



Fakultät für Elektrotechnik und Informationstechnik
der Technischen Universität München
Lehrstuhl für Integrierte Systeme

Evaluierung softwarebasierter Radiosignalverarbeitung auf general-purpose- und GPU-Prozessorarchitekturen

Lothar H. Stolz

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr.-Ing. habil. G. Rigoll

Prüfer der Dissertation:

1. apl. Prof. Dr.-Ing. habil. W. Stechele
2. Univ.-Prof. Dr.-Ing. habil. N. Wehn,
Universität Kaiserslautern

Die Dissertation wurde am 02.06.2014 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 24.03.2015 angenommen.

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation anhand von Marktsituation und Problemstellung	2
1.2	Zielsetzung und Erkenntnisbeitrag der Untersuchung	3
1.3	Untersucher Lösungsansatz und Allgemeingültigkeitswert der Ergebnisse . .	4
1.4	Aufbau der Arbeit	6
2	Themenfeldeinordnung und fachliche Grundlagen	7
2.1	Einordnung der Arbeit zum Stand der Technik	8
2.1.1	Existierende softwarebasierte Empfangslösungen	8
2.1.2	Kontext der GPU-Programmierung	11
2.2	Prozessorsysteme und Programmieretechniken	14
2.2.1	Logikschaltungen und Prozessoren	14
2.2.2	Methoden zur parallelen Datenverarbeitung in Prozessoren	16
2.2.3	Programmierung von Prozessoren	18
2.2.4	Designparadigmen zur effizienten Software-Implementierung	19
2.3	Komponenten eines Multistandard-Radiotransceivers	21
2.3.1	Frontend	21
2.3.2	Signalverarbeitung	22
2.3.3	Nutzdatenquelle/-senke	23
2.4	Umsetzung von Signalverarbeitung im Sende-/Empfangssystem	23
2.4.1	Analoge Signalverarbeitung	23
2.4.2	Digitale Signalverarbeitung	24
2.4.3	Der Begriff Software Defined Radio	24
2.4.4	Nutzung des Applikations- und Grafikprozessors	27
2.5	Konzepte der Radiosignalverarbeitung	29
2.5.1	Modulation und Basisbandsignalдарstellung	29
2.5.2	Repräsentation eines analogen Signals in der digitalen Domäne . . .	31
2.5.3	Digitale Übertragungsverfahren	33
3	Verfügbare Mikroarchitekturen für eingebettete Infotainmentsysteme	39
3.1	Übersicht der verfügbaren Mikroarchitekturen	40
3.2	Analyse der Mikroarchitekturen	41
3.2.1	Intel Bonell (Intel Atom)	41
3.2.2	AMD Bobcat (AMD Fusion)	43
3.2.3	ARM Cortex-A	44
3.2.4	Texas Instruments TMS320C64x	47
3.2.5	NVIDIA G9x (Geforce 9)	49
3.2.6	AMD Radeon R800 (AMD Radeon HD)	51

Inhaltsverzeichnis

3.2.7	Imagination Technologies SGX (PowerVR)	54
3.3	Einordnung der GPU als Prozessorklasse	56
3.3.1	Betrachtung der Prozessorarchitektur	56
3.3.2	Betrachtung des GPU-Programmierframeworks	57
3.3.3	Softwareoptimierungsprinzipien auf der GPU	59
3.3.4	Fazit zur GPU-Architektur	61
3.4	Architektur-Implementierungsvarianten	62
3.4.1	Betrachtung hinsichtlich theoretischen Maximaldurchsatzes	65
4	Entwicklung einer effizienten Signalverarbeitungskette für DAB-Empfang	67
4.1	Das DAB-System	68
4.1.1	Eigenschaften und Aufbau des Übertragungsverfahrens	68
4.1.2	Mathematische Formulierung des DAB-Signals	74
4.2	Entwicklung der Empfangsalgorithmen	75
4.2.1	Synchronisation: Signalakquisition	75
4.2.2	Synchronisation: Tracking	87
4.2.3	Demodulation	90
4.3	Nutzdatenhandling	100
4.3.1	Fast Information Channel	100
4.3.2	Main Service Channel	101
4.4	Softwaretechnische Umsetzung des Empfängersystems	101
4.4.1	Entwicklung des Programmcodes	101
4.4.2	Eliminierung von Datentransfers für Zwischenergebnisse	101
4.4.3	Übergang zur Festkommaarithmetik	103
4.4.4	Schnittstellen des Demodulatorsystems	103
4.5	Bewertung der Signalverarbeitungskette	103
4.5.1	Speicheranforderungen	103
4.5.2	Vermessung der Empfangsgüte	104
5	Evaluierung des Laufzeitverhaltens der x86-/GPU-Architekturen	107
5.1	Methodik der Laufzeitanalyse	108
5.1.1	Instrumentierung/Event-based Profiling	108
5.1.2	Statistisches Profiling	108
5.1.3	Profiling der CPU	109
5.1.4	Profiling der GPU	110
5.1.5	Auswahl relevanter Regionen für die Codeoptimierung	110
5.2	Intel Atom x86-/NVidia Ion GPU-Plattform	111
5.2.1	Hardwarearchitektur	112
5.2.2	Software und Betriebssystem	113
5.2.3	Evaluierung der Referenz-Hochsprachenimplementierung	116
5.2.4	Evaluierung der Variante mit SIMD-optimiertem SSE-Code	118
5.2.5	Evaluierung der Nutzung des GPU-Prozessors	125
5.3	AMD Fusion x86-/GPU-Plattform	135
5.3.1	Systemsoftware-Unterschiede zur Intel Atom-Plattform	136
5.3.2	Evaluierung der Hochsprachen-Referenzimplementierung	136
5.3.3	Evaluierung der Variante mit SIMD-optimiertem SSE-Code	138

5.3.4	Evaluierung der Nutzung des GPU-Prozessors	138
5.4	Zusammenfassung und Diskussion der x86-Plattformen	141
5.4.1	x86-/GPU-Plattform Intel Atom/NVidia Ion	141
5.4.2	x86-/GPU-Plattform AMD Fusion	141
5.4.3	Implementierungsempfehlung und erwarteter Ressourcenverbrauch	144
6	Evaluierung des Laufzeitverhaltens der ARM-/DSP-Architekturen	147
6.1	Evaluierung der ARM-Architektur	147
6.1.1	Software und Betriebssystem des ARM-Systems	147
6.1.2	Evaluierung der C-Referenzvariante	149
6.1.3	Evaluierung der Variante mit NEON-optimiertem Code	150
6.2	Nutzung des DSP-Subsystems	153
6.2.1	Interprozessorkommunikation im heterogenen ARM/DSP-System	153
6.2.2	Anpassung der Signalverarbeitungskette an die DSP-Architektur	153
6.2.3	Laufzeitanalyse des DSPs	156
6.3	Diskussion der Ergebnisse	157
7	Weitere Untersuchungen im Umfeld der entwickelten Signalverarbeitungskette	163
7.1	Betrachtung der elektrischen Leistungsaufnahme	164
7.1.1	Beschreibung des Messaufbaus	164
7.1.2	Untersuchung der low-cost x86/GPU-Plattform	164
7.1.3	Untersuchung der ARM-basierten Plattform	166
7.1.4	Fazit zur elektrischen Leistungsaufnahme	166
7.2	Einordnung der low-cost x86-CPU zu Workstation x86-CPUs	166
7.2.1	Die Intel Sandy Bridge-Plattform	166
7.2.2	Messungen und Laufzeitvergleich	167
7.3	Vergleich des prozessorbasierten Lösungsansatzes zu einem FPGA-basierten System	169
7.3.1	Einordnung eines FPGA-Entwurfs	169
7.3.2	Resümee zu einer möglichen Präferenz zwischen FPGA und Prozessor	171
7.4	Abschätzung eines Softwaredemodulators für DVB-T	171
7.4.1	Eigenschaften von DVB-T	172
7.4.2	Komplexität eines gewählten Übertragungsszenarios	173
7.4.3	Ergebnisse der Abschätzung	173
7.5	Prototypische Integration des Empfängers in ein Versuchsfahrzeug	176
7.5.1	Automotive Ethernet/IP	176
7.5.2	SDR-Demodulator und Audioquelle	177
7.5.3	Audiosenke und Bedieneinheit	178
7.5.4	IP-basiertes Audio Streaming	179
7.5.5	IP-basierte Middleware	181
7.5.6	Resümee zum Demonstrator-Gesamtfahrzeug	182
8	Zusammenfassung und Ausblick	183
8.1	Diskussion der CPU-Mikroarchitekturen mit integriertem SIMD-Coprozessor	183
8.2	Diskussion der GPU-Mikroarchitekturen	184

Inhaltsverzeichnis

8.3	Folgerungen für die Plattformauslegung und Produktentwicklung einer Automotive Entertainment-Plattform	189
8.4	Ausblick	191
	Abkürzungsverzeichnis	193
	Abbildungsverzeichnis	197
	Tabellenverzeichnis	201
	Literaturverzeichnis	203

1 Einleitung

1.1	Motivation anhand von Marktsituation und Problemstellung .	2
1.2	Zielsetzung und Erkenntnisbeitrag der Untersuchung	3
1.3	Untersuchter Lösungsansatz und Allgemeingültigkeitswert der Ergebnisse	4
1.4	Aufbau der Arbeit	6

Software Defined Radio soll die vollflexible Implementierung einer digitalen Signalverarbeitung für Funkgeräte bezeichnen, bei welcher sämtliche Verarbeitungsalgorithmen allein durch Firmware, also einen auswechselbaren Programmcode, bestimmt sind. Heute ist die natürliche Wahl für die Realisierung einer solchen Signalaufbereitung zum einen die Klasse der feldprogrammierbaren Logikbausteine (engl. Field Programmable Gate Arrays, FPGAs), zum anderen die Klasse der digitalen Signalprozessoren (engl. Digital Signal Processors, DSPs). Mit Hilfe dieser rekonfigurierbaren beziehungsweise programmierbaren Architekturen wird nach Stand der Technik typischerweise ein eigenständiges Subsystem allein zur Radiosignalverarbeitung entworfen, oft genannt der Basisbandprozessor. Es agiert zu einem sogenannten Applikationsprozessor als abgeschlossenes und untergeordnetes System. Als Applikationsprozessor wird derjenige Prozessor im Systemverbund bezeichnet, der sämtliche angebotenen Dienste verwaltet sowie die für den Benutzer erlebbare Funktionalität des Gerätes ausführt und beispielsweise über die Mensch-Maschine-Schnittstelle (engl. Human Machine Interface, HMI) darstellt. In der Regel besitzt er Steuerungshoheit über das Gesamtgerät und ist somit der Hauptprozessor (engl. Central Processing Unit, CPU) des Systems.

Heutige Infotainmentsysteme aus dem high-end Automobilbereich stellen faktisch eine leistungsstarke Allzweckrechnerplattform im Fahrzeug zur Verfügung. Neben den ursprünglichen Anwendungsgebieten, der Steuerung und Kontrolle von Audiobaugruppen sowie diverser generischer Fahrzeugfunktionen, wird heute durch das Infotainmentsystem eine Vielzahl komplexer Komfort- und Fahrerassistenzfunktionen bedient. Dazu gehören Algorithmen zur Navigationsplanung und -visualisierung, Video- und Kamerasysteme mit rechenintensiver Bildaufbereitung sowie die erste Generation von mobilfunkbasierten Online-Anwendungen. Es ist abzusehen, dass die kommenden Entwicklungen dieser Geräte sich in Richtung einer Netbook- oder Tablet-PC ähnlichen Architektur entwickeln werden.

1.1 Motivation anhand von Marktsituation und Problemstellung

Da direkt auf dem Applikationsprozessor dieser Plattformen in Zukunft vergleichsweise viel Rechenleistung vorhanden sein wird, soll in dieser Arbeit untersucht werden, ob es möglich ist, die Funktionalität des spezialisierten Basisbandprozessors auch innerhalb des general-purpose Applikationsprozessors abzubilden. Es wird hierzu das Potential der Applikationsprozessor-Mikroarchitekturen für softwarebasierte Radiosignalverarbeitung evaluiert.

Zunehmende Anzahl der Funktechnologien

Mit den Systemen der zweiten Generation des Mobilfunks (GSM, D-AMPS) konnten sich um die Jahrtausendwende digitale Kommunikationsverfahren im Massenmarkt durchsetzen. Seitdem nimmt die Anzahl der im Markt positionierten und folglich von den Geräteherstellern zu unterstützenden Funkübertragungsmethoden stetig zu. Gründe für diese Diversifikation sind in den wachsenden Anforderungen hinsichtlich des zu übermittelnden Datenvolumens bei gleichzeitigem Zwang zur Steigerung der Effizienz zu sehen. So äußern sich sämtliche Fortschritte im Bereich der elektronischen Bauelemente sowie Erkenntnisse im theoretischen Feld der Nachrichtentechnik stets in der umgehenden Umsetzung eines Übertragungsstandards. Auch existierende Standards erfahren während ihrer Lebenszeit Erweiterungen und Variationen zur Verbesserung der Dienstgüte und Unterstützung neuartiger Anwendungsszenarien (vor allem Daten- und Telematikdienste). Dementsprechend ist zu beobachten, dass sich global jüngst eine sehr inhomogene Struktur in der Telekommunikationslandschaft ausbildet, insbesondere an Ländergrenzen und zwischen den Gebieten der einzelnen Kontinente ohnehin. Die kommende technische Herausforderung für Gerätehersteller liegt deshalb darin, dem Kunden einen Zugriff auf ein Dienstportfolio zu ermöglichen, welches potentiell in schneller Abfolge über unterschiedlichste Funkübertragungsverfahren zur Verfügung gestellt wird (Abbildung 1.1).

Auswirkungen auf den Bereich des automotive Infotainment

Mobile Telekommunikation ist seit jeher stark mit dem Fahrzeug verbunden. Insbesondere Radioempfangssysteme der Unterhaltungselektronik fanden ab 1960 als Massenprodukt ihren Weg in das Fahrzeug. Für das Marktsegment blieb das Szenario eines analogen Rundfunkempfangs auf Basis der FM- und AM-Verfahren für mehrere Jahrzehnte stabil. Neben der Evolution des Mobilfunksektors stehen auch im Broadcasting-Bereich nun zum Zwecke der Digitalisierung des Rundfunks eine Vielzahl von Übertragungssystemen aktuell im weltweiten Wettbewerb.

