ein Wiederherstellungsprozess eingeleitet werden. Bei nicht kontinuierlich, aber vom Prozessor synchron verwendeten Modulen kann ein Modul vor seiner Verwendung in den Ruhezustand gesetzt werden, d. h. in den Zustand, der im fehlerfreien Fall vorherrscht. Dadurch können Auswirkungen reduziert werden, die aus Fehlern nach der letzten Zurücksetzung resultieren. Als Beispiel zu nennen ist hier das Zurücksetzen einer Bildkompressionseinheit vor jedem Bild.

Ist ein Fehler erkannt worden, so wird eine Wiederherstellung eingeleitet. Während des Bootprozesses werden nichtflüchtige Systemvariablen abgefragt. Sie beinhalten Informationen über die Ursache des aktuellen Bootprozesses (*Power-On* Reset, *Watchdog* Reset, …). Diese Werte liegen in der Systemkontrolle vor, die für das sichere Starten des Systems verantwortlich ist. Das Ziel des Wiederherstellungsprozesses ist es, den Instrumentenzustand vor Auftreten des Fehlers wiederherzustellen. Für eine erfolgreiche Wiederherstellung sind gute Kenntnisse über den Zustand des Instruments unmittelbar vor dem Reset notwendig. Hierzu sichert die DPU während des normalen Betriebs periodisch Systeminformationen in einem geschützten Speicherbereich, d. h. einem nach einem Reset (ausgenommen *Power-On* Reset) fehlerfrei zugänglichen Bereich, wie z. B. ein ständig mit Spannung versorgtes strahlungsfestes RAM oder Registerbänke in einem strahlungsfesten Baustein. Die periodisch abzulegenden System-informationen können sein:

- Instrumentenmodus
- Dateisystemtabellen
- Zustand der Schnittstellen (z. B. aktiver *SpaceWire*-Link)
- Aktive Messsequenz mit zuletzt bearbeiteter Messung

Innerhalb des Systementwurfes ist darauf zu achten, dass bei einem unbeabsichtigten Reset und der dazugehörigen Wiederherstellung möglichst wenige Daten beschädigt werden. Sollte das Instrument so eingesetzt werden, dass es während eines bestimmten Zeitraums sehr viele Bilder im Massenspeicher zwischenlagert, um sie zu einem späteren Zeitpunkt vollständig an die Raumsonde zu übertragen, so ist es nicht tolerierbar, wenn bei gefülltem Massenspeicher ein Systemzusammenbruch zum Verlust des gesamten Massenspeicherinhalts führt. Mit

entsprechender Verschaltung kann dieses unterbunden oder wesentlich unwahrscheinlicher gemacht werden. Ein Beispiel ist eine gesicherte Durchführung der *Refresh*-Zyklen zur Auffrischung der Ladung bei Verwendung einer Steuerungseinheit für SDRAM im strahlungsempfindlichen Baustein. Die *Refresh*-Überwachung ist strahlungsresistent aufzubauen und kann in einem strahlungsfesten Baustein oder aber als TMR-Design in einem strahlungsempfindlichen Baustein implementiert werden. Bei Ausbleiben des *Refresh*-Zyklus durch strahlungsbedingte Fehler in der SDRAM-Steuerungseinheit übernimmt hierbei die *Refresh*-Überwachung kurzfristig die Kontrolle über den dynamischen Speicher. Dieses erfolgt jedoch lediglich mit dem Ziel, periodische *Refresh*-Zyklen zu erzeugen, um damit die Datenerhaltung zu sichern. Mit der Wiederherstellungsprozedur geht die Kontrolle wieder an die SDRAM-Steuerung im strahlungsempfindlichen Baustein zurück. Die Alternativen zur *Refresh*-Überwachung können eine SDRAM-Steuerung im strahlungsfesten Baustein oder aber dessen Ausführung als TMR-Design in einem strahlungsempfindlichen Baustein sein.

Ergibt sich für ein Modul eine zu hohe Fehlerwahrscheinlichkeit, dann kann diese durch entsprechende Maßnahmen gezielt reduziert werden. Diese Maßnahmen können in beschränktem Maße auf der Einführung von Redundanz beruhen. Bei wichtigen Modulen wie z. B. der Überwachung von *Refresh*-Signalen, kann die Anwendung von TMR auf ein kleines Modul sinnvoll sein. Selbst bei Verzicht auf das Schrubben der Konfiguration, ist die Wahrscheinlichkeit sehr gering, dass mehr als eines der drei parallelen TMR-Bestandteile dieses kleinen Moduls beschädigt werden, wenn das Instrument in regelmäßigen Abständen, wie z. B. bei Venus Express / VMC einmal pro Tag, neu initialisiert wird.

4.6 Ausblick für verschiedene Architekturmöglichkeiten / Implementierungswege

Beim Aufbau eines Instrumentenrechners für eine kompakte Kamera können verschiedene Implementierungswege gewählt werden. Der traditionelle Aufbau eines Instrumentenrechners beinhaltet die Verwendung strahlungsfester Bauteile und wird in Kapitel 5 beschrieben. Die Auswahl strahlungsfester Prozessoren ist jedoch gering, ihre Performanz beschränkt. Zur Erweiterung der Funktionalität ist als Variante ein strahlungstoleranter Massenspeicher implementiert, was zu einer Aufweichung der Strahlungsfestigkeit führt und das strahlungsfeste Design in ein überwiegend strahlungsfestes Design überführt.

Die Aufweichung der Strahlungsfestigkeit durch den Massenspeicher führt dazu, eine Überführung auch anderer Baugruppen aus einem strahlungsfesten Design in ein strahlungstolerantes Design zu erwägen. In Kapitel 6 wird eine strahlungstolerante DPU beschrieben, die auf der Verfügbarkeit großer strahlungstoleranter SRAM-basierter RT-Virtex-II FPGA basiert. Dabei ist ein 32-Bit-Prozessor als VHDL-Modell zusammen mit zusätzlicher Logik, wie Fehlerkorrektur oder Schnittstellen, in einem strahlungsfesten FPGA implementiert. Die hohe SEU-Empfindlichkeit der strahlungstoleranten RT-Virtex-II FPGAs ist im Design zu beachten. Durch Implementierung von möglichst viel Logik innerhalb eines FPGA kann eine hohe

Packungsdichte in der DPU erreicht werden und es lässt sich eine kleine DPU aufbauen. Beispiele hierfür sind die Instrumente Venus Express / VMC [Mar04] und DAWN / DFC [Wit04]. Dieser Ansatz vereint innerhalb eines Bausteins die universelle Abarbeitung in einer Standard-CPU mit der spezialisierten, energieeffizienten Abarbeitung von Aufgaben in dedizierten Logik-Modulen und erlaubt damit ein großes Maß an Vielseitigkeit und an Performanzsteigerungen bei gleichzeitig kompakter Bauform. Mit zunehmender dedizierter Logik für spezielle Funktionen kann der Energiebedarf optimiert werden.

Die Verwendung kommerzieller Prozessoren in einem strahlungstoleranten Design bedeutet einen hohen Performanzgewinn. Kommerzielle Prozessoren sind häufig für hohe Performanz bei niedrigem Energiebedarf optimiert. Sie beinhalten neueste Technologie und sind den strahlungsfesten Prozessoren um etwa zwei Generationen voraus. Die Notwendigkeit, bei kommerziellen Prozessoren umfangreiche Qualifizierungsmaßnahmen (z. B. intensive Strahlungstests) durchzuführen und aufwendige Fehlerkorrekturmaßnahmen innerhalb des Systems bereitstellen zu müssen, führt dazu, dass sie im Weltraum weniger Verwendung finden. Zudem haben sie nur kurze Lebenszeiten – eine lange Verfügbarkeit der Bauteile ist nicht gegeben. Aus diesen Gründen wird dieser Ansatz hier nicht weiter verfolgt. Ein solcher Ansatz ist in [Ger01] zu finden.

Bei Kamerainstrumenten ist der erzielbare wissenschaftliche Nutzen häufig durch aufwendige Bildverarbeitungsfunktionen begrenzt. Eine schnelle Bilddatenkompression kann z. B. den Nutzen erhöhen. Zudem hat die Flexibilität im operationellen Betrieb Einfluss auf den wissenschaftlichen Nutzen. Sie wird u. a. durch die Menge an frei verfügbarem Speicher oder durch die Möglichkeit der Ausführung leicht anpassbarer, frei definierbarer Skripte bestimmt.

Am Beispiel der JPEG2000-Kompression werden daher in den Kapiteln 5 und 6 auf Basis des SPARC-V8-kompatiblen Prozessors folgende Implementierungsarten der Musterkonfiguration (siehe Abschnitt 4.1) besprochen:

- Strahlungsfeste DPU (Kapitel 5)
- Überwiegend strahlungsfeste DPU mit Massenspeicher (Kapitel 5)
- Strahlungstolerante DPU mit RT-Virtex-II FPGA (Kapitel 6).

Nach der anschließenden Vorstellung des entwickelten IDA JPEG2000 Core in Kapitel 7 wird die Implementierung dieser Hardwarebeschleunigung in die strahlungsfeste DPU ohne Massenspeicher sowie in die strahlungstolerante DPU beschrieben:

- Strahlungsfeste DPU mit IDA JPEG2000 Core (Kapitel 8.1)
- Strahlungstolerante DPU mit RT-Virtex-II FPGA und IDA JPEG2000 Core (Kapitel 8.2).

5 Strahlungsfeste DPU

Um Vergleichswerte zu einer strahlungstoleranten DPU herstellen zu können, wird in diesem Kapitel die Implementierung einer strahlungsfesten DPU nach traditionellem Ansatz mit Verwendung strahlungsfester Bauteile gezeigt und es werden typische Kennwerte analysiert. Eine Variante der strahlungsfesten DPU mit dem Einsatz eines Massenspeichers erhöht die operationelle Flexibilität.

5.1 Aufbau

Bild 5.1 zeigt einen typischen Aufbau, bei dem die strahlungsfesten Speicher direkt an den Prozessor angebunden sind und die missionsspezifischen Anforderungen wie z. B. die Schnittstellenlogiken innerhalb eines strahlungsfesten FPGA realisiert sind.

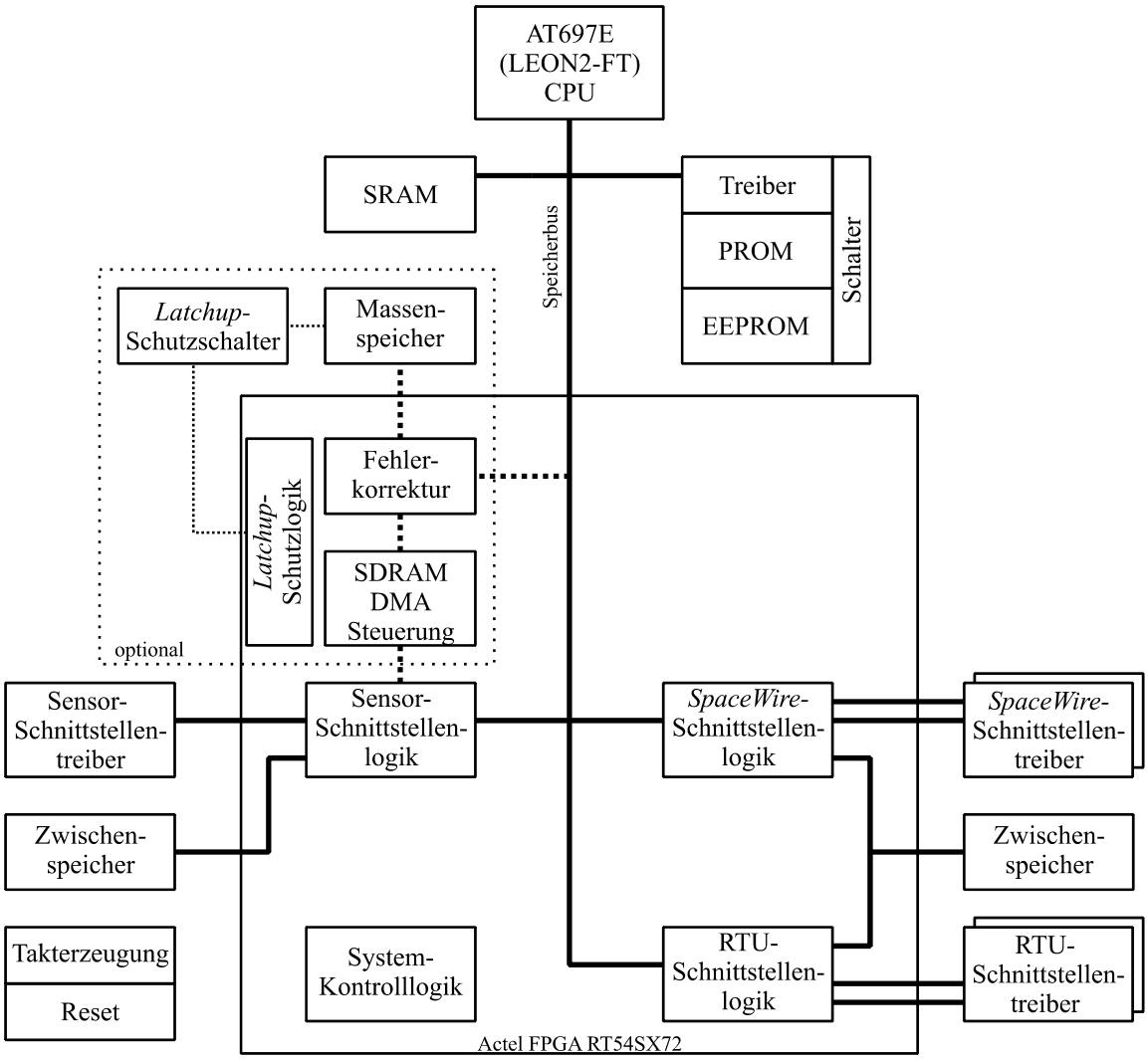

Bild 5.1: Aufbau der strahlungsfesten DPU;
gestrichelt eingerahmt: Variante unter Nutzung eines Massenspeichers

Kern der DPU ist der SPARC-V8-kompatible AT697E (LEON2-FT) Prozessor mit einer maximalen Taktfrequenz von 100 MHz. Die Logiken sowohl der RTU- als auch der *SpaceWire*-Schnittstelle sind in einem FPGA der RT54SX-S Familie der Firma Actel implementiert und über den Speicherbus mit dem Prozessor verbunden. Als strahlungsfester FPGA enthält dieser Baustein auch die Systemkontrolle u. a. bestehend aus *Power-On* Logik, *Watchdog*-Zähler und statischen Registern zum Steuern von z. B. dem NVRAM-Schalter. Da der FPGA nicht über genügend internen Speicher verfügt, sind die Schnittstellen zum temporären Speichern von Zwischenwerten und damit zur Entkopplung der Hardware von der Software mit kleinen externen Speicherbausteinen versehen. Diese sind als FIFO-Speicher betrieben. Es werden teilweise mehrere FIFO-Speicher mit der entsprechenden Steuerung im FPGA innerhalb eines Bausteins implementiert. Durch das Abbilden auf einem gemeinsamen Speicher können so die Hardwareressourcen verringert werden. Die nichtflüchtigen Speicher sind mit Schaltern sowie geeigneten Treiberbausteinen (d. h. ohne Eingangsschutzdioden) zum Schutz vor Rückspeisung versehen. Bei einer aktiven Zeit von z. B. 1 min pro Tag sind diese Speicher praktisch kaum aktiv. Da Fehler im Programmcode folgenschwere Auswirkungen haben können, ist der nichtflüchtige, veränderbare Speicher mit Fehler korrigierenden Maßnahmen geschützt, die in Software realisiert sind.

5.2 Kennwerte

Zur Beurteilung der Leistungsfähigkeit der strahlungsfesten DPU ist die Bestimmung von verschiedenen Kennwerten wie Performanz, Zuverlässigkeit, Einfluss strahlungsbedingter Fehler, Leistungsbedarf, Platzbedarf, Volumen und Masse erforderlich.

5.2.1 Performanz

Um für den Ansatz der strahlungsfesten DPU die Performanz zu bestimmen, in dieser Arbeit gemessen an der Durchführung der JPEG2000-Kompression, werden Testbilder mit Bittiefen von 12 bzw. 14 Bit komprimiert (siehe Bild 5.2 - Bild 5.5). Es wurde die JasPer-Implementierung (siehe Abschnitt 3.4) verwendet, die um eine *Rate Control* mit Einfluss auf *Tier-1* erweitert und von Fließkomma- auf Festkomma-Arithmetik umgestellt wurde.

In Tabelle 5.1 und Tabelle 5.2 sind die Ergebnisse der Kompression zusammengefasst. Für die verlustbehaftete Kompression zeigt Bild 5.6 die zur Kompression notwendige Zeit, die in Bild 5.7 und Bild 5.8 am Beispiel des Testbildes *Mars Full* weiter aufgeschlüsselt ist. Der Hauptanteil wird von *Tier-1* benötigt und verlagert sich mit zunehmendem Kompressionsfaktor zu der konstanten Zeit, die zur Durchführung der Transformation erforderlich ist. Zur Kompression von Bildern der Größe $1\text{k} \times 1\text{k} \times 14$ Bit werden bei verlustloser Kompression durchschnittlich 21 s benötigt. Dem steht eine gewünschte Bildrate von 1-2 s bei typischen Kamerainstrumenten gegenüber, so dass diese DPU stark eingeschränkten wissenschaftlichen Nutzen bringt. Bild 5.9 zeigt die Qualität der komprimierten Bilder anhand der PSNR-Metrik.

Da sich der strahlungsfeste AT697E (LEON2-FT) Prozessor momentan noch im Qualifizierungsprozess befindet und erst in Kürze verfügbar ist, erfolgten die Messungen in einer Testumgebung unter Verwendung der SPARC-V8 CPU in einem Virtex-II FPGA. Der AT697E (LEON2-FT) Prozessor wird durch einen LEON2-IP Core in einem Virtex-II FPGA abgebildet, so dass der Kern (Integer-Einheit) identisch ist. Der IP Core ist so konfiguriert, dass er mit dem LEON2-FT vergleichbar ist (z. B. gleiche *Cache*-Größen). Er wird mit 50 MHz betrieben und erlaubt mit Anpassung der *Waitstates* beim Speicherzugriff die Hochrechnung auf den AT697E (LEON2-FT) Prozessor. Die FPU wird während der Kompression nicht verwendet. Die maximale Größe eines *Tile* ist auf 256 × 256 festgelegt.

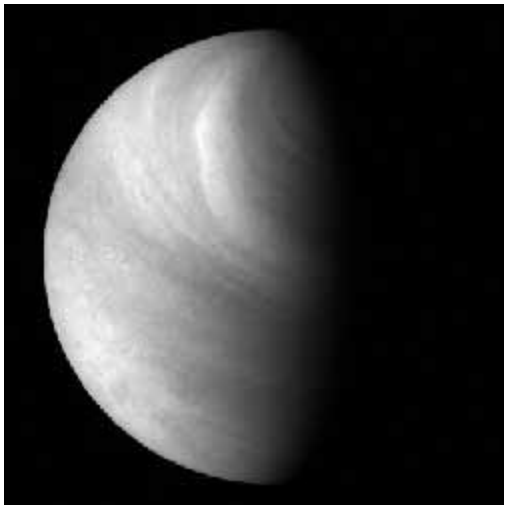

Bild 5.2: Venus 386 × 386 × 14 Bit (Bild von Venus Express / VMC)

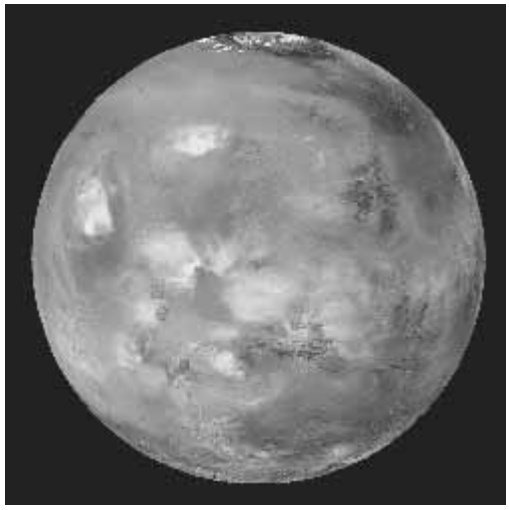

Bild 5.3: Mars Full (umgewandelt) 512 × 512 × 12 Bit [DLR02]

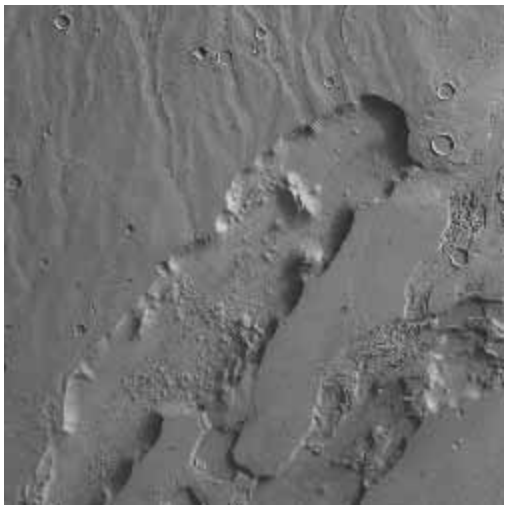

Bild 5.4: Mars DaoVallis (umgewandelt) 512 × 512 × 12 Bit [Neu04]

Bild 5.5: Mars Surface 256 × 248 × 12 Bit [NAS98]

Testbild	Kompressionsfaktor	Ausführungszeit [ms]	Durchsatz [MBit/s]
Venus	1,61	2.660	0,78
Mars Full	1,59	4.365	0,72
Mars DaoVallis	1,37	4.705	0,67
Mars Surface	1,28	1.270	0,62

Tabelle 5.1: Verlustlose Kompression

Kompressionsfaktor *c*	verlustlos	2	4	6	8	10	12	14	16
Durchsatz [MBit/s]	0,70	0,87	1,23	1,43	1,57	1,66	1,72	1,77	1,80

Tabelle 5.2: Durchschnittlicher Durchsatz der Kompression

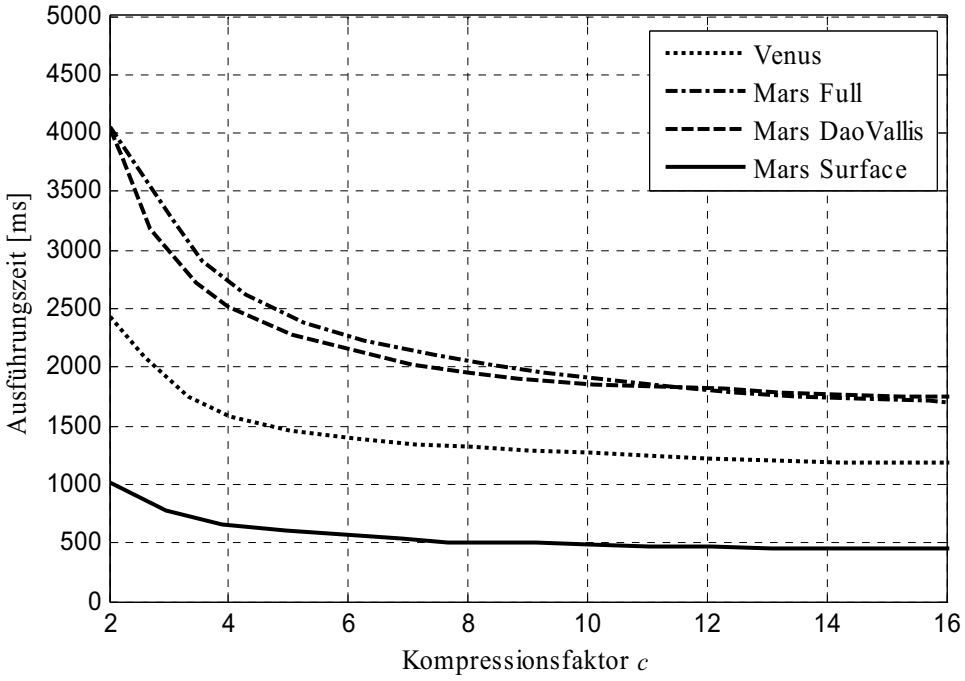

Bild 5.6: Ausführungszeit zur Kompression der Testbilder

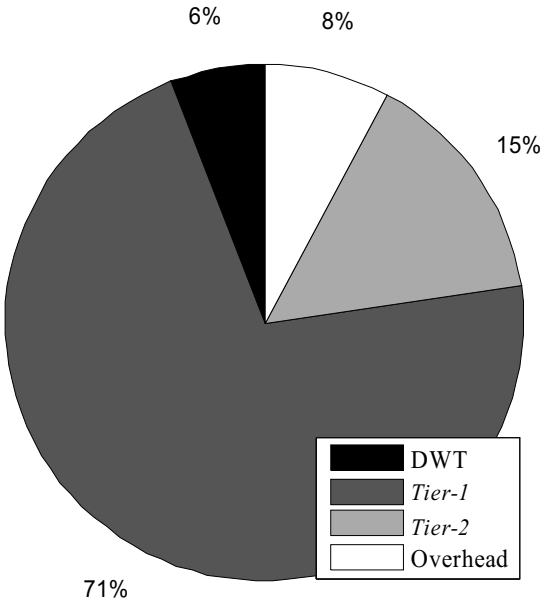

Bild 5.7: Aufteilung Ausführungszeit bei verlustloser Kompression

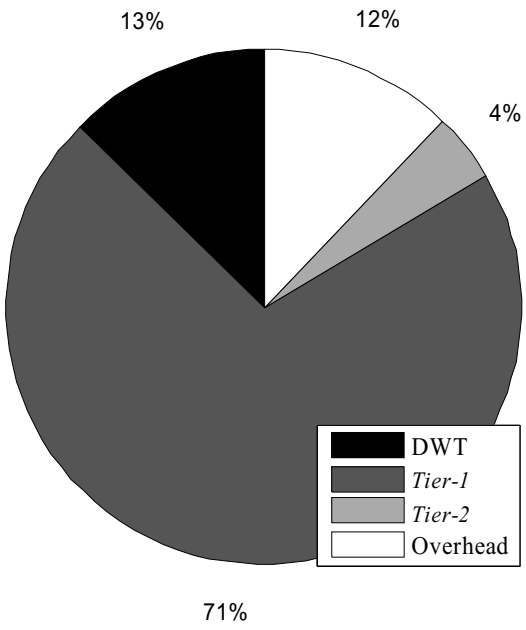

Bild 5.8: Aufteilung Ausführungszeit bei Kompressionsfaktor 10

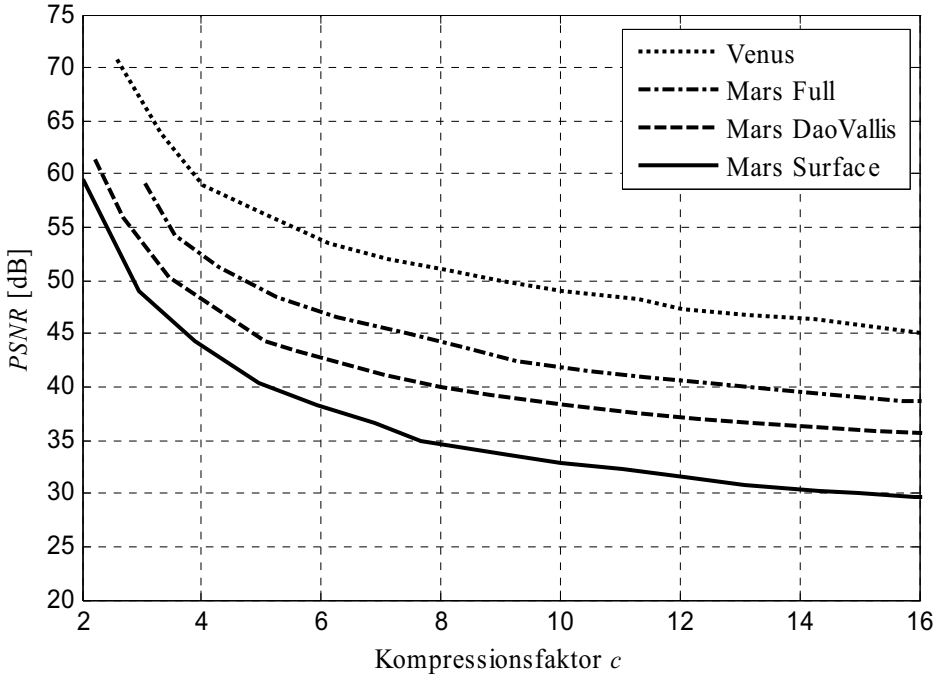

Bild 5.9: Kompressionsqualität Testbilder, *Tile*-Größe 256 × 256 Pixel

5.2.2 Zuverlässigkeit

Unterschiedliche Merkmale wie Qualitätsgrad und Anzahl der Bauteile aber auch Grad der eingebauten Redundanz und die Wartbarkeit bestimmen die Zuverlässigkeit. Für die strahlungsfeste DPU nach Bild 5.1 ergeben sich die in Tabelle 5.3 angegebenen Zuverlässigkeiten. Die Werte sind für eine Betriebsdauer von 5 Jahren bei einer Temperatur von 50 °C, sowie, in Anlehnung an Venus Express / VMC, einer aktiven Zeit von 6 h pro Tag berechnet. Es ist ersichtlich, dass die SRAM-Bausteine (ohne Fehlerkorrektur) und die Widerstände den größten Einfluss auf die Reduzierung der Zuverlässigkeit haben. Die Zuverlässigkeit des nichtflüchtigen, veränderbaren Speichers (EEPROM) ist aufgrund der verwendeten Fehlerkorrektur mit einer 1-aus-6 Redundanz sowie einer nur kurzen aktiven Zeit sehr hoch. In den Raumsonden-Schnittstellen befinden sich viele Widerstände, die aufgrund der einfach redundanten Auslegung nicht ins Gewicht fallen. Der genaue Berechnungsweg mit den zugrunde liegenden Daten befindet sich in Abschnitt A.2. Dabei ist für viele Bauteile der pessimistische Fall des vollständigen Funktionsausfalls eines Bausteins betrachtet, so dass die Berechnung dem *Worst Case* entspricht.

	Bauelement / Baugruppe	**Anzahl**	**Aktivität**	**R(*T*)**
Außerhalb redundanter Bereiche	AT697E (LEON2-FT) Prozessor	1	25%	0,998890
	RT54SX72S FPGA	1	25%	0,999687
	SRAM RH	10	25%	0,993756
	PROM RH	1	0,07%	0,999665
	Quarz-Oszillator	1	25%	0,999733
	Treiber	4	25%	0,998876
	Schmitt-Trigger	1	25%	0,999986
	MOSFET-Transistor	1	25%	0,999208
	PNP-Transistor	1	25%	0,999965
	Dioden	1	25%	0,999998
	Widerstände	15	25%	0,996306
	Keramik-Kondensatoren	86	25%	0,999033
	Tantal-Kondensatoren	18	25%	0,999197
	Stecker	2	25%	0,999990
	PCB	1	25%	0,998306
	EEPROM-Modul		0,07%	0,999998
	Raumsonden-Schnittstellen inkl. Stecker		25%	0,999919
	Gesamt			0,982636

Tabelle 5.3: Zuverlässigkeit nach T = 5 Jahren mit 6 h / Tag Betrieb und ϑ_A = 50 °C (Berechnungsweg siehe Abschnitt A.2)

5.2.3 Einfluss strahlungsbedingter Fehler

Die Berechnung der Strahlungsempfindlichkeit und damit die Bestimmung der zu erwartenden Rate für statische Fehler erfolgt exemplarisch für die Strahlungsbedingungen der ESA-Missionen Venus Express und ExoMars sowie für eine Mission im polaren *Low Earth Orbit* (LEO) als Beispiel für einen Planeten mit Magnetfeld. Es werden *Heavy Ions*-induzierte SEUs betrachtet, die, verglichen zu den Protonen-induzierten SEUs, in den gewählten Strahlungsumgebungen dominieren. In Tabelle 5.4 sind die entsprechenden Fehlerraten aufgeführt, in Tabelle 5.5 zusammengefasst und dort zusammen mit der Verfügbarkeit dargestellt. Bei der Fehlerberechnung gehen *Solar Flare*-Bedingungen in Form eines *Peak 5 Min* mit einer Wahrscheinlichkeit von 0,25 % ein und es wird wie bei der Bestimmung der Zuverlässigkeit mit einer aktiven Zeit von 6 h pro Tag gerechnet. Zur Verfügbarkeitsbestimmung wurde eine Wiederherstellungszeit von 10 s nach Auftreten eines Fehlers angesetzt. Zu beachten ist, dass die berechneten Ergebnisse häufig auf Angaben der Bauteilhersteller basieren, die für den ungünstigsten Fall gelten. Daher kann es für die verschiedenen Umgebungen zu ähnlichen Werten kommen (z. B. Fehlerrate des AT697E (LEON2-FT) Prozessors). Die Praxis zeigt, dass die tatsächlich auftretenden strahlungsbedingten Bauteil-Fehlerraten ein bis zwei Größenordnungen niedriger liegen.

	Fehlerrate [1/Tag]		
Baugruppe	**Venus Express**	**ExoMars**	**polarer LEO**
AT697E (LEON2-FT) Prozessor	2,51E-06	2,51E-06	2,51E-06
RT54SX72S FPGA	3,14E-10	1,15E-11	4,12E-11
SRAM	1,07E-06	2,31E-07	9,48E-07
PROM	3,02E-15	8,34E-16	3,52E-15
EEPROM	2,98E-14	1,34E-18	2,58E-15

Tabelle 5.4: Fehlerrate Baugruppen, aktive Zeit: 6 h / Tag

	Venus Express	**ExoMars**	**polarer LEO**
Fehlerrate [1/Tag]	3,58E-06	2,74E-06	3,45E-06
Fehlerrate [1/Jahr]	1,31E-03	9,99E-04	1,26E-03
Verfügbarkeit	0,99999999	0,99999999	0,99999999

Tabelle 5.5: Gesamtfehlerrate und Verfügbarkeit, Wiederherstellungszeit 10 s

5.2.4 Leistungsbedarf

Der zu erwartende Leistungsbedarf ist in Tabelle 5.6 für den Ruhezustand sowie für den Vollbetrieb (Akquisition, Speicherung und Kompression) dargestellt. Durch Implementierung eines Taktteilers kann die zur Verfügung gestellte Performanz und somit der notwendige

Leistungsbedarf der Situation angepasst werden. Dieses kann für Missionen mit Landeeinheiten interessant sein.

Leistungsprofil	Leistungsbedarf [W]
Ruhezustand	1,2
Vollbetrieb	1,4

Tabelle 5.6: Leistungsbedarf

5.2.5 Platzbedarf, Volumen und Masse

Die Vorgaben für die Platinengröße und das Format einer DPU ergeben sich i. Allg. aus den Vorgaben für den mechanischen Aufbau des gesamten Instruments. Die Ausführung der DPU kann abhängig von der jeweiligen Mission z. B. auf einer starren Platine oder als mehrfach gefaltete *Starrflex*-Platine erfolgen. Für die Ausführung auf starrer Platine bei doppelseitiger Bestückung mit 14 Ebenen ergeben sich für den Platzbedarf, das Volumen und die Masse die in Tabelle 5.7 aufgeführten Werte. Bei anderen Bauformen ist mit einer leichten Erhöhung dieser Werte zu rechnen.

Platzbedarf [cm^2]	Volumen [cm^3]	Masse [g]
160	205	205

Tabelle 5.7: Strahlungsfeste DPU, Bestückungshöhe 7 bzw. 4 mm

5.3 Variation unter Verwendung eines Massenspeichers

Das Fehlen eines größeren Speichers in Abschnitt 5.1 und 5.2 zur Speicherung von Bildern erlaubt es nicht, mehrere Bilder im Speicher zu halten um sie anschließend zu bearbeiten. Die Bearbeitung, z. B. die Kompression, erfolgt zeitlich direkt nach Akquisition eines jeden Bildes, um nach Übertragung der komprimierten Bilddaten an den Hauptspeicher der Raumsonde wieder Speicherplatz für ein neues Bild zur Verfügung zu stellen. Es ist eine möglichst schnelle Kompression erwünscht, die idealerweise in der geforderten Aufnahmegeschwindigkeit durchgeführt werden kann (Online-Kompression).

Die Aufnahmegeschwindigkeit, d. h. die maximal mögliche Geschwindigkeit schneller Bildabfolgen wird bei Nichtverfügbarkeit eines Massenspeichers u. a. beschränkt durch

- Kompressionsgeschwindigkeit und
- Dauer der Datenübertragung zur Raumsondenelektronik.

Die Nichtverfügbarkeit schnellen Massenspeichers führt zu einer sehr starken Einschränkung beim Einsatz der DPU und des Instruments. Hat die DPU Zugriff auf externen Massenspeicher

(z. B. Massenspeicher der Raumsonde), so können Bilder offline komprimiert werden. Der Zugriff auf einen externen Speicher unterliegt vielen Randbedingungen und die Möglichkeit des Zugriffs ist nicht immer gegeben.

Da die Verwendung eines DPU-externen Massenspeichers nicht immer möglich bzw. wenig attraktiv ist, kann die Implementierung eines DPU-internen Massenspeichers eine sinnvolle Ergänzung sein. Viele Instrumente beinhalten daher einen internen Massenspeicher (z. B. Venus Express / VMC, DAWN / DFC, Rosetta / OSIRIS). Für Massenspeicher werden aufgrund ihrer hohen Speicherdichte vielfach dynamische Speicher (z. B. SDRAM) verwendet. Sie basieren auf kommerziellen Produkten und sind nicht speziell für den Einsatz im Weltraum als strahlungsfeste Bausteine gefertigt. Dennoch können sie durch geeignete Maßnahmen für einen Einsatz innerhalb eines strahlungstoleranten Designs für Raumfahrtanwendungen qualifiziert werden. Bei Verwendung DPU-internen Massenspeichers werden die gestiegenen Möglichkeiten im Betrieb, wie Offline-Kompression oder Addition mehrerer Bilder, auf Kosten eines rein strahlungsfesten Designs erzielt.

Der in Bild 5.1 gestrichelt dargestellte Bereich stellt die Erweiterung der strahlungsfesten DPU um einen Massenspeicher dar. Die Einführung des Massenspeichers erfolgt mit Maßnahmen wie Fehlerkorrektur und Schutz vor *Latchup*. Die Speicherschnittstelle des Prozessors ermöglicht zwar die Erkennung eines Fehlers unter Zuhilfenahme einer Checksumme, erlaubt jedoch nicht die Korrektur des entsprechenden Fehlers. Daher ist für den Massenspeicher eine *Reed Solomon*-Hardware-Fehlerkorrektur im RT54SX-S FPGA implementiert, wodurch sich die Zugriffszeiten auf den Speicher etwas verlängern. Zur Akquisition von Bilddaten bei geringer CPU-Belastung ist eine *Direct Memory Access* (DMA)-Logik mit einer Steuerung für SDRAM-Bausteine integriert. Hierdurch werden die im Zwischenspeicher liegenden Bilddaten direkt in dem vorgesehenen Bereich im Massenspeicher abgelegt. Ein *Latchup*-Schutzschalter mit zugehöriger Logik im strahlungsfesten FPGA überwacht die Ströme der kommerziellen SDRAM-Bausteine und trennt diese bei Detektierung eines Überstroms von der Versorgungsspannung ab. Gleichzeitig werden *High*-Pegel auf Signalleitungen zu dem abgeschalteten Bereich unterbunden, um das selbstständige Löschen des *Latchup* zu ermöglichen. Mit Verwendung eines Massenspeichers kann gleichzeitig die Arbeitsspeichergröße reduziert werden.

5.3.1 Performanz

Durch die Einführung des Massenspeichers ergeben sich keine Änderungen in der durchschnittlichen Performanz. Mit der Speicherungsmöglichkeit im Massenspeicher kann allerdings eine begrenzte Anzahl von Bildern mit der gewünschten Rate von einem $1\text{k} \times 1\text{k} \times 14$ Bit-Bild je 1-2 s akquiriert werden. Nach Beendigung dieser Bildsequenz werden die Bilder anschließend jedoch mit niedrigem Durchsatz von durchschnittlich einem Bild je 21 s (verlustlos) offline komprimiert.

5.3.2 Zuverlässigkeit

Mit der Verwendung kommerzieller Speicherbausteine bleibt die Gesamtzuverlässigkeit sehr ähnlich zum strahlungsfesten Ansatz ohne Massenspeicher. Zwar reduziert der *Latchup*-Schutzschalter (im Wesentlichen dessen Widerstände) die Zuverlässigkeit, mit der reduzierten Anzahl an SRAM-Bausteinen wird dieses jedoch ausgeglichen (siehe Tabelle 5.8). Die geringe Bauteilzuverlässigkeit der SDRAM-Bausteine fällt aufgrund der implementierten Fehlerkorrektur mit einer 1-aus-6 Redundanz nicht ins Gewicht.

	Bauelement / Baugruppe	**Anzahl**	**Aktivität**	**R(*T*)**
Außerhalb redundanter Bereiche	AT697E (LEON2-FT) Prozessor	1	25%	0,998890
	RT54SX72S FPGA	1	25%	0,999687
	SRAM RH	6	25%	0,996249
	PROM RH	1	0,07%	0,999665
	Quarz-Oszillator	1	25%	0,999733
	Treiber	4	25%	0,998876
	Schmitt-Trigger	1	25%	0,999986
	Komparator	1	25%	0,999947
	MOSFET-Transistor	3	25%	0,997627
	PNP-Transistor	2	25%	0,999929
	Dioden	1	25%	0,999998
	Widerstände	26	25%	0,994096
	Keramik-Kondensatoren	87	25%	0,999010
	Tantal-Kondensatoren	18	25%	0,999197
	Stecker	2	25%	0,999990
	PCB	1	25%	0,998210
	EEPROM-Modul		0,07%	0,999998
	Raumsonden-Schnittstellen inkl. Stecker		25%	0,999919
	Massenspeicher		25%	0,999957
	Gesamt			0,981114

Tabelle 5.8: Zuverlässigkeit nach T = 5 Jahren mit 6 h / Tag Betrieb und ϑ_A = 50 °C (Berechnungsweg siehe Abschnitt A.2)

5.3.3 Einfluss strahlungsbedingter Fehler

Die erwarteten Fehlerraten sowie die Verfügbarkeit sind in Tabelle 5.9 und Tabelle 5.10 dargestellt. Die Fehlerrate im kommerziellen Massenspeicher ist als Datenfehler zu werten und separat angegeben. Es ist insbesondere ersichtlich, dass diese Fehlerrate aufgrund der Kombination Fehler korrigierender Code und Schrubben des Speichers (Schrubbintervall: 1 h) vergleichsweise niedrig liegt.

	Fehlerrate [1/Tag]		
Baugruppe	**Venus Express**	**ExoMars**	**polarer LEO**
AT697E (LEON2-FT) Prozessor	2,51E-06	2,51E-06	2,51E-06
RT54SX72S FPGA	3,14E-10	1,15E-11	4,12E-11
SRAM	5,35E-07	1,16E-07	4,74E-07
PROM	3,02E-15	8,34E-16	3,52E-15
EEPROM	2,98E-14	1,34E-18	2,58E-15

Tabelle 5.9: Fehlerrate Baugruppen

	Venus Express	**ExoMars**	**polarer LEO**
Fehlerrate [1/Tag]	3,04E-06	2,62E-06	2,98E-06
Fehlerrate [1/Jahr]	1,11E-03	9,57E-04	1,09E-03
Fehlerrate Massenspeicher [1/Jahr]	7,98E-07	1,55E-10	5,54E-08
Verfügbarkeit	0,99999999	0,99999999	0,99999999

Tabelle 5.10: Gesamtfehlerrate und Verfügbarkeit, Wiederherstellungszeit 10 s

5.3.4 Leistungsbedarf

Der Leistungsbedarf im Vollbetrieb erhöht sich mit Implementierung des Massenspeichers erheblich (siehe Tabelle 5.11). Dagegen ist der Leistungsbedarf im Ruhezustand mit Einführung des Massenspeichers nur leicht verändert.

Leistungsprofil	**Leistungsbedarf [W]**
Ruhezustand	1,3
Vollbetrieb	2,1

Tabelle 5.11: Leistungsbedarf

5.3.5 Platzbedarf, Volumen und Masse

Da mit Einführung des Massenspeichers die Arbeitsspeichergröße, bestehend aus strahlungsfesten Bauteilen, reduziert werden kann, ergeben sich im Vergleich zu Abschnitt 5.2 nur leichte Änderungen beim Platzbedarf, Volumen und bei der Masse (siehe Tabelle 5.12).

Platzbedarf [cm^2]	**Volumen [cm^3]**	**Masse [g]**
165	211	195

Tabelle 5.12: Strahlungsfeste DPU mit Massenspeicher, Bestückungshöhe 7 bzw. 4 mm

6 Strahlungstolerante DPU

In Kapitel 5 wurde auf die Einschränkungen beim Betrieb des Instruments mit strahlungsfester DPU hingewiesen, die sich aus der Nichtverfügbarkeit eines DPU-internen Massenspeichers ergeben. Mit Einführung eines Massenspeichers in Abschnitt 5.3 wird statt einer vollständig strahlungsfesten DPU ein strahlungstoleranter Bestandteil in Form des Massenspeichers eingeführt. Werden strahlungsfeste Baugruppen zunehmend in strahlungstolerante Baugruppen überführt, ergibt sich eine strahlungstolerante DPU. Das Hauptmerkmal dieser DPU ist die zunehmende Einführung von Fehlerkorrekturmaßnahmen, wie sie bei einem strahlungsfesten Entwurf nicht erforderlich sind. Hiermit kann ein fehlertolerantes System (Toleranz gegenüber strahlungsbedingten Fehlern) aufgebaut werden. Ziel ist es, eine hohe Performanz bei niedrigem Ressourcenbedarf (Energie, Größe und Masse) zu erreichen und dieses mit tolerierbaren Werten für die Zuverlässigkeit und die Verfügbarkeit, aber auch für die Datenintegrität.

Als Gegenentwurf zur strahlungsfesten DPU aus Kapitel 5 wird daher in diesem Kapitel eine strahlungstolerante DPU auf Basis eines SRAM-basierten strahlungstoleranten RT-Virtex-II FPGA vorgestellt. Um zahlreiche Einschränkungen im Betrieb zu vermeiden, beinhaltet die strahlungstolerante DPU einen Massenspeicher. Es werden verschiedene Kennwerte analysiert, mit denen ein Vergleich zu den Werten in Kapitel 5 ermöglicht wird.

6.1 Aufbau mit Soft CPU im RT-Virtex-II FPGA

Mit der Verfügbarkeit größerer FPGAs für Weltraumanwendungen kann eine hohe Integration beim Aufbau eines Instrumentenrechners erfolgen. Bei der vorgestellten DPU wird strahlungstoleranter Arbeits- und Massenspeicher eingesetzt sowie eine Kombination aus strahlungstolerantem EEPROM und Flash-Speicher für die nichtflüchtigen Daten. Ein Beispiel für die Implementierung ist die DPU für das Instrument Venus Express / VMC. Bild 6.1 zeigt die Implementierung einer strahlungstoleranten DPU für ein Kamerainstrument unter Verwendung eines XQR2V3000 FPGA und damit des mittelgroßen Bausteins der RT-Virtex-II Familie der Firma Xilinx. Kernstück ist dieser FPGA, der neben Sensorlogiken und Fehlerkorrektur einen 32-Bit Prozessor beinhaltet. Diese Implementierung verfolgt den Ansatz eines *System-On-Chip* (SoC) Designs, in dem möglichst viele Module zusammen mit dem Prozessor innerhalb eines Bausteins integriert sind. Beim verwendeten Prozessor handelt es sich um den SPARC-V8-kompatiblen LEON2-Prozessor als IP Core. Hiermit ist ein direkter Vergleich zur DPU mit Ausführung dieses Prozessors als strahlungsfester Baustein (AT697E) (siehe Kapitel 5) möglich. Der IP Core hat hohe Konfigurationsmöglichkeiten und kann entsprechend der Anwendung erstellt werden. Die *Cache*-Größen sind wie beim AT697E (LEON2-FT) Prozessor auf 32 kByte (Befehle) und 16 kByte (Daten) gesetzt. Im Gegensatz zum AT697E (LEON2-FT) Prozessor ist keine FPU eingebunden, da deren Ressourcenanforderungen sehr hoch sind. FPU-Operationen, die emuliert werden, sollten selten ausgeführt werden oder nach Möglichkeit durch Fixpunkt-Operationen ersetzt werden.

Sowohl der Arbeits- als auch der Massenspeicher erfordern den Einsatz einer Fehlerkorrektur in Hardware. Da sich die implementierte *Reed Solomon*-Fehlerkorrektur in demselben Baustein befindet wie der Prozessor, reduzieren sich die Laufzeiten zwischen beiden Modulen auf chipinterne Laufzeiten. Zusätzliche Laufzeiten von Ein- und Ausgangspuffern von Bausteinen und Kapazitäten auf PCB werden damit umgangen. Die *Refresh*-Überwachung für das SDRAM ist als TMR-Design im RT-Virtex-II FPGA ausgeführt. Für den nichtflüchtigen, veränderbaren Speicher (EEPROM und Flash) erfolgen die Fehler korrigierenden Maßnahmen über die Software, da hier nur ein geringer Durchsatz erforderlich ist. Dieser Speicher verfügt über Schalter. Zum Schutz der Eingangsschutzdioden unterbindet der FPGA *High*-Pegel an dem ausgeschalteten NVRAM-Speicher.

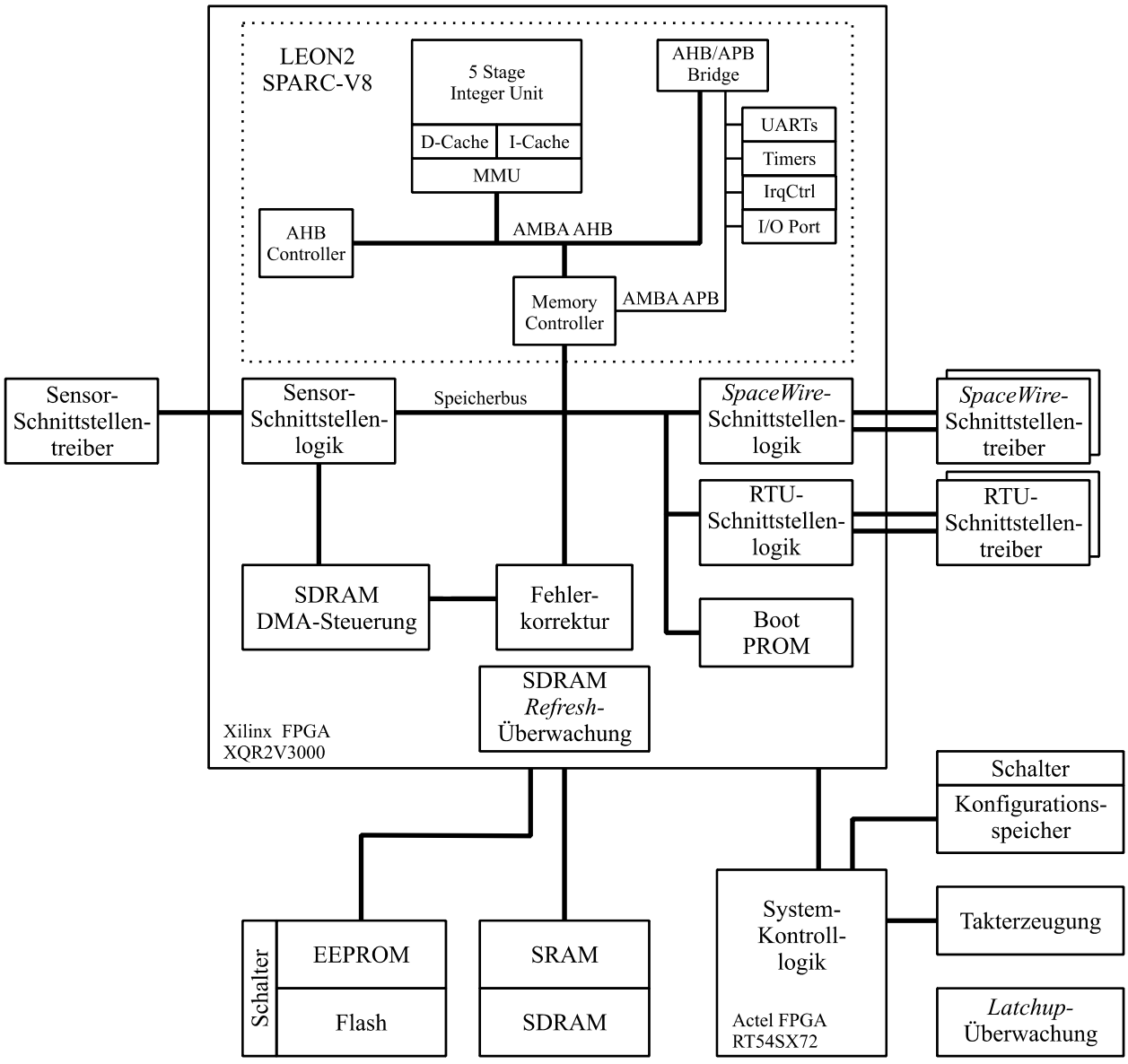

Bild 6.1: Aufbau der strahlungstoleranten DPU mit SRAM-basiertem FPGA und Soft CPU

Die Peripherielogiken (Sensor- und Raumsonden-Schnittstellen) können vollständig im FPGA implementiert werden. Zur Pufferung ihrer Daten wird FPGA-interner SRAM-Speicher verwendet und externer Zwischenspeicher wie in Kapitel 5 ist nicht notwendig. Die Daten vom Sensor werden über einen internen Zwischenspeicher mittels DMA-Transfer direkt in den vorgesehenen Bereich im Massenspeicher geschrieben. Die Bedienung der Raumsonden-Schnittstellen erfolgt unter Zuhilfenahme des Prozessors. Dieses ist darin begründet, dass die Daten zum einen teilweise zeitnah von dem Prozessor verarbeitet werden und zum anderen eine niedrigere Datenrate erforderlich ist als bei der Sensor-Schnittstelle.

Wird ein strahlungsempfindlicher SRAM-basierter RT-Virtex-II FPGA eingesetzt, so sind zusätzliche Maßnahmen für dessen Betrieb notwendig (siehe Abschnitt 4.5). Es muss z. B. die Konfiguration in den FPGA geladen und der korrekte Zustand des FPGA wegen des erhöhten Auftretens strahlungsbedingter Fehler regelmäßig überprüft werden (*Readback*-Verfahren). Zur Durchführung solcher Routinen wird ein strahlungsfester RT54SX-S FPGA der Firma Actel eingesetzt, der die Systemkontrolle übernimmt. Die Konfigurationsdaten werden in zwei Konfigurations-PROMs gehalten (*Bit*- und *Mask*-Datei), die in der Entwicklungsphase durch Flash-basierte Derivate ersetzt werden [Xil04c]. Die Konfigurations-PROMs sind mit Schaltern versehen und können bei Bedarf von der Versorgungsspannung abgetrennt werden. Dieses ist jedoch nur sinnvoll, wenn die *Readback*-Rate gering ist. Der *Watchdog*-Zähler, statische Register sowie die *Latchup*-Logik sind im strahlungsfesten RT54SX-S FPGA implementiert. Der *Latchup*-Schutzschalter selbst wirkt auf die kommerziellen SRAM, SDRAM und Flash Speicher. Das Design ist so ausgeführt, dass der *Bootloader* im FPGA-internen Speicher integriert ist. Die Anfangswerte der internen Speicher (*Block RAM*) sind im Konfigurations-PROM gespeichert und stehen dem System damit nach dem Schreiben der Konfiguration in den RT-Virtex-II FPGA zur Verfügung. Daher kann unter Nutzung des Konfigurations-PROMs auf ein dediziertes externes PROM für den *Bootloader* verzichtet werden. Wird das Design im XQR2V3000 FPGA der RT-Virtex-II Familie implementiert, so erzielt man für den Prozessor eine maximale Taktfrequenz von 50 MHz.

Die Verwendung eines großen FPGA ermöglicht es, eine Standard-CPU für allgemeine Aufgaben zusammen mit dedizierten Modulen für die effiziente Ausführung spezieller Aufgaben zu kombinieren. Da innerhalb des verwendeten FPGA noch viel Reserve zur Implementierung spezieller Module besteht, stellt die gezeigte strahlungstolerante DPU eine gute Grundlage für hohe Performanzsteigerungen dar.

6.2 Kennwerte

Zur Beurteilung der Leistungsfähigkeit der strahlungstoleranten DPU ist die Bestimmung von verschiedenen Kennwerten erforderlich.

6.2.1 Performanz

Die Performanz, gemessen anhand der JPEG2000-Kompression, wird mit der Software und den Testbildern bestimmt, wie sie in Abschnitt 5.2 beschrieben sind. Die Ergebnisse sind in Tabelle 6.1, Tabelle 6.2 sowie in Bild 6.2 ersichtlich und es zeigt sich die Reduzierung des Durchsatzes auf ca. 50 % gegenüber Kapitel 5, die aufgrund des CPU-Taktes zu erwarten war. Bei einem Bild der Größe 1k × 1k × 14 Bit sind durchschnittlich 42 s zur verlustlosen Kompression erforderlich. Auch hier kann eine gewünschte Akquisitionsrate von einem Bild je 1-2 s nur für eine begrenzte Zeit durch Zwischenspeicherung im Massenspeicher mit späterer, zeitintensiver Kompression erzielt werden.

Testbild	Kompressionsfaktor	Ausführungszeit [ms]	Durchsatz [MBit/s]
Venus	1,61	5.320	0,39
Mars Full	1,59	8.730	0,36
Mars DaoVallis	1,37	9.410	0,33
Mars Surface	1,28	2.540	0,31

Tabelle 6.1: Verlustlose Kompression

Kompressionsfaktor *c*	verlustlos	2	4	6	8	10	12	14	16
Durchsatz [MBit/s]	0,35	0,44	0,63	0,73	0,80	0,85	0,88	0,90	0,92

Tabelle 6.2: Durchschnittlicher Durchsatz der Kompression

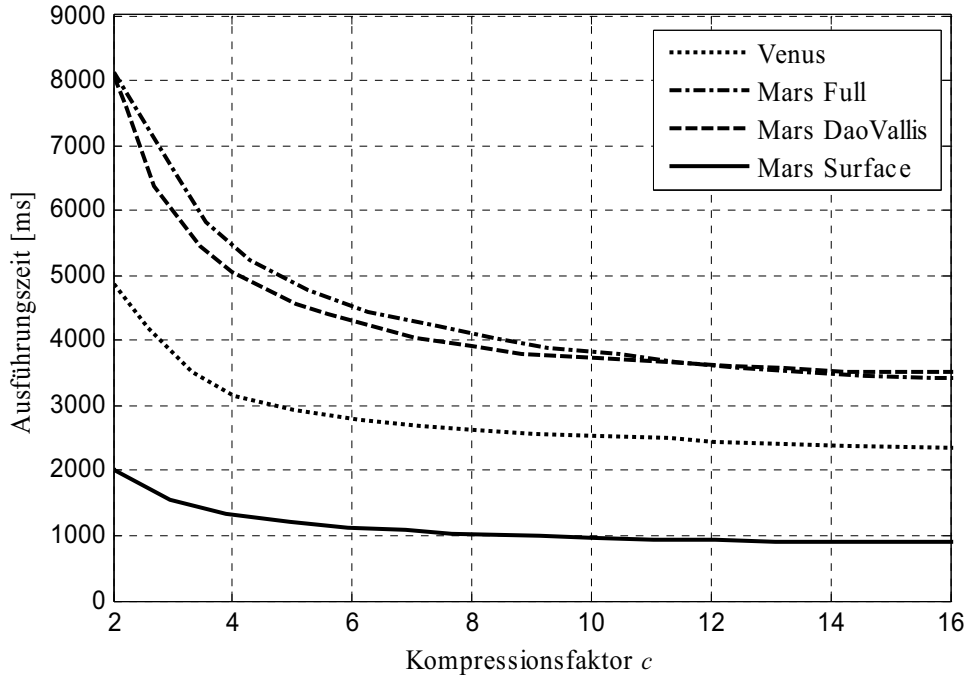

Bild 6.2: Ausführungszeit zur Kompression der Testbilder

6.2.2 FPGA Ressourcenbedarf

Der Ressourcenbedarf innerhalb der Kernkomponente RT-Virtex-II FPGA ist in Tabelle 6.3 aufgelistet und u. a. zur Bestimmung der Strahlungsfestigkeit erforderlich.

Modul	***Slices***		**Benutzer-Flipflops**		**18 Bit × 18 Bit Multipl.**		**18 kBit *Block RAMs***	
	abs.	**%**	**abs.**	**%**	**abs.**	**%**	**abs.**	**%**
LEON2	3.821	26,7	2.101	7,3	4	4,2	22	22,9
NVRAM-Schnittstelle	60	0,4	120	0,4	-	-	-	-
Fehlerkorrektur	133	0,9	260	0,9	-	-	-	-
Sensor-Schnittstelle	108	0,7	203	0,7	-	-	2	2,1
SDRAM DMA-Steuerung	69	0,5	77	0,3	-	-	-	-
SDRAM-Überwachung	100	0,7	120	0,4	-	-	-	-
RTU-Schnittstelle	134	0,9	254	0,9	-	-	1	1,0
SpaceWire-Schnittstelle	361	2,5	491	1,7	-	-	1	1,0
Boot-PROM	15	0,1	20	0,1	-	-	4	4,2
Summe	4.801	33,5	3.646	12,7	4	4,2	30	31,3

Tabelle 6.3: Ressourcenbedarf innerhalb des XQR2V3000 FPGA

6.2.3 Zuverlässigkeit

Tabelle 6.4 zeigt die Zuverlässigkeit der strahlungstoleranten DPU für den *Worst Case* (siehe Abschnitt A.2). Die Zuverlässigkeit der Widerstände hat dabei einen hohen Einfluss auf das gesamte System. Die geringere Bauteilzuverlässigkeit bei den Speicherbausteinen wird durch Redundanz ausgeglichen. Mit der Verwendung strahlungstoleranter statt strahlungsfester SRAM-Bausteine steigt die Zuverlässigkeit sogar leicht an. Dieses ist darin begründet, dass die strahlungsfesten SRAM-Bausteine, wie in Kapitel 5 gezeigt wurde, aufgrund sehr geringer SEU-Empfindlichkeit ohne Fehlerkorrektur betrieben werden können und dementsprechend keine Redundanz vorhanden ist. Dagegen kann mit der Fehlerkorrektur der strahlungstoleranten SRAM-Bausteine der Ausfall eines Bausteins abgefangen werden. Dieses führt zu einer erhöhten Zuverlässigkeit.

	Bauelement / Baugruppe	Anzahl	Aktivität	R(*T*)
Außerhalb redundanter Bereiche	XQR2V3000 FPGA	1	25%	0,999730
	RT54SX72S FPGA	1	25%	0,999687
	Konfig. PROM XQR17V16RH	2	25%	0,999516
	Quarz-Oszillator	1	25%	0,999733
	Treiber	1	25%	0,999719
	Schmitt-Trigger	1	25%	0,999986
	Komparator	1	25%	0,999947
	MOSFET-Transistor	4	25%	0,996837
	PNP-Transistor	3	25%	0,999894
	Dioden	1	25%	0,999998
	Widerstände	29	25%	0,993280
	Keramik-Kondensatoren	84	25%	0,999044
	Tantal-Kondensatoren	18	25%	0,999197
	Stecker	2	25%	0,999990
	PCB	1	25%	0,998063
	SRAM-Modul		25%	0,999948
	EEPROM-Modul		0,07%	0,999999
	Flash Modul		0,07%	0,999999
	Raumsonden-Schnittstellen inkl. Stecker		25%	0,999895
	Massenspeicher		25%	0,999970
	Gesamt			0,984535

Tabelle 6.4: Zuverlässigkeit nach T = 5 Jahren mit 6 h / Tag Betrieb und ϑ_A = 50 °C (Berechnungsweg siehe Abschnitt A.2)

6.2.4 Einfluss strahlungsbedingter Fehler

Der RT-Virtex-II FPGA mit seinen niedrigen LET-Schwellwerten bestimmt im Wesentlichen die Strahlungsfestigkeit der DPU. Nach Abschnitt 4.4 ist sie stark von den verwendeten Ressourcen abhängig, so dass Tabelle 6.3 zur Berechnung herangezogen wird. Die Datenmenge zur Beschreibung der Verdrahtung und Konfiguration wird darin als proportional zur jeweils verwendeten Ressourcenmenge angenommen. Für die Missionen Venus Express und ExoMars sowie für eine Mission im polaren LEO ergeben sich die in Tabelle 6.5 berechneten Raten für statische Fehler im RT-Virtex-II FPGA. Die *Solar Flare*-Fehlerrate (*Peak 5 Min*) geht mit einer Wahrscheinlichkeit von 0,25 % ein. Gegenüber dem RT-Virtex-II FPGA sind die Fehlerraten der übrigen Bausteine vernachlässigbar (siehe Tabelle 6.6) und die Fehlerrate des RT-Virtex-II FPGA kann als Fehlerrate für die gesamte DPU angesehen werden. Innerhalb des RT-Virtex-II FPGA ist bei der Ermittlung der Fehlerrate der LEON2-Prozessor aufgrund seines hohen Ressourcenbedarfs bestimmend. Beim LEON2-Prozessor wiederum sind mehr als die Hälfte der

Fehler durch SEUs im *Block RAM* bedingt (im Wesentlichen *Cache*) und können teilweise nur Datenfehler sein. Im pessimistischen Fall jedoch ist bei jedem Fehler mit einem Systemzusammenbruch und der notwendigen Wiederherstellung zur bestmöglichen Wiederherstellung des Systemzustands vor dem Fehler zu rechnen. Bei der DPU für Venus Express / VMC, die einen strahlungstoleranten Ansatz mit Verwendung SRAM-basierter FPGAs (RT-Virtex-I) beinhaltet, ist nach bisher ca. 1.000 h Betrieb noch kein Systemzusammenbruch aufgetreten.

In Tabelle 6.7 sind die Fehlerraten und die Verfügbarkeiten zusammengefasst. Aufgrund der SDRAM *Refresh*-Überwachung mit TMR ist ein größerer Datenverlust im Massenspeicher um etwa den Faktor 100 unwahrscheinlicher. Dieser Wert wird durch mögliche Fehler im Entscheider der Überwachung bestimmt.

	Fehlerrate [1/Tag]		
Baugruppe	**Venus Express**	**ExoMars**	**polarer LEO**
LEON2	6,14E-03	1,67E-03	1,73E-03
NVRAM-Schnittstelle	3,63E-05	9,97E-06	1,03E-05
Fehlerkorrektur	8,04E-05	2,21E-05	2,27E-05
Sensor-Schnittstelle	4,16E-04	1,13E-04	1,17E-04
SDRAM DMA-Steuerung	4,14E-05	1,14E-05	1,17E-05
SDRAM-Überwachung	1,79E-05	4,91E-06	5,05E-06
RTU-Schnittstelle	2,56E-04	6,99E-05	7,22E-05
SpaceWire-Schnittstelle	3,92E-04	1,07E-04	1,11E-04
Boot-PROM	3,29E-08	8,95E-09	9,24E-09
Zwischensumme	7,38E-03	2,01E-03	2,08E-03
SEFI	5,20E-07	1,30E-07	1,40E-07

Tabelle 6.5: Fehlerraten für den RT-Virtex-II FPGA

	Baugruppen Fehlerrate [1/Tag]		
Baugruppe	**Venus Express**	**ExoMars**	**polarer LEO**
RT54SX72S FPGA	3,14E-10	1,15E-11	4,12E-11
Konfigurations-PROM	6,96E-14	6,96E-14	6,96E-14
SRAM	1,06E-09	2,45E-13	8,02E-11
SDRAM	5,47E-10	1,06E-13	4,22E-11
EEPROM	1,43E-07	1,94E-08	5,48E-08
Flash	3,78E-08	6,13E-09	1,38E-08

Tabelle 6.6: Fehlerraten beim strahlungstoleranten Design

	Venus Express	ExoMars	polarer LEO
Fehlerrate [1/Tag]	7,38E-03	2,01E-03	2,08E-03
Fehlerrate [1/Jahr]	2,69E+00	7,35E-01	7,58E-01
Fehlerrate Massenspeicher [1/Jahr]	2,61E-02	7,16E-03	7,38E-03
Verfügbarkeit	0,99999915	0,99999977	0,99999976

Tabelle 6.7: Gesamtfehlerrate und Verfügbarkeit, Wiederherstellungszeit 10 s

6.2.5 Leistungsbedarf

Der zu erwartende Leistungsbedarf ist für die Profile Ruhezustand und Vollbetrieb in Tabelle 6.8 dargestellt. Mit der Nutzung des SRAM-basierten FPGA erhöht sich der Leistungsbedarf verglichen zur strahlungsfesten Implementierung.

Leistungsprofil	Leistungsbedarf [W]
Ruhezustand	1,7
Vollbetrieb	2,4

Tabelle 6.8: Leistungsbedarf

6.2.6 Platzbedarf, Volumen und Masse

Mit Einführung des RT-Virtex-II FPGA sowie weiterer strahlungstoleranter Baugruppen mit zunehmendem Anteil an Bausteinen mit Plastikgehäuse kann die DPU stärker integriert werden als beim strahlungsfesten Entwurf. Für die Ausführung als starre Platine bei doppelseitiger Bestückung mit 14 Ebenen ergeben sich für den Platzbedarf, das Volumen und die Masse folgende Werte:

Platzbedarf [cm^2]	Volumen [cm^3]	Masse [g]
110	119	120

Tabelle 6.9: Strahlungstolerante DPU, Bestückungshöhe 6 bzw. 3 mm

7 JPEG2000 Hardwarebeschleunigung

Verglichen mit kommerziellen Prozessoren bieten Prozessoren für Raumfahrtanwendungen eine sehr beschränkte Performanz. Bei DPUs für Kamerainstrumente macht sich dieses in der Ausführung von qualitativ hochwertigen Kompressionsalgorithmen in Software bemerkbar. Die Ausführung nimmt aufgrund ihrer hohen Anforderung an die Rechenleistung viel Zeit in Anspruch. Die in Kapitel 5 und 6 beschriebenen DPUs mit Kompression in Software benötigen am Beispiel der verlustlosen Kompression durchschnittlich 21 s bzw. 42 s zur Bearbeitung eines Bildes der Größe 1k × 1k × 14 Bit. Durch diese Kompressionszeit kann die erwünschte Bildrate von einem Bild je 1-2 s bei weitem nicht erzielt werden, mit der Folge eines stark reduzierten wissenschaftlichen Nutzens des Instruments. Idealerweise sollte die Kompression mit der Bilderzeugung Schritt halten können (Online-Kompression). Für typische Kamerainstrumente würde dieses eine Kompressionsgeschwindigkeit von einem Bild je 1-2 s und damit ein vielfaches der in Kapitel 5 und 6 erzielten Kompressionsgeschwindigkeiten bedeuten. Eine deutliche Beschleunigung der Kompression kann erreicht werden, indem der Kompressionsalgorithmus vollständig oder teilweise von der Software in die Hardware ausgelagert wird, um dort innerhalb eines Co-Prozessors schneller und energieeffizienter durchgeführt werden zu können. In diesem Kapitel wird daher die Implementierung des JPEG2000-Kompressionsalgorithmus in Hardware besprochen.