Speziell für die Automobilbranche stellt jedoch die jüngste Diversifizierung der Funktelekommunikation in Verbindung mit starkem Kostendruck eine große Herausforderung dar, welche sich nach klassischem Vorgehen nicht befriedigend bewältigen lässt: Zum einen ist eine hohe Anzahl von standardspezifischen Steuergeräten für sämtliche Ländervarianten zu entwickeln und vorzuhalten. Zum anderen ist, während die kurzlebigen Produkte der Consumer Electronic nur Intervallen von wenigen Monaten schritthalten können, der

1.2 Zielsetzung und Erkenntnisbeitrag der Untersuchung

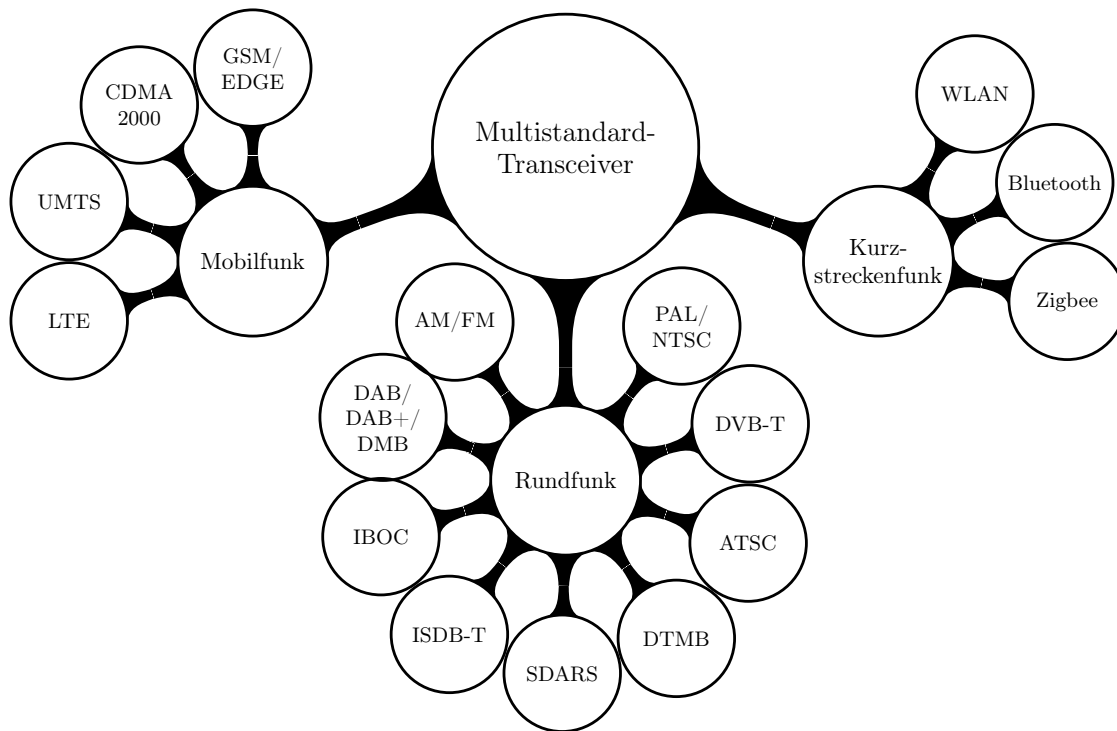


Abbildung 1.1: Visualisierung einer beispielhaften Auswahl der durch ein Multistandardkommunikationsgerät abzudeckenden Funkübertragungsverfahren.

typische Produktlebenszyklus eines Automobils vergleichsweise lang und auch dessen Entwicklungszyklus mit circa sieben Jahren weitaus länger als die Entwicklungszyklen heutiger Übertragungsstandards. Aus dieser Motivation heraus erwächst der Wunsch nach einer universellen Multistandard-Radioplattform, welche allein durch Austausch von Software eine funktionale Rekonfiguration und somit das Einspielen neuer Übertragungsstandards ermöglicht. Hierzu ist es notwendig, dass die digitale Signalverarbeitung des Gerätes vollständig in frei programmierbaren und somit in ihrer Funktion rekonfigurierbaren Schaltkreisen abgebildet wird [1]. Solange die dortigen Ressourcen ausreichend sind, könnten nach der Theorie stets neue Signalverarbeitungsalgorithmen ohne Tausch der zugrunde liegenden Hardware implementiert werden. Jedoch ist ein auf flexiblen Spezialbausteinen wie FPGAs oder dedizierten DSP-Subsystemen basierendes Radiosystem hinsichtlich des Preises im stark kostengetriebenen Markt der Automobilbranche als kritisch anzusehen. Die Implementierung sämtlicher Signalverarbeitungsalgorithmen dagegen auf der ohnehin vorhandenen general-purpose Mikroprozessorarchitektur des Applikationsprozessors im Infotainmentsystem hingegen könnte zur kosteneffizienten Realisierung eines flexiblen Multistandard-Funksystems beitragen.

1.2 Zielsetzung und Erkenntnisbeitrag der Untersuchung

Im Rahmen dieser Arbeit soll, als Alternative zum üblichen Ansatz mit einem dedizierten FPGA- oder DSP-Subsystem, die Realisierbarkeit von in Realzeit lauffähigen Radio-

1 Einleitung

signalverarbeitungsstrukturen direkt auf general-purpose Applikationsprozessorarchitekturen untersucht werden. So könnte der zusätzliche Prozessor des Radiosubsystems eingespart werden. Möglicherweise könnten sich zusätzlich durch Teilung der Prozessorressourcen mit anderen Usecases positive Synergieeffekte ergeben aufgrund der Tatsache einer nicht-simultanen Hardware-Nutzung. Dies könnte sich aufgrund besserer Auslastung auf Gesamtsystemebene aus Kostensicht weiterhin gewinnbringend auswirken.

Untersuchung der x86- und ARM-Architekturen für eingebettete Systeme

Es soll anhand von Fallstudien untersucht werden, in welchem Maße für den gewählten Anwendungsfall Radiosignalverarbeitung die Leistungsfähigkeit der beiden für Infotainmentsysteme des oberen Leistungssegmentes heute gebräuchlichsten low-cost Applikationsprozessorarchitekturen zu beanspruchen ist. Dabei handelt es sich zum einen um die aus dem Bereich der Personal Computer bekannte Klasse der x86-Prozessoren, die durch spezielle low-end Varianten jüngst zunehmend auch an Bedeutung für die Steuerung größerer eingebetteter Systeme gewonnen hat. Des weiteren wird die Klasse der ARM-basierten CPUs untersucht. Diese haben in den vergangenen Jahren leistungsmäßig auf das Niveau von PCs der Einstiegsklasse aufgeschlossen. Die Prozessorklasse ist im Consumerbereich in Smartphone- und Tablet-Geräten dominant.

Einordnung der CPU mit Multimediabefehlssatzerweiterung zum klassischen DSP

Moderne CPU-Kerne bieten in der Regel eine speziell zur beschleunigten Verarbeitung von Multimediadaten konzipierte integrierte Coprozessoreinheit. Diese Architekturermittlung lässt general-purpose Prozessoren zu einem hybriden Bauteil werden, das neben den klassischen Charakteristiken eines Mikroprozessors auch einzelne Fähigkeiten eines DSPs beinhaltet. Hieraus erwächst die in der Arbeit zusätzlich behandelte Fragestellung, inwieweit sich solche CPUs in der Praxis der ursprünglichen DSP-Anwendungsdomäne behaupten und ob sie diese überflüssig machen können.

Bewertung der GPU als Radiosignalprozessor

Besondere Beachtung soll auch der Idee gewidmet werden, die Rechenleistung eines Grafikbeschleunigers zweckentfremdet für die Radiosignalverarbeitung zu nutzen. Die 3D-Berechnungseinheiten vieler moderner Grafikprozessoren (engl. Graphics Processing Units, GPUs) sind nicht mehr funktional festgelegt, sie stellen vielmehr eine frei nutzbare Mikroarchitektur dar. Die Leistungsangaben der Hersteller sowie jüngste Veröffentlichungen in der Literatur lassen es als äußerst interessant erscheinen, die Nutzung der Prozessorklasse GPU für Radiosignalverarbeitung zu untersuchen. So könnte als naheliegende Idee eine low-cost CPU durch die im System ohnehin verbaute (Embedded-)GPU entlastet werden.

1.3 Untersucher Lösungsansatz und Allgemeingültigkeitswert der Ergebnisse

Im vorliegenden Dokument sollen die Untersuchungen exemplarisch anhand eines standardkonformen, vollständig implementierten Beispiels, einem Digitalradio-Empfangssystem für

den Rundfunkstandard DAB, durchgeführt werden.

Der Rundfunkstandard Digital Audio Broadcasting

Der Rundfunkstandard Digital Audio Broadcasting (DAB) nach [2] stellt in Europa das vorherrschende Konzept zur Modernisierung der durch analogen UKW Rundfunk geprägten Radiolandschaft dar. Nach langjähriger politischer Diskussionsphase erfahren die Derivate aus der DAB-Systemfamilie heute in Europa starken Nachhalt im Ausbau flächendeckender nationaler Sendernetze. Somit hat das System zum heutigen Zeitpunkt besonders für die Automobilindustrie als klassischem Nutzer des mobilen Radiodienstes eine hohe Relevanz.

Ein weiterer Grund, die vorliegende Studie auf Basis von Implementierungen des Systems DAB durchzuführen, sind in einer Risikoabwägung bezüglich der Erstellung eines lauffähigen Demonstrators begründet: DAB bietet, beispielsweise gegenüber dem technisch verwandten System Digital Video Broadcasting Terrestrial (DVB-T) für digitale Fernsehübertragung [3] bei moderaten Datenstromraten die Möglichkeit für eine ressourcenschonende Implementierung. Somit ist im Zuge der Projektuntersuchung bei gleichen fachlichen Untersuchungsgegenständen die tatsächliche Umsetzbarkeit eines bereits auf heutiger Hardware realzeitfähigen Demonstratorsystems bei DAB als eindeutig höher einzuschätzen gewesen als bei DVB-T.

Rundfunk versus bidirektionale und leitungsgebundene Kommunikation

Rundfunk als solches beschreibt speziell diejenige Klasse von Kommunikationstechniken, welche Punkt-zu-Multipunkt-Übertragung mit Hilfe von drahtloser Hochfrequenztechnik bereitstellen. Nichtsdestotrotz liegt mit Festlegung auf dieses Subset keine Beschränkung der Allgemeinheit hinsichtlich der erzielten Erkenntnisse im Umfeld der Realisierung einer Signalverarbeitung vor, da die untersuchten Teilalgorithmen und -probleme in identischer oder verwandter Art und Weise auch in anderen Applikationsbereichen der Nachrichtentechnik außerhalb des Rundfunks Anwendung finden:

- Mit intensiven Speicherzugriffsoperationen verbundener Transport der digitalisierten, hochratigen Repräsentation des analogen Basisbandsignals zur eigentlichen Prozessorinheit.
- Empfängersynchronisation mit Algorithmen der statistischen Schätzverfahren.
- Mehrträgerübertragungsverfahren Orthogonal Frequency Division Multiplex (OFDM), maßgeblich basierend auf dem Algorithmus der schnellen Fourier-Transformation (engl. Fast Fourier Transform, FFT).
- Rückführung einer abstrakten Signalkonstellation auf konkrete Bitwerte (Demapper).
- Untersuchung eines bekannten Vorwärtsfehlerkorrekturverfahrens (hier: Faltungscodierung).
- Verarbeitung eines Signalisierungsprotokolls der logischen Übertragungsschicht direkt im Anschluss an die physikalische Bitübertragungsschicht.

1 Einleitung

Vom Prinzip finden alle diese oder verwandte Verfahren neben dem Einsatz im Rundfunkumfeld ebenfalls in Applikationen moderner bidirektionaler Mobilfunk- (beispielsweise WLAN, LTE) als auch leitungsgebundener Übertragungsverfahren (beispielsweise DSL-Techniken) breite Anwendung.

Algorithmisches Referenzmodell und Ausleitung plattformspezifisch optimierter Implementierungen

Bei der praktischen Untersuchung der Schlüsselalgorithmen auf verschiedenen Rechenplattformen ist für einen Vergleich die wichtigste Voraussetzung der Anspruch, auf jeder der getesteten Rechnerarchitekturen durch intensive, architekturenspezifische Codeanpassungen nahe an das technisch erreichbare Optimum heranzukommen. Nur so kann die anschließende vergleichende Beurteilung und Einschätzung der Eignung einer Plattform als fair und somit nur dann als korrekt angesehen werden. Entsprechend werden in der Arbeit von einer Algorithmenbasis einzelne optimierte Referenzimplementierungen auf den untersuchten Zielplattformen abgeleitet. Spezielle Fähigkeiten der einzelnen Mikroarchitekturen werden untersucht und berücksichtigt. Dies schließt im Besonderen die bestmögliche Anpassung an parallele Datenverarbeitungseinheiten aller betrachteten Prozessorsysteme, sowohl auf der GPU als auch auf der CPU, ein.

1.4 Aufbau der Arbeit

In Kapitel 2 wird nach einem Überblick über existente SDR-Lösungen in die bearbeitete Thematik der Prozessorsysteme und Nachrichtentechnik eingeführt. Begriffe sollen eingeordnet und logisch abgegrenzt werden. Kapitel 3 beginnt mit einer Übersicht über die Technik des zur Fallstudie exemplarisch ausgewählten Übertragungsverfahrens DAB. Im Anschluss folgt die Beschreibung der in der Arbeit entwickelten mathematisch hocheffizienten Empfängeralgorithmenkette zur Digitalradiodemodulation. Kapitel 4 untersucht die Prozessorsysteme, welche für die Umsetzung in einer automotivetauglichen Infotainment-Plattform konkret verfügbar sind. Sie werden beschrieben, klassifiziert und dabei hinsichtlich ihrer Mikroarchitekturen genau betrachtet. Neben den Architekturen von gewöhnlichen Prozessorsystemen werden auch die Mikroarchitekturen von Grafikprozessoren beleuchtet. Für die erste Klasse der x86-basierten Systeme werden in Kapitel 5 spezifisch optimierte Implementierungen der Signalverarbeitung aus Kapitel 3 vorgestellt und anschließend vermessen. Kapitel 6 widmet sich der zweiten möglichen Systemklasse, welche auf ARM-Applikationsprozessoren basiert. Erneut werden unter Berücksichtigung architekturenspezifischer Merkmale Implementierungsvarianten der Radiosignalverarbeitungskette erarbeitet und das Empfangssystem evaluiert. Kapitel 7 schildert weitere im Rahmen der Arbeit durchgeführte Untersuchungen sowie den prototypischen Fahrzeug-Gesamtaufbau, mit welchem das Konzept der softwarebasierten Digitalradio-Demodulation in einem Infotainmentsystem funktional demonstriert werden konnte.