Die Ausführung einer Hardwarekompression auf einer Raumsonde ist z. B. in [Rab95] für das Instrument Huygens-Probe / DISR beschrieben. Die Kompression ist durch eine DCT-basierte Hardwarekompression auf Kosten hoher Ressourcenanforderungen beschleunigt (ältere Bausteine: mehrere Bausteine notwendig). Beim Aufbau eines Instrumentenrechners für kompakte Kameras darf eine hardwareunterstützte Kompression lediglich zu einem leichten Anstieg der Ressourcenanforderungen führen. Daher stellt sich die Frage, inwieweit die JPEG2000-Kompression als Beispiel für eine *Wavelet*-basierte Kompression durch verstärkte Hardwareimplementierung innerhalb eines kleinen Instrumentenrechners beschleunigt werden kann.

Die gegenüber der DCT-basierten Kompression hochwertigere DWT-basierte Kompression erfordert eine hohe Rechenleistung. DCT-basierte Verfahren für die Raumfahrt werden bereits eingesetzt, sei es als dedizierter weltraumtauglicher Chip oder als IP Core zum Einbinden in FPGAs und ASICs. Da DCT-basierte IP Cores weit weniger Ressourcen benötigen als *Wavelet*-basierte Verfahren, können sie auch in älteren und damit kleineren FPGAs effektiv eingesetzt werden. Hingegen ist die hochwertigere *Wavelet*-Kompression mit Hardwareunterstützung für die Anwendung innerhalb einer kleinen Instrumenten-DPU noch wenig verbreitet bzw. mit moderatem Ressourcenbedarf kaum vorhanden.

7.1 Dedizierter Chip für JPEG2000

Eine Möglichkeit, die JPEG2000-Kompression durch Hardware zu beschleunigen, ist der Einsatz eines dedizierten kommerziellen JPEG2000-Bausteins. Als Beispiel ist der ADV202 der Firma Analog Devices zu nennen [Ana05]. Im Wesentlichen für die Videoübertragung entwickelt, weist er einen hohen Durchsatz mit drei parallelen Codierungsstufen auf. Der ADV202 besitzt folgende Eigenschaften:

- Durchsatz: 130 Mbps bei 900 mW @150 MHz (max. Taktrate)
- Max. Bitanzahl pro Bildpunkt: 16 (verlustbehaftet), 14 (verlustlos)
- 9/7 und 5/3 *Wavelet*
- Keine Unterstützung für Pixel-basierte ROI-Codierung
- Bereits 500 mW Verbrauch im statischen Zustand
- Bauteilgröße: 13 mm × 13 mm

Während die Bauteilgröße praktisch keine Rolle spielt, sind vor allem der ständig vorhandene statische Leistungsbedarf von 500 mW und die Nichtverfügbarkeit einer Pixel-basierten ROI-Codierung problematisch. Zudem ist die Notwendigkeit für einen derart hohen Durchsatz bei kleinen Kameras i. Allg. nicht gegeben. Entscheidend ist jedoch, dass die Verwendung dieses kommerziellen Bausteins umfangreiche Qualifikationsmaßnahmen für den Einsatz im Weltraum erforderlich macht. Die Langlebigkeit dieses Bausteins ist nicht gewährleistet und die Erzeugung eines neuen *Die* erfordert erneute Qualifikationsmaßnahmen. Der Einsatz dieses Bausteins ist für Einzelstücke wie eine DPU für ein Kamerainstrument daher aus ökonomischer Sicht nicht machbar.

7.2 Kommerzielle JPEG2000 IP Cores

Eine weitere Möglichkeit der Implementierung einer JPEG2000-Beschleunigung ist es, kommerzielle JPEG2000 IP Cores in einem FPGA- oder ASIC-Baustein zu integrieren. Die Auswahl an kommerziellen JPEG2000 Cores ist stark eingegrenzt. Die drei bekanntesten JPEG2000 IP Cores sind in Tabelle 7.1 aufgeführt.

Die IP Cores sind häufig für Videoanwendungen ausgelegt und ermöglichen einen hohen Durchsatz. Dieses wird durch einen hohen Grad an interner Parallelität auf Kosten von Ressourcen erzielt. Die IP Cores sind konfigurierbar, u. a. ist die Performanz durch weitere parallele Stufen skalierbar. Jedoch benötigen die JPEG2000 Cores bereits in der kleinsten Konfiguration viele Ressourcen. Implementiert in einem RT-Virtex-II FPGA benötigen sie annähernd 200 % (Barco Core) der Größe des LEON2-Prozessors (siehe Abschnitt 6.2). Die notwendige Performanz für eine kleine Kamera liegt i. Allg. weit unter denen dieser IP Cores und es stellt sich die Frage, ob für die nicht notwendig hohe Performanz zusätzliche Ressourcen für ihre interne Parallelität zur Verfügung gestellt werden sollten. Eine mögliche

Implementierung, z. B. innerhalb der strahlungstoleranten DPU nach Kapitel 6, könnte das Einbinden des JPEG2000 Core in dem bereits vorhandenen FPGA sein. Dabei wäre der JPEG2000 Core als Co-Prozessor zusammen mit dem Prozessor in demselben FPGA integriert und dürfte nur moderate Ressourcenanforderungen aufweisen.

	Barco BA112JPEG20000 [Bar06]	**CAST JPEG2K_Encoder [Cas05]**	**Amphion CS6510 [Amp03]**
Max. Bittiefe verlustbehaftet	10	14	12
Max. Bittiefe verlustlos	12	12	12
Max. *Tile*-Größe	128 × 128	256 × 256	128 × 128
Max. *Codeblock*-Größe	32 × 32	64 × 64	32 × 32
Wavelet Typ	5/3 oder 9/7	5/3 oder 9/7	5/3 oder 9/7
Rate Control während *Tier-1*	nein	nein	nein
Durchsatz verlustlos *	ca. 80 MBit/s	ca. 80 MBit/s	ca. 128 MBit/s
Ressourcen bei minimaler Konfiguration *	6.153 *Slices* 45 *Block RAMs* (1 Kanal)	7.358 *Slices* 11 *Block RAMs* + *Tile Block RAM* (1 Kanal)	14.036 *Slices* *64 Block RAMs* (3 Kanäle)
Unterstützung von Pixel-basierter ROI-Codierung	nein	nein	nein

Tabelle 7.1: Übersicht kommerzieller JPEG2000 IP Cores
*** Virtex-II FPGA mit Geschwindigkeitsgrad 6**

An den drei gezeigten JPEG2000 Cores fällt auf, dass eine *Rate Control* erst in *Tier-2* eingreifen kann. Zwar werden *Rate Control* unterstützende Werte wie MSE schon während *Tier-1* erzeugt, diese haben jedoch noch keinen direkten Einfluss auf die tatsächlich zu verarbeitende Datenmenge im Entropie-Codierer. Erst mit schnellstmöglicher Auswertung dieser *Rate Control* unterstützenden Werte kann eine weitere Prozessierung der Daten unterbunden und noch Einfluss auf die Codierung während *Tier-1* genommen werden. Das wesentliche Argument gegen die Verwendung der vorgestellten Cores ist jedoch die geringe Bittiefe und die fehlende Unterstützung für Pixel-basierte ROI-Codierung. Lediglich der CAST JPEG2K_Encoder kann mit einer maximalen Anzahl von 14 Bit pro Bildpunkt arbeiten, dieses allerdings nur im verlustbehafteten Fall. Mit diesen IP Cores können somit die geforderten 14 Bit pro Bildpunkt nicht verarbeitet werden.

Die prinzipiellen Architekturmöglichkeiten für eine JPEG2000 Hardwarebeschleunigung zeigen die Notwendigkeit zur Entwicklung eines eigenen, skalierbaren JPEG2000 Core. Daher wurde

für die speziellen Anforderungen in einem Kamerainstrument der IDA JPEG2000 konzipiert, der im folgenden Abschnitt beschrieben ist.

7.3 Architektur des neu entwickelten IDA JPEG2000 Core

Ein dedizierter Chip wie der ADV202 entfällt aus den in Abschnitt 7.1 genannten Gründen für die Implementierung einer Hardwarebeschleunigung. Ebenso sind die kommerziell erwerbbaren JPEG2000 IP Cores aus Abschnitt 7.2 allein schon aufgrund ihrer zu geringen verarbeitbaren Bitanzahl pro Bildpunkt nicht für den Einsatz in einem Kamerainstrument mit den in Kapitel 1 aufgeführten typischen Kennwerten geeignet.

Die Entwicklung einer alternativen Implementierung der Hardwarebeschleunigung ist daher notwendig. Diese Implementierung zielt darauf ab, eine energieeffiziente Standardimplementierung des JPEG2000-Algorithmus mit Besonderheiten wie z. B. anpassbarer Bitanzahl pro Bildpunkt und moderatem Ressourcenbedarf zu beinhalten.

Für die Anwendung in einem Kamerainstrument weist der entwickelte IDA JPEG2000 Core folgende Eigenschaften auf:

- Min. 14 Bit pro Bildpunkt (verlustlos und verlustbehaftet)
- Min. 128 × 128 *Tile*-Größe
- 5/3 *Wavelet*
- Schlanke Grundkonfiguration u. a. für energieeffizienten Einsatz
- Skalierbar und damit anpassbar in Performanz und Ressourcenbedarf
- Durchsatz pro Ressourcenbedarf vergleichbar zu den kommerziellen Cores
- Optionale *Rate Control* während *Tier-1*
- Optionale Pixel-basierte ROI-Codierung

Der spezielle IDA JPEG2000 Core ist ausgelegt, um zusammen mit einem Prozessor und anderen klein- bis mittelgroßen Modulen wie Schnittstellenlogiken in einen strahlungstoleranten RT-Virtex-II FPGA implementiert zu werden. Innerhalb des FPGA sind noch genügend Ressourcen zur Implementierung weiterer spezieller Funktionen für Performanzsteigerungen vorhanden. Als Grundkonfiguration dient ein Aufbau mit dem mittelgroßen strahlungstoleranten XQR2V3000 FPGA der RT-Virtex-II Familie, so dass innerhalb dieser Familie noch genügend Reserve besteht. Während hier in Anlehnung an die strahlungstolerante DPU nach Kapitel 6 gezielt der strahlungstolerante RT-Virtex-II FPGA ausgewählt wurde, wird in Abschnitt 7.3.12 die Möglichkeit einer alternativen strahlungsfesten Implementierung gezeigt.

Um den Hardwarebedarf nicht unnötig zu erhöhen, stand bei der Entwicklung ein schlankes, skalierbares Design mit möglichst geringer Komplexität im Vordergrund. Auf nicht zwingend

notwendige Besonderheiten (z. B. *Precinct*, *Layer*, *Multicomponents*) wurde verzichtet. Gegenüber kommerziellen Cores hat die Entwicklung eines eigenen Core den wichtigen praktischen Vorteil, dass bei der Entwicklung einer DPU mit Einbindung des eigenen IDA JPEG2000 Core der vollständige Zugriff auf seine Quellen besteht. Damit können leicht Anpassungen an das jeweilige Projekt gemacht werden, wie z. B. höhere Bitanzahl pro Bildpunkt oder Änderungen an Schnittstellen. Für schnelle Datentransfers kann die Schnittstelle z. B. um einen Busmaster mit DMA-Funktionalität erweitert werden. Der IDA JPEG2000 Core kann der Anwendung entsprechend skaliert oder durch zusätzliche Funktionen ergänzt werden. Die Funktionalität „*Rate Control* während *Tier-1*“ beispielsweise beschleunigt die Verarbeitung verlustbehafteter Bilddaten.

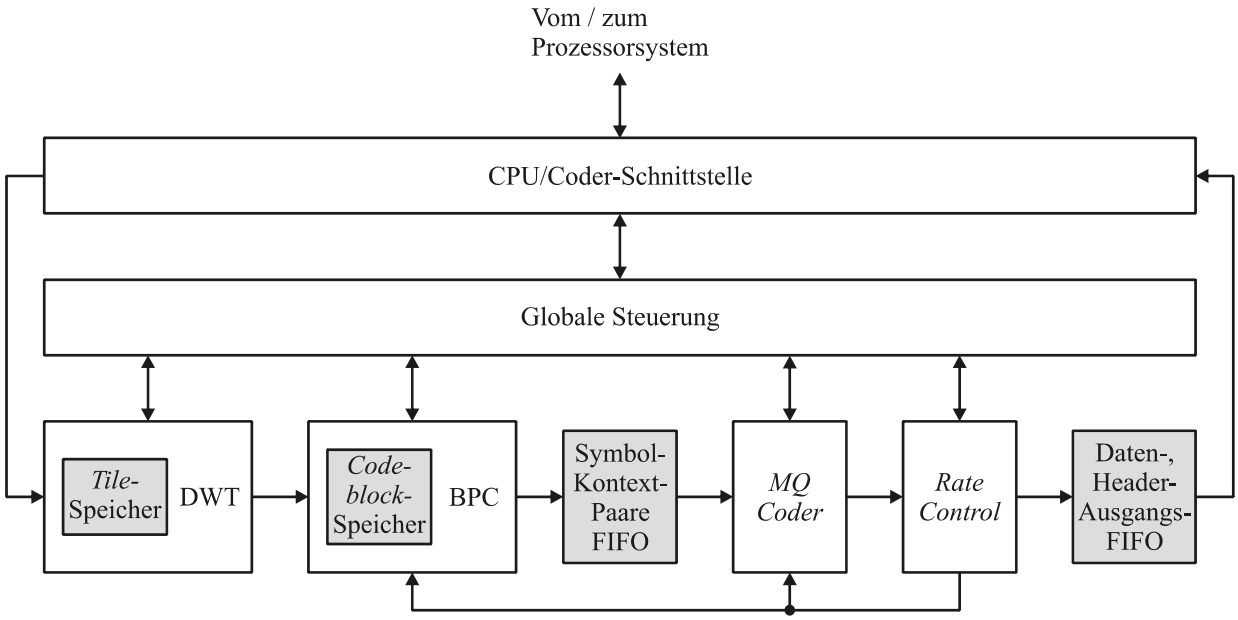

Bild 7.1: Übersicht IDA JPEG2000 Core

In Bild 7.1 ist der Aufbau des IDA JPEG2000 Core mit den einzelnen Modulen Diskrete *Wavelet* Transformation (DWT), *Bitplane Coder* (BPC), *MQ Coder* und *Rate Control* dargestellt. Weniger rechenintensive Funktionalitäten des JPEG2000-Algorithmus (z. B. Aufteilung in *Tiles*, Formatierung der Ausgangsdaten) werden durch den übergeordneten Prozessor in Software ausgeführt, wie es auch bei den kommerziellen JPEG2000 Cores vorgesehen ist. Dabei ist sicherzustellen dass die gewählte Partitionierung in Hardware und Software bestmöglich gewählt ist. Die Module haben jeweils eine eigene Steuerung und sind einer globalen Steuerungseinheit untergeordnet. Diese wiederum ist dem Prozessor unterstellt und kontrolliert die Zusammenarbeit der einzelnen Module. Sobald der Prozessor die Daten eines *Tile* des Gesamtbildes in den *Tile*-Speicher geschrieben hat und den IDA JPEG2000 Core startet, wird eine Transformation mit bis zu fünf Stufen durchgeführt. Anschließend werden die Koeffizienten eines jeden *Subbands* in vorzeichenlose Werte konvertiert und in den *Codeblock*-Speicher geschrieben. Es wird der BPC und *MQ Coder* zur Verarbeitung der *Codeblock*-Daten angestartet. Der BPC

bearbeitet die Daten des *Codeblocks* und schreibt die hieraus erzeugten Symbol-Kontext-Paare in einen als FIFO ausgeführten Zwischenspeicher. Hierauf greift der *MQ Coder* zu, der die komprimierten Daten in ein Ausgangs-FIFO schreibt. Sobald *Bitplane Coder* und *MQ Coder* die Daten eines *Codeblocks* bearbeitet haben, wird aus dem *Tile*-Speicher das nächste *Subband* konvertiert und in den *Codeblock*-Speicher geschrieben. Die *Rate Control* nimmt bei verlustbehafteter Kompression Einfluss auf die Menge der zu codierenden Daten und beschleunigt die Verarbeitung. Zusatzinformationen (*Header*-Daten) für die verschiedenen *Passes*, wie z. B. Nummer des *Pass*, der Bitebene, des *Subbands* sowie MSE-Informationen werden in einem zu den komprimierten Daten parallelen Ausgangs-FIFO abgelegt. Als interner Speicher wird *Dual Port* RAM verwendet, der einen praktischen Zugriff von zwei Seiten auf denselben Speicher erlaubt. Im Folgenden werden die einzelnen für den IDA JPEG2000 Core entwickelten, speziell optimierten Module näher beschrieben.

7.3.1 Diskrete *Wavelet* Transformation (DWT)

Die *Wavelet*-Transformation beim JPEG2000-Algorithmus beschränkt sich auf die Größe eines *Tile*. Je höher die *Tile*-Größe gewählt ist, umso besser ist die erzielbare Bildqualität. Die Basiskonfiguration des IDA JPEG2000 Core verwendet eine *Tile*-Größe von 128 × 128, kleinere *Tile*-Größen von z. B. 64 × 64 Pixel führen zu stärkeren Qualitätseinbußen [Kam04]. Für eine schnelle und lokal begrenzte Ausführung der Transformation ist es sehr wünschenswert, dass der *Tile*-Speicher innerhalb des FPGA liegt. Die Größe kann entsprechend der hierzu zur Verfügung stehenden FPGA Ressourcen angepasst werden.

Bei der für den IDA JPEG2000 Core entwickelten DWT ist für eine Anzahl von 14 Bit pro Bildpunkt eine maximale Tiefe von 17 Bit zur Speicherung der Koeffizienten erforderlich [Zho06]. Die Anzahl an Transformationsstufen ist parametrierbar und auf maximal fünf Stufen begrenzt. Bild 7.2 zeigt den Aufbau der entworfenen DWT. Die DWT-Steuerung enthält einen Adressgenerator, um die unterschiedlichen *Subbänder* zu adressieren. Zuerst wird eine zweidimensionale DWT auf das Original-*Tile* ausgeführt, anschließend rekursiv weitere zweidimensionale DWTs auf den LL Bereich der zuvor transformierten Werte. Eine zweidimensionale DWT erfolgt unter Verwendung eines Zwischenspeichers (Zeilen- / Spalten-Speicher) durch jeweils eine horizontale, mit anschließender vertikaler eindimensionaler DWT. Hierdurch ist ein klar strukturierter Aufbau der DWT gegeben.

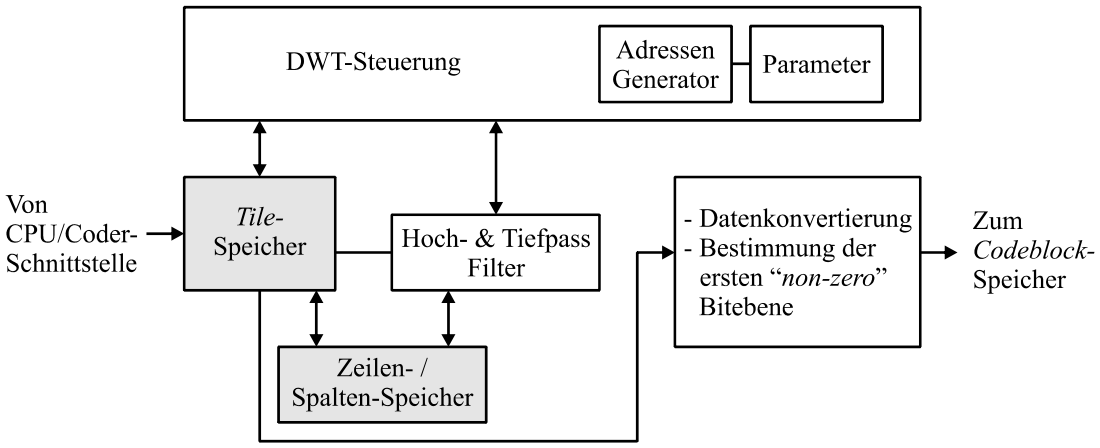

Bild 7.2: Implementierung DWT

Die Filterung der Daten mit Hochpass und Tiefpass erfolgt mit Hilfe des in Abschnitt 3.4 beschriebenen *Lifting*-Schemas. Die Umsetzung der zugehörigen Gleichungen (3.14) und (3.15) ist in Bild 7.3 ersichtlich. Sobald die Transformation eines *Subbands* bzw. eines einem *Codeblock* zugehörigen Bereichs abgeschlossen ist, werden die transformierten *Tile*-Daten zur weiteren Verarbeitung durch den *Bitplane Coder* in den *Codeblock*-Speicher übertragen. Innerhalb dieses Kopiervorgangs werden weitere Funktionen praktisch ohne Zeitverlust durchgeführt. Eine dieser Funktionen wandelt z. B. jeden Koeffizienten von einem vorzeichenbehafteten Wert in einen vorzeichenlosen Wert mit Vorzeichenbit um, wie es für die anschließende Codierung erforderlich ist. Die Bestimmung der höchstwertigen Bitebene mit Werten ungleich Null erfolgt in einer weiteren Funktion durch einfache ODER-Verknüpfung der transferierten vorzeichenlosen Werte. Hiermit wird dem anschließenden *Bitplane Coder* die Information zur Verfügung gestellt, mit welcher Bitebene begonnen werden soll.

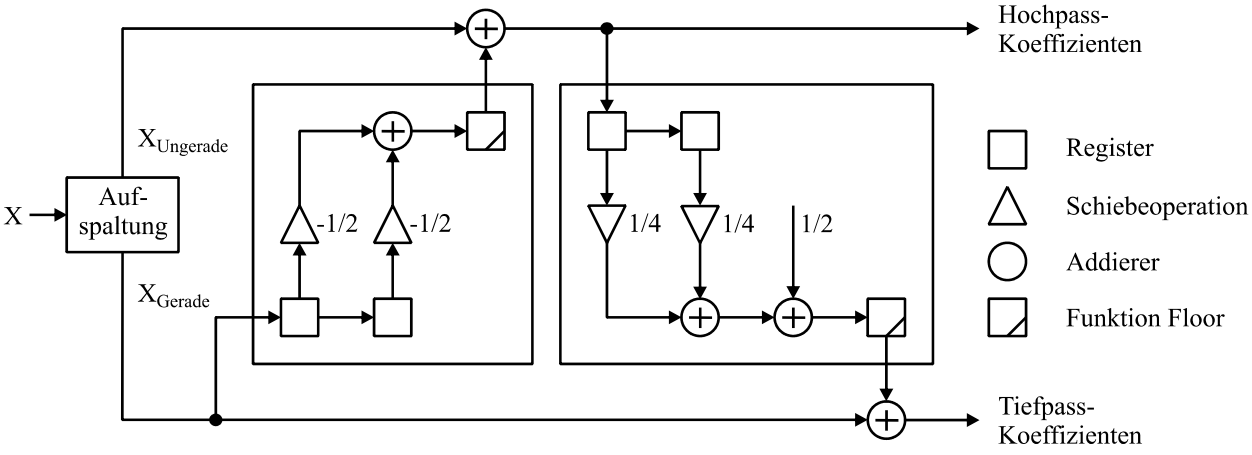

Bild 7.3: Implementierung der Filterung nach *Lifting*-Schema

Bei $N \times N$ großen *Tiles* werden zur Transformation einer Zeile oder Spalte $N + 5$ Takte für die Transformation und zusätzlich N Takte zum Speichern der Werte im *Tile*-Speicher zu insgesamt

$$2 \cdot N + 5 \quad \text{Takten} \tag{7.1}$$

benötigt. Für die zweidimensionale Transformation eines $N \times N$ großen Bereichs ergeben sich

$$4 \cdot N^2 + 10 \cdot N \quad \text{Takte}. \tag{7.2}$$

Hieraus ergibt sich die notwendige Taktanzahl zur Bestimmung einer 5-stufigen DWT zu

$$\sum_{i=0}^{4} \left(4 \cdot \left(\frac{N}{2^i} \right)^2 + 10 \cdot \frac{N}{2^i} \right) = \frac{341}{64} \cdot N^2 + \frac{310}{16} \cdot N. \tag{7.3}$$

Mit der Anzahl der notwendigen Takte zum Kopieren der Daten aus dem *Tile*-Speicher in den *Codeblock-Speicher* erhält man

$$\frac{341}{64} \cdot N^2 + \frac{310}{16} \cdot N + N^2 \quad \text{Takte}. \tag{7.4}$$

Für eine *Tile*-Größe von 128 × 128 ergeben sich 106.160 Takte. Die maximale Taktfrequenz für das Design liegt bei 90 MHz (XQR2V3000-4 FPGA). Der Ressourcenbedarf beträgt 380 *Slices* sowie 18 *Block RAMs* (17 für *Tile*-Speicher und 1 für Zeilen- / Spalten-Speicher).

7.3.2 Bitplane Coder (BPC)

In der Literatur befinden sich verschiedene Lösungswege für die Implementierung des *Bitplane Coder* (BPC) in Hardware [Chi02][Gan03]. Sie beinhalten vielfach ein hohes Maß an interner Parallelität auf Kosten eines komplexen Aufbaus. Für die Anwendung in einem kompakten Kamerainstrument dagegen ist zuviel interne Parallelität für ein schlankes Design hinderlich und aufgrund des zu erwartenden Durchsatzes auch nicht erforderlich.

In [And03] ist eine Hardwareimplementierung des *Bitplane Coder* (BPC) beschrieben, die sich direkt aus dem Algorithmus ergibt und weniger komplex aufgebaut ist, als die in [Chi02] und [Gan03] beschriebenen Implementierungen. Der *Codeblock* ist jedoch auf eine Größe von $N \times 4$ Koeffizienten und damit auf genau einen Streifen begrenzt. Dennoch kann diese Implementierung als Grundlage für den entworfenen *Bitplane Coder* angesehen werden. Die in dieser Arbeit verwendete Realisierung beinhaltet somit eine erweiterte Ausführung von [And03], bei der u. a. mehrere Streifen innerhalb eines *Codeblocks* bearbeitet werden können und damit die Limitierung auf einen Streifen aufgehoben ist. Die maximale *Codeblock*-Größe beträgt 64 × 64

Koeffizienten, so dass beim Bearbeiten der Streifen die Speicherung zusätzlicher Informationen aus dem vorangegangenen Streifen erforderlich ist. Eine weitere Ergänzung ist das gleichzeitige Auslesen mehrerer Daten eines Streifens zur Steigerung der Geschwindigkeit. Der Aufbau des BPC ist in Bild 7.4 gezeigt.

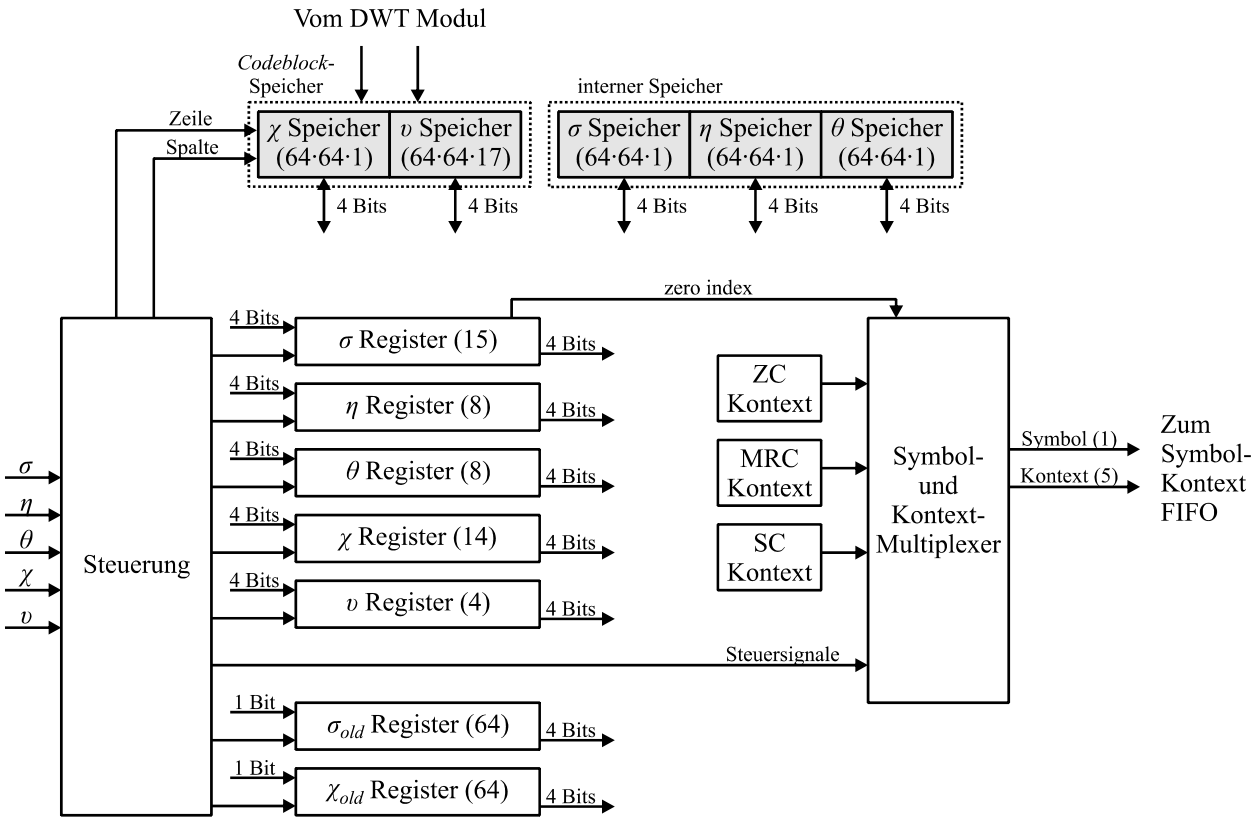

Bild 7.4: Blockschaltbild *Bitplane Coder*

Der *Bitplane Coder* benötigt je Koeffizienten-Position vier Zustandsinformationsbits und ein *Magnitude* Bit υ (Betrag des Koeffizienten). Die Zustandsinformationen bestimmen, in welchen *Pass* jedes Bit codiert wird, und werden zur Erzeugung des Symbols und Kontexts verwendet. Die vier Zustandsinformationsbits sind:

- *Significance* Bit σ: wird gesetzt, sobald das *Magnitude* Bit υ des zugehörigen *Subband*-Koeffizienten 1 wird.
- *Visited once* Bit η: wird gesetzt, sobald das Bit in einem *Pass* codiert wird.
- *Magnitude refinement coded* Bit θ: wird gesetzt, sobald die Grundoperation MRC verwendet wird.
- *Sign* Bit χ: ist 0 für positive und 1 für negative Zahlen und wird den *Subband*-Werten in ihrer Darstellung als vorzeichenloser Wert mit Vorzeichenbit entnommen.

Die η Bits werden am Ende einer jeden Bitebene zurückgesetzt, während die σ, θ und χ Bits erst mit neuen *Codeblock*-Daten zurückgesetzt werden und somit während der vollständigen Codierung eines *Codeblocks* gültig sind.

Die einzelnen Elemente des in Bild 7.4 gezeigten *Bitplane Coder* lassen sich folgendermaßen gruppieren:

- Kombinatorische Logik zur Erzeugung der Kontexte für ZC, MRC und SC (Kontexte für RLC sind im Multiplexer fest codiert).
- Mehrere logische Funktionen zur Erzeugung des aktuellen Kontexts aus den Zustandsvariablen und den *Subband*-Informationen.
- Fünf Schiebe-Register zur Speicherung der Variablen σ, η, θ, χ und υ mit unterschiedlichen Größen und Funktionalitäten.
- Zwei 64-Bit-Register χ_{old}, σ_{old} zur Speicherung der χ und σ Werte der letzten Zeile im vorherigen Streifen (siehe Bild 7.5). Dieses erlaubt es, vorherige Streifen-Werte mit in die Codierung des aktuellen Streifens einfließen zu lassen.
- Fünf Speicherblöcke jeweils mit der Größe $N \times N$, mit N als die Anzahl an Zeilen bzw. Spalten des aktuellen *Codeblocks*. Verglichen zu [And03] vergrößert sich die nutzbare *Codeblock*-Größe von $N \times 4$ auf $N \times N$ Koeffizienten. Diese Speicherblöcke sind zum Ablegen der *Magnitude* Bits und der vier zugehörigen Zustandsinformationsbits notwendig. Der *Codeblock*-Speicher besteht aus vier *Block RAMs* für die absoluten Koeffizientenwerte und einem *Block RAM* für die Vorzeicheninformation (siehe Bild 7.6). Durch die asymmetrische Struktur können mit einem Zugriff gleichzeitig vier Werte inkl. der Vorzeichenbits gelesen und der *Bitplane Coder* damit beschleunigt werden. Die Verwendung der *Block RAMs* als *Dual Port* RAM ermöglicht es, mit unterschiedlichen Formaten von zwei Seiten aus auf die Speicher zuzugreifen. Das Beschreiben des *Codeblock*-Speichers wird nach erfolgter DWT koeffizienten-orientiert, das Lesen jedoch streifen-orientiert durchgeführt. Die vier Zustandsinformationsbits sind jeweils in einem 4-Bit breiten *Block RAM* abgelegt, so dass jeweils gleichzeitig vier Werte streifen-orientiert ausgelesen werden können.
- Symbol- und Kontext-Multiplexer zur Auswahl des Symbols oder des Kontexts.
- Eine Steuerung bestehend aus einer Zustandsmaschine mit zugehörigen Zählern, die z. B. zur Bestimmung des aktuellen *Pass* oder der Bitebene notwendig sind. Die Steuerung führt die *Passes* SP, MRP, CP auf Basis der Grundoperationen ZC, RLC, SC, MRC (siehe Abschnitt 3.5.1) sowie der Initialisierungs- und Terminierungsoperationen durch [Zho05]. Es werden je *Pass* ein oder mehrere dieser Operationen eingesetzt.