2 Themenfeldeinordnung und fachliche Grundlagen

2.1	Einordnung der Arbeit zum Stand der Technik	8
2.1.1	Existierende softwarebasierte Empfangslösungen	8
2.1.2	Kontext der GPU-Programmierung	11
2.2	Prozessorsysteme und Programmieretechniken	14
2.2.1	Logikschaltungen und Prozessoren	14
2.2.2	Methoden zur parallelen Datenverarbeitung in Prozessoren . . .	16
2.2.3	Programmierung von Prozessoren	18
2.2.4	Designparadigmen zur effizienten Software-Implementierung . . .	19
2.3	Komponenten eines Multistandard-Radiotransceivers	21
2.3.1	Frontend	21
2.3.2	Signalverarbeitung	22
2.3.3	Nutzdatenquelle/-senke	23
2.4	Umsetzung von Signalverarbeitung im Sende-/Empfangssystem	23
2.4.1	Analoge Signalverarbeitung	23
2.4.2	Digitale Signalverarbeitung	24
2.4.3	Der Begriff Software Defined Radio	24
2.4.4	Nutzung des Applikations- und Grafikprozessors	27
2.5	Konzepte der Radiosignalverarbeitung	29
2.5.1	Modulation und Basisbandsignaldarstellung	29
2.5.2	Repräsentation eines analogen Signals in der digitalen Domäne .	31
2.5.3	Digitale Übertragungsverfahren	33

Im ersten Abschnitt des Kapitels wird der vorliegende Untersuchungsgegenstand zum Stand der Technik existierender Arbeiten eingeordnet. Im Anschluss wird eine kurze Einführung zu etablierten Konzepten des Aufbaus und der Nutzung von prozessorbasierten Systemen gegeben. Danach werden für die Zielanwendung eines Multistandard-Radiosystems die disjunkten Einzelthemenbereiche identifiziert und das Themenfeld der Basisbandsignalverarbeitung abgegrenzt. Zum Schluss des Kapitels wird in das Gebiet der allgemeinen Signalverarbeitung und der Radiosignalverarbeitung eingeleitet.

2.1 Einordnung der Arbeit zum Stand der Technik

Zuerst soll das Umfeld der in der Arbeit durchgeführten Untersuchungen exploriert werden. Neben der Begutachtung existierender Softwareradiorealisationen wird im Anschluss weiterhin auf Quellen zum Thema der Nutzung des GPU-Prozessors eingegangen.

2.1.1 Existierende softwarebasierte Empfangslösungen

Im Folgenden werden die bislang bekannten vollständig softwarebasierten Empfangslösungen für Digitalradio DAB dargestellt. Auch Lösungen für andere Standards finden Erwähnung.

Kommerzielle Implementierungen für Digitalradio DAB

Vom Chiphersteller Texas Instruments Incorp. existieren als Familie seit 2001 die Demodulator-Bausteine TMS320DRE200 [4] bis TMS320DRE310 [5]. Es handelt sich dabei prinzipiell um Signalprozessoren aus der bekannten Allzweck-DSP-Reihe TMS320C5000, jedoch mit vom Hersteller fest vorgegebenem Programmcode. Dieser, entwickelt von der Firma Radioscape Ltd., bedient über Softwaredemodulation die fixe Zielapplikation der DAB-Basisbandsignalverarbeitung. Für die jüngere TMS320C64x-Reihe wird von Seiten des Fraunhofer-Instituts für Integrierte Schaltungen IIS in Kooperation mit Texas Instruments Incorp. ein Softwarekit für DAB-Demodulation angeboten [6], welches der Endproduktentwickler auf gewöhnlichen C64x-DSPs ausführen kann. Detailinformationen zu beiden Systemen sind in öffentlicher Art und Weise für Dritte nicht verfügbar. Eine entsprechende, freiprogrammierbare TMS320C64x-Architektur soll in dieser Studie in Kapitel 6 exploriert werden.

Seit 2002 und nun als Marktführer (Marktanteil größer 70 %, Stand 2008, nach [7]) bietet Frontier Silicon Ltd. verschiedene System-on-Chips (SoC) mit intern softwarebasierter Verarbeitung als Lösung für DAB-Empfangsgeräte an. Dies sind beispielsweise FS1230 Chorus 3 und FS1235 Kino 3 [8]. Die geschlossenen Systeme basieren auf einem als Intellectual Property von Imagination Technologies lizenzierten DSP-Prozessorkern mit der Bezeichnung META122 sowie in ROM fest integrierter Firmware zur softwaredefinierten DAB-Basisbandsignalverarbeitung. Weiterhin sind im SoC die Analog/Digital- beziehungsweise Digital/Analog-Wandler für Zwischenfrequenz- und Audiosignale sowie weitere Peripherie-Interfaces integriert. Im Kontext der Arbeit erscheint besonders der Aspekt anmerkwert, dass ein verwandter, um grafikspezifische Einheiten erweiterter Prozessorkerntyp von Imagination Technologies in einem später betrachteten Grafikbeschleuniger eingesetzt zu werden scheint (PowerVR SGX, siehe Kapitel 5).

Die seit 2011 verfügbaren ICs SAF356x des Herstellers NXP zielen insbesondere auf das Automotive-Segment. Sie stellen ein Vektor- beziehungsweise SIMD-DSP-System dar, welches einen bereits digitalisierten Basisbanddatenstrom von einem Frontend annimmt und den decodierten PCM-Audiostrom digital für die Weiterverarbeitung in einem anderen Audio-IC übergibt [9]. Es sind keine Details über die interne DSP-Architektur veröffentlicht. Das multistandardfähige Grundkonzept wird durch unterschiedliche Firmware als Digitalradiodecoder für die Übertragungsverfahren DAB/T-DMB, DRM oder HD-Radio individualisiert vertrieben.

2.1 Einordnung der Arbeit zum Stand der Technik

Eine weitere kommerzielle softwarebasierte Demodulationskette für DAB ist von dem Chiphersteller Realtek seit dem Jahr 2011 bekannt. Sie ist vorgesehen zur Verwendung mit einem ursprünglich für Fernsehempfang im Consumer-PC-Markt entworfenen Chip namens RTL2832U [10]. Dieses IC integriert in seiner Hardware Analog/Digital-Wandler, TV-Demodulator nach DVB-T-Standard und USB-Schnittstelleninterface. Es ist jedoch unter gewissen Randbedingungen möglich, das digitalisierte Basisbandsignal unprozessiert am DVB-T-Demodulator vorbeizuleiten an den USB-Host, typisch einem PC. Dieser Sachverhalt wird für die Realisierung eines DAB-Produktes ausgenutzt. Das auf dem USB-Host ausgeführte Softwaredemodulatorsystem ist nur für die x86 Architektur unter dem Betriebssystem Microsoft Windows verfügbar, als Ressourcenanforderung wird in nicht näher spezifizierter Weise eine x86-CPU mit mindestens 1 GHz Taktfrequenz genannt. Weitere technische Daten sind nicht öffentlich verfügbar. Das low-cost PC-Radiosystem wird kommerziell unter dem Produktnamen Noxon DAB Stick durch die TerraTec Electronic GmbH vertrieben.

Quelloffene Implementierungen für Digitalradio DAB

Im Herbst 2011 veröffentlichte die Züricher Hochschule für Angewandte Wissenschaften eine Masterarbeit über einen softwarebasierten DAB-Demodulator [11]. Der frei zugängliche Code setzt auf dem ebenfalls quelloffenen USRP-/GNU-Radio-Laufzeitframework auf. GNU-Radio stellt Mittel zu einer komponentenweisen Modellierung von Funksystemen bereit, das resultierende System beinhaltet auf Kosten der einfachen Modellierbarkeit jedoch viel konzeptbedingten Overhead. Entsprechend kann das System Echtzeitdemodulation nur auf leistungsstarken Maschinen erzielen. Prozessorlastanalysen sind in der Arbeit der Züricher Hochschule selbst nicht enthalten. In [12] wird berichtet hinsichtlich Laufzeit bei Portierung der Implementierung auf einen ARM Cortex A8 Prozessor (vgl. Kapitel 6), dass 13 s Rechenzeit für 1 s Realzeit-Datenstrom aufzuwenden sind¹.

Eine weitere quelloffene SDR-Software ist SDR-J [13], welche seit Ende 2012 neben FM-Demodulation auch DAB unterstützt. Das Projekt verwendet die frei verfügbaren, allgemeingültigen Algorithmenblöcke der Fourier-Transformations-Bibliothek FFTW [14] und des automatischen Viterbi-Decoder-Codegenerators Spiral [15]. Zum Ressourcenverbrauch gibt die Seite des Projektes an, dass zur mängelfreien Demodulation eine Dualcore-CPU Intel Core i5 mit 2.5 GHz Taktfrequenz nötig ist und die Ausführung "mehr oder weniger" auf einer 2.0 GHz Dualcore-CPU möglich sei.

Weitere Implementierungen für Digitalradio DAB

Aus dem wissenschaftlichen Bereich existiert mit Tseng et. al [16] ein Softwaredemodulator für DAB, welcher auf einer 1.5 GHz Intel Centrino CPU eine Gesamtprozessorlast von 29 % verursacht. Dabei werden auch die SIMD-Befehlssatzerweiterungen der x86-CPU (Streaming SIMD Extensions, SSE) genutzt. Da der Fokus der Arbeit auf der nachrichtentechnischen Untersuchung iterativer Decodieralgorithmen liegt, sind keine detaillierteren Laufzeitmessungen oder Code veröffentlicht.

¹Dies gilt bereits bei Anweisung an den Compiler zur Generierung von Code mit den später betrachteten NEON-SIMD-Befehlssatzerweiterungen.

Xiong et. al. [17] führt ein Softwareradio für echtzeitfähige DAB-Demodulation mit einem ARM Prozessor vor. Jedoch stellt sich heraus, dass die Demodulation nicht auf dem Prozessor, sondern in den Logikgruppen eines FPGAs erfolgt. Der ARM-Kern ist an diese Signalverarbeitungseinheiten über AMBA-Bus angebunden. Er kontrolliert und steuert nur diese, er führt die Signalverarbeitung nicht selbst aus.

Lösungen für andere digitale Funkstandards

Über eine vollständig softwarebasierte Demodulationskette für das Satellitennavigationssystem GPS und Galileo berichtet K. Borre et al. [18]. Es wird ein entsprechender Programmcode für die MATLAB-Simulationsumgebung vorgestellt.

Der Übertragungsstandard Digital Radio Mondiale (DRM) ist im Vergleich zu DAB ein schmalbandigeres Digitalradiosystem (bis zu 20 kHz anstelle von 1.5 MHz Kanalbandbreite), entwickelt zum Einsatz im Frequenzbereich der Kurz- und Mittelwellen. Aufgrund der beschränkten Bandbreite sowie Datenrate stellt das DRM-System für Empfang in Realzeit vergleichsweise geringe Anforderungen an die Leistungsfähigkeit des Demodulationsprozessors. Für die Demodulation von DRM-Signalen ist die kommerzielle Softwarelösung Fraunhofer Software Radio [19] verfügbar. Das Projekt Dream [20] stellt eine quelloffene DRM-Implementierung dar.

Einen kompletten standardkonformen Softwareempfänger für Demodulation des Digitalfernsehstandards DVB-T auf x86-Prozessoren zeigt V. Pellegrini et al. [21]. Als wesentliches Element der durchgeführten Optimierungstechniken der Codeumsetzung wird eine Segmentierung der Algorithmen sowie die Nutzung von vorberechneten Tabellen genannt. Für den Block des OFDM-Demodulators ist auf die Fourier-Transformation des bereits erwähnten FFTW-Projektes zurückgegriffen. Für ein DVB-T-Übertragungsszenario mit moderater Datenrate 11.612 Mb/s wird eine Auslastung unter 50 % der Ressourcen eines Intel Q9400 Systems (4 CPU-Kerne, 2.66 GHz Taktfrequenz) angegeben. Auch wird die Demodulation einer DVB-T-Übertragung mit einem höhervolumigen Datenstrom von 24.13 Mb/s demonstriert. Eine kommerzielle Lösung zur x86-softwarebasierten DVB-T-Demodulation wurde von Mirics Limited unter der Bezeichnung FlexiStream vorgestellt [22]. Als Mindestanforderung an das zu verwendende Hardwaresystem ist ein Zweikernprozessor Intel Core2 mit 2.0 GHz Taktfrequenz angegeben.

Ein vollständiges Softwareradiosystem mit Nutzung eines GPU-Prozessors wurde von S. Choi et al. in [23] und [24] für den WiBro beziehungsweise WiMAX-Standard vorgestellt. Laufzeiten der einzelnen GPU-Algorithmenblöcke sind aufgezeigt, jedoch gibt die Untersuchung noch keine Analyse über die verbleibende Prozessorlast auf der Host CPU oder eine vergleichende Evaluation einer nur-CPU Signalverarbeitungskette. In [23] wird durch die GPU eine Beschleunigung um annähernd den Faktor 90 berichtet im Vergleich zur Ausführung auf einem state-of-the-art DSP-basierten System, welches den DSP Texas Instruments TMS320C64 nutzt².

²Es soll zu dieser Quelle zusätzlich angemerkt werden, dass das dort betrachtete GPU-System NVidia 9800 GTX eine mittlere thermische Verlustleistung (Thermal Design Power, TDP) von 140 W spezifiziert [25], wohingegen der verglichene DSP Texas Instruments TMS320C6416 für eine typische Anwendung (85 % Prozessorlast, 600 MHz) mit 1.5W spezifiziert ist [26].

Abgrenzung zu den verfügbaren kommerziellen Implementierungen

Anhand verfügbarer proprietärer Lösungen, welche allerdings ohne Quellcode vorliegen, sind Untersuchungen und hiermit verbundene Modifikationen nicht möglich. Nur dann können explizit die Performanceauswirkungen einer Erweiterung von Prozessoren um SIMD-Techniken oder der Nutzung der GPU-Architektur exploriert werden. Dies wird erst durch vollständigen Zugriff auf die Quellcodebasis möglich. Die Erarbeitung einer eigenen Signalverarbeitungskette im Rahmen der vorliegenden Arbeit dient deshalb dazu, Detailanalysen vornehmen zu können sowie dabei vergleichende Portierung und Untersuchung auf mehreren verschiedenen Zielprozessorarchitekturen durchzuführen.

Abgrenzung zu den verfügbaren quelloffenen Implementierungen

Bezüglich des Resultats grenzt sich die hier gezeigte Umsetzung zu den genannten quelloffenen Projekten dadurch ab, dass durch hohen Optimierungsgrad die Ausführung auch auf low-cost CPUs in Realzeit möglich ist. Bereits low-end x86-Prozessoren (Intel Atom N270 1.6 GHz, Singlecore, mit einfachem Mikroarchitekturaufbau) reichen zum Betrieb aus, dabei wird nur geringe Prozessorlast (12.1%, siehe Kapitel 5) gefordert. Auf einer zu dem im vorigen Absatz erwähnten System vergleichbaren x86-Dualcore-CPU aus dem Performance-Segment wird nur marginale Last hervorgerufen (nach Kapitel 7.2 weniger als 2% für ein Dualcore-System). Ähnliches gilt für den ARM-Prozessortyp Cortex-A8: Hier werden nur 0.3s Rechenzeit für 1s Quelldatenstrom benötigt (siehe Kapitel 6).

Das vorgestellte System mit intensiver Algorithmenoptimierung sowie anwendungsfall-spezifischer und händischer Codeoptimierung hat den Anspruch, durch Ressourcenverbrauchsoptimierung möglichst nahe an das auf der jeweiligen Mikroarchitektur und bezüglich des Anwendungsfalls erreichbare Optimum heranzureichen. Der Zweck hiervon ist, valide Eignungs- und Ressourcenverbrauchseinschätzungen hinsichtlich der für den Anwendungsfall SDR verfügbaren Prozessorklassen geben zu können. Sämtliche Overhead verursachenden abstrakten Modellierungen und allgemeingültige, nicht auf anwendungsspezifisches Optimierungspotential hin erstellten Bibliotheks-Routinen werden deshalb im Softwareentwurf explizit vermieden, es werden stattdessen maßgeschneiderte Funktionsblöcke programmiert.

2.1.2 Kontext der GPU-Programmierung

Im Allgemeinen beschäftigt sich die Literatur im Themenfeld der GPU vorrangig mit der Portierung von Code zur Lösung berechnungsintensiver Probleme auf die GPU und der Beschreibung einer dabei erzielten Laufzeitreduzierung, das heißt der Zeitspanne, bis zu der Berechnungsergebnisse des Datensatzes präsentiert werden. Diese Zeitdauer wird in Kontext zur Ausgangsimplementierung, meist einer CPU-Umsetzung, gestellt.

Neue Algorithmen beziehungsweise Modifikation von Algorithmen

Eine hohe Zahl an Dokumenten, vor allem im Themenfeld der Bildverarbeitung, hat die Abwandlung bekannter Algorithmen mit Hinblick auf eine bestmögliche Anpassung an die GPU-Architektur zum Thema. Bei einer Modifikation eines Algorithmus ist ein verändertes Rechenergebnis zu erwarten. Dies ist dennoch oft akzeptabel, wenn trotzdem der Zweck

2 Themenfeldeinordnung und fachliche Grundlagen

der anvisierten Anwendung erfüllt und dabei eine effiziente Berechnung ermöglicht wird. Eine reine Umsetzungsvariante berührt im Gegensatz zu einer Modifikation den vorliegenden Algorithmus an sich nicht, in jedem Falle liegen stets die ursprünglichen Paare aus Eingangsdatensatz und Ergebnis vor. Umsetzungsvarianten können dennoch plattformspezifisch angepasst, das heißt optimiert, sein.

In demjenigen Umfeld der Kommunikationstechnik, welche sich nicht mit Entwicklung neuer Übertragungsverfahren sondern der praxistauglichen Umsetzung existierender Verfahren beschäftigt, müssen Szenarien für etablierte und fest standardisierte Kommunikationsverfahren exakt umgesetzt werden. Dort sind Stimuli und zugehöriges Berechnungsergebnis bitgenau festgelegt. Eine Systemverbesserung, welche allerdings diese Bitgenauigkeit verletzt, führt zu Verlust der Standardkonformität und erfüllt somit nicht den eingangs gestellten Verwendungszweck. Die Freiheit, Algorithmen anzupassen oder gar gegen für die Prozessorarchitektur besser geeignete Arten auszutauschen, ist entsprechend beschränkt. So wird auch in der vorliegenden Arbeit ein praxisrelevanter Beitrag über die Umsetzbarkeit in einem Infotainmentsystem-Produkt nur durch Implementierungsvarianten zu leisten sein, die bitgenaue Standardkonformität im Übertragungsweg weiterhin einhalten.

Allgemeine Algorithmen mit Relevanz für die Radiosignalverarbeitung

Eine Vielzahl an Literatur deckt die einzelnen für Radiosignalverarbeitungsapplikationen relevanten Algorithmen ab: Eine ausführliche Studie über die Abbildung der schnellen Fourier-Transformation FFT auf die GPU gibt Volkov et al. [27]. Dort wurde für ein Computersystem gezeigt, dass eine GPU die FFT für die ausgewählte Transformationslänge $N = 512$ mehr als zehnmals schneller ausführen kann als die verwendete CPU.

Der als Fehlerschutzdecoder für Faltungscodes einsetzbare Viterbi-Algorithmus wurde im Detail auf einem GPU Subsystem durch Zhang et al. [28] und Lin et al. [29] behandelt. Zhang beschreibt die parallele Berechnung der Viterbi-Pfadmetriken auf den einzelnen SIMD-Rechenwerken der GPU. Lin beschreibt, wie mittels Tiling-Verfahrens, dem Aufsplitten einer Codewortsequenz, die Parallelität für die GPU erhöht werden kann. Aufgrund des Zusatzaufwandes beim Zusammenführen der Teilergebnisse scheint die Anwendung dieses Tiling-Verfahrens jedoch nur für sehr lange Sequenzen effizient. Die vorgestellten GPU-Implementierungen wurden mit rein sequentiell implementiertem, nicht-SIMD nutzendem CPU-Code verglichen. Dafür werden Geschwindigkeitsgewinne vom Faktor drei beziehungsweise 14 zugunsten der GPU berichtet.

Für Fehlerschutz mittels Low Density Parity Check (LDPC)-Codes existieren Untersuchungen hinsichtlich der Umsetzung eines Decoders auf der GPU von Chang et al. [30] und Ji et al. [31]. Chang stellt einen auf Belief-Propagation/Sum-Product-Algorithmus basierenden Decoder auf der GPU vor und schließt mit der Beobachtung, dass die vorgestellte GPU-Implementierung gegenüber der CPU einen 271fachen Geschwindigkeitsgewinn liefert. Verwendet wurde eine GPU mit NVidia GT200-Chip. Jedoch findet sich im Text über die als Vergleich herangezogene CPU Implementierung, dass sie nur einen der vier vorhandenen Prozessorkerne in der Vergleichs-CPU Intel Xeon E5504 nutzt. Während der GPU-Code mit Fließkommaarithmetik einfacher Genauigkeit (32 bit) rechnet, wird außerdem die verglichene CPU mit doppelter Genauigkeit betrieben (64 bit). Der Code der CPU wird ferner als rein sequentielle Implementierung in der Programmiersprache C beschrieben. In der weiteren LDPC-Decoderimplementierung auf der GPU von Ji werden sowohl der

2.1 Einordnung der Arbeit zum Stand der Technik

Sum-Product- als auch der Min-Sum-Algorithmus evaluiert. Als Hardwareplattform kommt ein System aus einer Intel Core 2 Quad CPU mit 2.42 GHz (mutmaßlich Intel Q6600) und NVidia GT200b-basiertem Grafikkadaper (NVidia GTX285) zum Einsatz. Je nach Szenario werden durch die GPU Beschleunigungsfaktoren von 13.4 bis 40.5 angegeben. Der Autor gibt jedoch an, dass die CPU-Implementierung die datenparallelen SIMD-Instruktionen der CPU nicht nutzt und nur auf einem der vier CPU-Kerne ausgeführt ist.

Vollständige Radiosysteme unter Nutzung der GPU

Als vollständig lauffähige Radiosysteme mit Nutzung der GPU sind die bereits in Abschnitt 2.1.1 genannten Veröffentlichungen [23] und [24] von Choi et al. bekannt. Hinsichtlich der GPU-Programmierung erfolgt eine Analyse der Kernlaufzeiten auf der GPU. Eine Gesamtsystemanalyse, welche das Hostsystem im Detail einschließt, wird nicht gegeben.

Bewertung des Vergleichs zwischen GPU zur CPU

Es fällt auf, dass viele Literaturquellen zu GPU-beschleunigten Algorithmen eine grundlegende Referenzimplementierung auf der x86-CPU mit einer massiv optimierten Codevariante auf dem GPU-Prozessor vergleichen. Architekturspezifische Optimierungen hinsichtlich der CPU, insbesondere die Nutzung von deren SIMD-Fähigkeiten zur datenparallelen Verarbeitung von Multimediainhalten, scheinen meist nicht berücksichtigt, es wird der CPU-Vergleich oft mit simplem, sequentiell arbeitendem Code durchgeführt. Auf diesen Umstand wurde auch bereits 2010 von Vuduc et al. [32] hingewiesen. Vuduc untersuchte selbst drei einschlägig bekannte Elementaralgorithmen der linearen Algebra, welche häufig im wissenschaftlichen Bereich angewendet werden (Sparse Matrix-Vector Multiply, Cholesky-Faktorisierung, Fast Multipole Method). Es wurden publizierte GPU-Implementierungen Dritter referenziert sowie von Vuduc äquivalente CPU-Varianten neu entwickelt, optimiert und gegenübergestellt. Dabei wurde gezeigt, dass auf der CPU eine Performance erreicht werden konnte, welche entweder diejenige der in anderen Quellen angegebenen GPU-Implementierungen erreicht oder diese gar übertrifft. Die entsprechende Schlussfolgerung der Untersuchung weist darauf hin, dass für einen fairen Vergleich zwischen CPU und GPU es unbedingt einer realistischen Anwendungsfällen entsprechenden Workload, das heißt Datensatzmenge, erfordert. Auch müsse zum anderen für beide gegeneinander evaluierten Codevarianten der CPU und der GPU hinsichtlich Optimierungen der gleiche Entwicklungsaufwand bei der Programmerstellung investiert werden.

Fazit zum Kontext der GPU-Untersuchungen

Die im Rahmen der Arbeit gewonnenen Schlüsselerkenntnisse gliedern sich ein zu der Position von Vuduc [32]. Ein Architekturvergleich kann nur in fairer und unverzerrter Weise erfolgen, wenn ein bestmöglicher Optimierungsgrad der Algorithmen auf allen miteinander verglichenen Architekturen erreicht ist. Deshalb müssen die erarbeiteten Codeportierungen für Grafikprozessoren einer intensiven Codeoptimierung unterzogen werden, bei der die seitens der GPU-Hersteller als auch der im betreffenden wissenschaftlichen Umfeld bekannten Methodiken effizienter Implementierungen bestmöglich berücksichtigt sind. Aber auch die als Vergleich ausgeführten Portierungen auf Host-CPUs/Applikationsprozessoren müssen

2 Themenfeldeinordnung und fachliche Grundlagen

intensiv nach Herstellerratschlägen und den allgemein bekannten Konzepten der Softwaretechnologie optimiert werden. Speziell hier erscheinen bei Betrachtung des Literaturumfeldes der GPU/CPU-Vergleiche Lücken zu existieren, zu welchen die vorliegende Arbeit durch explizite Mitevaluierung der CPU-SIMD-Fähigkeiten für den Bereich der Radiosignalverarbeitungsalgorithmen beziehungsweise vollständiger Radiosysteme einen Beitrag liefern soll.

2.2 Prozessorsysteme und Programmiertechniken

Das folgende Unterkapitel soll eine Abgrenzung des Prozessors zu der ebenfalls für die Thematik der rekonfigurierbaren Signalverarbeitung als Zielplattform gebräuchlichen reprogrammierbaren Logik geben, die Parallelitätskonzepte moderner Prozessoren einschließlich der im späteren wichtigen Vektorprocessorfähigkeit vorstellen und auf generell gültige Programmierparadigmen eingehen, welche zur effizienten Umsetzung der untersuchten Signalverarbeitung grundlegend und unbedingt zu beachten sind.

2.2.1 Logikschaltungen und Prozessoren

Die natürliche Implementierungsvariante einer Logikfunktionalität setzt diese über einen Halbleiterherstellungsprozess direkt und unveränderlich um in eine applikationsspezifische integrierte Schaltung (engl. Application-Specific Integrated Circuit, ASIC). Damit jedoch auch nach dem Halbleiterherstellungsprozess funktional rekonfigurierbare Logiksysteme ermöglicht werden, müssen Gatterfunktionen als auch -verbindungen ad-hoc rekonfigurierbar angelegt sein. Hierfür wurden als Zielarchitektur feldprogrammierbare Logikbausteine (Field Programmable Gate Array, FPGA) geschaffen, welche auf einem Chip eine Vielzahl universeller Logikgrundelemente mit der Möglichkeit zur beliebigen internen Verschaltung durch Multiplexerkaskaden bereitstellen. Die konkrete Schaltungsfunktionalität wird dann durch Abbildung auf das System dieser Logikzellen implementiert.

In der Arbeit sollen die unterschiedlichen Mikroarchitekturen einzelner Prozessorsysteme hinsichtlich des Anwendungsfeldes der Programmierung für Radiosignalverarbeitung untersucht werden. Die Möglichkeit der Implementierung solcher Systeme mittels Logikschaltungsentwurf, beispielsweise mittels FPGAs, ist nicht Teil der Arbeit.