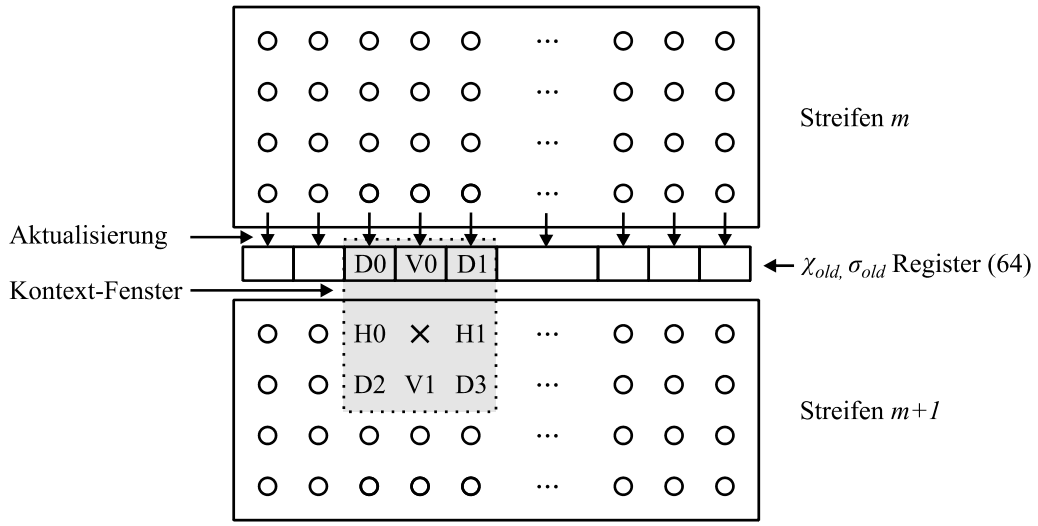

Bild 7.5: Zwischenspeicherung von Daten bei Streifenbearbeitung

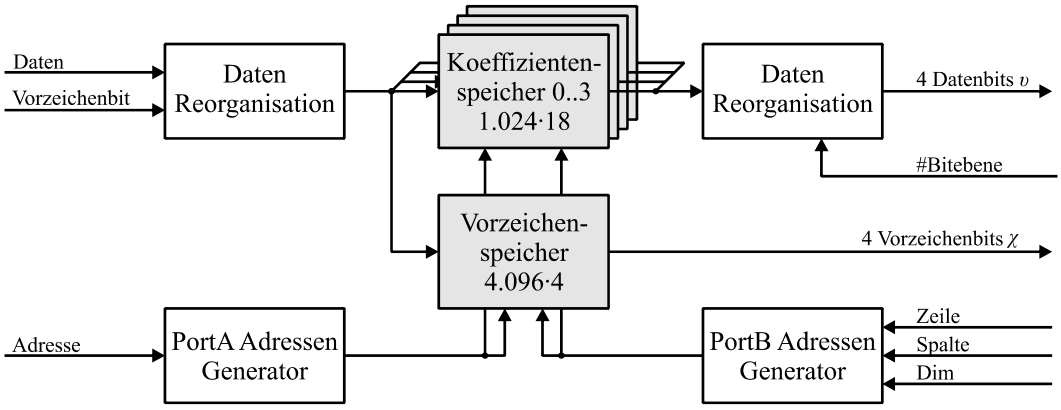

Bild 7.6: Organisation *Codeblock*-Speicher für schnellen Zugriff

Nach der Transformation liegt die oberste Bitebene mit Werten ungleich Null meistens unterhalb der höchsten Bitebene. Daher ist eine vollständige Codierung aller Koeffizientenebenen bei allen *Codeblöcken* (z. B. 17 Ebenen bei einer Anzahl von 14 Bit pro Bildpunkt) i. Allg. nicht erforderlich. Bei $N \times N$ großen *Tiles* mit B Bit pro Bildpunkt wird jeder Pixel im Durchschnitt $3 \cdot B$ mal bearbeitet.

Die vier übereinander liegenden Werte eines $N \times 4$ großen Streifens, die jeweils gleichzeitig ausgelesen werden können, bilden eine 4er-Gruppe. Ein Teil der 4er-Gruppen wird während des Durchlaufs eines *Pass* codiert, ein anderer Teil (ca. 50 %) übersprungen. Aus dem Aufbau der Zustandsmaschine ergibt sich für die codierten 4er-Gruppen eine durchschnittliche Anzahl von zehn Takten, für die übersprungenen 4er-Gruppen dagegen eine reduzierte Anzahl von fünf Takten.

Für die codierten 4er-Gruppen sind

$$3 \cdot 10 \cdot B \cdot \frac{N^2}{4} \cdot 50\,\% = 3,75 \cdot B \cdot N^2 \quad \text{Takte,} \tag{7.5}$$

für die nicht codierten 4er-Gruppen

$$3 \cdot 5 \cdot B \cdot \frac{N^2}{4} \cdot 50\,\% = 1,875 \cdot B \cdot N^2 \quad \text{Takte} \tag{7.6}$$

erforderlich. Die Gesamtzahl an Takten ergibt sich zu

$$3,75 \cdot B \cdot N^2 + 1,875 \cdot B \cdot N^2 = 5,625 \cdot B \cdot N^2. \tag{7.7}$$

Die tatsächliche Taktanzahl ist abhängig von den Daten selbst. Bei einer *Tile*-Größe von 128×128 mit einer Bittiefe B von 14 Bit ergeben sich durchschnittlich 1.290.240 Takte. Die maximale Taktfrequenz für das Design liegt bei 90 MHz (XQR2V3000-4 FPGA). Der Ressourcenbedarf beträgt ca. 640 *Slices* sowie drei *Block RAMs*. Für den *Codeblock*-Speicher werden fünf *Block RAMs* benötigt.

7.3.3 MQ Coder

Wie in Abschnitt 3.5.2 beschrieben, kann der *MQ Coder* als eine Zustandsmaschine angesehen werden, die eine Reihe von Eingangswerten mit den zugehörigen Kontexten auf ein Codewort abbildet. Die hier entworfene Implementierung ist direkt aus dem JPEG2000-Standard abgeleitet und in Bild 7.7 dargestellt [Bol00][Tau02].

Die einzelnen Elemente des *MQ Coder* sind:

- Kontextzustandstabelle zur Speicherung des Index und MPS für den aktuellen Kontext
- *Q* Tabelle (Wahrscheinlichkeitstabelle) mit Werten für Q_e, *NMPS*, *NLPS*
- Zähler für die bereits im aktuellen *Pass* erzeugten Bytes
- Zähler für die erzeugten Bits
- Verschiedene Register: 16-Bit-Register *Q* für den aktuellen Wahrscheinlichkeitswert Q_e, 16-Bit-Register *A* (Intervall Länge) mit Schiebe-Funktion, 32-Bit-Register *C* (untere Grenze des Intervalls), 32-Bit temporäres *C* Register und 8-Bit-Register *B* (für das komprimierte Byte)
- Speicher zum Ablegen von Informationen zur Berechnung des mittleren quadratischen Fehlers (MSE)
- Steuerung mit 21 Zuständen zur Steuerung des Ablaufes

Innerhalb des *MQ Coder* Moduls wird für jeden *Pass* der akkumulierte Fehler MSE ermittelt, der zur Bestimmung der Verzerrung bei verlustbehafteter Kompression verwendet wird. Hierzu werden für jedes codierte Bit mittels der abgelegten *Lookup*-Tabelle dessen MSE bestimmt und diese Werte für einen *Pass* aufsummiert.

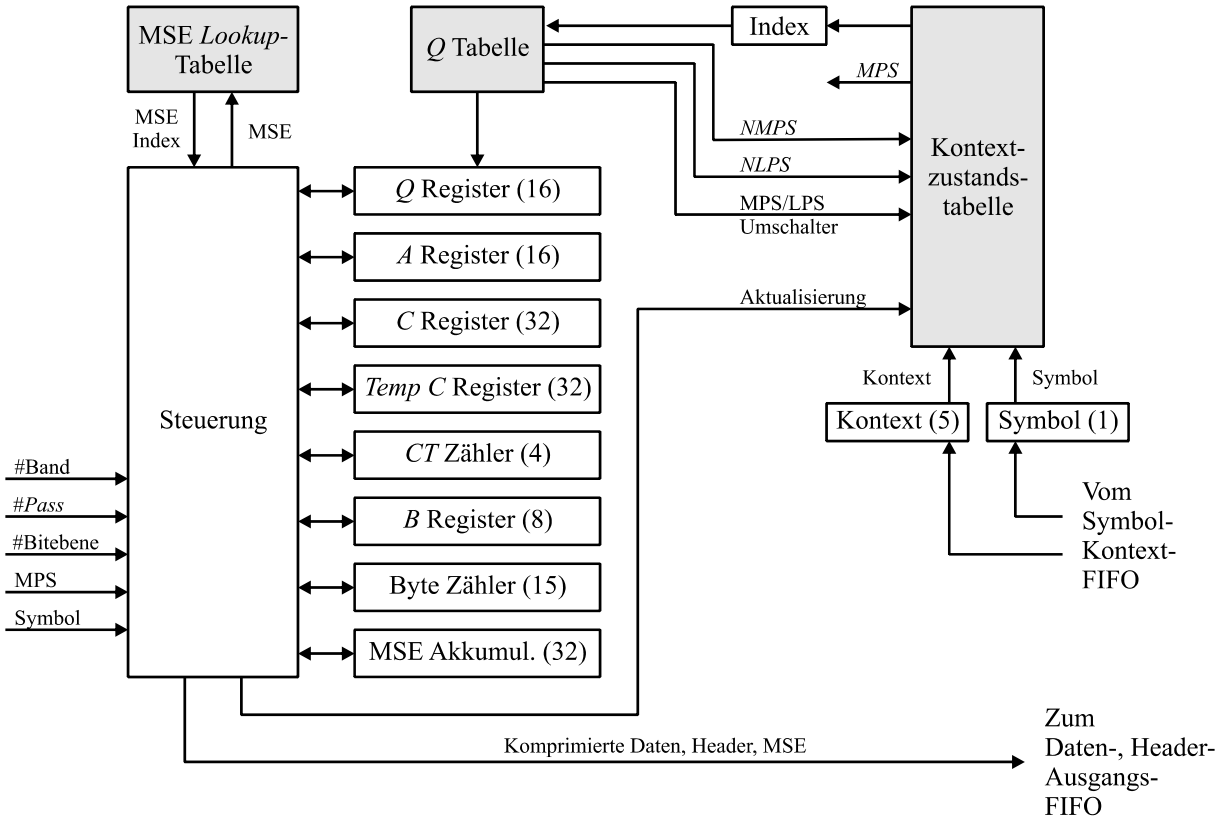

Bild 7.7: Implementierung *MQ Coder*

Die Anzahl notwendiger Takte zur Erzeugung der Codewörter eines *Tile* ist abhängig von der Anzahl der vom *Bitplane Coder* erzeugten Symbol-Kontext-Paare. Für ein *Tile* der Größe $N \times N$ mit B Bit pro Bildpunkt werden durchschnittlich $N \times N \times B$ Symbol-Kontext-Paare erzeugt [Zho05]. Zur Erzeugung eines Codewortes werden zwischen vier und neun Takte, in den allermeisten Fällen jedoch sechs Takte (Arithmetische Codierung) benötigt. Hiermit ergibt sich die für ein *Tile* notwendige Anzahl der Takte zu:

$$6 \cdot B \cdot N^2. \tag{7.8}$$

Bei einer *Tile*-Größe von 128×128 mit einer Bittiefe B von 14 Bit ergeben sich insgesamt 1.376.256 Takte und damit ein Wert, der vergleichbar ist zu der vom *Bitplane Coder* benötigten Anzahl. Beim BPC und beim *MQ Coder* liegt die Anzahl der Takte somit weit über der, die zur Transformation notwendig ist (> Faktor 10). Die maximale Taktfrequenz für das Design beträgt 90 MHz (XQR2V3000-4 FPGA). Der Ressourcenbedarf liegt bei ca. 700 *Slices* sowie einem *Block RAM*.

7.3.4 *Rate Control*

Zur Beschleunigung der Kompression bei verlustbehafteten Bildern enthält der IDA JPEG2000 Core eine *Rate Control*, die bereits Einfluss nehmen kann auf die Codierung während *Tier-1*.

Der Entwurf der *Rate Control* erfolgte in Anlehnung an den in [Cha03] beschriebenen Algorithmus mit Rückkopplung (siehe Bild 7.8) und führt zu einer ratenabhängigen Kompressionszeit. Viele andere in der Literatur angegebene Hardwareverfahren sind entweder komplexer oder aber qualitativ weniger hochwertig. Bei der hier verwendeten Implementierung handelt es sich um eine kombinierte *Rate Control*, die sowohl in Hardware als auch in Software umgesetzt wird. Der erste Teil wird in Hardware durchgeführt und beendet die Codierung, sobald genügend qualitativ hochwertige Daten zur Weiterverarbeitung vorhanden sind. Beim zweiten Teil erfolgt in Software eine Beschneidung der zuvor erzeugten Daten, deren Menge im Gegensatz zu PCRD (siehe Abschnitt 3.5.3) von der gewünschten Rate abhängig ist.

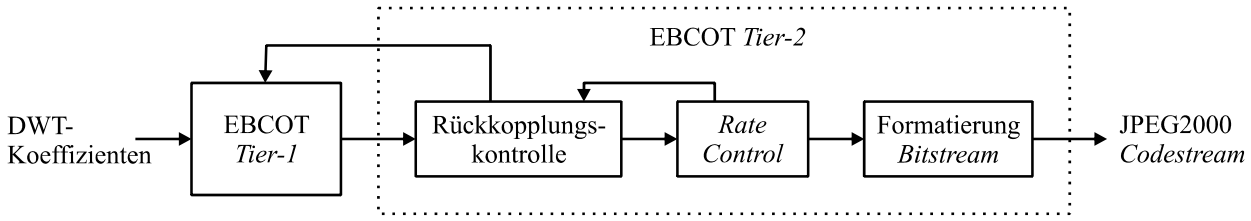

Bild 7.8: *Rate Control* mit Rückkopplung

In Bild 7.9 ist der Aufbau des Hardwareanteils dargestellt. Der Algorithmus basiert darauf, dass für die Steigungswerte (*Slope*-Werte, siehe Abschnitt 3.5.3) eine *Slope*-Byte-Tabelle aufgebaut wird, die die akkumulierte Anzahl an Bytes für jeden *Pass* für jeweils einen *Slope*-Wert enthält. Implementiert ist eine Genauigkeit von 1.024 *Slope*-Werten. Der Index für die Einträge in dieser Tabelle ist der *Slope* selbst. Nach jeder Bearbeitung eines *Codeblocks* werden die Einträge in der *Slope*-Byte-Tabelle untersucht. Die Anzahl an Bytes wird dabei vom höchsten *Slope* bis zum niedrigsten *Slope* akkumuliert, bis sie die vorgegebene Ziellänge überschreitet. Dieser zuletzt akkumulierte *Slope* wird für den nächsten *Codeblock* als Schwellwert (*Target Slope*) verwendet. Liegt der *Slope*-Wert eines *Pass* beim nächsten *Codeblock* unter dem des Schwellwertes, dann wird die Codierung der darauf folgenden *Passes* innerhalb des *Codeblocks* übersprungen. Zur Berechnung der Steigungen beinhaltet die *Rate Control* eine Multiplikations- und eine Dividiereinheit. Die Codierung arbeitet von der höchsten bis zur niedrigsten Transformationsstufe. Dieses führt dazu, dass die Daten bis zum erstmaligen Erreichen der Ziellänge vollständig codiert (*Slope*-Schwellwert ist 0) und die Codierung überflüssiger Daten erst ab diesem Zeitpunkt unterbunden wird. Hierdurch werden durchschnittlich um Faktor 2,5 mehr Daten erzeugt als nach der vollständigen Erzeugung der komprimierten Daten eines *Tile* nach *Tier-2* insgesamt noch verwendet werden. In der Summe kann bei vorgegebener Rate eine geringere Verzerrung erzielt werden als es bei reinen Hardwareverfahren der Fall ist. Solche Verfahren

brechen die Codierung z. B. ab, sobald die erwünschte Datenmenge erreicht ist und sind dadurch schneller.

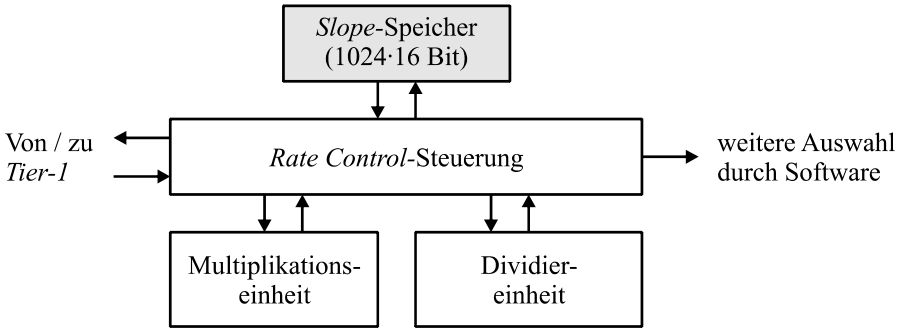

Bild 7.9: Implementierung *Rate Control* in Hardware

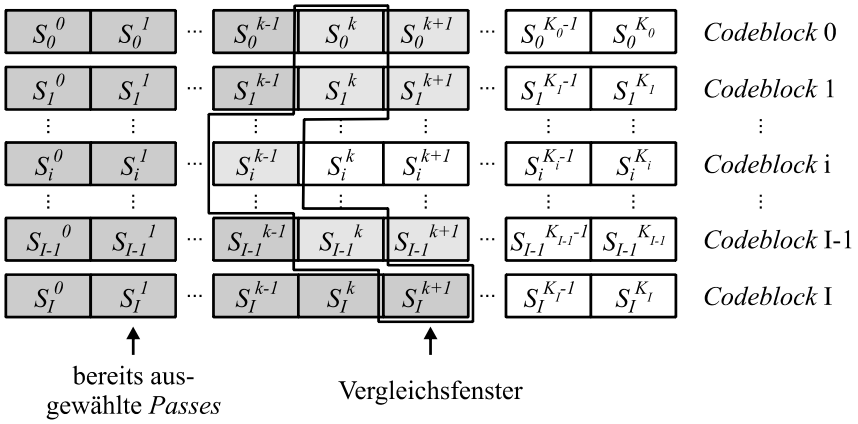

Bild 7.10: Auswahl der *Slopes* S_i^k mit den Abbruchstellen *k* unter Ausnutzung der Monotonie innerhalb eines *Codeblocks i*

Beim Auslesen der codierten Daten aus dem IDA JPEG2000 Core überprüft die Software, ob die hintereinander ausgelesenen Werte zu einer monoton abfallenden *Rate Distortion*-Kurve führen. Bei abweichenden Werten setzt sie den *Slope*-Wert auf den des Vorgängers. Sind alle *Codeblock*-Daten eines *Tile* vorhanden, werden zuerst die *Slope*-Werte aller *Passes* mit dem aus der Hardware gelieferten Schwellwert verglichen und diejenigen *Passes* für den endgültigen *Codestream* ausgewählt, deren *Slope*-Werte über dem Schwellwert liegen. Hierzu ist ein Vergleich je *Pass* durchzuführen. Die Verwendung des unpräzisen Schwellwerts zur Auswahl der *Passes* führt dazu, dass die zur Verfügung stehende Datenmenge nicht vollständig ausgenutzt wird. Daher ist eine weitere Routine erforderlich, die die restlichen zur Verfügung stehenden Daten mit Werten aus den bisher noch nicht ausgewählten *Passes* auffüllt. Die *Slope*-Werte der *Passes* werden *Codeblock*-übergreifend verglichen (siehe Bild 7.10). Aufgrund des monotonen Kurvenverlaufs der *Rate Distortion*-Kurve ist zur Auswahl eines *Pass* der Zugriff auf lediglich einen Wert pro *Codeblock* erforderlich. Der auf den ausgewählten *Pass* folgende Wert innerhalb des *Codeblocks* wird anschließend zur Bestimmung des nächsten Wertes herangezogen.

Diese optionale rückgekoppelte *Rate Control* arbeitet ebenfalls mit 90 MHz und benötigt mit ca. 500 *Slices* vergleichsweise viele Ressourcen. Als interner Speicher hingegen ist lediglich ein *Block RAM* für den *Slope*-Speicher erforderlich.

7.3.5 *Region of Interest*-Codierung

Spricht man von einer *Region of Interest* (ROI)-Codierung, so ist in den meisten Fällen eine Pixel-basierte ROI-Codierung gemeint. Zur Vereinfachung könnte eine ROI-Codierung auch auf größere Einheiten ausgeführt werden, z. B. auf *Tiles*. Eine ROI-Codierung kann damit z. B. in zwei Genauigkeitsstufen unterteilt werden:

- *Tile*-basiert
- Pixel-basiert

Bei *Tile*-basierter ROI-Codierung werden den *Tiles* einzelne Kompressionsfaktoren zugeteilt. Durch geringere Kompression der interessanten *Tiles,* verglichen zu den weniger interessanten *Tiles*, kann die Gesamtdatenrate für ein Bild reduziert werden. Da die Hardwarebeschleunigung bereits auf Basis von *Tiles* arbeitet, würde diese ROI-Codierung lediglich eine Anpassung an der Software erforderlich machen. Sie ist somit ebenfalls bei den kommerziellen JPEG2000 Cores denkbar. Die Zielgröße des komprimierten Datenstroms für das jeweilige *Tile* muss zuvor bestimmt werden. Ein Nachteil der *Tile*-basierten ROI-Codierung ist in den diskreten Fenstergrößen und -positionen zu sehen. Ein kleiner ROI-Bereich (z. B. 150 × 150 Pixel), zentriert über einem *Tile* mit Überlappung zu allen Seiten, erfordert bei einer *Tile*-Größe von 128 × 128 die vollständige Codierung von neun *Tiles* mit geringem Kompressionsfaktor. Die ROI würde damit in diesem Beispiel künstlich auf 384 × 384 Koeffizienten angehoben werden, d. h. um den Faktor 6. Durch die *Tile*-basierte ROI-Codierung kann die Datenrate daher nur ungenau und mit großem Verschnitt reduziert werden. Interessant kann die *Tile*-basierte ROI-Codierung sein, wenn das Verhältnis Bildgröße zu *Tile*-Größe sehr groß ist und lediglich eine Softwareimplementierung benötigt wird.

Mit Pixel-basierter ROI-Codierung, wie sie im JPEG2000-Standard vorgeschlagen wird, können frei wählbare Pixelbereiche gezielt qualitativ hervorgehoben werden. Für die hier verwendete *Maxshift*-Methode (siehe Abschnitt 3.6) muss die ROI-Maske im Frequenzbereich aus einer vorgegebenen ROI-Maske im Zeitbereich berechnet werden. Diese Bestimmung der anzuhebenden Koeffizienten erfolgt für ein festes ROI-Muster einmalig. Die Verschiebung der Koeffizienten zu höheren Bitebenen erfordert bei der Codierung eine Verdopplung der zu verarbeitenden Bittiefe (z. B. 34 Bit anstelle von 17 Bit) und hat damit Einfluss auf den Speicherbedarf und die Codierungsgeschwindigkeit (z. B. + 12,5 % bei acht 128 × 128 *Tiles* mit ROI-Bits innerhalb eines 1k × 1k Bildes).

Das Verschieben der ROI-Koeffizienten zu höheren Bitebenen erfolgt mit minimalen zusätzlichen Laufzeiten durch einfache Schiebe-Operationen während des Kopiervorganges aus dem

Tile-Speicher in den *Codeblock*-Speicher (siehe Bild 7.11). Die Größe des *Tile*-Speichers bleibt unverändert, während der vergleichsweise kleine *Codeblock*-Speicher eine höhere Speichertiefe zur Unterbringung der ROI-Bitebenen aufweisen muss. Mit jedem neuen *Tile*, das die Software in den *Tile*-Speicher des Coders transferiert, beschreibt sie zudem den Speicher für die zugehörige ROI-Maske im Frequenzbereich, bestehend aus 1-Bit-Werten. Ihre Einträge entscheiden darüber, ob die ROI-Koeffizienten beim Kopieren aus dem *Tile*-Speicher in den *Codeblock*-Speicher angehoben werden.

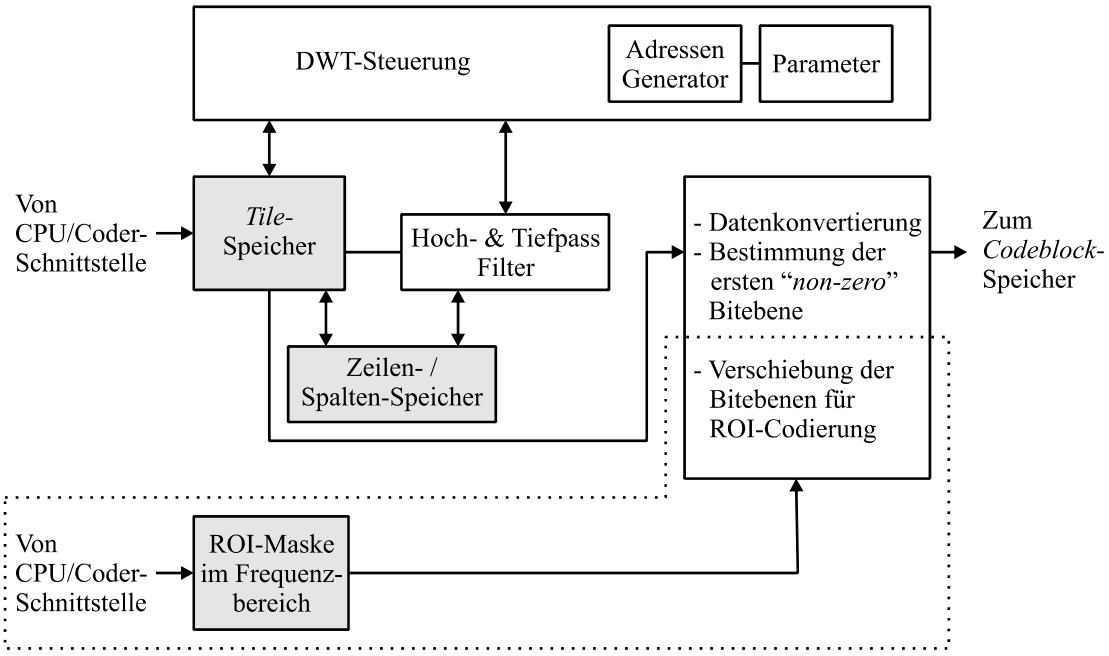

Bild 7.11: Erweiterung DWT-Modul zur Verwendung der ROI-Codierung

Die Implementierung der optionalen ROI-Codierung kann wie die DWT mit 90 MHz (XQR2V3000-4 FPGA) betrieben werden und erfordert insgesamt ca. 170 *Slices* sowie 6 *Block RAMs*. Hiervon sind die meisten *Block RAMs* für die Erweiterung des *Codeblock*-Speichers aufgrund der erhöhten Koeffiziententiefe erforderlich.

7.3.6 Steuerung

Den Modulen DWT, BPC, *MQ Coder* und *Rate Control* ist eine globale Steuerung übergeordnet, die den verschachtelten Ablauf zwischen den einzelnen Modulen und deren Funktionsweise steuert. Sie beinhaltet mehrere Konfigurationsregister, auf die von einem übergeordneten Prozessor zugegriffen werden kann. Die Steuerung ist mit einer umfangreichen Interrupt-Funktionalität ausgestattet, die es bei unterschiedlichen Bedingungen erlaubt, dem Prozessor mittels Interrupt aktuelle Ereignisse des IDA JPEG2000 Core anzuzeigen, um eine schnelle Reaktion des Prozessors zu ermöglichen. Als Beispiele zu nennen sind hier die Füllzustände der FIFO oder die Erzeugung eines Interrupts, sobald der letzte *Codeblock* aus dem *Tile*-Speicher in

den *Codeblock*-Speicher transferiert wurde. Hierdurch ist ein hoher Verschachtelungsgrad der Hardware- und Softwareroutinen realisierbar. Um Zugriffe durch den Prozessor zu ermöglichen, sind dessen Zugriffssignale auf die der globalen Steuerung abgebildet. Dieses erfolgt innerhalb der CPU/Coder-Schnittstelle. Da der IDA JPEG2000 Core eine andere Taktfrequenz als der Prozessor verwenden kann (z. B. $f_{JPEG2000Core} = 90$ MHz, $f_{CPU} = 50$ MHz), beinhaltet die CPU/Coder-Schnittstelle eine Synchronisierung dieser beiden Taktfrequenzen. Bei hohen Datenraten, wie sie beim Beschreiben des *Tile* oder dem Auslesen der erzeugten Daten anfallen, werden die zugehörigen als *Dual Port* RAM ausgeführten Speicher aus Geschwindigkeitsgründen ohne Verwendung der globalen Steuerung direkt von der CPU/Coder-Schnittstelle angesprochen (siehe Bild 7.1). Teilweise erfolgt die Anpassung der Daten (z. B. Umwandlung in vorzeichenbehaftete Werte durch Subtrahierung um 2^{B-1} und Darstellung als 2er-Komplement) bereits in der CPU/Coder-Schnittstelle und nicht erst im Coder selbst.

7.3.7 Verifikation des IDA JPEG2000 Core

Der Core wurde unter Verwendung des Mentor Graphics HDL Designer 1004.1b entworfen. Zur Simulation wurde ModelSim SE 6.0 und zur Synthese Precision RTL 2004c.45 verwendet. Die Platzierung sowie Verdrahtung auf dem verwendeten FPGA erfolgte mit Xilinx ISE 7.1i. Zur funktionalen Überprüfung des Designs wurden die Ausgaben des IDA JPEG2000 Core mit Referenzdaten verglichen, die durch eine JasPer-Software erzeugt worden sind. Die JasPer-Software ist Teil des JPEG2000-Standards.

Die Verifikation in Hardware erfolgte unter Verwendung eines Evaluation Board mit einem Virtex-II Pro FPGA (XC2VP20). Innerhalb des FPGA ist dem Core als steuernde Einheit ein LEON2-Prozessor übergeordnet worden. Die Ausgaben des Core werden an den PC übertragen und dort mit den Referenzdaten verglichen. Diese stammen ebenfalls aus der JasPer-Software.

7.3.8 Software

Die implementierten Interrupts erlauben es, die Hardware- und Softwareroutinen eng miteinander zu verschachteln. Dazu muss das Prozessorsystem imstande sein, genügend schnell zu reagieren und Aktionen mit dem erforderlichen Durchsatz durchzuführen. Die Aufgaben der Software lassen sich bei der gewählten Hardware / Software Partitionierung zusammenfassen in:

- Steuerungsaufgaben
- Datentransfer zum und vom Coder
- *Tier-2* mit *Rate Control,* Erzeugung eines JPEG2000 *Codestream*

In Bild 7.12 ist das Zusammenspiel des IDA JPEG2000 Core mit dem Prozessorsystem gezeigt. Während die Steuerungsaufgaben eine geringe Belastung des Prozessorsystems darstellen, sollten insbesondere der Datentransfer sowie *Tier-2* und die Erzeugung eines JPEG2000

Codestream genügend schnell ausgeführt sein, damit es nicht zu Wartezeiten innerhalb des Coder kommt. Wie in Bild 7.12 ersichtlich, kann die Transformation der *Tiles*, mit Ausnahme des ersten *Tile* eines Bildes, parallel zur Codierung des letzten *Codeblocks* des vorherigen *Tile* erfolgen. Mit 64 *Tiles* bei einem 1k × 1k Bild kann daher die Zeit, die zur Transformation des ersten *Tile* benötigt wird, vernachlässigt werden. Die Aufteilung eines Bildes in *Tiles* erfolgt in Zusammenhang mit dem Datentransfer und wirkt sich kaum auf den Durchsatz aus.

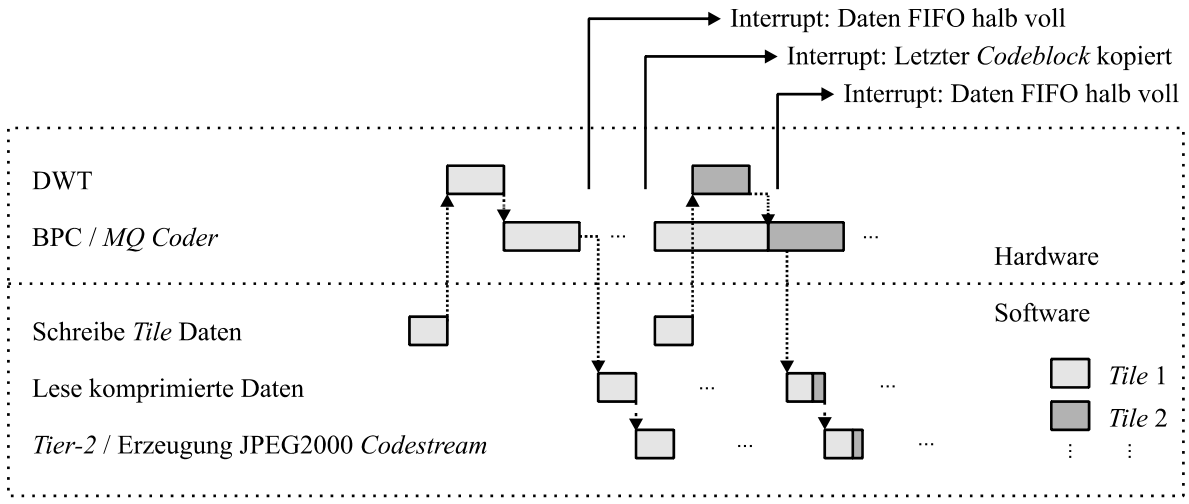

Bild 7.12: Hardware- und Softwareroutinen

Die Frage, ob die Softwareroutinen genügend schnell durchgeführt werden können, um den maximal möglichen Durchsatz des IDA JPEG2000 Core zu garantieren und damit eine richtige Partitionierung des JPEG2000-Algorithmus in Hardware und Software zu bestätigen, kann nur im Zusammenhang mit dessen Anbindung an ein Prozessorsystem beantwortet werden. Dieses erfolgt in den Abschnitten 8.1 und 8.2.

7.3.9 Zusammenfassung der Laufzeiten und des FPGA-Ressourcenbedarfs

Die in Abschnitt 7.3.2 und 7.3.3 genannten Taktzahlen der Module sind berechnete Werte, die auf Messungen mit Testbildern beruhen. Die ermittelte Gesamtanzahl an Takten für ein *Tile* der Größe 128 × 128 mit 14 Bit pro Bildpunkt ergibt sich zu:

Modul	Taktanzahl	Beitrag
DWT	106.160	0%
BPC	1.290.240	0%
MQ Coder	1.376.256	100%
Sicherheit	68.813	100%
Summe	1.445.069	

Tabelle 7.2: Zusammenfassung Taktanzahl

Die Bereiche eines *Tile* werden von hohen Transformationsstufen zu niedrigen Transformationsstufen, und deshalb mit ansteigender Größe des *Codeblocks,* bearbeitet. Sobald der letzte *Codeblock* der niedrigsten Transformationsstufe (großer *Codeblock*) in den *Codeblock*-Speicher übertragen wurde, steht der *Tile*-Speicher für weitere Daten zur Verfügung. Die Codierung dieses *Codeblocks* dauert aufgrund seiner Größe verhältnismäßig lange, so dass während dieser Zeit die Transformation neuer *Tile*-Daten erfolgen und abgeschlossen werden kann. Durch die Verschachtelung von *Tile*-Transformation mit der Codierung von *Codeblock*-Daten aus der vorangegangenen Transformation kann der Beitrag der DWT zur gesamten Anzahl erforderlicher Takte für ein *Tile* minimiert werden (nur für das erste *Tile* eines Bildes wird zusätzliche Zeit benötigt, die vernachlässigbar ist). Da der *MQ Coder* zeitlich nahezu parallel zum BPC arbeitet, ist der *MQ Coder* das für die Performanzbestimmung maßgebende Modul. Die Bearbeitungszeiten für DWT und BPC sind somit durch die des *MQ Coder* verdeckt. Bei der Bestimmung der Taktanzahl ist eine Sicherheit von 5 % der *MQ Coder*-Takte eingerechnet. Diese wird u. a. von der Schnittstelle BPC – *MQ Coder* für mögliche Wartezeiten aufgrund Datenknappheit beansprucht. Mit der maximalen Taktfrequenz für DWT, BPC und *MQ Coder* von 90 MHz (XQR2V3000-4 FPGA) beträgt die rechnerisch ermittelte Laufzeit für ein *Tile* der Größe 128 × 128 mit 14 Bit pro Bildpunkt bei verlustloser Kompression 16 ms. Dieses entspricht einem Durchsatz von 14,3 MBit/s. Die tatsächlichen Werte variieren in Abhängigkeit von den Daten.