Übergang zum Prozessor

Logikschaltungen erlauben selbstverständlich auch die Erstellung von Zustandsautomaten einfachen bis komplexen Umfanges. Prozessoren, als komplexe Zustandsautomaten gekoppelt mit einem Datenpfad und Rechenwerk, stellen somit eine Untergruppe der Logikschaltungen dar. Der Prozessor besitzt eine oder eine gegebene Anzahl an Rechenwerkeinheiten, in der Regel bezeichnet als Arithmetisch-Logische Einheit (Arithmetic-Logic Unit, ALU) mit mehreren funktionalen Konfigurationen. Die Datensätze werden in zyklischem Datenfluss durch diese stets gleichen, in jedem Durchgang aber unterschiedlich parametrisierten Allzweckrechenheiten geleitet, bis final der gewünschte Algorithmus vollständig abgearbeitet ist und das Ergebnis zur Verfügung steht. Als Zwischenablage für die Datensätze während der einzelnen Zyklen dienen prozessorintern Registerbänke oder externe Speicher. Kontrollsignale, welche im Steuer- oder Leitwerk des Prozessors kontinuierlich in jedem

Taktzyklus aus den Instruktionen eines Programmablaufplans gewonnen werden, steuern die Konfiguration der Recheneinheiten, bestimmen damit deren Funktion und adressieren den momentanen Datensatz. In der Registermaschine wird also die verfügbare Recheneinheit von verschiedenen Datensätzen im Zeitmultiplexverfahren mehrfach genutzt. Durch Ablage der Instruktionenfolge in einem reprogrammieren Speicher wird der Prozessor trotz fester Schaltungsverdrahtung zu einer funktional rekonfigurierbaren Plattformarchitektur.

Imperative versus datenstromorientierte Programmierung

Die datenstromorientierte Programmierung stellt auf allgemeinen Logikschaltungen die naheliegende Entwicklungsmethodik für Signalverarbeitungsalgorithmen dar. Hierbei werden auf dem Weg der Daten durch die Schaltung ähnlich einem Fließbandprozess spezialisierte Recheneinheiten an den jeweiligen Stellen der, gegebenenfalls verzweigten, Signalpfade implementiert. Dadurch, dass für jeden Verarbeitungsschritt eine dedizierte Recheneinheit vorgesehen ist, kann sowohl aufgrund der Parallelität beziehungsweise Nebenläufigkeit der Verarbeitung als auch der funktionalen Spezialisierung der einzelnen Rechenwerke ein Schaltungssystem mit hohem Datendurchsatz realisiert werden. Im Gegensatz zur datenstromorientierten Programmentwicklung legt sich der speichergekoppelte Prozessor als Zustandsautomat auf das Paradigma der imperativen Programmierung fest, bei der eine sequentielle Folge von Anweisungen schrittweise abgearbeitet wird. Beide Systemkonzepte führen für die Umsetzung einer identischen Aufgabenstellung zu mitunter stark unterschiedlichen Implementierungsvarianten.

Applikationsspezifische Prozessoren

Prozessorsystemimplementierungen für allgemeine Anwendungen (General-Purpose Processor, GPP) versuchen eine Rechenplattform mit universell-effizientem Instruktionssatz zur Verfügung zu stellen, wobei keine Auslegung auf bestimmte spätere Zielanwendungen den Entwurf maßgeblich beeinflusst. Hingegen ist dies der Fall für applikationsspezifische Prozessoren (Application Specific Instruction-Set Processor, ASIP). Der digitale Signalprozessor (DSP) stellt einen Mittelweg dar, er ist durch spezielle arithmetische Fähigkeiten teiloptimiert für das Anwendungssegment der digitalen Signalverarbeitung. Speziell soll die Arbeit sich der Untersuchung von general-purpose Prozessorarchitekturen widmen. Die Arbeit wird aber zur Einordnung auch Ergebnisse für eine bekannte DSP-Architektur vorstellen.

Die Grafikkarte als Prozessor: Unified Shading und GPGPU

Als Spezialfall soll weiterhin die zweckentfremdete Nutzung moderner Grafikprozessoren (Graphics Processing Units, GPUs) für Radiosignalverarbeitung untersucht werden. Deren ASIP-Architekturen sollen in Bezug zu bekannten Prozessorsystemen gestellt werden. Um eine 3D-Szene zu zeichnen muss die Grafikkarte mehrere unterschiedliche Operationen für jedes Element dieser Szene durchführen. Dies sind typischerweise die Transformation von Objekten und Licht (Vertex Shader), Beschneiden und Fragmentieren der Polygone (Geometry Shader) sowie letztendlich das Pixel Rendering inklusive Textur-Mapping

2 Themenfeldeinordnung und fachliche Grundlagen

und Interpolation (Pixel Shader). Traditionell wurde für jede Stufe der Grafikprozessierung ein auf die entsprechende Aufgabe hin entwickeltes Subsystem implementiert, mittels fester Datenpfadstruktur zwischen diesen Elementen wurde die sogenannte Shader Pipeline gebildet. Im Zuge stetig wachsender Vielfalt der Anforderungen des Themenfeldes moderner 3D-Grafikbeschleunigung erfolgte die phasenweise Migration von einem ASIC mit hinsichtlich Datenfluss und Funktionalität festgelegter Verarbeitungspipeline zu einem vollständig programmierbaren ASIP-Prozessorsystem. Die zugehörige Grafikbeschleuniger-Architektur wird als Unified Shading bezeichnet. Da diese Prozessoren prinzipiell flexibel auf Daten jeglicher Art rechnen können, ist ein moderner Grafikadapter an sich nicht auf die Bewältigung grafischer Problemstellungen limitiert. General-Purpose Computing on GPU (GPGPU) beschreibt den Ansatz, die GPU zur Berechnung nichtgrafischer Problemstellungen zu nutzen.

2.2.2 Methoden zur parallelen Datenverarbeitung in Prozessoren

Zur Erhöhung des Datendurchsatzes pro Prozessortaktzyklus haben sich verschiedene Konzepte etabliert, welche in das klassisch rein sequentielle Grundentwurfsdesign eines Prozessorsystems Elemente der Parallelität einführen.

Pipelining

Je nach Implementierung des Prozessors treten bei der Abarbeitung Muster stets identischer Teilbearbeitungsschritte auf. Hierzu gehört beispielsweise das Schema der Bereitstellung der Datenoperanden über Leseoperationen, die Durchführung der logischen Operation selbst sowie das Rückschreiben der Ergebnisse in Zielspeicherbereiche. Wird auf Subinstruktionsebene dem klassischerweise vollständig imperativen Abarbeitungsverhalten des Prozessors ein datenflussähnliches Verarbeitungskonzept zur Erledigung dieser elementaren Schrittfolge unterlagert, so können die einzelnen Phasen der typischen Befehlsabarbeitung nebenläufig ausgeführt und Latenzen verdeckt werden. Dieses Architekturkonzept wird als Pipelining bezeichnet.

VLIW und Superskalarität

Komplexere Prozessorkerne besitzen für unterschiedliche Instruktionstypen in der Regel verschiedene Arten funktionspezifischer Verarbeitungswerke nebeneinander integriert. Dies können zum Beispiel Logik für Speicherzugriffe, arithmetisch-/logische Operationen und dedizierte Multipliziererschaltungen sein. Nach klassischem Betriebsablauf würde stets pro Zyklus nur ein Befehl bearbeitet sowie vom Leitwerk nur die Nutzung des jeweils geeigneten Rechenwerks instruiert. Besteht zwischen den Operanden benachbarter Instruktionen keine mathematische Abhängigkeit, welche streng sequentielle Abarbeitung erfordert, so kann Parallelität auf Instruktionsebene (Instruction Level Parallelism, ILP) ausgenutzt werden und Befehle zugleich auf unterschiedliche Teilrechenwerke im Prozessor zugewiesen werden. Zur Nutzung müssen im Prozessor mehrere Datenpfade vorhanden sein.

Das Very Long Instruction Word (VLIW)-Konzept spezifiziert hierfür in jedem Befehlsword für sämtliche im Prozessor enthaltenen Rechenwerkseinheiten direkt ihre momentane Funktion (gegebenenfalls auch Inaktivität). So kann auf einfache Weise, allerdings auf

Kosten erhöhten Speicherbedarfs für den Programmablaufplan, befehlsparallele Ausführung durch statische Encodierung im Befehlsstrom umgesetzt werden. Das VLIW-Konzept implementiert ein Multiple Instruction Multiple Data (MIMD)-Konzept.

Als alternative Architektur zur Nutzung von ILP übernimmt in superskalaren Prozessoren eine dedizierte Logik die Aufgabe, mehrere Standardinstruktionen dynamisch zu aggregieren und diese auf die $n > 1$ Verarbeitungseinheiten des Prozessorkerns zu verteilen. Hierbei muss die Logik ad-hoc prüfen, ob zwischen den potentiell parallel zugeteilten Instruktionen Datenabhängigkeiten existieren, die eine parallele Abarbeitung verbieten würden. Zur Unterstützung der Effizienz der Superskalarität existieren zwei verbreitete Ansätze. Zum einen existiert die Technologie der Out-of-Order-Execution. Hier werden mehrere Instruktionen im Voraus betrachtet und dynamisch umsortiert, um höchstmögliche Befehlsunabhängigkeit zu erzielen. Zum anderen bieten Prozessoren mit hardwaregestützter Multithreading-Unterstützung die Möglichkeit, abwechselnd Instruktionen aus unterschiedlichen Programmablaufplänen abzuarbeiten³. Da unterschiedliche Programme in der Regel auf unterschiedlichen Datensätzen arbeiten, wird im gemeinsamen gemischten Instruktionsstrom der Grad an Interinstruktionsdatenabhängigkeiten verringert. Während das VLIW-Konzept auf die Reduzierung der Logik des Leitwerks auf Kosten einer Speicherbedarfserhöhung für den Programmablaufplan zielt, erhöhen die dynamischen Methoden der Superskalarität die Schaltungskomplexität des Prozessorsteuerwerks bei verringertem Codevolumen.

Multicore-Systeme

Im Multicore-System werden zur Erhöhung des maximalen Durchsatzes mehrere Prozessorkerne nebeneinander in das System integriert. Dies können sowohl in homogener Weise gleichartige Prozessortypen sein (symmetrisches Multicoresystem) als auch Prozessorkerne, welche sich hinsichtlich Eigenschaften und Fähigkeiten heterogen unterscheiden (asymmetrisches Multicoresystem).

Vektor- und SIMD-Prozessoren

Während im klassischen Multicore-System jedes Rechenwerk sein eigenes Leitwerk besitzt, werden bei Vektorprozessoren $n > 1$ Rechenwerke gemeinsam durch nur ein einzelnes Leitwerk simultan gesteuert. Bei stets gleichem Maximaldatendurchsatz wird verglichen mit einem n -Prozessor-Multicoresystem der Schaltungsaufwand entsprechend $n - 1$ Steuerwerken reduziert und so können Chipfläche und Kosten gespart werden. Aufgrund des gemeinsamen Steuerwerkes erlaubt die Architektur nur in jedem Taktzyklus stets einen identischen Instruktionstyp auf sämtlichen Rechenwerken, dort allerdings mit unterschiedlichen Daten, auszuführen. Eine solche Vektorarchitektur wird als Single Instruction Multiple Data (SIMD) bezeichnet. Sie nutzt Datenparallelität der gegebenen Problemstellung: Die Vektorelemente gehören stets gleichartigen Datensätzen an. Folglich müssen die auszuführenden Algorithmen für diese Architektur geeignet sein und ihre Berechnungsterme ausreichend mathematische Datenparallelität bieten, um das Potential des Maximaldatendurchsatzes in der Praxis erreichen zu können.

³Der Hersteller Intel nutzt hierfür die Bezeichnung beziehungsweise den Markennamen Hyperthreading.

2.2.3 Programmierung von Prozessoren

Um eine konkrete Funktionalität auf einem Prozessor zu implementieren ist durch den Entwickler eine entsprechende Folge imperativer Instruktionen zu erzeugen.

Hochsprache

Anstelle der direkten Encodierung der Maschineninstruktionsfolge ist die Programmentwicklung unter Nutzung von Abstraktionsebenen und Methoden des computergestützten Entwurfsprozesses gebräuchlich. Formulierungen der Algorithmen durch Text (oder auch grafische Diagramme) in formalen, problemorientierten Programmiersprachen werden durch sogenannte Compiler automatisiert in Maschinencode überführt. So kann die Entwicklungskomplexität für den Programmierer drastisch gesenkt und auch plattformunabhängiger Code erstellt werden. Jedoch muss stets bedacht werden, dass mit zunehmender Einführung von Schemata zur Abstrahierung im Softwareentwurf, bezogen auf das technische Optimum, tendenziell suboptimale Ergebnisse auftreten können. Für eingebettete, ressourceneffiziente Anwendungen stellt die Hochsprache C, standardisiert beispielsweise nach ISO in der Revision C99 [33], die dominierende Hochsprache dar. Die Programmiersprache ermöglicht durch nahe Orientierung an den zugrunde liegenden technischen Konzepten ressourcenschonende und damit laufzeiteffiziente Programmierung.

Maschinensprache und Assembler

Für sehr laufzeitkritische Programmabschnitte lohnt sich oft trotz im Gesamten guter Ergebnisse bei Verwendung eines Compilers die direkte Manipulation der Maschineninstruktionsfolge des Prozessors durch den Programmierer. Da Maschinensprachebefehle die triviale Arbeitsweise des Prozessors aufzeigen, eröffnet der Blick auf diese eine direkte Sicht auf Potentiale und Schwachstellen des Zusammenspiels von Hardware und Softwarealgorithmus. Entsprechend folgt konkludent die Möglichkeit zur zielorientierten Modifikation des Codes. So können Algorithmen bestmöglich, allerdings verbunden mit hohem Arbeitsaufwand, auf die Strukturen und Rechenwerke der Prozessorarchitektur abgebildet werden. Assembler führt durch sogenannte Mnemonics eine für Menschen besser verständliche Darstellung der Maschinensprache ein. Ein wesentlicher Beitrag der Arbeit soll für die laufzeitkritischen Codeelemente die Gegenüberstellung der Ergebnisse aktueller C-Entwicklungswerkzeuge zu dem Versuch der bestmöglichen händischen Anpassung an die Mikroarchitektur mittels Assembler-Programmcode sein.