Für einen Performanz-Vergleich des IDA JPEG2000 Core mit den kommerziellen Cores aus Tabelle 7.1 wird exemplarisch der Barco B112JPEG2000E Core verwendet. Dessen Datenblatt bietet unter den kommerziellen Cores die meisten Informationen für einen verlässlichen Vergleich, der hier anhand einer verlustlosen Kompression durchgeführt wird. Zur Schaffung gleicher Bedingungen sind zwei Korrekturfaktoren zu bestimmen. Beim Barco B112JPEG2000E Core ist ein Durchsatz von ca. 80 MBit/s bei einem 8-Bit Bild angegeben, wobei mit durchschnittlich fünf gefüllten Bitebenen gerechnet wurde. Der IDA JPEG2000 Core liefert einen Durchsatz von ca. 14,3 MBit/s. Bei der Berechnung geht ein Bild mit 14 Bit pro Bildpunkt mit durchschnittlich 14 gefüllten Ebenen der insgesamt 17 Ebenen ein. Dieses ergibt einen ersten Korrekturfaktor von $14/14 \cdot 8/5 = 1{,}6$. Des weiteren wurde der Durchsatz beim Barco B112JPEG2000E Core für einen Virtex-II FPGA mit dem Geschwindigkeitsgrad 6 berechnet, während der Durchsatz des IDA JPEG2000 Core für einen strahlungstoleranten RT-Virtex-II FPGA gültig ist, für den nur der langsamere Geschwindigkeitsgrad 4 zur Verfügung steht. Dieses ergibt einen weiteren Korrekturfaktor von 1,25 [Xil05b]. Um für den Barco B112JPEG2000E Core die gleichen Verhältnisse zu schaffen, wie sie bei der Berechnung des Durchsatzes des IDA JPEG2000 Core verwendet wurden, müssen die beiden Korrekturfaktoren angewendet werden und man erhält für den Barco B112JPEG2000E Core den korrigierten Durchsatz von 40 MBit/s. Setzt man den Durchsatz ins Verhältnis zum Ressourcenbedarf (an Kombinatorik), dann erzielt man beim Barco Core mit 40 MBit/s bei 6.153 *Slices* einen Durchsatz von 6,5 MBit/s für 1.000 *Slices*. Beim IDA JPEG2000 Core sind es 14,3 MBit/s bei 2.219 *Slices* (ebenfalls ohne *Rate Control* während *Tier-1*) und damit 6,4 MBit/s für 1.000 *Slices*. Am Beispiel des Barco B112JPEG2000E Core und verlustloser Kompression ist damit gezeigt, dass der auf die

verbrauchten Ressourcen (Kombinatorik) normierte Durchsatz des IDA JPEG2000 Core gleichwertig zu dem der kommerziellen JPEG2000 Cores ist.

Der FPGA-Ressourcenbedarf des IDA JPEG2000 Core ist in Tabelle 7.3 aufgeführt. Die Anzahl an *Slices* ist bei BPC und *MQ Coder* ähnlich hoch. Beim internen Speicher ist die Größe des *Tile*-Speichers maßgebend. Erwägt man eine Erhöhung der *Tile*-Größe von 128×128 auf 256×256 und damit eine Vervierfachung des erforderlichen Speichers, so wären für den IDA JPEG2000 Core insgesamt bereits mehr als 100 % an *Block RAMs* des XQR2V3000 FPGA erforderlich und die Verwendung des größten Bausteins der RT-Virtex-II Familie wäre notwendig.

Modul	***Slices***		**Benutzer-Flipflops**		**18 Bit × 18 Bit Multipl.**		**18 kBit *Block RAMs***	
	abs.	**%**	**abs.**	**%**	**abs.**	**%**	**abs.**	**%**
CPU/Coder-Schnittstelle	132	0,9	267	0,9	-	-	-	-
DWT	371	2,6	394	1,4	-	-	1	1,0
BPC	636	4,4	344	1,2	-	-	3	3,1
MQ Coder	683	4,8	496	1,7	1	1,0	1	1,0
Rate Control	484	3,4	663	2,3	2	2,1	1	1,0
Tile-Speicher	21	0,1	61	0,2	-	-	19	19,8
Codeblock-Speicher	34	0,2	18	0,1	-	-	5	5,2
Symbol-Kontext-FIFO	33	0,2	33	0,1	-	-	1	1,0
Ausgangs-FIFO	68	0,5	104	0,4	-	-	3	3,1
Steuerung etc.	243	1,7	326	1,1	-	-	-	-
ROI-Codierung	171	1,2	186	0,6	-	-	6	6,3
Grundkonfiguration: Einfache Codierungskette, *Rate Control*, keine ROI-Codierung	2.703	18,9	2.706	9,4	3	3,1	34	35,4

Tabelle 7.3: IDA JPEG2000 Core: Ressourcenbedarf innerhalb des XQR2V3000 FPGA

7.3.10 Abänderungen der Grundkonfiguration

Der IDA JPEG2000 Core ist mit der Vorgabe entwickelt worden, in der Grundkonfiguration niedrige Anforderungen an den FPGA-Ressourcenbedarf zu stellen.

Die Grundkonfiguration des IDA JPEG2000 Core mit einfacher Codierungskette bestehend aus 1× DWT, 1× BPC, 1× *MQ Coder* und 1× *Rate Control* kann abgeändert werden für z. B.:

1. Parallelisierung zur Erhöhung des Durchsatzes
2. Erhöhte *Tile*-Größe (z. B. 256 × 256) zur Qualitätssteigerung
3. Entfernung der *Rate Control* während *Tier-1*.

Die ersten beiden Abänderungen erfolgen unter Inkaufnahme eines höheren Ressourcenbedarfs (siehe Tabelle 7.4) und ggf. eines größeren erforderlichen RT-Virtex-II FPGA. Zu beachten ist, dass bei einem erhöhten Durchsatz ebenso der Datentransfer mit dem Core ansteigt und dass diese Erhöhung vom Prozessorsystem unterstützt werden muss. In Bild 7.13 ist die Konfiguration des IDA JPEG2000 Core mit einfacher Codierungskette ohne *Rate Control* dargestellt. Einen höheren Durchsatz liefert die in Bild 7.14 dargestellte Konfiguration mit dreifacher Codierungskette.

Konfiguration	***Slices***		**18 kBit *Block RAMs***		**Durchsatz [MBit/s]**	
	abs.	**%**	**abs.**	**%**	**verlustlos**	***c* = 6**
Grundkonfiguration	2.703	18,9	34	35,4	14,3	21,0
1-fache Codierungskette ohne *Rate Control*	2.219	15,5	33	34,4	14,3	14,3
1-fache Codierungskette ohne *Rate Control*, mit ROI-Codierung und *Tile*-Größe 256 × 256	2.390	16,7	99	103,1	14,3	14,3
3-fache Codierungskette ohne *Rate Control*	5.125	35,7	59	61,5	42,0	42,0

Tabelle 7.4: Kennwerte für verschiedene Konfigurationen des IDA JPEG20000 Core innerhalb des XQR2V3000 FPGA

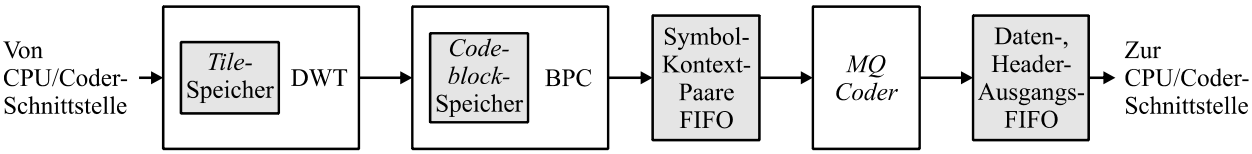

Bild 7.13: Konfigurationsbeispiel mit einfacher Ausführung von *Tier-1* ohne *Rate Control* in Hardware

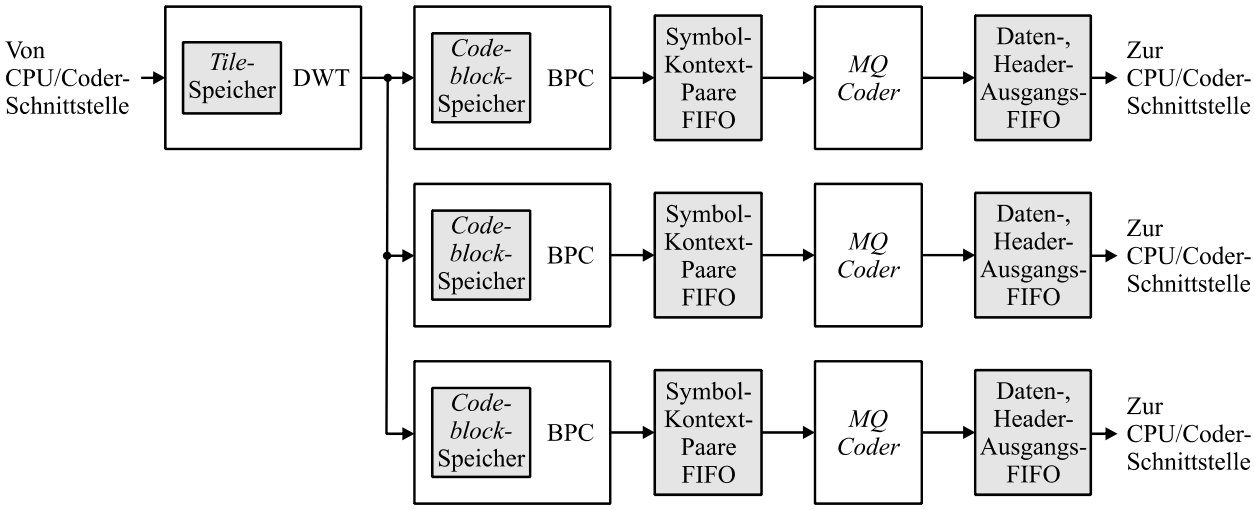

Bild 7.14: Konfigurationsbeispiel mit dreifacher Ausführung von *Tier-1* ohne *Rate Control*

7.3.11 Performanzmessungen

Eine enge Verzahnung der Hardware- und Softwareroutinen des JPEG2000-Algorithmus führt zu einer schnellen Kompression. Sind die Softwareroutinen verglichen zu den Hardwareroutinen schnell genug, so werden die Ausführungszeiten durch diese Hardwareroutinen bestimmt. Für die Kompression der vier Testbilder (siehe Bild 5.2 bis Bild 5.5) ergeben sich die in Tabelle 7.5, Tabelle 7.6 und Bild 7.15 gezeigten Werte. Die Zeit zur Durchführung der Kompression wird durch die parallel arbeitenden Module *Bitplane Coder* bzw. *MQ Coder* bestimmt. Die erzielten Bildqualitäten, gemessen anhand der PSNR-Metrik, sind in Bild 7.16 ersichtlich. Zur visuellen Beurteilung der Kompression sind die Testbilder für einige Kompressionsfaktoren in Bild 7.17 bis Bild 7.20 dargestellt.

Testbild	Kompressionsfaktor	Ausführungszeit [ms]	Durchsatz [MBit/s]
Venus	1,56	123	16,3
Mars Full	1,57	213	14,8
Mars DaoVallis	1,35	224	14,0
Mars Surface	1,26	59	13,3

Tabelle 7.5: Verlustlose Kompression mit dem IDA JPEG2000 Core

Kompressionsfaktor *c*	verlustlos	2	4	6	8	10	12	14	16
Durchsatz [MBit/s]	14,6	14,9	18,4	21,5	23,6	25,9	27,5	29,2	30,6

Tabelle 7.6: Durchschnittlicher Durchsatz der Kompression

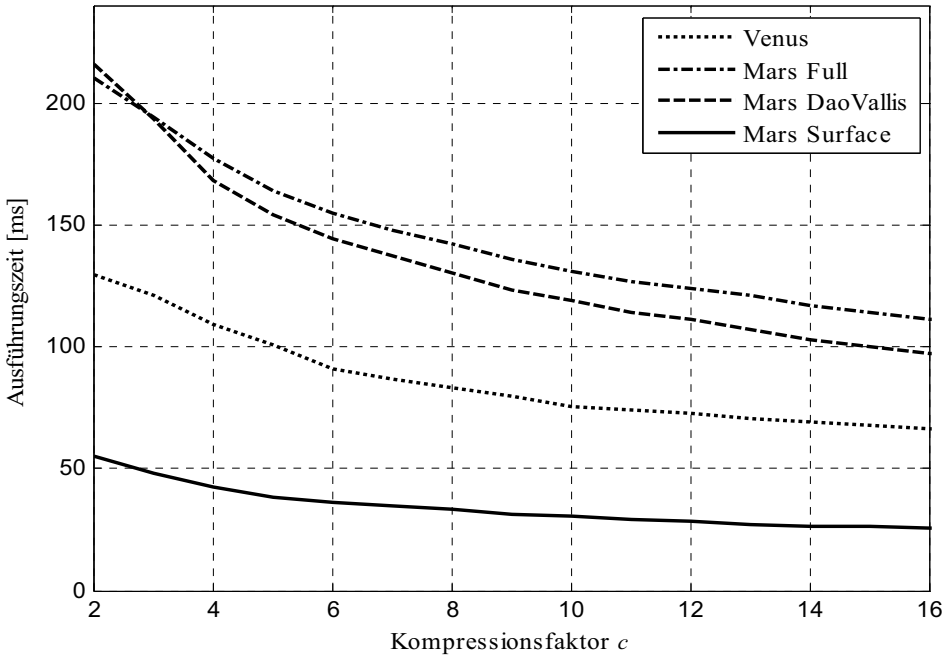

Bild 7.15: Ausführungszeit zur Kompression der Testbilder

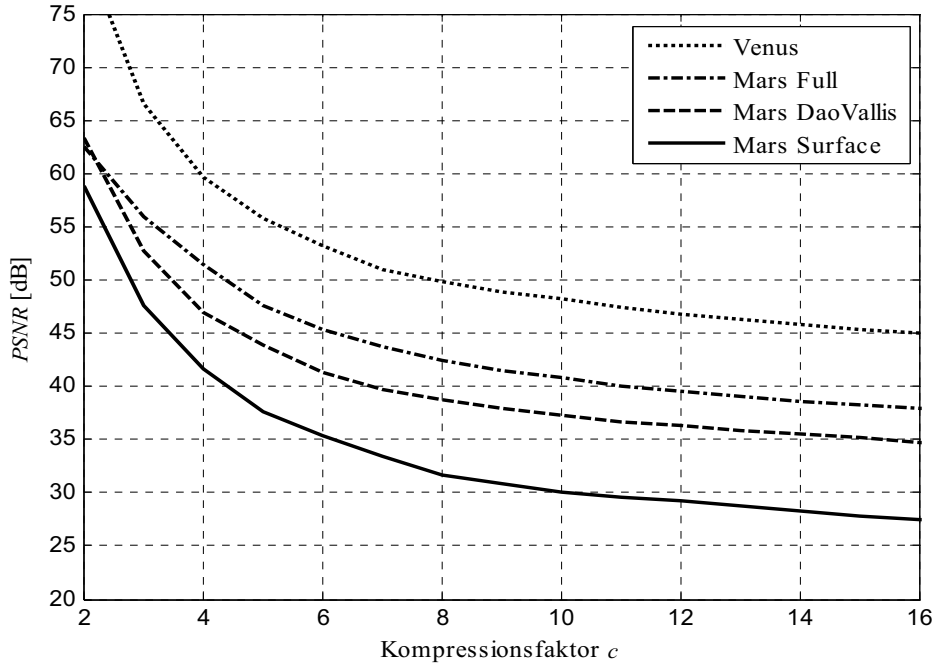

Bild 7.16: Bildqualität, *Tile*-Größe 128 × 128

Bild 7.17: Kompressionsergebnisse für Testbild Venus

Bild 7.18: Kompressionsergebnisse für Testbild Mars Full

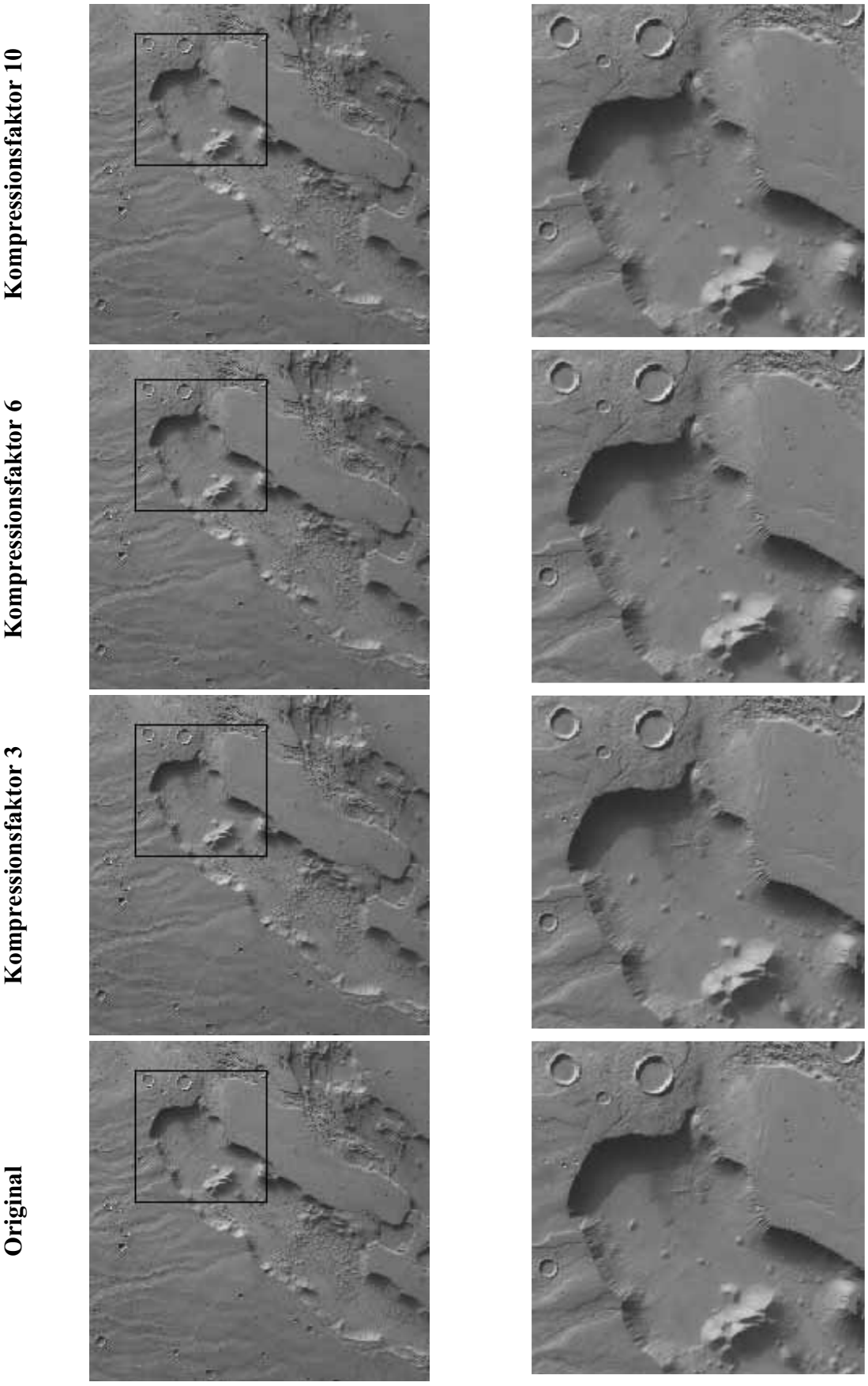

Bild 7.19: Kompressionsergebnisse für Testbild Mars DaoVallis

Bild 7.20: Kompressionsergebnisse für Testbild Mars Surface

7.3.12 Implementierung in strahlungsfestem Baustein

Erwägt man die Verwendung des IDA JPEG2000 Core in der strahlungsfesten DPU nach Kapitel 5, so stellt sich die Frage nach dem zu verwendenden Baustein. Aufgrund der häufigen Zugriffe auf den *Codeblock*-Speicher ist es für einen sinnvollen Einsatz sehr wünschenswert, diesen Speicher vollständig in dem Baustein zu integrieren, in dem auch die verarbeitende Logik integriert ist. Dieses gilt ebenfalls für andere häufig verwendete Werte. Da eine ASIC Implementierung aus praktischen Gründen eher die zweite Wahl darstellt, wird hier der RTAX4000S FPGA aus der RTAX-S Familie von Actel ausgewählt. Gegenüber strahlungsfesten FPGAs der Firma UTMC bietet er ein Vielfaches mehr an internem Speicher und gegenüber den RT-Virtex-II FPGAs die Möglichkeit der strahlungsfesten Implementierung der Logik ohne Einbau von Redundanz. Letzteres ist vor allem maßgebend bei der Auswahl des RTAX4000S FPGA innerhalb der strahlungsfesten DPU. Bei Verwendung dieses Bausteins sind Synergieeffekte mit dem in Kapitel 5 implementierten RT54SX-S FPGA zu erwarten. Es ist zu beachten, dass der interne Speicher sehr strahlungsempfindlich und deshalb mit einer *Hamming*-Fehlerkorrektur versehen ist [Wan03]. Damit kann der Anspruch für ein strahlungsfestes Design erhalten werden.

Während der *Codeblock* neben anderen temporären Daten im FPGA gehalten werden kann, ist für die Speicherung der *Tile*-Daten nicht genügend FPGA-interner Speicher vorhanden. Der *Tile*-Speicher muss daher ausgelagert und in einem externen strahlungsfesten Speicherbaustein implementiert werden. Die *Tile*-Daten werden zeilen- und spaltenweise in den FPGA eingelesen und dort transformiert, um anschließend wieder zurück geschrieben zu werden. Dieses führt zu einem erhöhten externen Datenstrom.

Insgesamt besteht die strahlungsfeste Implementierung des IDA JPEG2000 Core aus dem RTAX4000S FPGA und einem externen Zusatzspeicher. Zur Abschätzung der Performanz auf Basis der in Abschnitt 7.3.9 und 7.3.11 erzielten Werte ist der Einfluss des externen *Tile*-Speichers sowie die Fehlerkorrektur bei Zugriff auf den internen Speicher zu prüfen. Aufgrund zusätzlicher Laufzeiten benötigt der Transfer der *Tile*-Daten zur Transformation in den FPGA-internen Zeilen- / Spalten-Speicher mehr Zeit als bei FPGA-internem *Tile*-Speicher. Da die Transformation, verglichen zur nahezu parallel ausführbaren Codierung, insgesamt jedoch nur wenig Zeit in Anspruch nimmt, macht sich die Verwendung des externen *Tile*-Speichers kaum negativ auf die Performanz bemerkbar. Jedoch wird die Codiergeschwindigkeit mit Implementierung der Fehlerkorrektur für den internen Speicher durch verlängerte Zugriffe heruntergesetzt. Beim *Bitplane Coder* nach Abschnitt 7.3.2 erfolgen innerhalb von 7,5 Taktzyklen durchschnittlich zwei Zugriffe auf den internen Speicher, jeweils mit der Dauer eines Taktzyklus. Bei einer Zugriffsverlängerung von einem Taktzyklus auf den korrigierten internen Speicher erhöht sich damit die Anzahl notwendiger Taktzyklen um ca. 30 %. Beim *MQ Coder* erfordert lediglich die MSE-Berechnung Zugriff auf korrigierten Speicher. Diese kann jedoch parallel zur eigentlichen Bearbeitung erfolgen und führt daher nicht zu einer Verringerung des Durchsatzes.

Mit der Annahme, dass der IDA JPEG2000 Core im hier verwendeten RTAX4000S FPGA mit derselben Taktfrequenz betrieben werden kann wie im RT-Virtex-II FPGA, sind insgesamt ca. 30 % mehr Zeit zur Kompression erforderlich. Im Gegensatz zu Abschnitt 7.3 ist die maximal mögliche *Tile*-Größe nicht vom stark begrenzten internen FPGA-Speicher abhängig, sondern von der Größe des externen Zusatzspeichers. Mit typischen Speicherkapazitäten für strahlungsfeste SRAM-Bausteine von 4 MBit können z. B. *Tiles* mit einer Größe von 256 × 256 problemlos gespeichert werden.

8 Implementierung des IDA JPEG2000 Core und Vergleich

Der entwickelte IDA JPEG2000 Core aus Kapitel 7 wird im Folgenden sowohl in die strahlungsfeste DPU aus Kapitel 5 als auch in die strahlungstolerante DPU aus Kapitel 6 implementiert und es werden verschiedene Kennwerte analysiert. In einem Vergleich wird deutlich, inwieweit der in dieser Arbeit als Beispiel verwendete JPEG2000-Algorithmus unter Zuhilfenahme des IDA JPEG2000 Core zu einer Steigerung der Performanz in der Raumfahrt eingesetzt werden kann.

8.1 Strahlungsfeste DPU mit IDA JPEG2000 Core

Die strahlungsfeste DPU nach Abschnitt 5.1 kann effektiver verwendet werden, wenn eine schnelle Kompression eingefügt wird. Ist sie genügend schnell, d. h. es erfolgt eine Online-Kompression, so ist die Aufnahmegeschwindigkeit der Bilder nicht mehr von der Kompression abhängig, was zu einer sinnvollen Erweiterung der strahlungsfesten DPU führt. Zur Beschleunigung der Kompression mit Hilfe des IDA JPEG2000 Core wird die auf der RTAX-S Familie basierende Implementierung nach Abschnitt 7.3.12 gegenüber der RT-Virtex-II basierten Implementierung bevorzugt. Dieses ist in der geringeren Strahlungsempfindlichkeit der RTAX-S FPGAs begründet, da nur so der Anspruch eines strahlungsfesten Designs erhalten werden kann.

Da die Funktionalitäten des RT54SX-S FPGA nach Abschnitt 5.1 (z. B. die Schnittstellenlogiken) mit in den weitaus größeren RTAX-S FPGA integriert werden können, kann der RTAX-S FPGA den kleineren RT54SX-S FPGA vollständig ersetzen. Bei der Implementierung der Sensor- und Raumsonden-Schnittstellen kann auf die ursprünglich notwendigen externen Zwischenspeicher verzichtet werden. Diese werden durch FPGA-internen Speicher ersetzt. Bild 8.1 zeigt den Aufbau der strahlungsfesten DPU mit JPEG2000 Hardwarebeschleunigung und dem hierzu notwendigen Speicherbaustein zur Zwischenspeicherung der *Tile*-Daten. Die Anbindung des IDA JPEG2000 Core an das System kann auf verschiedenen Wegen erfolgen. Der Core kann mit einem eigenen kleinen Prozessor versehen werden, der die Kompression der Bilder autonom vom LEON2-Prozessor steuert und notwendige Berechnungen durchführt. Diese Kompressionseinheit könnte bereits im Pfad der empfangenen Daten liegen und eine Online-Kompression durchführen. Da der JPEG2000-Standard auf Basis von *Tiles* arbeitet, muss bereits eine bestimmte Menge an Bilddaten vorhanden sein, bevor mit der Kompression begonnen wird (z. B. $1\text{k} \times 256$ Pixel bei einem $1\text{k} \times 1\text{k}$ Bild und einer *Tile*-Größe von 256×256). FPGA-interner Speicher steht in dieser Größe nicht zur Verfügung, so dass ein externer Zwischenspeicher notwendig ist. Die Verwendung eines dedizierten Prozessors für den IDA JPEG2000 Core würde zudem zusätzliche Ressourcen im FPGA erfordern. Bevor ein Bild komprimiert wird, sind vorverarbeitende Funktionen wie die *Flat Field* Korrektur (siehe Abschnitt 3.1) notwendig. Daher ist es ohnehin erforderlich, dass das Rohbild ohne Kompression im Bildspeicher abgelegt wird. Die Verwendung eines eigenen Prozessors für den IDA JPEG2000 Core würde keinen merklichen Performanzgewinn bringen, jedoch erhebliche Nachteile im Gesamtaufwand (z. B. Synchronisierung mit der Bildaufnahme). Aus diesen Gründen ist eine Kompression der Daten im Empfangspfad zwischen Sensor-Schnittstelle und

Bildspeicher für den Aufbau einer DPU für eine kompakte Kamera nicht sinnvoll. Vielmehr ist der entwickelte IDA JPEG2000 Core als Co-Prozessor hardwareunterstützend zum Prozessor anzusehen und Teile des Algorithmus (z. B. Aufteilung in *Tiles*, *Tier-2*, Formatierung) sind aufgrund der gewählten Hardware / Software Partitionierung auf dem Standard-Prozessor in Software durchzuführen. Gesteuert durch den LEON2-Prozessor kann die Kompression auf einfachem Weg in den Ablauf „Akquisition – Vorverarbeitung – Kompression – Speichern / Senden“ eingebunden werden. Die hierzu notwendige Anbindung des IDA JPEG2000 Core an das Prozessorsystem (siehe Abschnitt 7.3.6, CPU/Coder-Schnittstelle) z. B. zum Schreiben der *Tile*-Daten oder zum Lesen der komprimierten Werte erfolgt durch die Ankopplung des Core als I/O Modul auf den Speicherbus. Hierbei sind zur Übertragung eines einzelnen Wertes ca. zehn Taktzyklen notwendig. Dieses ist für die in Kapitel 1 aufgelisteten Kennwerte mit einer Akquisitionsrate von einem Bild der Größe 1k × 1k × 14 Bit alle 1-2 s ausreichend.

Das Fehlen eines DPU-internen Massenspeichers kann trotz schneller Kompression stärkere Einschränkungen im operationellen Betrieb bedeuten. Operationen, die auf die gleichzeitige Verfügbarkeit mehrerer Bilder innerhalb der DPU angewiesen sind, sind weiterhin nicht möglich.

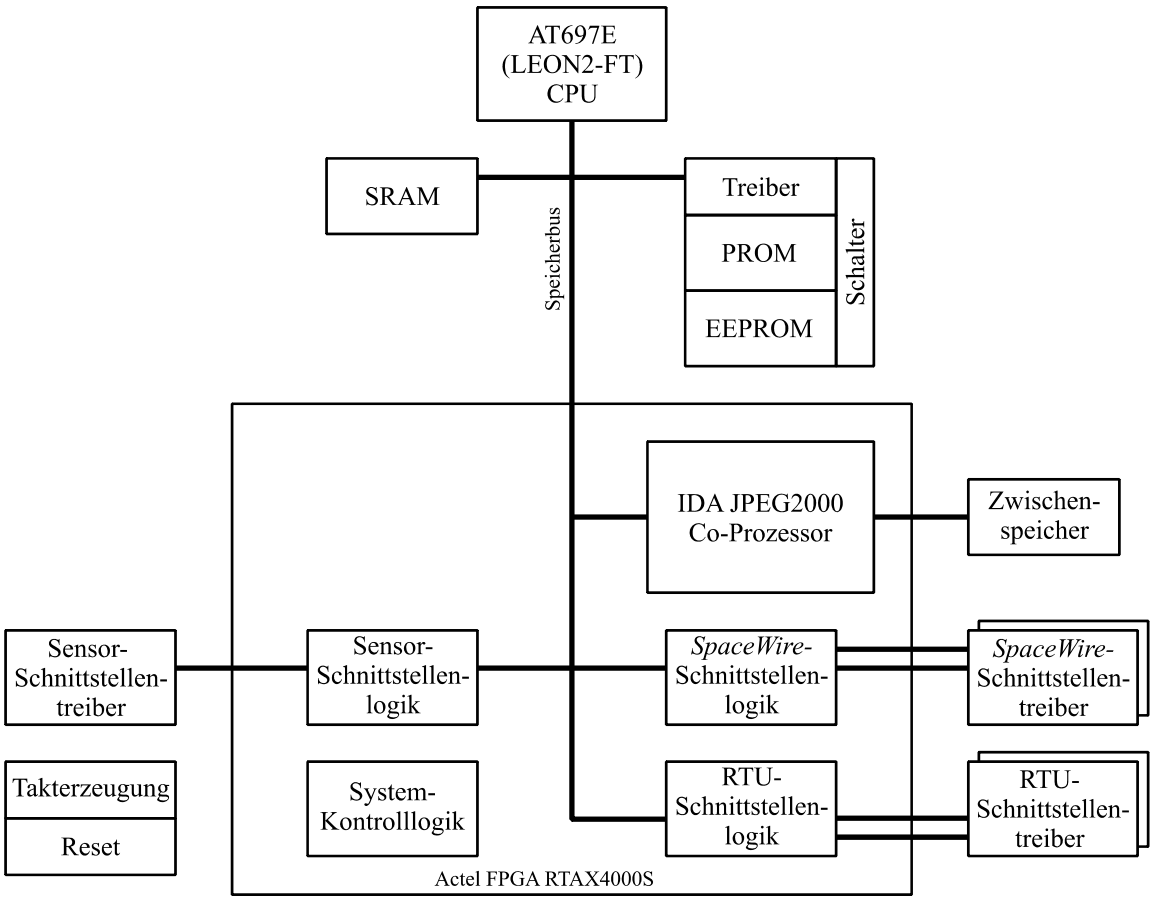

Bild 8.1: Strahlungsfeste DPU mit IDA JPEG2000 Core

8.1.1 Performanz

Die Performanz der Hardwarebeschleunigung ist bereits in Kapitel 7 ermittelt worden. Es stellt sich im Zusammenhang mit dem Prozessor die Frage, wie schnell die dem IDA JPEG2000 Core zugehörigen Softwareroutinen ausgeführt werden können. Die Implementierung der zur Kompression notwendigen Softwareroutinen sollte zu dem Ziel führen, dass die Softwareroutinen entsprechend Bild 7.12 mit den Hardwareroutinen des JPEG2000-Algorithmus Schritt halten können. Nur so kann eine richtige Partitionierung in Hardware und Software bestätigt werden. Die Geschwindigkeit wird im Wesentlichen durch den Datentransfer von und zum Coder sowie durch *Tier-2* und die Erzeugung des JPEG2000 *Codestream* bestimmt. *Tier-2* sowie die Erzeugung des JPEG2000 *Codestream* sind für eine kurze Verarbeitungszeit ausgeführt. Hierbei spielt es eine große Rolle, dass die *Rate Control* bereits während *Tier-1* eingreift. Die Hardware liefert für den in Software durchzuführenden Teil der *Rate Control* innerhalb von *Tier-2* bereits wichtige Informationen (z. B. *Slope*-Werte), so dass die entsprechenden Softwareroutinen gegenüber einer reinen Softwarelösung weit schneller ausgeführt werden können. Zudem hat die durch die Hardware vorgegebene begrenzte Genauigkeit von 1.024 *Slope*-Werten Einfluss auf die erhöhte Verarbeitungsgeschwindigkeit. Bei reinen Softwarelösungen ist die Genauigkeit häufig höher gewählt. Die Ausführungszeiten der JPEG2000 Softwareroutinen wurden durch einen LEON2-Prozessor bei einer Taktfrequenz von 50 MHz im Virtex-II FPGA mit anschließender Hochrechnung auf den AT697E (LEON2-FT) Prozessor bestimmt und sind in Tabelle 8.1 aufgeführt.