Programmierung der GPU und zugehörige Frameworks

Bei den Programmierframeworks der beiden GPU-Hersteller handelt es sich stets um eine Kombination aus einem Compilersystem zur Formulierung der nebenläufigen GPGPU-Datenprozesse auf der GPU sowie einer Laufzeitbibliothek zur Steuerung des GPU-Subsystems von der hierarchisch übergeordneten CPU aus. Die GPGPU-Subroutinen werden typischerweise als Kernel bezeichnet, der GPGPU-Prozessor stellt das sogenannte Device dar. Allen vorliegenden Frameworks ist gemein, dass sie zur Beschreibung der Kernelfunktionalität einen auf der C-Hochsprache basierenden Sprachdialekt nutzen. Dabei wird

das Paradigma des expliziten Parallelismus verwendet, bei dem der Programmierer dem Compiler die Parallelisierbarkeit einer Coderegion durch Syntax explizit signalisiert.

NVidia stellt das Common Unified Device Architecture (CUDA) Software Development Kit (SDK) mit dem Sprachkonstrukt CUDA C für die Programmierung ihrer Grafikprozessoren bereit [34]. Dazu gehören sowohl diverse Bibliotheken für bestimmte Standardalgorithmen wie die Fourier-Transformation und diverse Matrixoperationen als auch ein Tool zur Laufzeitanalyse der GPU. Neben dem herstellereigenen Sprachdialekt CUDA bietet das SDK seit geraumer Zeit ebenfalls Unterstützung für das später eingeführte OpenCL. Für die Softwareentwicklung auf den AMD GPUs wird das AMD Accelerated Parallel Processing SDK [35] angeboten. Das initial als Stream SDK auf der Sprache Brook, ebenfalls einer Ableitung von C, basierende Framework wurde seitens des Herstellers zugunsten einer dem OpenCL-Standard konformen SDK-Umgebung weiterentwickelt. AMD stellt mit den AMD Accelerated Parallel Processing Math Libraries [36] fertige Komponentenbibliotheken für bestimmte Algorithmen zur Verfügung. OpenCL (Open Computing Language) an sich beschreibt eine normierte Formulierungs- und Nutzungsweise für die parallele Programmierung heterogener Systeme [37]. Die Sprach- und Schnittstellendefinition ist weder an einen Hersteller gebunden noch auf die GPU-Hardwarearchitektur an sich beschränkt. Das zugrunde liegende Normungsgremium verfolgt den Anspruch einen Industriestandard zu schaffen. Wie dargestellt bieten beide GPU-Hersteller AMD und NVidia in ihren SDKs heute Unterstützung für diesen Standard an.

2.2.4 Designparadigmen zur effizienten Software-Implementierung

Damit eine Implementierung von Signalverarbeitungsalgorithmen auf einer Prozessorarchitektur performantes Laufzeitverhalten erreicht, sind bei der Erstellung des Programmcodes verschiedene generell allgemeingültige Designparadigmen zu beachten. Diese finden sich in der bekannten Literatur über effiziente Softwareentwicklung, beispielsweise in [38] und [39], [40] und [41] oder auch [42]. Die wesentlichen Aussagen hinsichtlich der Optimierungskonzepte, welche bei der Umsetzung aber auch bereits der vorangehenden Auswahl der Algorithmen zu berücksichtigen sind, lassen sich wie folgt gruppieren und zusammenfassen:

Meidung ineffizienter Prozessorbefehle

Instruktionen, welche auf der Prozessorhardware im Vergleich zu anderen Befehlen hohe Ausführungszeit besitzen, sind zu vermeiden. Als typisches Beispiel ist eine Divisionsoperation auf einem Prozessor in der Regel laufezeitineffizienter als Multiplikationsoperationen. Entsprechend sind Algorithmen zu meiden, die intensiven Gebrauch von solchen Befehlsoperationen machen. Folglich beginnt die Laufzeitoptimierung bereits vor der Programmierung bei der Auswahl beziehungsweise der Entwicklung der Signalverarbeitungsalgorithmen.

Minimierung der Datentransfermenge

Da zum Datentransport nötige Lade- und Speicheroperationen eines Prozessors typischerweise nicht weniger Zeitaufwand bedeuten als mathematische Rechenoperationen selbst,

2 Themenfeldeinordnung und fachliche Grundlagen

gilt auch der Vermeidung unnötiger Datentransfers im Arbeitsspeicher Aufmerksamkeit. Eine alleinige Optimierung hin zu einer minimalen Anzahl arithmetischer Operationen ist ungenügend. Um Speicherbandbreite zu sparen kann es sich als vorteilhaft erweisen, separate Funktionsblöcke prinzipiell unabhängiger Algorithmen entgegen der logischen Partitionierungsweise zu gemeinsamen Funktionsblöcken zu verschmelzen. Dies führt zu einer Softwarearchitektur, welche die Steigerung der Ausführungsperformance durch Schaffung einer möglichst hohen Datenlokalität über das Paradigma eines klar strukturierten Hierarchieaufbaus stellt.

Datenflussanpassung an hardware-spezifische Speicherzugriffsmuster

Eng verwandt mit dem vorhergehenden Punkt ist die Beachtung der hardware-spezifischen Datenflusskonzepte der Prozessorarchitektur. Die Beachtung natürlicher Speicherzugriffsmuster direkt im Algorithmus vermeidet Effizienzverluste durch den Zwang, vor der eigentlichen Berechnung Operationen zur Datenumsortierung durchzuführen. Der Aspekt ist insbesondere bei SIMD-Datenprozessierung mit Anordnung der Daten in Vektoren von Relevanz.

Zugriffszeitorientierte Nutzung der Speicherhierarchie

Im externen Speichersystem implementieren die heutigen Computerarchitekturen Hierarchien unterschiedlich schneller Speicherbereiche. Hierzu zählen beispielsweise Caches, verteilt typischerweise in bis zu drei kaskadierte Stufen der sogenannten L1- bis L3-Caches, oder schnelle Scratchpad-SRAM-Speicher nahe dem Prozessorkern. Es erweist sich für den Programmierer als selbstverständlich, zuerst die internen Registerbänke im Prozessorkern effizient auszunutzen, bevor Plätze in prozessorkernexternen Speichern genutzt werden, und anschließend darauf zu achten, Speicher mit kurzer Zugriffszeit in Bevorzugung zu langsamen Speichern verstärkt zu nutzen.

Nutzung vorberechneter Tabellen

Soweit die Mächtigkeit eines während der Laufzeit aufwändig zu berechnenden Funktionsterms es zulässt, kann es sinnvoll sein, anstelle der Berechnung einen Zugriff auf eine vorab berechnete und im Speicher hinterlegte Ergebnistabelle zu implementieren. Zur Vorteilhaftigkeit muss die benötigte Speicherzugriffszeit unter der für die Berechnung benötigten Prozessorzeit liegen. Jedoch müssen auch sekundäre Effekte des Caching berücksichtigt werden, da durch eine Look-Up-Table (LUT) andere Nutzdaten aus dem schnellen Cachespeicher des Prozessors verdrängt werden können. Ein typisches Beispiel für eine LUT stellt auf Prozessoren ohne entsprechende Spezialrechenwerke die Berechnung trigonometrischer Funktionen dar. Oft kann auch die Nutzung von LUTs für ganze Teilfunktionsterme eines Algorithmus nutzbringend angewendet werden.

Übergang von Fließkomma- zur Festkommaarithmetik

Die Technologie der Fließkommaarithmetik stellt durch zusätzliche Logik zur adaptiven Anpassung der numerischen Präzision einen virtuell vergrößerten Dynamikbereich im Zahlenraum zur Verfügung. Der Einsatz von Fließkommaarithmetik gegenüber der Festkomma-

2.3 Komponenten eines Multistandard-Radiotransceivers

arithmetik zeigt für den vorliegenden Anwendungszweck der Signalverarbeitungskette selbst prinzipiell keinen mathematischen Vorteil, da im System ein stets vorausbestimmbares Verhältnis sämtlicher Signalpegel zueinander besteht. Die Nutzung der Möglichkeit des Übergangs von einer Implementierung in Fließkommaarithmetik zu einer Berechnungskette mit Festkommaarithmetik birgt oft Vorteile:

- Je nach Prozessor werden Festkommaoperationen meist schneller als Fließkommaoperationen ausgeführt.
- Es besteht die Möglichkeit zur Realisierung auf günstigeren Prozessoren ohne jegliche Fließkommaarithmetikeinheit (Floating Point Unit, FPU).

Daneben ermöglichen geringere Bitbreiten der Daten weitere Vorteile. Fließkommawerte einfacher Präzision besitzen eine Wortbreite von 32 bit. Festkommawerte können dagegen auch mit 16 bit oder gar 8 bit definiert werden.

- Dies birgt Potential zur Reduzierung der nötigen Transferbandbreite zwischen Prozessor und Arbeitsspeicher.
- Im SIMD-Betrieb ist oft eine höhere Packungsdichte pro eines Vektors festgelegter Bitbreite möglich, somit kann eine höhere Parallelität und damit ein höherer Durchsatz an Operationen pro Taktzyklus erreicht werden.

Die Wortbreite sämtlicher Instruktionsoperanden sollte deshalb stets so gewählt sein, dass numerische Genauigkeitsanforderungen nicht übererfüllt werden.

2.3 Komponenten eines Multistandard-Radiotransceivers

Der Untersuchungsgegenstand der Arbeit ist motiviert durch den Wunsch nach flexiblen, multistandardfähigen Radiogeräten. Abbildung 2.1 zeigt in beispielhafter, abstrahierender Form auf der ersten Unterebene die drei Hauptthemenfelder auf, die der Entwurf eines multistandardfähigen Systems umfassen muss.

2.3.1 Frontend

Das Frontend dient als Bindeglied der Signalverarbeitung zum Antennensystem. Es hat im Empfangsfall die Aufgabe, ein elektrisches Hochfrequenzsignal dem Signalverarbeitungssystem in einer verwertbaren Form aufzubereiten. Für den komplementären Sendefall ist aus angelieferter Signalverlaufsinformation entsprechende elektrische Hochfrequenzleistung zu generieren. Ein Frontend muss stets die folgenden Funktionsgruppen beinhalten, welche zur Realisierung eines multistandardfähigen Frontend meist auch mehrfach ausgeführt werden müssen:

- Elektrische Anbindung an das jeweilige Antennensystem.
- Vorselektion des Nutzsignalanteils mittels Kanal- und/oder Bandfiltern⁴.

⁴Ungenügender Einsatz von Filtern äußert sich im Empfänger in dem Auftreten von Phantomsignalen bis hin zu sogenannten Zustopfeffekten. Sie entstehen im Empfänger selbst durch Kreuzmodulation an Nichtlinearitäten und sind im Weiteren nicht mehr von tatsächlich existierenden Signalen zu unterscheiden. Im Sendefall äußert sich mangelnde Filterung in der Störung anderer Funkdienste.

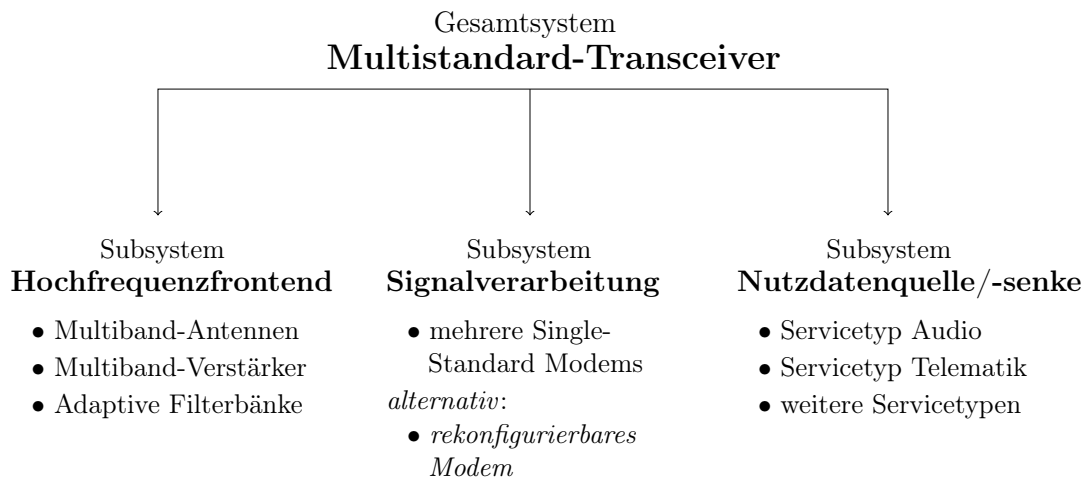


Abbildung 2.1: Übersicht der einzelnen Komponenten beziehungsweise disjunkten Themengebiete, welche für das Produkt eines vollständigen multistandardfähigen Funksystems allesamt zu bearbeiten sind.

- Sende- und/oder Empfangsverstärker⁵ mit dynamischer Pegelregelung.
- Signaltransformationen als Vorverarbeitungsschritte, um über eine niederfrequente Repräsentation des Funksignals (ein sogenanntes Zwischenfrequenz- oder Basisbandsignal) an die eigentliche Signalverarbeitungselektronik angebunden werden zu können.

Es soll betont werden, dass kein praxistaugliches Radiofrontend vollständig flexibel sein oder allein durch nachgeschaltete (Software-)Algorithmen in der Stufe der digitalen Signalverarbeitung fehlende Flexibilität und Leistungsfähigkeit des Frontends kompensiert werden kann. Auch heute ist nicht absehbar, dass in Zukunft ein hinsichtlich Hochfrequenz (HF)-Signalen spektral sauberes, serientaugliches und zugleich in wirtschaftlicher Weise produzierbares Sende- oder Empfangssystem ohne einzelne funktionsfeste Baugruppen in analoger Schaltungstechnik auskommen könnte. Die Technologie des Frontends ist nicht Gegenstand der vorliegenden Arbeit.

2.3.2 Signalverarbeitung

Die eigentliche Radiosignalverarbeitungskette wird auch als Basisbandaufbereitung bezeichnet. Im Sendefall ist die Aufgabe der Signalverarbeitung, die abzustrahlende Wellenform festzulegen. Die Wellenform stellt stets eine Funktion der zu übertragenden Nutzdaten dar. Im Empfangsfall sind analog dazu aus der bei der Übertragung gestörten Empfangswellenform die einst gesendeten Nutzdaten fehlerfrei zu schätzen. In Anlehnung an den Vorgang der Modulation und Demodulation wird eine kombinierte Sende- und Emp-

⁵Je nach Übertragungsszenario entstehen teilweise extrem unterschiedliche Anforderungen an Dynamikumfang und Linearität des Systems.

2.4 Umsetzung von Signalverarbeitung im Sende-/Empfangssystem

fangssignalverarbeitung mit dem Akronym Modem bezeichnet. Im Schichtenmodell nach OSI [43] ist die Basisbandaufbereitung als Teil der physikalischen Schicht einzuordnen.

Die vorliegende Arbeit befasst sich mit Varianten der Umsetzung beziehungsweise Realisierung solcher digitaler Signalverarbeitung. Die Entwicklung neuer mathematischer Wellenformen zur Nachrichtenübertragung ist hingegen nicht Gegenstand der Arbeit. Ein Gerät, welches mehrere Wellenformen unterstützt, kann auf folgende Weise implementiert sein:

- Umschaltbare Bank aus mehreren standardspezifischen Modems.
- Modem mit intern rekonfigurierbarer Signalverarbeitung.

Die in der Arbeit durchgeführten Untersuchungen sollen den Themenbereich der Entwicklung von Modems mit via Softwaretausch rekonfigurierbarer Signalverarbeitung unterstützen.

2.3.3 Nutzdatenquelle/-senke

Für sämtliche Dienstetypen muss im Multistandard-Kommunikationsgerät ein entsprechendes Subsystem zur geeigneten Darstellung der Nutzdaten vorhanden sein. Der Funktionsbereich soll als Nutzdatenquelle oder -senke bezeichnet werden. In der Betrachtungsweise des Schichtenmodells nach OSI finden sich diese Schichten oberhalb der physikalischen Schicht. Sie sind somit klar nicht Gegenstand der vorliegenden Arbeit: Der Sinngehalt sowie das zugehörige Codierungsformat von Daten, welche über die physikalische Datenverbindung des Modems übertragen werden, sind von der Radiosignalverarbeitung selbst vollständig unabhängig und bei der logischen Systembetrachtung strikt zu trennen, denn es können verschiedene Dienstetypen mit unterschiedlichsten Nutzdaten (Audio, Video, Telematik, ...) auf einer identischen Bitübertragungsschicht realisiert werden. Eine Rückwirkung auf die technischen Konzepte der Rohdatenübertragung existiert jedoch nicht.

2.4 Umsetzung von Signalverarbeitung im Sende-/Empfangssystem

Es soll in die grundlegenden Methoden der Signalverarbeitung für Radioanwendungen eingeführt werden. Auch soll die Thematik der Nutzung des Applikationsprozessors erläutert werden.

2.4.1 Analoge Signalverarbeitung

Mit Hilfe der analogen Schaltungstechnik kann nur solche Signalverarbeitung realisiert werden, deren zugrunde liegende Mathematik sich hinreichend genau und direkt durch physikalisch-elektrische Eigenschaften von verfügbaren elektrischen Bauteilen abbilden lässt. So sind Baugruppen in analoger Technik typischerweise aus vergleichsweise funktional einfachen Subsystemen wie Mischer, spannungsgesteuerten Oszillatoren oder Filtern aufgebaut⁶. Während die analoge Signalverarbeitung im Segment des niederfrequenten Basis-

⁶Hiermit können für AM- und FM-Nachrichtensysteme (De)Modulatoren bereits vollständig implementiert werden.

bandes sowie auf der Zwischenfrequenzebene praktisch vollständig durch digitale Signalverarbeitung abgelöst wurde, ist sie stets essentielles Element einer jeden Hoch- und Höchstfrequenzbaugruppe und nach wie vor unverzichtbar: Als ein Beispiel für analoge Signalverarbeitung kann die Verschiebung des Signals im Frequenzbereich zwischen Trägerfrequenz und Basisband gesehen werden, welche im HF-Frontend angesiedelt ist.

2.4.2 Digitale Signalverarbeitung

Für die digitale Signalverarbeitung werden die kontinuierlichen analogen Signale in eine diskrete Wertefolge überführt und anschließend in ein Digitalrechensystem gespeist. Es wird eine Trennung zwischen der elektrischen Implementierung und der logischen Datenverarbeitung eingeführt. Als Digitalrechensystem dienen Logikschaltungen oder Prozessorsysteme. Die Wandlung zwischen analoger und digitaler Domäne erfolgt über Analog-Digital- (ADC) und Digital-Analog-Wandler (DAC). Durch die nun gegebene Möglichkeit zur vollkommen abstrakt-mathematischen Betrachtung des Signalverlaufs erlaubt die digitale Signalverarbeitung die Anwendung beliebiger Algorithmen auf das Signal. Für den Anwendungsfall der Radiosignalverarbeitung eröffnet sich durch die digitale Signalverarbeitung die Möglichkeit zur Realisierung von Modulationsverfahren mit mathematisch komplexem Hintergrund. Im Speziellen eröffnet die digitale Signalverarbeitung gegenüber der analogen Technik beispielsweise die folgenden wichtigen Methoden:

- Exakte, quasi ideale Rechenoperationen.
- Implementierung von idealen Laufzeit-/Verzögerungsgliedern.
- Signalraumtransformationen, beispielsweise Fourier-Transformation.
- Adaptive Modifikation von Funktionsparametern aller Art, beispielsweise adaptive Filtersysteme.

Weitere Informationen zur Theorie der digitalen Signalverarbeitung können unter anderem bei Proakis [44] gefunden werden. Die Nutzung volldigitaler Signalverarbeitung an sich für die Basisband- oder eine Zwischenfrequenz-Domäne ist heute bereits etablierter Stand der Technik, sowohl für moderne Digitalübertragungsverfahren ohnehin als auch für aktuelle Geräte zur Verwendung mit betagteren Analogübertragungsverfahren wie beispielsweise FM oder AM.

2.4.3 Der Begriff Software Defined Radio

Da das Umfeld der vorliegenden Arbeit aus dem Bereich Radiosignalverarbeitung stark durch den jüngst aufgekommenen Begriff Software Defined Radio (SDR) geprägt ist, soll intensiv auf diesen Begriff eingegangen werden. Es soll dabei aber auch verdeutlicht werden, dass das durch diese populäre Bezeichnung definierte Szenario vom existenten Stand der Technik der digitalen Signalverarbeitung aus für sich alleine keine eigenständige, neuartige Entwurfsmethodik darstellt.

Gegenstand eines Softwareradios und Nutzen

SDR ist als eine Ausprägungsform der Implementierung einer digitalen Radiosignalverarbeitung anzusehen. SDR beschreibt das Paradigma, sämtliche Signalverarbeitung eines Radiomodems auf einer Rechenplattform abzubilden, welche unprogrammierbar ist [1]. Gemäß den bekannten Eigenschaften von Software ist Funktionalität der Radiosignalverarbeitungseinheit aktualisier- und austauschbar, alleine durch das Einspielen unterschiedlicher Software-Module. Durch diese Idee eröffnet das Konzept einen hohen Grad an Flexibilität. Das SDR-basierte Modem kann verschiedenste Algorithmen ad-hoc darstellen und so seine Signalverarbeitung potentiell an viele, darunter auch nach dem eigentlichen Entwicklungszeitpunkt nachträglich eingeführte Übertragungsprotokolle anpassen. Die Übertragungshardware des verwendeten Gerätes ist somit vom letztendlich eingesetzten Übertragungsprotokoll abstrahiert. Es stellt somit eine interessante Implementierungsvariante für Multistandard-Transceiver dar.

Als Merkmal kann festgehalten werden, dass in einem Software Defined Radio keine für einen speziellen Algorithmus funktional festgelegten Verarbeitungselemente in der Signalverarbeitungskette des Systems existent sind.

Implementierung

SDR an sich ist folglich eine auf Flexibilität optimierte Realisierungsvariante einer digitalen Radiosignalverarbeitung. Für die Implementierung eines solchen Systems mittels Logikschaltung stehen rekonfigurierbare feldprogrammierbare Logikbausteine (FPGA) zur Verfügung. Alternativ können Mikroprozessoren, vornehmlich digitale Signalprozessoren (DSP), genutzt werden. Eine klare Differenzierung von SDR hin zu klassischer DSP-Signalverarbeitung erweist sich in diesem Fall als besonders schwierig bis im praktischen Fall unmöglich, da in beiden Fällen im Grunde schlicht die Ausführung von Programmcode auf einem Prozessor zur Signalverarbeitung beschrieben wird. Als notwendige Anforderung für SDR muss, wie bereits angedeutet, vorgesehen sein, dass das Programm des Prozessors im Prinzip austauschbar ist.

Abgrenzung

SDR ist selbst, im Gegensatz zur digitalen Signalverarbeitung, kein Enabler für einzelne neue Übertragungstechnologien, denn jeglicher durch ein SDR implementierte Algorithmus kann auch über eine nicht-umkonfigurierbare Digitalrechnerschaltung dargestellt werden. SDR erstreckt sich alleine auf den Bereich eines Systems, welcher die Implementierung der physikalischen Bitübertragungsschicht darstellt. Unbedingt klar abzugrenzen ist die Domäne höherer Protokollschichten, insbesondere der Quellencodierung. Die flexible Nutzung und der Tausch mehrerer Protokollmechanismen, welche auf einer identischen physikalischen Schicht aufsetzen, sind nicht mit SDR zu verwechseln. Folglich ist im partitionierenden Gesamtentwurf eines Radiosystems die Signalverarbeitung in der Nutzdatenquelle beziehungsweise -senke, welche den Strom der unmodulierten Nutzdaten verarbeitet und interpretiert⁷, vom Bereich des SDR-Radiosystems logisch strikt abzutrennen.

⁷Dies können Audio- oder Video-Codecs sein.

Grenzen der Flexibilität

In der Theorie ermöglicht das Konzept weite Flexibilität der Signalverarbeitung. Jedoch erweist sich diese in der Produktrealität als durch unumgängliche Randbedingungen eingeschränkt. Aus wirtschaftlichen Gesichtspunkten wird die im SDR-System existente Rechenleistung in stark kostengetriebenen Massenprodukten nahe an das für den initial gegebenen Anwendungsfall nötige Minimum reduziert sein. Die Fähigkeit, neuartige Funkstandards umzusetzen, endet, sobald die Rechnerressourcen des Systems für die Behandlung der zugehörigen Algorithmen erschöpft sind. Auch ist die Signalverarbeitung nur eine von mehreren Komponenten eines Funksystems. Speziell soll an dieser Stelle auf das HF-Frontend hingewiesen werden. Es ist zwangsweise als feste Schaltung zu implementieren und seine Möglichkeiten stellen eine physikalische Grenze für die Fähigkeiten des Gesamtsystems dar. Des Weiteren sind unter Umständen auch regulatorische Gesichtspunkte in Betracht zu ziehen. Nach den Regeln der Federal Communications Commission (FCC) ist bei Modifikation der Signalverarbeitung eines Radiogerätes eine Erneuerung der Produktzulassung erforderlich. Entsprechend verliert das Gerät seine bestehende Zertifizierung. Hersteller sind unter Umständen mithaftbar, falls unautorisierte Software auf der SDR-Hardware ausgeführt wird und hiergegen keine Schutzmechanismen vorgesehen wurden [46].

Auswirkungen auf Schaltungskomplexität und Kosten

Es hält sich, vor allem im populärwissenschaftlichen Umfeld, die Auffassung, der Einsatz von SDR verringere generell den schaltungstechnischen Aufwand im Radiotransceiver gegenüber konventionell implementierter Signalverarbeitung. So führt auch Reed [45] zu Beginn in das Thema ein mit der These, dass ein Softwareradio-Ansatz den Gehalt an Hochfrequenz- und sonstigen Bauteilen im Radiogerät im Vergleich reduziere. Diese These ist kritisch zu hinterfragen:

- Es ist allgemein bekannt, dass eine flexible Implementierung von Logikschaltungen stets komplexer ist als ein applikationsspezifischer, auf eine einzelne Zielanwendung optimierter Logikschaltungsentwurf.
- Die digitale Basisbandsignalverarbeitung stellt wie erläutert eine logisch klar trennbare Einheit dar. Die Realisierungsvariante der Basisbandsignalverarbeitungsalgorithmen, das heißt ob als konventionelle digitale Signalverarbeitungslogik oder als Softwareradio, ist unabhängig von und somit ohne Einfluss auf dritte Bereiche der Gesamtschaltung. Deshalb besteht eine Auswirkung auf den Komplexitätsanteil anderer Baugruppen (wie dem HF Frontend oder der Nutzdatenquelle/-senke) des Radiotransceivers nicht.

So ergibt sich die Folgerung, dass für eine SDR-Implementierung in der Produktion höhere variable Stückkosten zu erwarten sind als für eine konventionelle Implementierung. Eine positive wirtschaftliche Bilanz des SDR-Ansatzes kann dennoch möglich sein:

- Durch Rückgriff auf programmierbare Standardbauteile kann der Designaufwand möglicherweise, zum Beispiel gegenüber einem ASIC Entwurf, reduziert werden. Die höheren variablen Stückkosten über den Produktionszeitraum müssen also durch die potentiell niedrigeren Fixkosten beim Entwurf kompensiert werden.

- Auch können Synergieeffekte hinsichtlich der zeitlich getrennten Mehrfachnutzung der vorhandenen Rechenplattform den Business-Case positiv beeinflussen. Bei der Betrachtung sämtlicher für das Zielgerät spezifizierten Radio-Funktionalitäten muss für die Entscheidung zugunsten SDR ein Kollektiv aus mehreren separaten, funktional fest implementierten Baugruppen teurer sein als die Hardwareplattform, welche als flexible Alternative sämtliche geforderten funktionalen Überdeckungen zugleich erfüllen kann.

Folglich ist der Ansatz einer Software-Basisbandverarbeitung vor allem für Multistandard-Geräte mit nicht-simultanen Betriebsmodi sowie für Kleinserien mit geringen Stückzahlen wirtschaftlich interessant.