	Zeit [ms]	
	verlustlos	***c* = 6**
Beschreiben des *Tile*-Speichers	20	20
Lesen der komprimierten Daten (inkl. 1 Vergleich je *Pass*)	25	13
Tier-2 / *Rate Control*		
Byte-Anpassung	-	4
Neuberechnung des Schwellwertes	-	3
Durchsuchen aller *Passes* (1 Vergleich je *Pass*)	-	2
Auffüllen der verbleibenden *Codestream*-Daten (16 Vergleiche je *Pass*)	-	2
Erzeugung des JPEG2000 *Codestream* (inkl. Kopieren der Daten)	20	8
Summe	65	52

Tabelle 8.1: Ausführungszeiten der Softwareroutinen bei Verwendung des IDA JPEG2000 Core am Testbild Mars Full (512 × 512 × 12 Bit)

Abgesehen von der Zeit für das Beschreiben des *Tile*-Speichers sind die angegebenen Zeiten vom Kompressionsfaktor abhängig. Es zeigen die Messungen, dass die Software schnell genug ausgeführt werden kann und die Hardware damit der bestimmende Faktor bei der Kompressions-

zeit ist. Eine weitere Ausgliederung von Softwareroutinen in Hardwareroutinen ist somit nicht erforderlich und eine richtige Partitionierung in Hardware und Software kann bestätigt werden.

Zur Kompression der Testbilder ergeben sich die in Tabelle 8.2 gezeigten Werte. Sie resultieren aus dem Herunterskalieren der Ergebnisse aus Abschnitt 7.3.11 mit den Annahmen aus Abschnitt 7.3.12. Ein Bild der Größe 1k × 1k × 14 Bit kann durchschnittlich innerhalb von 1,31 s verlustlos komprimiert werden, so dass die geforderte Bildrate von 1-2 s mit der Hardwarebeschleunigung weitgehend online komprimiert werden kann.

Kompressionsfaktor *c*	verlustlos	2	4	6	8	10	12	14	16
Durchsatz [MBit/s]	11,2	11,5	14,2	16,5	18,2	19,9	21,2	22,5	23,5

Tabelle 8.2: Durchschnittlicher Durchsatz der Kompression

8.1.2 Zuverlässigkeit

Verglichen zur strahlungsfesten DPU ohne Hardwarebeschleunigung gibt es nur geringe Veränderungen bei der Zuverlässigkeit. Der RT54SX-S FPGA wird durch einen RTAX-S FPGA mit sehr ähnlicher Zuverlässigkeit ersetzt, wodurch Zwischenspeicher für die Sensor-Schnittstelle und Raumsonden-Schnittstellen entfallen können. Verglichen zu Kapitel 5 ist eine leicht angestiegene Zuverlässigkeit auf 0,983450 zu verzeichnen.

8.1.3 Einfluss strahlungsbedingter Fehler

Tabelle 8.3 fasst die Auftrittswahrscheinlichkeiten für die statischen Fehler sowie die Verfügbarkeit zusammen. Die Werte der Fehlerraten nach Abschnitt 5.2 sind sehr ähnlich.

	Venus Express	**ExoMars**	**polarer LEO**
Fehlerrate [1/Tag]	3,58E-06	2,74E-06	3,45E-06
Fehlerrate [1/Jahr]	1,31E-03	9,99E-04	1,26E-03
Verfügbarkeit	0,99999999	0,99999999	0,99999999

Tabelle 8.3: Gesamtfehlerrate und Verfügbarkeit, Wiederherstellungszeit 10 s

8.1.4 Leistungsbedarf

Der Leistungsbedarf ergibt sich zu:

Leistungsprofil	**Leistungsbedarf [W]**
Ruhezustand	1,4
Vollbetrieb	2,1

Tabelle 8.4: Leistungsbedarf

8.1.5 Platzbedarf, Volumen und Masse

Für die Ausführung als starre Platine bei doppelseitiger Bestückung mit 14 Ebenen ergeben sich für den Platzbedarf, das Volumen und die Masse folgende Werte:

Platzbedarf [cm^2]	Volumen [cm^3]	Masse [g]
170	218	215

Tabelle 8.5: Strahlungsfeste DPU mit Hardwarebeschleunigung, Bestückungshöhe 7 bzw. 4 mm

8.2 Strahlungstolerante DPU mit IDA JPEG2000 Core

Alternativ zu einer Implementierung in der strahlungsfesten DPU, kann der IDA JPEG2000 Core in die strahlungstolerante DPU nach Kapitel 6 eingesetzt werden. Deren Kernkomponente, der RT-Virtex-II FPGA, beinhaltet noch ungenutzte Ressourcen, so dass die RT-Virtex-II basierende Hardwarebeschleunigung aus Abschnitt 7.3 ohne Änderungen am generellen DPU-Aufbau implementiert werden kann. Die verwendeten Bauteile sind mit denen aus Kapitel 6 identisch und es ergeben sich die gleichen mechanischen Kennwerte sowie die gleiche Zuverlässigkeit. Der in Bild 8.2 dargestellte Aufbau enthält als wesentlichen Unterschied zu Bild 6.1 den IDA JPEG2000 Core innerhalb des SRAM-basierten RT-Virtex-II FPGA zur Steigerung der Performanz. Der IDA JPEG2000 Core wird hier in seiner Grundkonfiguration verwendet. Der LEON2-Prozessor wird mit einer Taktfrequenz von 50 MHz, der IDA JPEG2000 Core mit einer Taktfrequenz von 90 MHz betrieben.

Die Anbindung des IDA JPEG2000 Core an das Prozessorsystem (siehe Abschnitt 7.3.6, CPU/Coder-Schnittstelle), u. a. zum Schreiben der *Tile*-Daten oder Lesen der komprimierten Werte, kann auf verschiedenen Wegen erfolgen. Bei dem verwendeten LEON2-Prozessor kann die Ankopplung des Core als I/O Modul auf den Speicherbus erfolgen, wie in Abschnitt 8.1 bei der strahlungsfesten Implementierung geschehen. Mit Verwendung des LEON2-Prozessors als IP Core erhält man zudem die Möglichkeit, den Core z. B. als Busmaster mit DMA-Funktionalität an den AMBA AHB Systembus anzubinden. Mit dieser Implementierung ergeben sich gegenüber der Implementierung als I/O Modul schnellere Transferraten zwischen dem IDA JPEG2000 Core und dem Prozessorsystem. Bei Implementierung der CPU/Coder-Schnittstelle als Busmaster kann dieser schnelle *Burst*-Übertragungen initiieren. Gegenüber der Implementierung als Slave auf den Bus bringt die Anbindung eines Masters den wichtigen Vorteil, dass dieser selbst entscheidet, was er zu unterstützen hat. Eine Kombination Anbindung über I/O Bereich für Steuerungszwecke mit einer Anbindung als Busmaster für den hohen Datentransfer ist ebenfalls denkbar. Trotz langsameren Speicherbusses verglichen zur strahlungsfesten DPU ist eine vollständige I/O Implementierung für den Betrieb des IDA JPEG2000 Core in seiner Grundkonfiguration schnell genug und die Einführung einer Busmaster-Implementierung ist nicht erforderlich. Es ist demnach nicht sinnvoll, auf Kosten

eines aufwendigeren Aufbaus und zusätzlich notwendiger Ressourcen die Anbindung als Busmaster zu realisieren.

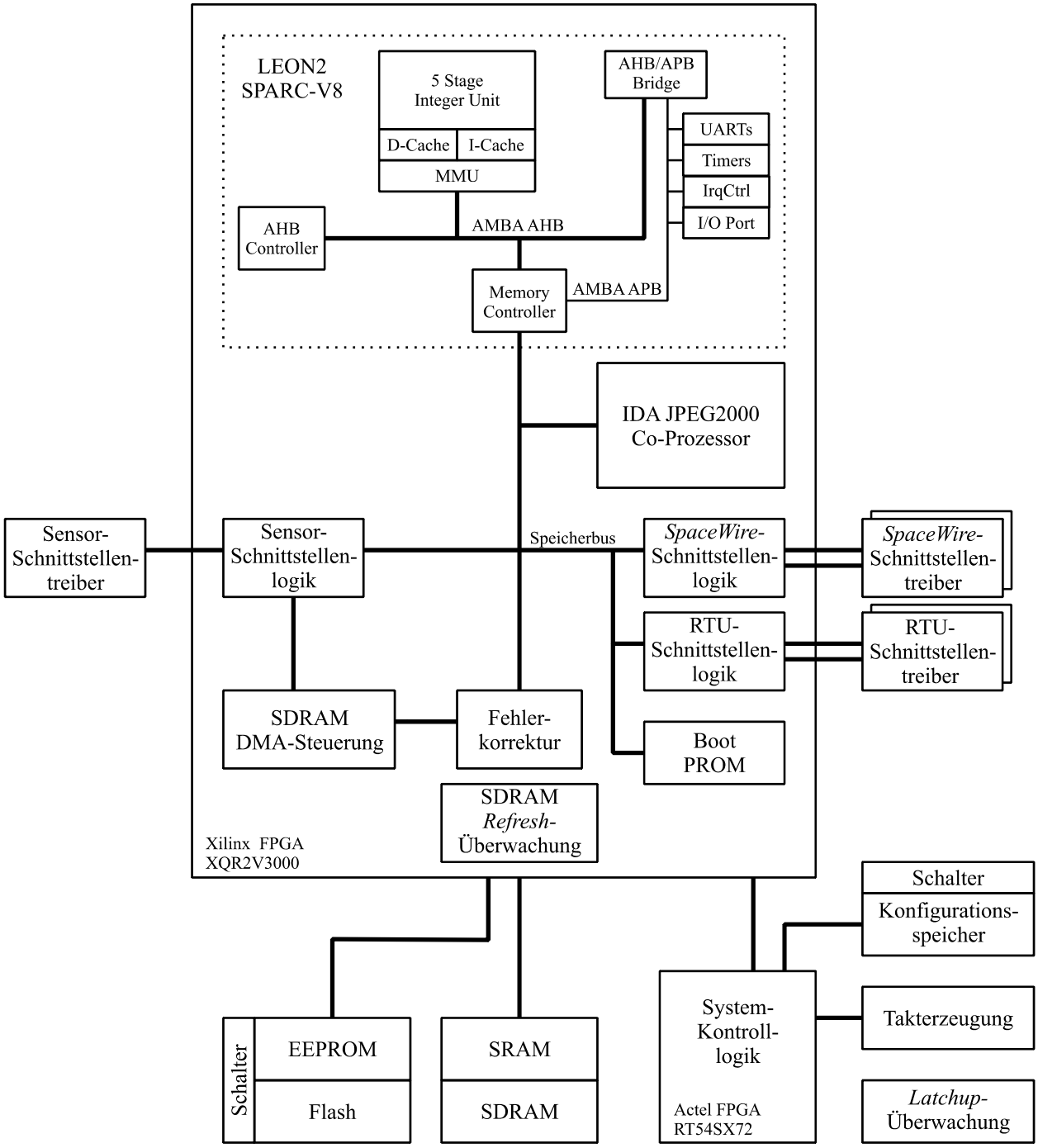

Bild 8.2: Strahlungstolerante DPU mit IDA JPEG2000 Core

8.2.1 Performanz

Bei der Bestimmung der Performanz ist es auch hier entscheidend, ob die durch die Software auszuführenden Routinen genügend schnell ausgeführt werden können, um zu verhindern, dass die Ausführungszeit des IDA JPEG2000 Core durch Wartezeiten verringert wird. Die Ausführungszeiten für die Softwareroutinen sind mit dem Aufbau eines Prozessors mit 50 MHz ermittelt und die hochgerechneten Werte sind in Tabelle 8.1 für die strahlungsfeste DPU mit dem AT697E (LEON2-FT) Prozessor für eine Taktfrequenz von 100 MHz dargestellt worden. Trotz halber Prozessor-Taktfrequenz und damit einer Reduzierung auf ca. 50 % beim Durchsatz der Softwareroutinen ist die Ausführungsgeschwindigkeit noch hoch genug, so dass sich für die Performanz der strahlungstoleranten DPU mit IDA JPEG2000 Core die in Abschnitt 7.3 gezeigten Werte ergeben (Tabelle 7.5, Tabelle 7.6 und Bild 7.15). Die richtig gewählte Partitionierung des JPEG2000-Algorithmus in Hardware und Software kann auch für den hier verwendeten Prozessor bestätigt werden. Die verlustlose Kompression eines Bildes der Größe 1k × 1k × 14 Bit kann innerhalb von durchschnittlich 1,01 s erfolgen. Die geforderte Bildrate von 1-2 s kann folglich online komprimiert werden.

8.2.2 FPGA-Ressourcenbedarf

Der Ressourcenbedarf innerhalb der Kernkomponente RT-Virtex-II FPGA ist in Tabelle 8.6 aufgelistet und zur Bestimmung der Strahlungsfestigkeit erforderlich. Es zeigt sich, dass innerhalb des FPGA noch genügend Ressourcen für weitere Performanzsteigerungen verfügbar sind.

Modul	***Slices***		**Benutzer-Flipflops**		**18 Bit × 18 Bit Multipl.**		**18 kBit *Block RAMs***	
	abs.	**%**	**abs.**	**%**	**abs.**	**%**	**abs.**	**%**
LEON2	3.821	26,7	2.101	7,3	4	4,2	22	22,9
NVRAM-Schnittstelle	60	0,4	120	0,4	-	-	-	-
Fehlerkorrektur	133	0,9	260	0,9	-	-	-	-
Sensor-Schnittstelle	108	0,7	203	0,7	-	-	2	2,1
SDRAM DMA-Steuerung	69	0,5	77	0,3	-	-	-	-
SDRAM-Überwachung	100	0,7	120	0,4	-	-	-	-
RTU-Schnittstelle	134	0,9	254	0,9	-	-	1	1,0
SpaceWire-Schnittstelle	361	2,5	491	1,7	-	-	1	1,0
Boot-PROM	15	0,1	20	0,1	-	-	4	4,2
IDA JPEG2000 Core - Grundkonfiguration	2.703	18,9	2.706	9,4	3	3,1	34	35,4
Summe	7.503	52,3	6.352	22,2	7	7,3	64	66,7

Tabelle 8.6: Ressourcenbedarf innerhalb des XQR2V3000 FPGA

8.2.3 Zuverlässigkeit

Die bestimmenden Faktoren zur Berechnung der Zuverlässigkeit sind mit Implementierung einer Hardwarebeschleunigung in die strahlungstolerante DPU nach Kapitel 6 nicht verändert worden und es ergibt sich dieselbe Zuverlässigkeit.

8.2.4 Einfluss strahlungsbedingter Fehler

Analog zu Kapitel 6 wird die Fehlerrate der DPU im Wesentlichen durch den RT-Virtex-II FPGA mit den verwendeten Ressourcen entsprechend Tabelle 8.6 bestimmt. Für die Hauptmodule des RT-Virtex-II FPGA sind die Werte für die betrachteten Missionen in Tabelle 8.7 dargestellt und in Tabelle 8.8 zusammengefasst. Es ist ersichtlich, dass der IDA JPEG2000 Core mehr als 50 % Einfluss auf die Fehlerrate hat. Davon wiederum entfallen ca. 80 % auf Datenfehler im *Block RAM*, die nicht zu einem Systemzusammenbruch führen müssen. Aufgrund dieser Datenfehler sowie aufgrund möglicher Datenfehler im LEON2-Modul dürfte die tatsächliche Rate für einen Systemzusammenbruch insgesamt niedriger liegen.

	Modul Fehlerrate [1/Tag]		
Modul	**Venus Express**	**ExoMars**	**polarer LEO**
LEON2	6,14E-03	1,67E-03	1,73E-03
IDA JPEG2000 Core	7,58E-03	2,07E-03	2,13E-03

Tabelle 8.7: Fehlerrate für den IDA JPEG2000 Core und den LEON2-Prozessor innerhalb des XQR2V3000 FPGA

	Venus Express	**ExoMars**	**polarer LEO**
Fehlerrate [1/Tag]	1,50E-02	4,08E-03	4,17E-03
Fehlerrate [1/Jahr]	5,46E+00	1,49E+00	1,52E+00
Fehlerrate Massenspeicher [1/Jahr]	2,61E-02	7,16E-03	7,30E-03
Verfügbarkeit	0,99999827	0,99999953	0,99999952

Tabelle 8.8: Gesamtfehlerrate und Verfügbarkeit, Wiederherstellungszeit 10 s

8.2.5 Leistungsbedarf

Der notwendige Leistungsbedarf ist in Tabelle 8.9 aufgeführt. Ein Großteil der Leistung ist für den LEON2-Prozessor notwendig. Wenn eine Reduzierung der Kompressionsperformanz in Kauf genommen werden kann, so kann es sinnvoll sein, den IDA JPEG2000 Core anstelle von 90 MHz mit demselben Takt zu versehen, mit dem der Prozessor betrieben wird (z. B. 50 MHz). Dieses führt zu einer Reduzierung des Leistungsbedarfs.

Leistungsprofil	Leistungsbedarf [W]
Ruhezustand	1,9
Vollbetrieb	2,8

Tabelle 8.9: Leistungsbedarf

8.2.6 Platzbedarf, Volumen und Masse

Es ergeben sich dieselben Werte wie für die strahlungstolerante DPU ohne Beschleunigung.

Platzbedarf [cm^2]	Volumen [cm^3]	Masse [g]
110	119	120

Tabelle 8.10: Strahlungstolerante DPU mit IDA JPEG2000 Core, Bestückungshöhe 6 bzw. 3 mm

8.3 Vergleich der Implementierungswege

Im Folgenden werden die besprochenen Implementierungsarten

- Strahlungsfeste DPU (ohne Massenspeicher, siehe Kapitel 5)
- Strahlungstolerante DPU (siehe Kapitel 6)
- Strahlungsfeste DPU mit IDA JPEG2000 Core (siehe Abschnitt 8.1)
- Strahlungstolerante DPU mit IDA JPEG2000 Core (siehe Abschnitt 8.2)

hinsichtlich der Auswahlkriterien Performanz, Zuverlässigkeit, strahlungsbedingte Fehlerrate, Größe, Masse und Leistungsbedarf, aber auch bzgl. der Erweiterungsmöglichkeiten miteinander verglichen. Der Vergleich ermöglicht die Auswahl der passenden Konfiguration für eine Anwendung.

8.3.1 Performanz

Tabelle 8.11, Tabelle 8.12 und Bild 8.3 zeigen die gravierenden Unterschiede in der Performanz bei Verwendung einer Hardwarebeschleunigung, gemessen an der Kompression eines Bildes mit dem JPEG2000-Algorithmus. Am Beispiel des Kompressionsfaktors 8 kann mit Verwendung der Hardwarebeschleunigung die Kompressionsgeschwindigkeit bei der strahlungsfesten DPU um den Faktor 11, bei der strahlungstoleranten Version sogar um den Faktor 29 erhöht werden. Eine

erwünschte durchschnittliche Bildakquisitionsrate von einem Bild der Größe 1k × 1k × 14 Bit alle 1-2 s kann lediglich mit Verwendung einer Hardwarebeschleunigung für eine Online-Kompression erzielt werden. Eine Kompression in Software führt dagegen zu einer drastischen Verringerung der durchschnittlichen Bildakquisitionsraten, mit der Folge einer starken Reduzierung des wissenschaftlichen Nutzens des Instruments. Bild 8.4 zeigt den Unterschied der Bildqualität anhand zweier Testbilder. Sie liegen im Bereich von 1 dB und sind vergleichbar zu Untersuchungen der Abhängigkeit der Bildqualität von der *Tile*-Größe [Kam04]. Bei den Softwareimplementierungen sowie in der strahlungsfesten Hardwareimplementierung mit externem *Tile*-Speicher ist eine *Tile*-Größe von 256×256, in der strahlungstoleranten Hardwareimplementierung eine Größe von 128×128 gewählt. Die Verwendung einer Hardwarebeschleunigung ohne externen *Tile*-Speicher erlaubt es nicht, auf einfachem, Ressourcen schonendem Weg, die *Tile*-Größe und damit eine erhöhte Bildqualität zu erreichen. Bei der RT-Virtex-II basierten Hardwarebeschleunigung kann eine Erweiterung des *Tile*-Speichers z. B. durch Übergang zu größerem Baustein mit größerem internen Speicher oder aber durch Auslagerung des DWT-Speichers in einen externen Speicher, vergleichbar zur strahlungsfesten Beschleunigung, erfolgen.

	Durchsatz [MBit/s]			
Implementierungsart	**Venus**	**Mars Full**	**Mars DaoVallis**	**Mars Surface**
Strahlungsfeste DPU	0,78	0,72	0,67	0,62
Strahlungstolerante DPU	0,40	0,37	0,34	0,32
Strahlungsfeste DPU mit IDA JPEG2000 Core	12,50	11,40	10,80	10,20
Strahlungstolerante DPU mit IDA JPEG2000 Core	16,30	14,80	14,00	13,30

Tabelle 8.11: Vergleich durchschnittlicher Durchsatz bei verlustloser Kompression

Implementierungsart	**Ausführungszeit [s]**	**Bildrate [1/s]**
Strahlungsfeste DPU	20,97	0,048
Strahlungstolerante DPU	41,94	0,024
Strahlungsfeste DPU mit IDA JPEG2000 Core	1,31	0,763
Strahlungstolerante DPU mit IDA JPEG2000 Core	1,01	0,995

Tabelle 8.12: Vergleich verlustlose Kompression von Bildern der Größe 1k × 1k × 14 Bit

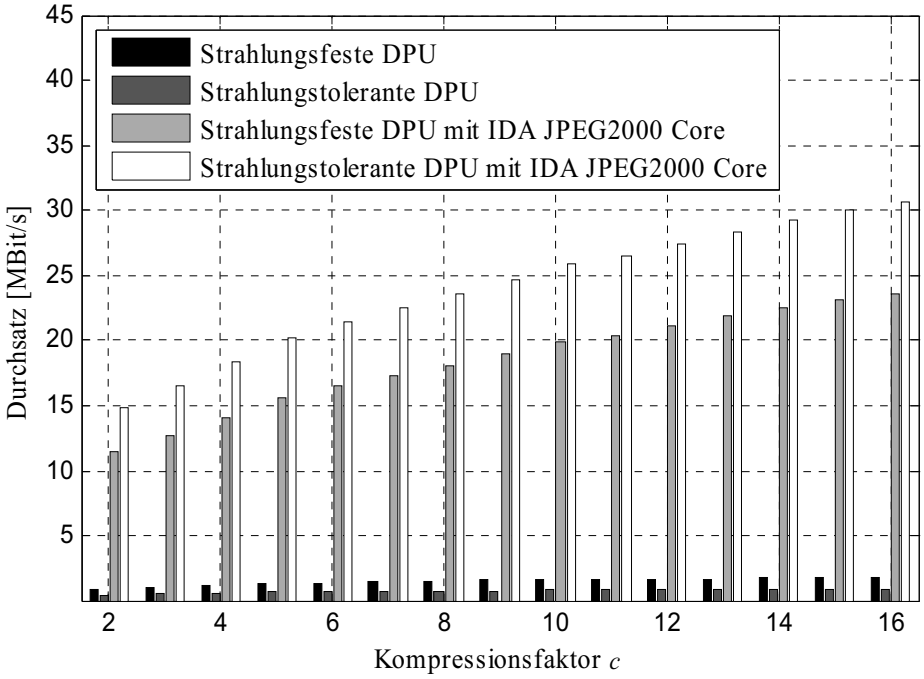

Bild 8.3: Vergleich durchschnittlicher Durchsatz bei verlustbehafteter Kompression

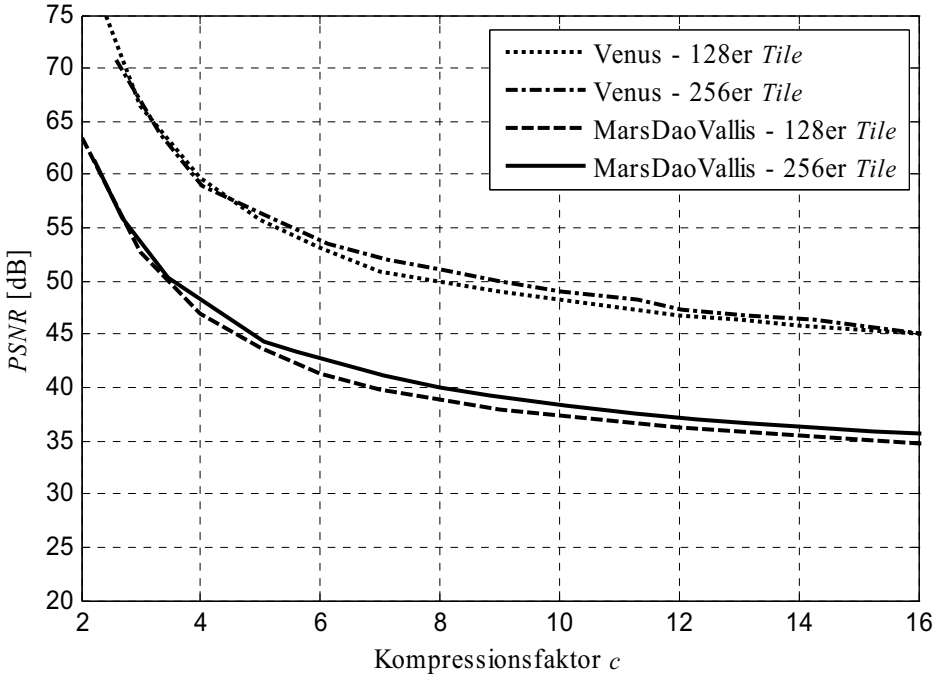

Bild 8.4: Vergleich Bildqualität, *Tile*-Größe 128 × 128 bzw. 256 × 256

8.3.2 Zuverlässigkeit

In Tabelle 8.13 ist die Zuverlässigkeit der verschiedenen Implementierungsarten dargestellt. Es ist ersichtlich, dass es keine signifikanten Unterschiede gibt und die strahlungsfesten Implementierungen nicht unbedingt eine höhere Zuverlässigkeit aufweisen müssen. Zusätzlich führen Bauteile, wie sie z. B. für den *Latchup*-Schutzschalter erforderlich sind, bei den strahlungstoleranten Implementierungen zu einer reduzierten Zuverlässigkeit. Mit der Einführung von Fehlerkorrektur und Redundanz kann jedoch eine hohe Baugruppen-Zuverlässigkeit erzielt werden. Die verwendeten SRAM-Bausteine verdeutlichen dieses sehr gut. Ein defekter strahlungstoleranter Baustein kann im Einsatz mit einer Fehlerkorrektur toleriert werden, so dass eine höhere Baugruppen-Zuverlässigkeit als bei den strahlungsfesten Bausteinen erreicht wird. Die strahlungsfesten SRAM-Bausteine weisen zwar eine höhere Bauteilzuverlässigkeit auf, sie werden aufgrund sehr geringer SEU-Empfindlichkeit jedoch ohne Fehlerkorrektur betrieben und der Ausfall eines Bausteins kann folglich nicht abgefangen werden.

Implementierungsart	Zuverlässigkeit
Strahlungsfeste DPU	0,982636
Strahlungstolerante DPU	0,984535
Strahlungsfeste DPU mit IDA JPEG2000 Core	0,983450
Strahlungstolerante DPU mit IDA JPEG2000 Core	0,984535

Tabelle 8.13: Vergleich Zuverlässigkeit

Es ist hervorzuheben, dass die strahlungstolerante Implementierung der DPU mit entsprechenden Maßnahmen die gleiche Zuverlässigkeit erreicht wie eine strahlungsfeste Implementierung.

8.3.3 Sicherheit gegen strahlungsbedingte Fehler

Tabelle 8.14 zeigt die jährliche Fehlerrate, die durch statische Fehler im System erzeugt werden. Es ist ein starker Unterschied zwischen den strahlungsfesten und strahlungstoleranten Ansätzen erkennbar. Dieses ist in der hohen Strahlungsempfindlichkeit der verwendeten RT-Virtex-II FPGA bei strahlungstolerantem Ansatz begründet. Da unter den Fehlern auch Datenfehler sind, führt nicht jeder Fehler zu einem Systemzusammenbruch und die tatsächliche temporäre Ausfallrate ist niedriger. Aufgrund der Verwendung einer *Refresh*-Überwachung sind in den strahlungstoleranten Implementierungen Fehler im Bildspeicher um den Faktor 100 unwahrscheinlicher. Hierdurch wird vermieden, dass es bei einem Systemzusammenbruch zu einem Verlust von gespeicherten Bildern kommt.

Bei dem strahlungstoleranten Ansatz sind die Erkennung eines Fehlers und die Durchführung einer Wiederherstellungsprozedur mit bestmöglichen Wiederherstellungserfolgen von hoher Bedeutung. Es ergeben sich die in Tabelle 8.15 gezeigten Verfügbarkeiten der einzelnen

Implementierungsarten. Beim strahlungstoleranten Ansatz kann trotz der vergleichsweise hohen Strahlungsempfindlichkeit mit einer recht hohen Verfügbarkeit gerechnet werden. Diese liegt mit Werten größer $1\text{-}10^{-6}$ im Bereich typischer fehlertoleranter Systeme [Sho02]. Für den schlechtesten Fall ergibt sich eine Nichtverfügbarkeit von lediglich 1 min pro Jahr, was für viele Missionen tolerierbar sein sollte.

	Fehlerrate [1/Jahr]		
Implementierungsart	**Venus Express**	**ExoMars**	**polarer LEO**
Strahlungsfeste DPU	1,31E-03	9,99E-04	1,26E-03
Strahlungstolerante DPU	2,69E+00	7,35E-01	7,58E-01
Strahlungsfeste DPU mit IDA JPEG2000 Core	1,13E-04	7,91E-05	7,91E-05
Strahlungstolerante DPU mit IDA JPEG2000 Core	5,46E+00	1,49E+00	1,54E+00

Tabelle 8.14: Vergleich Gesamtfehlerrate

	Verfügbarkeit		
Implementierungsart	**Venus Express**	**ExoMars**	**polarer LEO**
Strahlungsfeste DPU	0,99999999	0,99999999	0,99999999
Strahlungstolerante DPU	0,99999915	0,99999977	0,99999976
Strahlungsfeste DPU mit IDA JPEG2000 Core	0,99999999	0,99999999	0,99999999
Strahlungstolerante DPU mit IDA JPEG2000 Core	0,99999827	0,99999953	0,99999952

Tabelle 8.15: Vergleich Verfügbarkeit, Wiederherstellungszeit 10 s

8.3.4 Leistungs- und Energiebedarf

Die Unterschiede im Leistungsbedarf der vorgestellten Implementierungswege sind in Tabelle 8.16 zusammengefasst. Es zeigt sich, dass alle Ansätze einen vergleichsweise hohen Leistungsbedarf im Ruhezustand aufweisen. Die strahlungstoleranten Ansätze weisen gegenüber den vergleichbaren strahlungsfesten Ansätzen einen höheren Leistungsbedarf auf, der insbesondere in der Verwendung eines SRAM-basierten FPGA und der Implementierung eines Massenspeichers begründet ist. Ist ein einfacher Prozessor aus Performanzgründen nicht ausreichend und wird aus diesem Grunde eine Kompression in Hardware verwendet, dann führt dieses auch zu einer Erhöhung der Grundlast.

	Leistungsbedarf [W]	
Implementierungsart	**Ruhezustand**	**Vollbetrieb**
Strahlungsfeste DPU	1,2	1,4
Strahlungstolerante DPU	1,7	2,4
Strahlungsfeste DPU mit IDA JPEG2000 Core	1,4	2,1
Strahlungstolerante DPU mit IDA JPEG2000 Core	1,9	2,8

Tabelle 8.16: Vergleich Leistungsbedarf

Unter der Annahme, dass die Kompressionszeit bestimmend ist und die DPU stets im Vollbetrieb genutzt wird, ergeben sich folgende Werte für den Energiebedarf zur Kompression eines Bildes:

Implementierungsart	**Energiebedarf pro Bild [Ws]**
Strahlungsfeste DPU	29,36
Strahlungstolerante DPU	100,66
Strahlungsfeste DPU mit IDA JPEG2000 Core	2,75
Strahlungstolerante DPU mit IDA JPEG2000 Core	2,82

Tabelle 8.17: Vergleich Energiebedarf zur Kompression eines Bildes der Größe 1k × 1k × 14 Bit

Am wenigsten effektiv ist die strahlungstolerante DPU gefolgt von der strahlungsfesten DPU, beide ohne Hardwarebeschleunigung. Die DPUs mit IDA JPEG2000 Core benötigen gegenüber der strahlungsfesten und strahlungstoleranten DPU ohne Hardwarebeschleunigung weit weniger, d. h. um ca. den Faktor 10 bzw. 35 weniger Energie zur Durchführung der Kompression. Tatsächlich werden die DPUs auch in weniger energieeffizienten Zuständen, z. B. im Ruhezustand betrieben. Die Verfügbarkeit einer schnellen Kompression eröffnet jedoch die Möglichkeit, den Energiebedarf stark zu optimieren. Die Grundlast und die Last bei Kompression sind mit einem möglichst hohen zeitlichen Anteil an effektiver Betriebsweise zu verbinden. Der zeitliche Anteil des weniger effizienten Betriebs, wie der Ruhezustand, sollte minimiert oder aber durch Reduzierung des Systemtaktes energieeffizienter gestaltet werden. Im Betrieb bedeutet dieses z. B., dass die DPU nur kurze Messsequenzen mit hohem Durchsatz ausführt und anschließend ausgeschaltet oder in einen energiereduzierten Ruhezustand überführt wird. Die Möglichkeit, bei Verfügbarkeit einer Hardwarebeschleunigung effiziente Energieprofile zu durchfahren, kann für Missionen mit sehr wenig verfügbarer Energie, z. B. für Missionen mit Landeeinheiten, von hohem Wert sein. Würde man nichtflüchtige Varianten des Massenspeichers einsetzen, z. B. mit Verwendung von Flash-Speicher, so ergäben sich noch weitere Wege den Energiebedarf zu optimieren. Mit der Möglichkeit, die DPU ohne Verlust der Bilddaten im Massenspeicher auszuschalten, kann die zeitliche Abhängigkeit von Arbeitsschritten wie Akquisition, Bildverarbeitung, Kompression stark reduziert werden. Dieses kann für einen energieeffizienten Betrieb genutzt werden.

Insgesamt zeigt sich, dass die Verwendung einer Hardwarebeschleunigung aufgrund der weit geringeren Menge der zu bewegenden Ladung hinsichtlich des Energiebedarfs sehr effektiv ist. Hiermit kann ein sehr energiesparendes System aufgebaut werden und es ergeben sich viele Möglichkeiten, die zur Verfügung stehende Energie optimal zu nutzen.

8.3.5 Platzbedarf, Volumen und Masse

Am Beispiel einer starren Platine mit doppelseitiger Bestückung zeigt Tabelle 8.18 die Unterschiede im Platzbedarf, Volumen und in der Masse. Für andere Bauformen ist mit einem leichten Anstieg dieser Werte zu rechnen. Gegenüber den strahlungstoleranten Implementierungen benötigen die strahlungsfesten Ansätze trotz fehlendem Massenspeicher 45 % bzw. 54 % mehr Platz, 72 % bzw. 83 % mehr Volumen und weisen eine um 71 % bzw. 79 % erhöhte Masse auf. Dieses ist vor allem darin begründet, dass die strahlungstoleranten Versionen viele kleine und leichte Bauteile mit Plastikgehäuse beinhalten.

Implementierungsart	Platzbedarf [cm^2]	Volumen [cm^3]	Masse [g]
Strahlungsfeste DPU	160	205	205
Strahlungstolerante DPU	110	119	120
Strahlungsfeste DPU mit IDA JPEG2000 Core	170	218	215
Strahlungstolerante DPU mit IDA JPEG2000 Core	110	119	120

Tabelle 8.18: Vergleich Platzbedarf, Volumen und Masse bei starrer Platine

8.3.6 Flexibilität

Zur Beurteilung der Flexibilität der verschiedenen Implementierungsarten können verschiedene Gruppierungen vorgenommen werden:

- Flexibilität zur Steigerung der Performanz
- Flexibilität im Betrieb
- Flexibilität während der Entwicklung

Eine **Steigerung der Performanz** kann für die strahlungsfeste DPU ohne IDA JPEG2000 Core im Wesentlichen durch die Software erfolgen, d. h. durch effizientere Ausführung von Routinen. Die erzielbare Erhöhung der Performanz ist hierbei jedoch sehr begrenzt. Dagegen ist mit der Verfügbarkeit großer FPGAs (RT-Virtex-II oder RTAX-S) durch Implementierung von z. B. missionsspezifischen Berechnungen in Hardware eine hohe Performanzsteigerung möglich. Zu nennen ist z. B. eine hardwareunterstützte Durchführung einer *Flat Field*-Korrektur oder die Durchführung von Faltungsoperationen. Die DPU kann sehr gezielt optimiert werden und

deutliche Performanzgewinne bei optimiertem Energiebedarf erreichen. Verglichen zum RTAX-S FPGA, der sehr begrenzten internen Speicher aufweist, bietet der RT-Virtex-II FPGA im strahlungstoleranten Ansatz selbst mit Implementierung des IDA JPEG2000 Core noch sehr viele Ressourcen. Zudem bietet er schnelle fest verdrahtete Multiplikationseinheiten, die nicht erst durch programmierbare Logik aufgebaut werden müssen. Für schnelle Übertragungsraten erlaubt der strahlungstolerante Ansatz die direkte Anbindung einer Hardwarebeschleunigung an den internen Prozessorbus. Diese Möglichkeit ist bei den strahlungsfesten DPUs nicht vorhanden, da der LEON2-Prozessor in Form eines dedizierten Bausteins eingesetzt ist und hierbei kein Zugriff auf den internen Prozessorbus besteht. Bei den strahlungstoleranten Ansätzen sind daher die besten Möglichkeiten zur Steigerung der Performanz zu erwarten.

Im **Betrieb** können Anpassungen lediglich durch Änderungen der Software erfolgen. Die rein auf Software basierten Lösungen bieten die größte Flexibilität z. B. bei der Anpassung der Kompressionsalgorithmen. Andererseits erhält man mit der Verfügbarkeit einer hardware-unterstützten schnellen Kompression eine hohe Flexibilität im operationellen Betrieb. Diese wird mit Vorhandensein eines DPU-internen Massenspeichers und eines großen nichtflüchtigen Speichers, wie in den strahlungstoleranten DPUs gegeben, verstärkt. Eine Nichtverfügbarkeit eines Massenspeichers würde einige Operationen nicht oder nur sehr eingeschränkt ermöglichen.

Während der **Entwicklung** ist ein großer zeitlicher Anteil zur Implementierung der Software und der FPGA-Funktionen notwendig. Bei Verwendung von SRAM-basierten FPGAs wie dem RT-Virtex-II FPGA können aufgrund der Mehrfachprogrammierbarkeit mit moderatem Aufwand neue Funktionen implementiert und getestet werden. Im Gegensatz dazu ist die Implementierung eines großen Designs in den nur einmal programmierbaren RTAX-S FPGA weniger anwendungsfreundlich. Zwar wird durch intensive Nutzung von Simulationen sowie ggf. Entwicklung mit vergleichbaren mehrfach programmierbaren Bausteinen die Fehlerwahrscheinlichkeit stark reduziert, oft treten aber noch Fehler auf, häufig im Zusammenspiel mit FPGA-externen Komponenten, und ein Austausch des einmal programmierbaren Bauteils ist erforderlich. Der Nutzen der Mehrfachprogrammierbarkeit ist umso größer, je mehr Ressourcen der FPGA bietet und je größer das FPGA-Design damit sein kann.

9 Zusammenfassung und Ausblick

Traditionell beinhaltet der Entwurf eines Rechners für wissenschaftliche Instrumente auf Raumsonden die Verwendung strahlungsfester Bauteile. Aufgrund ihrer geringen Verfügbarkeit und ihrer geringeren Performanz bei höherer Masse werden zunehmend Baugruppen aus einer strahlungsfesten in eine strahlungstolerante Implementierung mit zugehörigen Schutzmaßnahmen (z. B. Fehlerkorrektur beim Speicher) überführt. Hiermit erhöht sich die Anzahl der Möglichkeiten beim Entwurf eines Instrumentenrechners.

Das Ziel bei der Implementierung eines Instrumentenrechners für eine kompakte Kamera ist es, bei tolerierbaren Werten für die Zuverlässigkeit und die Fehlertoleranz (Toleranz gegenüber strahlungsbedingten Fehlern) sowie geringen Ressourcenanforderungen von Platzbedarf, Volumen, Masse und Energie eine hohe wissenschaftliche Ausbeute beim Betrieb der Kamera zu erzielen. Häufig ist der wissenschaftliche Nutzen durch die Zeit zur Durchführung einer hochwertigen Bildkompression beschränkt.

Unter Berücksichtigung der Besonderheiten für Elektronik im Weltraumeinsatz sowie der Besonderheiten des hochwertigen *State of the Art* Kompressionsalgorithmus JPEG2000, wurde sowohl ein strahlungsfester als auch strahlungstoleranter Aufbau eines Instrumentenrechners für eine kompakte Kamera vorgestellt. Während der strahlungsfeste Ansatz auf einem strahlungsfesten Prozessor basiert, bildet beim strahlungstoleranten Entwurf ein neuartiger weltraumtauglicher SRAM-basierter FPGA mit vielen Ressourcen die Kernkomponente der DPU. Durch Integration möglichst vieler Module samt Prozessor in diesem FPGA kann eine hohe Packungsdichte erzielt werden. Sowohl für die strahlungsfeste als auch die strahlungstolerante DPU wurde die Performanz exemplarisch an der Ausführung des JPEG2000 Kompressionsalgorithmus ermittelt.

Zur Steigerung der Performanz ist ein spezieller IDA JPEG2000 Core entwickelt worden, der die Kompressionsgeschwindigkeit energieeffizient um ein vielfaches erhöht. Der Core ist für die Anforderungen innerhalb eines Kamerainstruments entworfen worden und beinhaltet mit einer schlanken Grundkonfiguration ein u. A. in Performanz und Ressourcenverbrauch skalierbares Design.

Nach Implementierung dieser Hardwarebeschleunigung auf die strahlungsfeste und die strahlungstolerante Version wurden die Ansätze unter verschiedenen Kriterien miteinander verglichen. Unter den vorgestellten Ansätzen zeichnet sich die strahlungstolerante DPU mit IDA JPEG2000 Core durch die höchste Performanz bei gleichzeitig kleinstem Aufbau aus. Mit Verwendung des IDA JPEG2000 Core kann die Kompressionsgeschwindigkeit um bis zu Faktor 30 ansteigen. Für typische Bilderzeugungsraten von einem Bild der Größe 1k × 1k × 14 Bit je 1-2 s wird mit dem IDA JPEG2000 Core durch eine Online-Kompression eine hohe Ausbeute des Instruments erzielt. Die energieeffiziente Ausführung der Kompression kann für einen energieoptimierten Betrieb der DPU genutzt werden, wie es zum Beispiel auf Missionen mit Landefahrzeugen notwendig ist. Hier kann die zur Verfügung stehende Energie gezielt eingesetzt

werden. Bei den strahlungstoleranten Ansätzen kann eine Zuverlässigkeit erreicht werden, die vergleichbar ist zu den strahlungsfesten Ansätzen. Die strahlungsfesten Ansätze weisen erwartungsgemäß die geringsten Raten strahlungsbedingter Fehler auf. Bei den strahlungstoleranten DPUs kann für die ausgewählten Missionen eine moderate Strahlungsfestigkeit (1 min Nichtverfügbarkeit pro Jahr) erzielt werden. Die Nutzbarkeit größerer Speicher, sowohl in Form des flüchtigen Massenspeichers als auch in Form des Flash-basierten nichtflüchtigen Speichers führt zu einer starken Erhöhung der operationellen Flexibilität. Die strahlungstolerante DPU vereint die Ausführung sowohl allgemeiner Funktionen mit Standard-CPU als auch spezieller Funktionen (hier z. B. exemplarisch die JPEG2000-Kompression) mit hoher Effektivität durch dedizierte Logik innerhalb eines FPGA-Bausteins und dieses bei kompaktem Aufbau des Instrumentenrechners. Es bestehen innerhalb des FPGA noch umfangreiche Reserven zur weiteren Steigerung der Performanz und damit zur Erhöhung des wissenschaftlichen Nutzens des Instruments (z. B. Implementierung spezieller Filter).

Die strahlungstolerante DPU mit IDA JPEG2000 Core ist ein sinnvoller Ansatz für Kamerainstrumente auf vielen Missionen, bei denen strahlungsbedingte Nichtverfügbarkeit in der Größenordnung von wenigen Minuten pro Jahr toleriert werden kann. Der Ansatz vereint eine hohe Performanz mit einer kompakten Bauform und einem hohen Grad an Flexibilität und energieoptimierten Betriebsmöglichkeiten für eine kompakte Kamera.

In Zukunft ist mit einer verstärkten Integration von Modulen (modulare Arbeitsweise mit IP Cores) in zunehmend größeren FPGAs zu rechnen. Rechenintensive Operationen, die bisher in Software durchgeführt werden, werden in steigendem Maße durch dedizierte Hardwaremodule ersetzt und sowohl schneller als auch energieeffizienter ausgeführt. Bei SRAM-basierten FPGAs ist zu erwarten, dass zunehmend Mechanismen verwendet werden, die eine Anpassung der FPGA-Implementierung noch während des Einsatzes im Weltraum erlauben. Der Entwicklung solcher Mechanismen steht schon jetzt prinzipiell nichts im Wege. Durch eine partielle Rekonfigurierbarkeit können z. B. FPGA-interne Module an definierten Schnittstellen ausgetauscht werden und es wird eine hohe Flexibilität vergleichbar der von reinen Softwarelösungen erzielt. Zur Erhöhung der Performanz könnte in Zukunft die kommerzielle Virtex-II Pro Familie Verwendung finden, die fest verdrahtete Prozessoren mit frei definierbarer Logik innerhalb eines Bausteins vereint. Es kann erwartet werden, dass in Bezug auf die Fehlerdetektierung besondere Vorgehensweisen erforderlich sind. Bevor sie für den Einsatz im Weltraum in Betracht kommen, sind umfangreiche Qualifikationsmaßnahmen, wie die Bestimmung des Verhaltens unter Strahleneinfluss, notwendig.

A Anhang

A.1 Strahlungskurven und Fehlerberechnung

Zur Ermittlung der Häufigkeit von strahlungsbedingten Fehlern sind die missionsabhängigen LET-Spektren sowie die Wirkungsquerschnitte der verwendeten Bauteile erforderlich. Es stehen jedoch nicht immer Wirkungsquerschnitte zur Verfügung. Insbesondere bei hoch qualifizierten Bauteilen geben Hersteller vielfach die Häufigkeit strahlungsbedingter Fehler für den ungünstigsten Fall an.

Bei der Berechnung der strahlungsbedingten Fehler wurden folgende Annahmen gemacht:

- Einschaltdauer: 6 h pro Tag
- Schirmung: 1 g/cm^2
- *Solar Flare* Bedingungen gehen in Form des *Peak 5 Min* (CREME96 Model basierend auf dem extremen *Solar Flare* im Okt. 1989) mit Auftrittswahrscheinlichkeit von 0,25 % ein
- Auswirkungen von Protonen für die betrachteten Missionen sind vernachlässigbar
- Lesezugriffe auf NVRAM dominieren gegenüber Schreibzugriffen

LET-Spektren für verschiedene Missionen

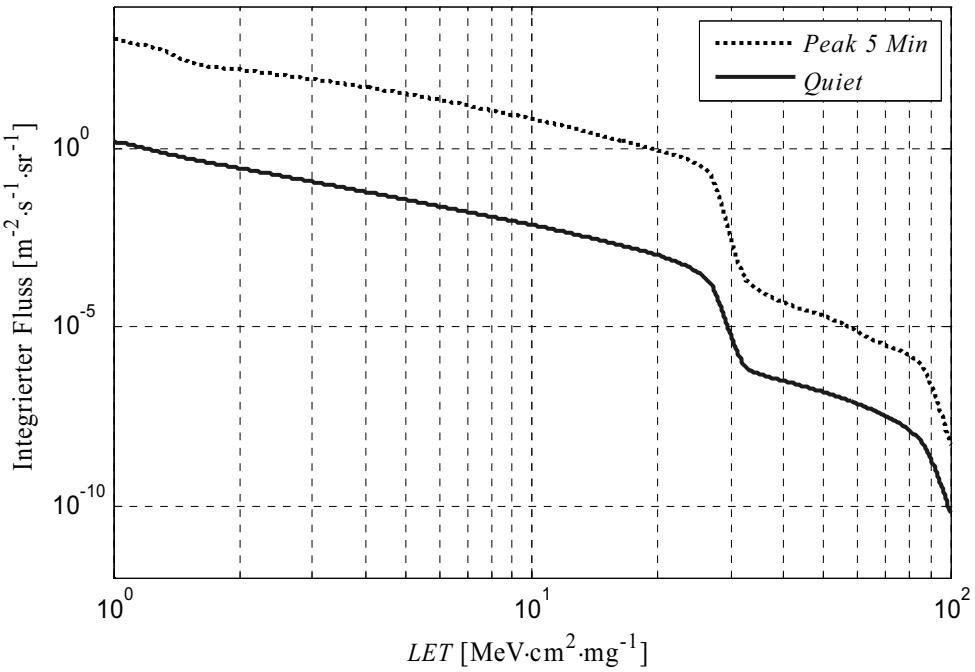

Bild A.1: LET-Spektrum für Venus Express [Soe02]

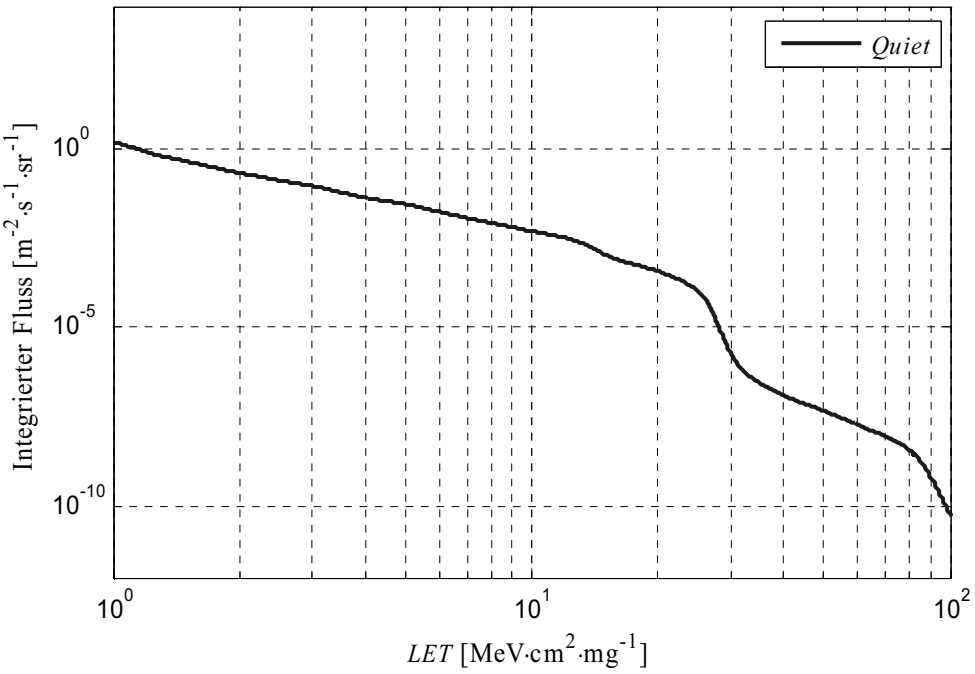

Bild A.2: LET-Spektrum für ExoMars [CNE04]

In Bild A.2 sind nur GCR dargestellt, da *Solar Flare Heavy Ions* so stark abgeschirmt sind, dass die GCR dominierend sind. Die Mars Atmosphäre entspricht einer Abschirmwirkung von ca. 65,7 mm Al [CNE04].

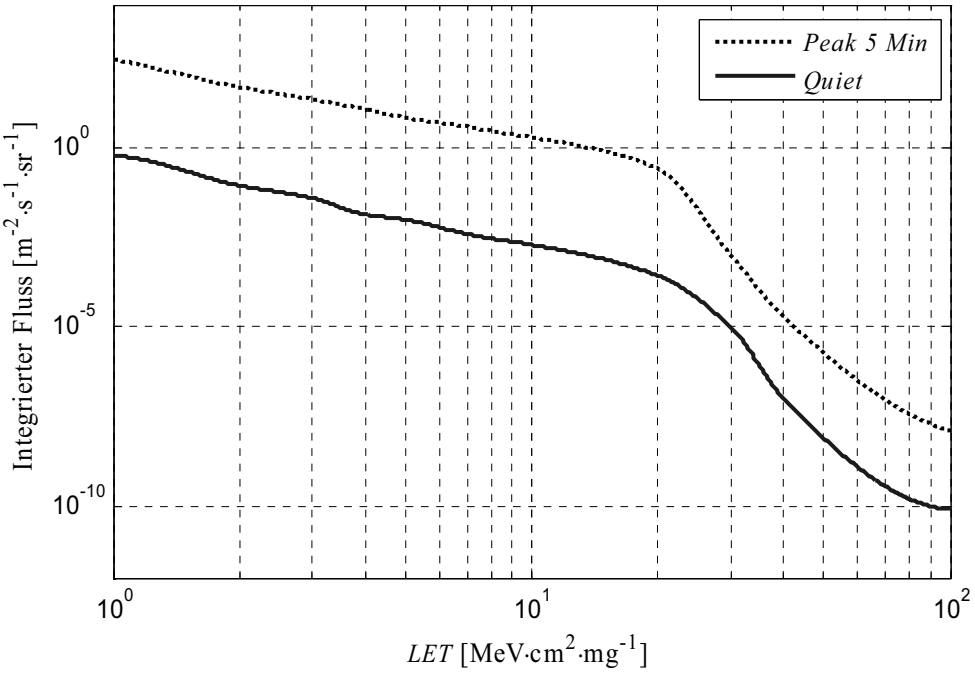

Bild A.3: LET-Spektrum für einen polaren LEO in 900 km [Hey01], (Protonen unberücksichtigt)

Wirkungsquerschnitte für verwendete Bauteile

Testbild	σ_{sat}	LET_{th}	*Width*	*Power*	pro …
Actel_FF_10MHz[1]	3,0E-09	30,0	15,0	1,0	Bit
Virtex-II BRAM[2]	4,2E-08	1,0	17,0	0,9	Bit
Virtex-II Konfiguration[2]	3,8E-08	1,0	33,0	0,8	Bit
Virtex-II LUT[2]	3,8E-08	1,0	33,0	0,8	Bit
Virtex-II FF[2]	3,8E-08	1,0	33,0	0,8	Bit
Virtex-II POR SEFI[2]	2,5E-06	1,5	22,0	1,2	Bauteil
Virtex-II SMAP SEFI[2]	1,7E-06	1,5	17,0	1,0	Bauteil
Virtex-II JCFG SEFI[2]	2,5E-07	1,5	17,0	1,0	Bauteil
SRAM 4 MBit UT8R512X8 RH[3]	1,7E-07	46,0	18,2	2,0	Bit
SRAM 4 MBit kommerziell[4]	7,8E-09	1,4	18,0	2,0	Bit
PROM RH UT28F256LVQL[5]	7,6E-12	34,0	18,0	1,8	Bit
EEPROM 28LV010 Lesen[6]	3,0E-12	30,0	18,0	2,1	Bit
EEPROM 28LV010 Schreiben[6]	5,0E-09	10,0	18,0	2,0	Bit
EEPROM 79LV0408 Lesen[6,7]	3,0E-12	90,0	15,0	2,0	Bit
EEPROM 79LV0408 Schreiben[6,7]	3,0E-09	18,0	15,0	2,0	Bit
Flash 528 Byte Zwischenspeicher[8]	3,5E-07	7,0	18,2	2,0	Bit
SDRAM 256 MBit[9]	2,0E-09	5,0	50,0	1,0	Bit
SDRAM 256 MBit SEFI[9]	5,0E-04	18,0	10,0	1,0	Bauteil

Tabelle A.1: Wirkungsquerschnitte mit Weibull-Parameter (siehe Gleichung A.1)

[1] J. J. Wang, Actel, RT54SX72SU Heavy Ion Beam Test Report, August 2004

[2] G. M. Swift, JPL, Xilinx Single Event Effects, 1st Consortium Report: Virtex-II Static SEU Characterization, JPL, Single Event Effects Consortium, Technical report, Januar 2004 (TAM Werte)

3 Aeroflex, Inc, UT8Q512K8 512Kx8 SRAM, Datenblatt, März 2003

[4] Aeroflex, Inc, Single Event Effects Qualification, UT8Q512K8 (RQ02), Lot 6ZVL04 4Mbit SRAM, Juli 2003 (Samsung)

[5] Aeroflex, Inc, UT28F256LVQL Radiation-Hardened 32k x 8 PROM, Datenblatt, Juni 2005 (Logic SEU Onset LET)

[6] Maxwell Technologies, 28LV010 3.3V 1 Megabit (128k x 8-Bit) EEPROM, Datenblatt, Rev 6, März 2003

[7] Maxwell Technologies, 79LV0408 Low Voltage 4 Megabit (512k x 8Bit) EEPROM, Datenblatt, Rev 7, Januar 2005

[8] D. N. Nguyen, S. M. Guertin, G. M. Swift and A. H. Johnston, Radiation Effects on Advanced Flash Memories, IEEE Transactions on Nuclear Science, 46 (6), 1744 (1999); Dezember 1999 (128 MBit, Samsung)

[9] Maxwell Technologies, SEE Radiation Test Report, HM5225405B, April 2003 (256 MBit, Elpida)

Weibull-Verteilung

$$\sigma(LET) = \sigma_{Sat} \cdot (1 - e^{-(\frac{LET - LET_{th}}{Width})^{Power}}) \tag{A.1}$$

mit

- σ_{sat}: Sättigungswert des Wirkungsquerschnitts
- LET_{th}: LET Schwellwert
- *Width*: Kurvenparameter
- *Power*: Kurvenparameter

Reed Solomon-Fehlerkorrektur

Mit Verwendung der symbol-orientierten *Reed Solomon*-Fehlerkorrektur zusammen mit regelmäßigem Schrubben des Speichers wird die tatsächliche Anzahl von Fehlern im Datenwort stark reduziert. Die Bitfehlerrate BER (*Bit Error Rate*) für nicht korrigierbare Fehler innerhalb eines Schrubbintervalls ergibt sich nach [Fic98] zu:

$$BER \approx \frac{b}{2} \cdot (n-1) \cdot \alpha^2{}_{SEU} \cdot T_S \tag{A.2}$$

mit

- n: Anzahl Symbole pro Wort
- b: Anzahl Bits pro Symbol
- α_{SEU}: SEU Fehlerrate
- T_S: Schrubbintervall.

A.2 Zuverlässigkeitsbestimmung

Zur Bestimmung der Zuverlässigkeiten der DPUs wurden folgende Annahmen gemacht:

- Betriebsdauer: $T = 5$ Jahre
- Umgebungstemperatur: $\vartheta_A = 50$ °C
- Einschaltdauer: 6 h pro Tag
- *Dormancy* Faktor: 0,1

Berechnung

Die für die Berechnung der Zuverlässigkeit bestimmenden Bauteilewerte sind den Datenblättern oder Qualitätsreports der Hersteller entnommen. Stehen diese nicht zur Verfügung, so wird ein Modell zur Bestimmung der Ausfallrate nach dem MIL-Handbook [DOD91] verwendet.

Der Lambda-Wert λ_{All} für N Bauteilkategorien ergibt sich zu:

$$\lambda_{All} = \sum_{i=1}^{N} k_{FM,i} \cdot k_{D,i} \cdot n_i \cdot \lambda_i \tag{A.3}$$

mit

- Ausfallrate λ_i eines Bauteils i
- Anzahl n_i an Bauteilen innerhalb einer Bauteilkategorie
- Korrekturfaktor $k_{FM,i}$ aus der *Failure Mode and Effects Analysis*
- Korrekturfaktor $k_{D,i}$, der sich aus dem *Derating* (Unterbeanspruchung) des jeweiligen Bauteils ergibt.

Bei der Berechnung der System-Überlebenswahrscheinlichkeit $R_{All}(T)$ wird die Auslastung $k_{Active,i}$ der i-ten Bauteilkategorie berücksichtigt:

$$R(T) = \exp\left(-T \cdot 10^{-6} \cdot \sum_{i=1}^{N} k_{FM,i} \cdot k_{D,i} \cdot n_i \cdot \lambda_i \cdot k_{Active,i}\right) \tag{A.4}$$

mit

- Einschaltdauer/T $k_{Active,i}$ der jeweiligen Bauteilkategorie.

Für die redundanten Bereiche werden die Überlebenswahrscheinlichkeiten separat berechnet. Sie gehen anschließend als Baugruppen-Überlebenswahrscheinlichkeit mit in die Berechnung der

System-Überlebenswahrscheinlichkeit ein. Zur Bestimmung des redundanten Einflusses werden folgende Berechnungsvorschriften verwendet:

Für 1-aus-2 Redundanz (z. B. für Raumsonden-Schnittstelle) [Sho02]:

$$R_{red} = 2 \cdot R - R^2 \tag{A.5}$$

und für 1-aus-6 Redundanz (z. B. für Massenspeicher) [Sho02]

$$R_{red} = R^6 + 6 \cdot R^5 \cdot (1 - R). \tag{A.6}$$

Außerhalb der redundanten Bereiche wird ein Serienmodell zur Bestimmung der Zuverlässigkeit verwendet.

Failure Mode and Effects Analysis

Veränderte Bauteileigenschaften außerhalb der Spezifikation führen nicht zwangsläufig zum Ausfall der gesamten DPU. Der Faktor $k_{FM,i}$ gibt die Gewichtung eines Bauteils an der Ausfallwahrscheinlichkeit der DPU an. Die *Failure Mode and Effects Analysis* reduziert $k_{FM,i}$ entsprechend des Ausfallmechanismus:

$$k_{FM,i} = 1 - \sum_{k=1}^{M} k_{P,i,k} \cdot P_{FM,i,k} \tag{A.7}$$

mit

- Anzahl M unterschiedlicher Bauteilfunktionen
- Anteil $k_{P,i,k}$ der Bauteile entsprechend der Bauteilfunktion k
- Anteil $P_{FM,i,k}$ eines Fehlers der Bauteilfunktion k an der Bauteilausfallwahrscheinlichkeit

Die *Failure Mode and Effects Analysis* wird bei der DPU für die Widerstände und Kondensatoren durchgeführt. Der Bereich der Raumsonden-Schnittstellen, die in allen DPUs gleich aufgebaut sind, ist separat ausgeführt, da diese Schnittstellen redundant aufgebaut sind und viele Widerstände und Kondensatoren enthalten. Für die anderen Bauteile wird mit $k_{FM,i} = 1$ ein Totalausfall und damit der *Worst Case* betrachtet. Die in Klammern dargestellten Werte werden als nicht kritisch angesehen.

		Pull up / Pull down	Serien	Übrige
	P_{Open}	0,44	0,44	0,44
	P_{Change}	(0,56)	(0,56)	0,56
Strahlungsfeste DPU	k_P	0,53	0,33	0,13
	k_{FM}	= 1- (0,53·0,56+0,33·0,56) = 0,52		
Strahlungsfeste DPU mit Massenspeicher	k_P	0,38	0,54	0,08
	k_{FM}	= 1- (0,38·0,56+0,54·0,56) = 0,48		
Strahlungstolerante DPU	k_P	0,44	0,47	0,08
	k_{FM}	= 1- (0,44·0,56+0,47·0,56) = 0,49		
Strahlungsfeste DPU mit IDA JPEG2000 Core	k_P	0,62	0,23	0,15
	k_{FM}	= 1- (0,62·0,56+0,23·0,56) = 0,52		
Raumsonden-Schnittstellen	k_P	0,08	0,92	0,00
	k_{FM}	= 1- (0,08·0,56+0,92·0,56) = 0,44		

Tabelle A.2: Widerstände

		Abblock	Übrige
	P_{Open}	(0,18)	0,18
	P_{Change}	(0,03)	0,03
	P_{Short}	0,78	0,78
Strahlungsfeste DPU	k_P	0,98	0,02
	k_{FM}	= 1- (0,98·0,18+0,98·0,03) = 0,79	
Strahlungsfeste DPU mit Massenspeicher	k_P	0,97	0,03
	k_{FM}	= 1- (0,97·0,18+0,97·0,03) = 0,80	
Strahlungstolerante DPU	k_P	0,97	0,03
	k_{FM}	= 1- (0,97·0,18+0,97·0,03) = 0,80	
Strahlungsfeste DPU mit IDA JPEG2000 Core	k_P	0,98	0,02
	k_{FM}	= 1- (0,98·0,18+0,98·0,03) = 0,79	
Raumsonden-Schnittstellen	k_P	0,50	0,50
	k_{FM}	= 1- (0,50·0,18+0,50·0,03) = 0,90	

Tabelle A.3: Keramik-Kondensatoren

		Abblock
	P_{Open}	(0,36)
	P_{Change}	(0,17)
	P_{Short}	0,46
Strahlungsfeste DPU	k_P	1,00
	k_{FM}	= 1- (1,00·0,36+1,00·0,17) = 0,47
Strahlungsfeste DPU mit Massenspeicher	k_P	1,00
	k_{FM}	= 1- (1,00·0,36+1,00·0,17) = 0,47
Strahlungstolerante DPU	k_P	1,00
	k_{FM}	= 1- (1,00·0,36+1,00·0,17) = 0,47
Strahlungsfeste DPU mit IDA JPEG2000 Core	k_P	1,00
	k_{FM}	= 1- (1,00·0,36+1,00·0,17) = 0,47
Raumsonden-Schnittstellen	k_P	1,00
	k_{FM}	= 1- (1,00·0,36+1,00·0,17) = 0,47

Tabelle A.4: Tantal-Kondensatoren
(gleiche Werte, da nur als Abblockkondensator verwendet)

Derating

Die Widerstände und Kondensatoren werden unterbeansprucht und es ergeben sich hieraus entsprechende *Derating*-Faktoren, die zur Bestimmung der Zuverlässigkeit eingesetzt werden.

- **Widerstände**

Für die Widerstände ergibt sich ein *Stress*-Faktor *S* im ungünstigsten Fall von

$$S = \textit{Applied Power} / \textit{Rated Power} = 33\ \text{mW} / 100\ \text{mW} = 0{,}33. \tag{A.8}$$

Hiermit ergibt sich nach [DOD91] ein *Derating*-Faktor $k_D = 1/3$.

- **Keramik-Kondensatoren**

Für die Keramik-Kondensatoren ergibt sich ein *Stress*-Faktor *S* im ungünstigsten Fall von

$$S = \textit{Applied Power} / \textit{Rated Power} = 5\ \text{V} / 50\ \text{V} = 0{,}1. \tag{A.9}$$

Hiermit ergibt sich nach [DOD91] ein *Derating*-Faktor $k_D = 1/100$.

- **Tantal-Kondensatoren**

Für die Tantal-Kondensatoren ergibt sich ein *Stress*-Faktor *S* im ungünstigsten Fall von

$$S = \textit{Applied Power} / \textit{Rated Power} = 3{,}3\ \text{V} / 10\ \text{V} = 0{,}33. \tag{A.10}$$

Hiermit ergibt sich nach [DOD91] ein *Derating*-Faktor $k_D = 1/15$.

Ergebnisse

Die Resultate der Zuverlässigkeitsbestimmung sind in den Tabellen der folgenden Seiten dargestellt.

Bauteil	Anzahl	Qualität	Spezif.	Lambda	Quelle	Aktivität	FM	*Derating*	R(T)
AT697E	1	QML-Q	MIL-PRF-38535	0,078000	Qualitätsreport	25,00%	1,00	1,00	0,998890
RT54SX72S	1	QML-Q	MIL-PRF-38535	0,022000	Qualitätsreport	25,00%	1,00	1,00	0,999687
UT8R512X8	10	QML-Q	MIL-PRF-38535	0,044000	Qualitätsreport	25,00%	1,00	1,00	0,993756
UT28F256LVQL	1	QML-Q	MIL-PRF-38535	0,076000	Qualitätsreport	0,07%	1,00	1,00	0,999665
Quarz-Oszillator	1	Class B	MIL-PRF-55310	0,018746	MIL-HDBK-217F	25,00%	1,00	1,00	0,999733
UT4ACS162245S	4	QML-Q	MIL-PRF-38535	0,019752	MIL-HDBK-217F	25,00%	1,00	1,00	0,998876
54HCTS14	1	QML-V	MIL-PRF-38535	0,001002	MIL-HDBK-217F	25,00%	1,00	1,00	0,999986
IRHF9130	1	JANS	MIL-PRF-19500	0,055642	MIL-HDBK-217F	25,00%	1,00	1,00	0,999208
2N2907	1	JANS	MIL-PRF-19500	0,002487	MIL-HDBK-217F	25,00%	1,00	1,00	0,999965
1N4150UR-1	1	JANS	MIL-PRF-19500	0,000118	MIL-HDBK-217F	25,00%	1,00	1,00	0,999998
Widerstände	15	R	MIL-PRF-55342	0,100000	Datenblatt	25,00%	0,52	0,33	0,996306
Keramik-Kondensatoren	86	R	MIL-PRF-55681	0,100000	Datenblatt	25,00%	0,79	0,01	0,999033
Tantal-Kondensatoren	18	R	MIL-PRF-55365	0,100000	Datenblatt	25,00%	0,47	0,07	0,999197
Stecker	2	---	MIL-P-45204	0,000353	MIL-HDBK-217F	25,00%	1,00	1,00	0,999990
79LV0408 EEPROM MCM	1	Maxwell Class K	MIL-STD-883	0,081197	MIL-HDBK-217F	0,07%	1,00	1,00	0,999642
EEPROM Modul (6 x EEPROM)									0,999998
S/C I/F 26CLV31RH	1	QML-Q	MIL-PRF-38535	0,095113	MIL-HDBK-217F	25,00%	1,00	1,00	0,998647
S/C I/F 26CLV32RH	2	QML-Q	MIL-PRF-38535	0,018218	MIL-HDBK-217F	25,00%	1,00	1,00	0,999481
S/C I/F UT54LVDS031LV	1	QML-Q	MIL-PRF-38535	0,095113	MIL-HDBK-217F	25,00%	1,00	1,00	0,998647
S/C I/F UT54LVDS032LV	1	QML-Q	MIL-PRF-38535	0,018218	MIL-HDBK-217F	25,00%	1,00	1,00	0,999741
S/C I/F Widerstände	26	R	MIL-PRF-55342	0,100000	Datenblatt	25,00%	0,44	0,33	0,994586
S/C I/F Keramik-Kondensatoren	10	R	MIL-PRF-55681	0,100000	Datenblatt	25,00%	0,90	0,01	0,999872
S/C I/F Stecker	1	---	MIL-P-45204	0,001084	MIL-HDBK-217F	25,00%	1,00	1,00	0,999985
S/C Schnittstelle einfach									0,990983
S/C Schnittstelle									0,999919
PCB	1	QML	MIL-PRF-55110	0,119094	MIL-HDBK-217F	25,00%	1,00	1,00	0,998306
Gesamt									0,982636

Tabelle A.5: Strahlungsfeste DPU

Bauteil	Anzahl	Qualität	Spezif.	Lambda	Quelle	Aktivität	FM	*Derating*	R(T)
AT697E	1	QML-Q	MIL-PRF-38535	0,078000	Qualitätsreport	25,00%	1,00	1,00	0,998890
RT54SX72S	1	QML-Q	MIL-PRF-38535	0,022000	Qualitätsreport	25,00%	1,00	1,00	0,999687
UT8R512X8	6	QML-Q	MIL-PRF-38535	0,044000	Qualitätsreport	25,00%	1,00	1,00	0,996249
UT28F256LVQL	1	QML-Q	MIL-PRF-38535	0,076000	Qualitätsreport	0,07%	1,00	1,00	0,999665
Quarz-Oszillator	1	Class B	MIL-PRF-55310	0,018746	MIL-HDBK-217F	25,00%	1,00	1,00	0,999733
UT4ACS162245S	4	QML-Q	MIL-PRF-38535	0,019752	MIL-HDBK-217F	25,00%	1,00	1,00	0,998876
54HCTS14	1	QML-V	MIL-PRF-38535	0,001002	MIL-HDBK-217F	25,00%	1,00	1,00	0,999986
HS9-139RH	1	QML-V	MIL-PRF-38535	0,003732	MIL-HDBK-217F	25,00%	1,00	1,00	0,999947
IRHF9130	3	JANS	MIL-PRF-19500	0,055642	MIL-HDBK-217F	25,00%	1,00	1,00	0,997627
2N2907	2	JANS	MIL-PRF-19500	0,002487	MIL-HDBK-217F	25,00%	1,00	1,00	0,999929
1N4150UR-1	1	JANS	MIL-PRF-19500	0,000118	MIL-HDBK-217F	25,00%	1,00	1,00	0,999998
Widerstände	26	R	MIL-PRF-55342	0,100000	Datenblatt	25,00%	0,48	0,33	0,994096
Keramik-Kondensatoren	87	R	MIL-PRF-55681	0,100000	Datenblatt	25,00%	0,80	0,01	0,999010
Tantal-Kondensatoren	18	R	MIL-PRF-55365	0,100000	Datenblatt	25,00%	0,47	0,07	0,999197
Stecker	2	---	MIL-P-45204	0,000353	MIL-HDBK-217F	25,00%	1,00	1,00	0,999990
79LV0408 EEPROM MCM	1	Maxwell Class K	MIL-STD-883	0,081197	MIL-HDBK-217F	0,07%	1,00	1,00	0,999642
EEPROM Modul (6 x EEPROM)									0,999998
S/C I/F 26CLV31RH	1	QML-Q	MIL-PRF-38535	0,095113	MIL-HDBK-217F	25,00%	1,00	1,00	0,998647
S/C I/F 26CLV32RH	2	QML-Q	MIL-PRF-38535	0,018218	MIL-HDBK-217F	25,00%	1,00	1,00	0,999481
S/C I/F UT54LVDS031LV	1	QML-Q	MIL-PRF-38535	0,095113	MIL-HDBK-217F	25,00%	1,00	1,00	0,998647
S/C I/F UT54LVDS032LV	1	QML-Q	MIL-PRF-38535	0,018218	MIL-HDBK-217F	25,00%	1,00	1,00	0,999741
S/C I/F Widerstände	26	R	MIL-PRF-55342	0,100000	Datenblatt	25,00%	0,44	0,33	0,994586
S/C I/F Keramik-Kondensatoren	10	R	MIL-PRF-55681	0,100000	Datenblatt	25,00%	0,90	0,01	0,999872
S/C I/F Stecker	1	---	MIL-P-45204	0,001084	MIL-HDBK-217F	25,00%	1,00	1,00	0,999985
S/C Schnittstelle einfach									0,990983
S/C Schnittstelle									0,999919
SDRAM	1	kommerziell	---	0,119000	IEEE Beitrag	25,00%	1,00	1,00	0,998307
Massenspeicher (6 x SDRAM)									0,999957
PCB	1	QML	MIL-PRF-55110	0,125841	MIL-HDBK-217F	25,00%	1,00	1,00	0,998210
Gesamt									0,981114

Tabelle A.6: Strahlungsfeste DPU mit Massenspeicher

Bauteil	Anzahl	Qualität	Spezif.	Lambda	Quelle	Aktivität	FM	*Derating*	R(T)
XQR2V3000	1	QML-Q	---	0,019000	Qualitätsreport	25,00%	1,00	1,00	0,999730
RT54SX72S	1	QML-Q	MIL-PRF-38535	0,022000	Qualitätsreport	25,00%	1,00	1,00	0,999687
XQR17V16RH	2	QML-Q	MIL-STD-883	0,017000	Qualitätsreport	25,00%	1,00	1,00	0,999516
Quarz-Oszillator	1	Class B	MIL-PRF-55310	0,018746	MIL-HDBK-217F	25,00%	1,00	1,00	0,999733
UT4ACS162245S	1	QML-Q	MIL-PRF-38535	0,019752	MIL-HDBK-217F	25,00%	1,00	1,00	0,999719
54HCTS14	1	QML-V	MIL-PRF-38535	0,001002	MIL-HDBK-217F	25,00%	1,00	1,00	0,999986
HS9-139RH	1	QML-V	MIL-PRF-38535	0,003732	MIL-HDBK-217F	25,00%	1,00	1,00	0,999947
IRHF9130	4	JANS	MIL-PRF-19500	0,055642	MIL-HDBK-217F	25,00%	1,00	1,00	0,996837
2N2907	3	JANS	MIL-PRF-19500	0,002487	MIL-HDBK-217F	25,00%	1,00	1,00	0,999894
1N4150UR-1	1	JANS	MIL-PRF-19500	0,000118	MIL-HDBK-217F	25,00%	1,00	1,00	0,999998
Widerstände	29	R	MIL-PRF-55342	0,100000	Datenblatt	25,00%	0,49	0,33	0,993280
Keramik-Kondensatoren	84	R	MIL-PRF-55681	0,100000	Datenblatt	25,00%	0,80	0,01	0,999044
Tantal-Kondensatoren	18	R	MIL-PRF-55365	0,100000	Datenblatt	25,00%	0,47	0,07	0,999197
Stecker	2	---	MIL-P-45204	0,000353	MIL-HDBK-217F	25,00%	1,00	1,00	0,999990
SRAM	1	---	---	0,131000	MIL-HDBK-217F	25,00%	1,00	1,00	0,998137
SRAM Modul (6 x SRAM)									0,999948
EEPROM	1	---	---	0,041000	MIL-HDBK-217F	0,07%	1,00	1,00	0,999819
EEPROM Modul (2 x EEPROM)									0,999999
Flash	1	---	---	0,080000	Qualitätsreport	0,07%	1,00	1,00	0,999647
Flash Modul (2 x Flash)									0,999999
S/C I/F 26CLV31RH	1	QML-Q	MIL-PRF-38535	0,095113	MIL-HDBK-217F	25,00%	1,00	1,00	0,998647
S/C I/F 26CLV32RH	2	QML-Q	MIL-PRF-38535	0,018218	MIL-HDBK-217F	25,00%	1,00	1,00	0,999481
S/C I/F UT54LVDS031LV	1	QML-Q	MIL-PRF-38535	0,095113	MIL-HDBK-217F	25,00%	1,00	1,00	0,998647
S/C I/F UT54LVDS032LV	1	QML-Q	MIL-PRF-38535	0,018218	MIL-HDBK-217F	25,00%	1,00	1,00	0,999741
S/C I/F Widerstände	26	R	MIL-PRF-55342	0,100000	Datenblatt	25,00%	0,44	0,33	0,994586
S/C I/F Keramik-Kondensatoren	10	R	MIL-PRF-55681	0,100000	Datenblatt	25,00%	0,90	0,01	0,999872
S/C I/F Stecker	1	---	MIL-P-45204	0,001084	MIL-HDBK-217F	25,00%	1,00	1,00	0,999985
S/C Schnittstelle einfach									0,990983
S/C Schnittstelle									0,999919
SDRAM Modul	1	kommerziell	---	0,119000	IEEE Beitrag	25,00%	1,00	1,00	0,998307
Massenspeicher (6 x SDRAM)									0,999957
PCB	1	QML	MIL-PRF-55110	0,136201	MIL-HDBK-217F	25,00%	1,00	1,00	0,998063
Gesamt									0,984535

Tabelle A.7: Strahlungstolerante DPU

Bauteil	Anzahl	Qualität	Spezif.	Lambda	Quelle	Aktivität	FM	*Derating*	R(T)
AT697E	1	QML-Q	MIL-PRF-38535	0,078000	Qualitätsreport	25,00%	1,00	1,00	0,998890
RT-AX4000	1	QML-Q	MIL-PRF-38535	0,007800	Qualitätsreport	25,00%	1,00	1,00	0,999889
UT8R512X8	8	QML-Q	MIL-PRF-38535	0,044000	Qualitätsreport	25,00%	1,00	1,00	0,995002
UT8R128X32	1	QML-Q	MIL-PRF-38535	0,044000	Qualitätsreport	25,00%	1,00	1,00	0,999374
UT28F256LVQL	1	QML-Q	MIL-PRF-38535	0,076000	Qualitätsreport	0,07%	1,00	1,00	0,999665
Quarz-Oszillator	1	Class B	MIL-PRF-55310	0,018746	MIL-HDBK-217F	25,00%	1,00	1,00	0,999733
UT4ACS162245S	4	QML-Q	MIL-PRF-38535	0,019752	MIL-HDBK-217F	25,00%	1,00	1,00	0,998876
54HCTS14	1	QML-V	MIL-PRF-38535	0,001002	MIL-HDBK-217F	25,00%	1,00	1,00	0,999986
IRHF9130	1	JANS	MIL-PRF-19500	0,055642	MIL-HDBK-217F	25,00%	1,00	1,00	0,999208
2N2907	1	JANS	MIL-PRF-19500	0,002487	MIL-HDBK-217F	25,00%	1,00	1,00	0,999965
1N4150UR-1	1	JANS	MIL-PRF-19500	0,000118	MIL-HDBK-217F	25,00%	1,00	1,00	0,999998
Widerstände	13	R	MIL-PRF-55342	0,100000	Datenblatt	25,00%	0,52	0,33	0,996798
Keramik-Kondensatoren	106	R	MIL-PRF-55681	0,100000	Datenblatt	25,00%	0,79	0,01	0,998809
Tantal-Kondensatoren	22	R	MIL-PRF-55365	0,100000	Datenblatt	25,00%	0,47	0,07	0,999019
Stecker	2	---	MIL-P-45204	0,000353	MIL-HDBK-217F	25,00%	1,00	1,00	0,999990
79LV0408 EEPROM MCM	1	Maxwell Class K	MIL-STD-883	0,081197	MIL-HDBK-217F	0,07%	1,00	1,00	0,999642
EEPROM Modul (6 x EEPROM)									0,999998
S/C I/F 26CLV31RH	1	QML-Q	MIL-PRF-38535	0,095113	MIL-HDBK-217F	25,00%	1,00	1,00	0,998647
S/C I/F 26CLV32RH	2	QML-Q	MIL-PRF-38535	0,018218	MIL-HDBK-217F	25,00%	1,00	1,00	0,999481
S/C I/F UT54LVDS031LV	1	QML-Q	MIL-PRF-38535	0,095113	MIL-HDBK-217F	25,00%	1,00	1,00	0,998647
S/C I/F UT54LVDS032LV	1	QML-Q	MIL-PRF-38535	0,018218	MIL-HDBK-217F	25,00%	1,00	1,00	0,999741
S/C I/F Widerstände	26	R	MIL-PRF-55342	0,100000	Datenblatt	25,00%	0,44	0,33	0,994586
S/C I/F Keramik-Kondensatoren	10	R	MIL-PRF-55681	0,100000	Datenblatt	25,00%	0,90	0,01	0,999872
S/C I/F Stecker	1	---	MIL-P-45204	0,001084	MIL-HDBK-217F	25,00%	1,00	1,00	0,999985
S/C Schnittstelle einfach									0,990983
S/C Schnittstelle									0,999919
PCB	1	QML	MIL-PRF-55110	0,125442	MIL-HDBK-217F	25,00%	1,00	1,00	0,998216
Gesamt									0,983450

Tabelle A.8: Strahlungsfeste DPU mit IDA JPEG2000 Core

A.3 Liste der verwendeten Bauteile

Bauteil	Hersteller	Bezeichnung	Gehäusetyp	Masse [g]	Platz-bedarf [cm^2]
LEON2-FT Prozessor	Atmel	AT697E	MCGA349	9,00	6,35
RT54SX72S FPGA	Actel	RT54SX72S	CQ208	8,80	12,25
RT-AX4000 FPGA	Actel	RT-AX4000	CQ352	27,40	29,16
Virtex-II-3000 FPGA	Xilinx	XQR2V3000	CG717	13,30	12,25
Konfig. PROM	Xilinx	XQR17V16RH	VQ44	1,44	0,40
SRAM RH 512k8	Aeroflex	UT8R512X8	FP36	3,76	5,26
SRAM RH 128k32	Aeroflex	UT8R128X32	QFP68	9,20	9,85
SRAM 512k8	IDT	IDT71V424S.	TSOP36	0,45	2,21
SDRAM 32M8	Samsung	K4S560832D	TSOP(II)54	0,53	2,68
PROM RH 32k8	Aeroflex	UT28F256LVQL	FP28	2,40	3,76
EEPROM 512k8	Maxwell	79LV0408	LDFP40 MCM	10,00	6,16
EEPROM 128k8	Maxwell	28LV010	Rad Pack 32	6,00	4,01
Flash 128 MBit	Samsung	KM29U128T	TSOP48	0,53	2,48
Treiber	Aeroflex	UT4ACS162245S	CFP48	1,60	2,78
Schmitt-Trigger	Intersil	54HCTS14MS	CFP14	0,55	1,20
Komparator	Intersil	HS9-138RH-T	CFP14	0,55	1,20
RS422-Treiber	Intersil	HS-26CLV31RH	CFP16	0,63	1,60
RS422-Empfänger	Intersil	HS-26CLV32RH	CFP16	0,63	1,60
LVDS-Treiber	Aeroflex	UT54LVDS031LV	CFP16	0,63	1,60
LVDS-Empfänger	Aeroflex	UT54LVDS032LV	CFP16	0,63	1,60
Quarz-Oszillator	EuroQuartz	EQXO-2000BM	DIL-8	2,45	1,64
P-Kanal MOSFET	IRF	IRHF9130	TO-39	1,15	0,80
PNP-Transistor	Microsemi	JANS2N2907	TO-18	0,29	0,42

Tabelle A.9: Verwendete aktive Bauteile

A.4 Abkürzungen

AMBA	**A**dvanced **M**icrocontroller **B**us **A**rchitecture
ASIC	**A**pplication **S**pecific **I**ntegrated **C**ircuit
AU	**A**stronomical **U**nit
BPC	Bitplane Coder
bpp	Bits per Pixel
CCD	**C**harged **C**oupled **D**evice
CLB	**C**onfigurable **L**ogic **B**locks
CP	**C**leanup **P**ass
CPU	**C**entral **P**rocessing **U**nit
DCM	**D**igital **C**lock **M**anager
DCT	**D**iscrete **C**osinus **T**ransformation **(**Diskrete Kosinus Transformation)
DDR SDRAM	**D**ouble **D**ata **R**ate **S**ynchronized **D**ynamic **R**andom **A**ccess **M**emory
DFC	**D**awn **F**raming **C**amera
DMA	**D**irect **M**emory **A**ccess
DPU	**D**ata **P**rocessing **U**nit
DWT	**D**iscrete **W**avelet **T**ransformation
EBCOT	**E**mbedded **B**lock **C**oding with **O**ptimized **T**runcation
EEPROM	**E**lectrically **E**rasable **P**rogrammable **R**ead **O**nly **M**emory
ESA	**E**uropean **S**pace **A**gency
FIFO	**F**irst **I**n **F**irst **O**ut
FIR	**F**inite **I**mpulse **R**esponse
FPGA	**F**ield **P**rogrammable **G**ate **A**rray
FPU	**F**loating **P**oint **U**nit
GCR	**G**alactic **C**osmic **R**ays
HI	**H**eavy **I**ons
I/F	Interface
I/O	**I**nput-**O**utput
IC	**I**ntegrated **C**ircuit (Integrierter Schaltkreis)
IDA	Institut für Datentechnik und Kommunikationsnetze
IP	**I**ntellectual **P**roperty
ISO	International Organization for Standardization
JPEG	**J**oint **P**hotographic **E**xperts **G**roup
LEO	**L**ow **E**arth **O**rbit
LET	**L**inear **E**nergy **T**ransfer (Massenbremsvermögen)
LPS	**L**east **P**robable **S**ymbol
LU	**L**atch**u**p
LUT	Lookup-Tabelle
LVDS	**L**ow **V**oltage **D**ifferential **S**ignaling
MIPS	**M**illion **I**nstructions **P**er **S**econd
MPS	**M**ore **P**robable **S**ymbol

MRC	**M**agnitude **R**efinement **C**oding
MRP	**M**agnitude **R**efinement **P**ass
MSE	**M**ean **S**quared **E**rror
NASA	**N**ational **A**eronautics and **S**pace **A**gency
NLPS	**N**ext **L**east **P**robable **S**ymbol
NMPS	**N**ext **M**ore **P**robable **S**ymbol
OCL	**O**n-board **C**ommand **L**anguage
OSIRIS	**O**ptical, **S**pectroscopic and **I**nfrared **R**emote **I**maging **S**ystem
PCB	**P**rinted **C**ircuit **B**oard
POR	**P**ower-**O**n **R**eset
PROM	**P**rogrammable **R**ead **O**nly **M**emory
PSNR	**P**eak **S**ignal to **N**oise **R**atio
RH	**R**adiation-**H**ardened
RLC	**R**un-**L**ength **C**oding
ROI	**R**egion **O**f **I**nterest
ROSINA	**R**osetta **O**rbiter **S**pectrometer for **I**on and **N**eutral **A**nalysis
RT	**R**adiation **T**olerant (strahlungstolerant)
RTE	**R**adiative **T**ransfer **E**quation
RTOS	**R**eal-**T**ime **O**perating **S**ystem
S/C	**S**pace **C**raft (Satellit, Sonde)
S/W	Software
SC	**S**ign **C**oding
SDRAM	**S**ynchronized **D**ynamic **R**andom **A**ccess **M**emory
SEE	**S**ingle **E**vent **E**ffect
SEFI	**S**ingle **E**vent **F**unctional **I**nterrupt
SEL	**S**ingle **E**vent **L**atchup
SEP	**S**ingle **E**nergetic **P**articles
SEU	**S**ingle **E**vent **U**pset
SoC	**S**ystem **O**n **C**hip
SOI	**S**ilicon **O**n **I**nsulator
SP	**S**ignificance **P**ass
SPARC	**S**calable **P**rocessor **ARC**hitecture
SPIHT	**S**et **P**artitioning **i**n **H**ierarchical **T**rees
SRAM	**S**tatic **R**andom **A**ccess **M**emory
TC	Telecommand
TM	Telemetrie
TMR	**T**riple **M**odule **R**edundancy
VHDL	**V**ery **H**igh Speed Integrated Circuit Hardware **D**escription **L**anguage
VMC	**V**enus **M**onitoring **C**amera
WFT	**W**indowed **F**ourier **T**ransformation
ZC	**Z**ero **C**oding

A.5 Formelzeichen

a	Skalierungsfaktor (*Wavelet*-Transformation)
A	Intervallbreite
b	Verschiebungswerte (*Wavelet*-Transformation)
B	Bitanzahl pro Bildpunkt
c	Kompressionsfaktor
C	Intervallbasis
d	Wanddicke
d	Bitmenge komprimierter Bilddaten
D	Dosis
D	Verzerrung (*Distortion*)
ΔD_i^k	Beitrag des *Pass k* vom *Codeblock i* zur Reduzierung der Verzerrung (*Distortion*)
ΔR_i^k	Beitrag des *Pass k* vom *Codeblock i* zur Länge eines *Bitstream* (*Rate*)
E	Energie
F	Fluenz
F_{HI}	*Heavy Ions*-Fluenz
F_{Pr}	Protonen-Fluenz
Φ	Partikelfluss
H_0	Entscheidungsgehalt
LET	Linear Energy Transfer
m	Masse
M	Bildgröße Y-Richtung
N	Bildgröße X-Richtung
$N_{SEU,\ HI}$	Fehlerrate (SEU, *Heavy Ions*)
$N_{SEU,\ Pr}$	Fehlerrate (SEU, Protonen)
p	Wahrscheinlichkeit
Q_e	LPS-Wahrscheinlichkeit
R	Länge eines *Bitstream* (*Rate*)
R_{SEU}	SEU-Rate
$R_{SEU,\ HI}$	SEU-Rate durch HI bedingt
$R_{SEU,\ Pr}$	SEU-Rate durch Protonen bedingt
ρ	Dichte
$\Psi_{a,b}$	Mutter-*Wavelet*
S	Steigung (*Slope*)
$\sigma_{SEU,\ HI}$	Wirkungsquerschnitt (SEU, *Heavy Ions*)
$\sigma_{SEU,\ Pr}$	Wirkungsquerschnitt (SEU, Protonen)
ϑ_A	Umgebungstemperatur

A.6 Literaturverzeichnis

[Ada00] M. D. Adams, H. Man, F. Kossentini, T. Ebrahimi
JPEG 2000: The Next Generation Still Image Compression Standard
Contribution to ISO/IEC JTC 1/SC 29/WG 1 N 1734, Juni 2000

[Ada05] M. D. Adams
The JasPer Project Home Page
http://www.ece.uvic.ca/~mdadams/jasper/, 2005

[Aer03] Aeroflex, Inc.
Single Event Effects Qualification, UT8Q512K8 (RQ02), Lot 6ZVL04 4Mbit SRAM
Aeoroflex, Inc., Plainview, NY, USA, Juli 2003

[Amp03] Amphion Semiconductor Ltd.
CS6510 JPEG2000 Encoder
Amphion Semiconductor Ltd., Belfast, Nordirland, UK, Januar 2003

[Ana05] Analog Devices, Inc.
ADV202, JPEG2000 Video Codec, Datasheet
Analog Devices, Inc., Norwood, MA, USA, 2005

[And03] K. Andra, C. Chakrabarti, T. Acharya,
A High-Performance JPEG2000 Architecture
IEEE Transactions on Circuits and Systems for Video Technology,
Vol. 12, No. 3, März 2003

[Atm06] Atmel Corp.
AT697E, Rad-Hard 32 bit SPARC V8 Processor, Datasheet
Atmel Corp., San Jose, CA, USA, Rev. 4226D-AERO-02/06, Februar 2006

[Bar06] Barco-Silex
BA112JPEG2000E, JPEG2000 Encoder, Factsheet
Barco-Silex, Louvain-La-Neuve, Belgien, Mai 2006

[Bol00] M. Boliek, C. Christopoulos, E. Majani
JPEG2000 Part I Final Committee Draft Version 1.0
ISO / IEC JTC/SC29/WG1 N1646R, März 2000

[Brä89] D. Bräunig
Wirkung hochenergetischer Strahlung auf Halbleiterbauelemente
Springer-Verlag, Berlin / Heidelberg, 1989

[Brü06] M. Brüggemann, H. Schmidt, D. Walter, F. Gliem, R. Harboe-Sørensen, P. Roos, M. Stähle
SEE Tests of NAND Flash Memory Devices for Use in a Safeguard Data Recorder
Radiation Effects on Components and Systems (RADECS) 2006 Workshop, Athen, Griechenland, September 2006

[Car01] C. Carmichael, E. Fuller, J. Fabula, F. Lima
Proton Testing of SEU Mitigation Methods for the Virtex FPGA
Proc. of Military and Aerospace Applications of Programmable Logic Devices (MAPLD), Washington D. C., USA, September 2001

[Car06] C. Carmichael
Triple Module Redundancy Design Techniques for Virtex FPGAs
Xilinx, Inc., XAPP197 Version 1.0.1, Juli 2006

[Car99] C. Carmichael, E. Fuller, P. Blain, M. Caffrey
SEU Mitigation Techniques for Virtex FPGAs in Space Applications
MAPLD '99, Laurel, MD, USA, Post Paper, September 1999

[Cas05] Cast, Inc.
JPEG2K_E, JPEG 2000 Encoder Core
Cast, Inc., Woodcliff Lake, NJ, USA, Dezember 2005

[Cha03] T.-H. Chang, C.-J. Lian, H.-H. Chen, J.-Y. Chang, L.-G. Chen
Effective Hardware-Oriented Technique for the Rate Control of JPEG2000 Encoding
IEEE Int. Symp. Circuits and Systems 2003, Bangkok, Thailand, Mai 2003

[Chi02] J. S. Chiang, Y. S. Lin, C. Y. Hsieh
Efficient Pass-Parallel Architecture for EBCOT in JPEG2000
IEEE Int. Symp. on Circuits and Systems, Mai 2002

[Chr00] C. Christopoulos, A. Skodras, T. Ebrahimi
The JPEG2000 Still Image Coding System: An Overview
IEEE Transactions on Consumer Electronics, Vol. 46, No. 4, pp. 1103-1127, November 2000

[CNE04] CNES, JPL, ESA
Mars Missions, Environments Specifications
CNES-JPL-ESA Document, MARS-TS-00-003-JOINT, Mai 2004

[DLR02] DLR
Bild-Datenbank
http://bilddb.rb.kp.dlr.de, Mars Global Surveyor, 2002

[DOD91] U. S. Department of Defense
Reliability Prediction of Electronic Equipment (MIL-HDBK-217F)
U. S. Department of Defense, Dezember 1991

[eCo06] eCos
The eCos Home Page
http://ecos.sourceware.org/, 2006

[ECS02] ECSS
SpaceWire - Links, nodes, routers and networks
Draft ECSS-E-50-12A, November 2002

[Far05] C. L. Farage, K. A. Pimbblet
Evaluation of Cosmic Ray Rejection Algorithms on Single-shot Exposures
Publications of the Astronomical Society of Australia, Volume 22, Issue 3, 2005

[Fey93] J. Feynman, G. Spitale, J. Wang, S. Gabriel
Interplanetary proton fluence model- JPL 1991
Journal of Geophysical Research, Vol. 98, No. A8, 13281-13294, 1993

[Fic98] T. Fichna, M. Gärtner, F. Gliem, F. Rombeck
Fault-Tolerance of Spaceborne Semiconductor Mass Memories
Proceedings of the Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing, FTCS, 408-413, 1998

[Fry05] T. W. Fry, S. A. Hauck
SPIHT Image Compression on FPGAs
IEEE Transactions on Circuits and Systems for Video Technology, Vol. 15, No. 9, September 2005

[Gai05] Gaisler Research
LEON2 Processor User's Manual, XST Edition
Gaisler Research, Göteborg, Schweden, Version 1.0.30, Juli 2005

[Gan03] M. Gangadhar, D. Bhatia
FPGA based EBCOT Architecture for JPEG 2000
IEEE International Conference on Field-Programmable Technology 2003

[Ger01] B. Gerlach
Miniaturisierung von Instrumentenrechnern für die Weltraumfahrt
Dissertation, Cuvillier Verlag, Göttingen, August 2001

[Gli05] F. Gliem
Raumfahrtelektronik I, Teil A: Strahlungsumgebung, Strahlungsschäden
Vorlesung, Institut für Datentechnik und Kommunikationsnetze,
Technische Universität Braunschweig, Draft Version, April 2005

[Gra03] P. Graham et al.
Consequences and Categories of SRAM FPGA Configuration SEUs
6th Military and Aerospace Programmable Logic Devices (MAPLD)
International Conference, Washington, D. C., USA, 2003

[Hey01] D. Heynderickx et al.
SPENVIS : Space Environment Information System, Chapter 9
Online, http://www.spenvis.oma.be/spenvis/ecss/ecss09/ecss09.html, 2001

[Hol02] A. Holmes-Siedle, L. Adams
Handbook of Radiation Effects (2nd ed)
Oxford University Press, Oxford, UK, Januar 2002

[Kam04] K. Kamm
Implementierung einer JPEG2000-Bilddatenkompression für die Venus Monitoring Kamera
Diplomarbeit, Institut für Datentechnik und Kommunikationsnetze,
Technische Universität Braunschweig, September 2004

[Mar04] W.J. Markiewicz, D. Titov, H.U. Keller, R. Jaumann, H. Michalik,
D. Crisp, L. Esposito, S.S. Limaye, S. Watanabe, N. Ignatiev, N. Thomas
VENUS Monitoring Camera for VENUS EXPRESS
European Geosciences Union, 1st General Assembly, Nizza, April 2004

[Mat00] Matra Marconi Space
Mars Express, Annex A (E-IDS) Electrical Interface Requirement Specification
MEX-MMT-SP-007, V2.1, Juli 2000

[Mic05] H. Michalik
Entwurf fehlertoleranter Rechnersysteme
Vorlesung, Institut für Datentechnik und Kommunikationsnetze, Technische Universität Braunschweig, März 2005

[Mit02] T. Mitrov
Wavelet-Kodierung von Standbildern: Set Partitioning in Hierarchical Trees (SPIHT)
Seminar Fehlerresistente Übertragungssysteme, Institut für Informatik V, Universität Bonn, 2002

[NAS98] NASA
Mars Pathfinder - IMP Imager for Mars Pathfinder
EDR Archive Volume 1 of 3, USA_NASA_PDS_MPIM_0001 Version, Juli 1998

[NAS99] *NASA*
Outer Planets Program, Environmental Requirements
http://centauri.larc.nasa.gov/outerplanets/Envir_rqts.pdf, August 1999

[Neu04] G. Neukum
Mars Express HRSC Bildserie #067 - Dao Vallis (Orbit 0528)
http://www.geoinf.fu-berlin.de, Fachrichtung Planetologie und Fernerkundung, Freie Universität Berlin, Juni 2004

[Oar06] OAR Corporation
The RTEMS Home Page
OAR Corporation, http://www.rtems.org, Huntsville, USA, 2006

[Pen88] W. B. Pennebaker, J. L. Mitchell, G. G. Langdon Jr., R. B. Arps
An overview of the basic principles of the Q-coder adaptive binary arithmetic coder
IBM, J.RES, Develop, Vol. 32, No. 6, November 1988

[Rab95] F. Rabe
Schnelle Bilddatenkompression für Raumfahrtanwendungen
Dissertation, Dr. Kovač Verlag, Hamburg, Juli 1995

[Rit02] J. Ritter
Wavelet based image compression using FPGAs
Dissertation, Martin-Luther-Universität Halle-Wittenberg, Dezember 2002

[Saa04] Saab Ericsson Space, F.Sturesson, S.Mattsson
Application-Like Radiation Test of XTMR and FTMR Mitigation Techniques for Xilinx Virtex-II FPGA
Final Presentation of ESTEC Contract No. 11407/95/NL/MV, CCN-5, COO-7, November 2004

[Sai96] A. Said and W. A. Pearlman
A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees
IEEE Transactions on Circuits and Systems for Video Technology, Vol. 6, No. 3, Juni 1996

[Shi02] P. Shivakumar, M. Kistler, S.W. Keckler, D. Burger, L. Alvisi
Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic
Proceedings of the 2002 International Conference on Dependable Systems and Networks, 2002

[Sho02] M. L. Shooman
Reliability of Computer Systems and Network
John Wiley & Sons, Inc, New York, Oktober 2002

[Sib02] C. Sibilla, Astrium
Venus Express, Generic TM/TC Interface Control Document
Astrium, VEX-T.ASTR-ICD-00326, V1.0, Dezember 2002

[Soe02] J. Sørensen, ESA/ESTEC/TOS-EMA
Venus Express Radiation Environment
ESA ESTEC, EMA-02-029/JS, März 2002

[Soe05] J. Sørensen, ESA/ESTEC/TOS-EMA
Solar Orbiter Environment Specification
ESA ESTEC ,TOS-EMA-03-034/JS, Issue 1.3, März 2005

[Sol05] Solar Orbiter assessment team, ESA-ESTEC
Solar Orbiter. Payload Definition Document
ESA-ESTEC, SCI-A/2004/175/AO, Version 4.1, August 2005

[Ste04] M. Stettler, M. Caffrey, P.Graham, J. Krone
Radiation effects and mitigation strategies for modern FPGAs
Los Alamos National Laboratory, USA, Report No. LA-UR-04-6341, 2004

[Swi04] G. M. Swift, JPL
Xilinx Single Event Effects, 1st Consortium Report: Virtex-II Static SEU Characterization
JPL, Single Event Effects Consortium, Technical report, Januar 2004

[Tan06] A. S. Tanenbaum
Computerarchitektur. Strukturen – Konzepte - Grundlagen
Pearson Studium, 5. Aufl., Dezember 2006

[Tau02] D. S. Taubmann, M. W. Marcellin
JPEG2000 Image Compression Fundamentals, Standards and Practice
Kluwer Academic Publishers, 2002

[Tri95] C. Tribble
The Space Environment, Implications for Spacecraft Design
Princeton University Press, 1995

[Wan03] J.J. Wang, W. Wong, S. Wolday, B. Cronquist, J. McCollum, R. Katz, I. Kleyner
Single Event Upset and Hardening in 0.15 µm Antifuse-Based Field Programmable Gate Array
IEEE Transactions of Nuclear Science, Vol. 50, No. 6, Dezember 2003

[Weg06] M.S. Wegmueller, D. Perels, T. Blaser, S. Senn, P. Stadelmann, N. Felber, W. Fichtner
Silicon Implementation of the SPIHT Algorithm for Compression of ECG Records
Submitted to: 49th IEEE Midwest Symposium on Circuits and Systems (MWSCAS), Puerto Rico, August 2006

[Wit03] T. Wittrock, K.-U. Reiche, K. Stöckner, H. Michalik, F. Gliem
Flexible Operational Sequencing of Complex Spaceborne Instruments – The Software System OCL
54th Int. Astronautical Congress, Bremen, September 2003

[Wit04] H. Witte, R. Jaumann, H. U. Keller
DAWN, DAWN Factsheet
DLR, Februar 2004

[Xil01] Xilinx, Inc.
Virtex 2.5 V Field Programmable Gate Arrays
Xilinx, Inc., Product Specification, DS003-2 (v2.5) April, 2001

[Xil04a] Xilinx, Inc.
QPro Virtex-II 1.5V Radiation Hardened QML Platform FPGAs
Xilinx, Inc., Product Specification, DS124 (v1.1), Januar 2004

[Xil04b] Xilinx, Inc.
Xilinx TMRTool BETA Class, Principles of Xilinx Triple Module Redundancy
Xilinx, Inc., Presentations CD, 2004

[Xil04c] Xilinx, Inc.
Platform Flash In-System Programmable Configuration PROMs
Xilinx, Inc., Preliminary Product Specification, DS123 (v2.5), Oktober 2004

[Xil05a] Xilinx, Inc.
Virtex-II Platform FPGA User Guide
Xilinx, Inc., UG002 (v2.0), März 2005

[Xil05b] Xilinx, Inc.
Virtex-II Platform FPGAs: Complete Data Sheet
Module 3: DC and Switching Characteristics
Xilinx, Inc., DS031 (v3.4), März 2005

[Yui02] C. Yui, G. Swift and C. Carmichael
Single Event Upset Susceptibility Testing of the Xilinx Virtex II FPGA
MAPLD International Conference 2002, Laurel, MD, USA, September 2002

[Zho05] G. Zhou
Hardware Accelerator of JPEG2000 Coder
Diplomarbeit, Institut für Datentechnik und Kommunikationsnetze, Technische Universität Braunschweig, September 2005

[Zho06] G. Zhou, H. Michalik
Discrete Wavelet Transform: Dynamic Range Analysis
Technical Note, Institut für Datentechnik und Kommunikationsnetze, Technische Universität Braunschweig, Mai 2006