2.4.4 Nutzung des Applikations- und Grafikprozessors

Traditionell wird für prozessorbasierte digitale Signalverarbeitung ein eigenes Subsystem genutzt. Der dortige, oft funktionsspezifisch optimierte Prozessor steht exklusiv der Radiosignalverarbeitung zur Verfügung, somit kann Echtzeitfähigkeit auf unproblematische Weise garantiert werden. Der Fokus der vorliegenden Arbeit liegt darauf, die Verarbeitungsleistung ausgewählter general-purpose Prozessoren für den Anwendungsfall der Radiosignalverarbeitung zu vergleichen und zu untersuchen, ob die Laufzeitanforderungen einer Digitalradiosignalverarbeitungskette auf der entsprechenden Mikroarchitektur prinzipiell erfüllt werden können. Neben dieser grundlegenden Frage werden für die konkrete Produktumsetzung mittels Prozessoren, die als geteilte Ressource von weiteren Diensten genutzt werden, Aspekte hinsichtlich Softwarearchitektur und -stabilität dort in weiteren Untersuchungen zu klären sein. Insbesondere sind auf einem Multitasking-System wie dem Applikationsprozessor im periodischen Betrieb stets gegenseitige Beeinflussungen der Anwendungs-Tasks, das heißt Schwankungen (engl. Jitter) für den zeitlichen Ablauf sowie der Prozessierungslatenz zu erwarten.

Klassifikation der Echtzeit

Einen Abriss über die Thematik des Echtzeitbetriebs und dessen Anforderungen findet sich beispielsweise bei Sha et al. [47]. Für echtzeitfähige Systeme muss eine spezifizierte Zeitdauer ΔT_{RT} zu garantieren sein, nach welcher stets die am Eingang des Systems angelegten Daten am Ausgang in verarbeiteter Form vorliegen. Diese Echtzeitanforderung, eine Zeitschranke, wird auch als Deadline bezeichnet. Die Korrektheit der Ergebnisse des Systems hängt also nicht nur von seinem Berechnungswert oder der isoliert betrachteten Rechendauer/-geschwindigkeit ab, sondern auch davon, ob diese Berechnung garantiert bis zu einem festgelegten Zeitpunkt vollständig durchgeführt wurde. Ein System ohne eine fest spezifizierte Latenz-Zeitschranke ΔT_{RT} ist definitionsgemäß kein Echtzeitsystem.

Nach den Auswirkungen auf das Gesamtsystem, welche durch eine einmalige zeitliche Verletzung dieser Deadline nach sich gezogen werden, kann ein Echtzeitsystem weiter klassifiziert werden:

- Bei einem harten Echtzeitsystem tritt bei Verletzung der Zeitschranke ein totales, katastrophales Systemversagen auf.

2 Themenfeldeinordnung und fachliche Grundlagen

- Bei einem weichen Echtzeitsystem ist nur der Nutzen eines verzögerten Ergebnisses reduziert. Trotzdem kann das System mit geringerer Quality-of-Service, das heißt beispielsweise mit temporären Ausfällen, weiter stabil laufen.

Für beide Varianten eines Echtzeitsystems⁸ folgt aus der Forderung nach einer konstanten Latenz zwischen Eingang und Ausgang, dass der Takt der Zeitdimension an Systemeingang und -ausgang nicht zwingend identisch, aber frequenzstarr gekoppelt ist.

Echtzeit im Kontext Radiosignalverarbeitung

Angewandt für den Anwendungsfall softwarebasierter Radiosignalverarbeitung lassen sich die folgenden Effekte bei einer Verletzung der Echtzeit-Deadline ableiten:

- Im Falle des Nicht-Verlustes des Synchronitätstaktes kurzzeitige (De)Modulationsfehler im Datenstrom, in Mediendatenströmen erkennbar durch Aussetzer/Stummschaltung, Knacken/Störartefakte.
- Im Empfangsfall Instabilität des Trackings zur Signalverfolgung, in Folge Verlust der Empfangssynchronität mit zeitaufwändiger Neusynchronisation.
- Im Sendefall Verletzungen von spezifizierten Zeitablauf-/Rahmenschemen der physikalischen Schicht.
- Im Sendefall spektrale Störungen in Nachbarkanälen durch Diskontinuitäten im fehlerbehafteten Signalverlauf.

Die ersten beiden Punkt mögen bei vereinzeltm Auftreten noch als grenzwertig störend und so tolerierbar angesehen werden, soweit eine kundenorientierte Beurteilung dies zulässt. Verletzungen nach Punkt 3 und 4 sind als katastrophal einzustufen, da sie aus legal-regulatorischer Sicht im Sinne der kooperativen und störungsfreien Frequenznutzung nicht tragbar sind.

Anforderungen für die Laufzeitumgebung

Als Folge ist die verlässliche zeitliche Allokation von Rechenressourcen, das heißt eine entsprechende hart echtzeitfähige Laufzeitumgebung, auf geteilt genutzten Applikationsprozessoren eine zwingende Grundvoraussetzung an die verwendete (Automotive-Entertainment-)Softwaregesamtplattform für das Ziel einer Integration der Radiosignalverarbeitungsroutinen in die Domäne eines Applikations- oder auch eines GPGPU-fähigen Grafikprozessors. Ansonsten kann dieses Vorhaben nicht in Erwägung gezogen werden, Zeitverletzungen durch dritte Applikations-Tasks können nicht toleriert werden. Der Hinweis erfolgt explizit hinsichtlich der im Rahmen der Arbeit mituntersuchten verfügbaren GPGPU-Frameworks, welche zum aktuellen Zeitpunkt in ihren SDKs noch keine Unterstützung harter Echtzeit bieten können.

⁸Oft wird als Mittelweg eine weitere Stufe mit der englischen Bezeichnung *firm* geführt, welche als Verschärfung der weichen Echtzeit besagt, dass die Nützlichkeit eines Ergebnisses nach der Zeitschranke nicht nur reduziert sondern gleich Null ist.

2.5 Konzepte der Radiosignalverarbeitung

Zum Verständnis sollen die wichtigsten Konzepte der im Weiteren verwendeten nachrichtentechnischen Methoden kurz umrissen werden. Hierzu folgt ein Überblick über die Signalmodulation, die digitale Signalrepräsentation und das Konzept eines digitalen Übertragungssystems. Ausführliche Informationen über die eingeführten Methoden finden sich beispielsweise bei Proakis/Salehi [48].

2.5.1 Modulation und Basisbandsignaldarstellung

Durch Modulation können Telekommunikationsverfahren hinsichtlich Störresistenz und genutztem Frequenzbereich an das jeweilige genutzte physikalische Übertragungsmedium bestmöglich angepasst werden. Auch kann durch Multiplexverfahren ein gemeinsames Übertragungsmedium zwischen mehreren Diensten geteilt und so Mehrfachübertragungen realisiert werden. Der Modulator bezeichnet die elektrische beziehungsweise logische Baugruppe zur Wandlung der Signale im Sender. Der inverse Funktionsblock wird am Empfänger als Demodulator benannt.

Definition der Modulation

Modulation bezeichnet die Veränderung eines Schwingungsprozesses in Abhängigkeit von einem kontinuierlichen oder diskreten Nutzdatensignal $q(t)$ oder $q[k]$. Das modulierte hochfrequente Signal $s_{\text{HF}}(t)$ orientiert sich dabei grundsätzlich an der sogenannten Trägerschwingung $s_0(t) = \cos(2\pi f_0 t)$ mit der Trägerfrequenz f_0 . Bezüglich dieses Bezugssignals werden Amplitude und Phasenwinkel von $s_{\text{HF}}(t)$ ständig variiert. Dabei besteht eine mathematisch eindeutige Abbildung zwischen dem Verlauf der instantanen Signalparameter des Schwingungsprozesses von $s_{\text{HF}}(t)$ sowie dem Funktionsverlauf des Nutzdatensignals $q(t)$ beziehungsweise $q[k]$. Somit ist durch eine inverse Operation, die sogenannte Demodulation, stets die vollständige Rekonstruktion des ursprünglichen Nutzdatensignals aus dem modulierten Signal $s_{\text{HF}}(t)$ möglich. Im Leistungsdichtespektrum entstehen durch die Modulation Seitenbänder um die Trägerfrequenz f_0 .

Äquivalentes komplexes Basisband

Es existieren im Feld der Nachrichtentechnik eine Vielzahl von Schaltungs- und Signaldarstellungsvarianten zur Modulation eines Signals, welche je nach Anwendungsfall spezielle Vorteile bieten. Im Folgenden soll das universelle Konzept einer Modulation mit Hilfe der intermediären Signaldarstellungen des äquivalenten komplexen Basisbandsignals beschrieben werden.

Für die Beschreibung des Amplitudenwertes und des Phasenwinkels der modulierten hochfrequenten Trägerschwingung $s_{\text{HF}}(t)$ über die Zeit bietet es sich an, dies über den Phasor eines komplexwertigen Zeitsignals $\underline{s}(t)$, des äquivalenten komplexen Basisbandsignals, zu beschreiben. Die Nutzinformation $q(k)$ oder $q[k]$ ist vom Datenmodulator auf eine solche komplexe Einhüllende $\underline{s}(t)$ abzubilden. Sodann ist diese auf die hochfrequente Trägerschwingung zu übertragen.

$$s_{\text{HF}}(t) = \|\underline{s}(t)\| \cdot \cos(2\pi f_0 t + \arg\{\underline{s}(t)\}) \quad (2.1)$$

2 Themenfeldeinordnung und fachliche Grundlagen

Dies resultiert in der Frequenzdomäne in einer Verschiebung des zweiseitigen Spektrums des komplexwertigen Basisbandsignals hin zur gewünschten Trägerfrequenz f_0 und Abbildung auf ein reellwertiges Signal s_{HF} .

Inphase/Quadratur-Mischung

Die Polardarstellung des Basisbandsignals gemäß Amplitude $\|\underline{s}(t)\|$ und Phase $\arg\{\underline{s}(t)\}$ lässt sich alternativ auch entsprechend der kartesischen Betrachtung der komplexen Ebene als eine Überlagerung zweier 90° phasenverschobener Komponenten Inphase $s_I(t)$ und Quadratur $s_Q(t)$, entsprechend Realteil und Imaginärteil, deuten.

$$\|\underline{s}(t)\| \cdot e^{j\arg\{\underline{s}(t)\}} = s_I(t) + js_Q(t) \quad (2.2)$$

Das Signal $s_{\text{HF}}(t)$ mit variablem Phasenwinkel kann über diese Darstellung auf schaltungs-technisch einfache Weise direkt aus zwei phasenfesten Einzelschwingungen synthetisiert werden, ohne dass ein direkter Eingriff in den Phasenwinkel des Schwingungsprozesses nötig ist. Dazu erfolgt das Multiplizieren der reellen Teilkomponente $s_I(t)$ mit $\cos(2\pi f_0 t)$. Selbiges wird für die imaginäre Teilkomponente $s_Q(t)$ durchgeführt mit der zweiten, phasenstarr gekoppelten Hilfsträgerschwingung $\sin(2\pi f_0 t)$. Da diese jedoch um exakt 90° phasenverschoben ist, bleibt die Orthogonalität bei der additiven Überlagerung der Produktterme erhalten.

$$s_{\text{HF}}(t) = s_I(t) \cdot \cos(2\pi f_0 t) + s_Q(t) \cdot \sin(2\pi f_0 t) \quad (2.3)$$

Sind diese beiden Mischsignale $s_I(t)$ und $s_Q(t)$ bandbegrenzt jeweils mit der oberen Grenzfrequenz f_g , so belegt das kombinierte Signal $s_{\text{HF}}(t)$ im Spektrum den zweiseitigen Frequenzbereich $(f_0 - f_g) < f < (f_0 + f_g)$. In der technischen Schaltungspraxis ist diese Art der Mischung bekannt als Inphase/Quadratur (IQ)-Mischverfahren. Am Empfänger können aus dem ankommenden Signal $s_{\text{HF}}(t) \approx \tilde{s}_{\text{HF}}(t)$ die beiden ursprünglichen Signalkomponenten durch einen inversen Mischprozess wieder getrennt werden. Der inverse Mischprozess verlangt, dass der ursprüngliche sendeseitige Trägerschwingungsprozess im Empfänger durch eine Synchronisationseinrichtung phasenkohärent geschätzt und nachgebildet wird. Zur Rekonstruktion der Signale $\tilde{s}_I(t)$ und $\tilde{s}_Q(t)$ ist ein Tiefpassfilter nötig, um durch den Mischprozess entstehende Mischprodukte aus Spiegelfrequenzanteilen zu unterdrücken. Bild 2.2 zeigt das theoretische Blockschaltbild eines IQ-Mischsystems für Sender und Empfänger.

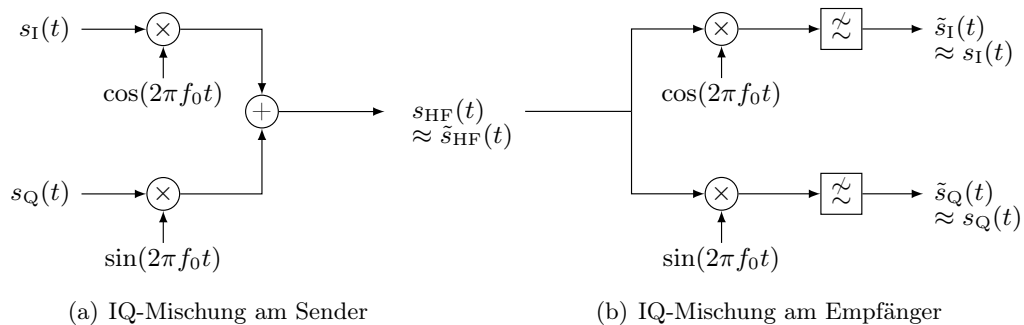


Abbildung 2.2: Grundschtaltung der IQ-Mischverfahren.

Die über dieses Standardverfahren empfangsseitig gewonnene äquivalente komplexwertige Basisbandsignalrepräsentation eines Funksignals $\tilde{s}(t) = \tilde{s}_I(t) + j\tilde{s}_Q(t)$ soll in einer digitalisierten Darstellung ihres Real- und Imaginärteils im Weiteren als Schnittstellensignale der untersuchten Basisbandsignalverarbeitungskette zum Frontend dienen.

2.5.2 Repräsentation eines analogen Signals in der digitalen Domäne

Der folgende Abschnitt soll die allgemeinen Grundlagen zur Wandlung zwischen analogen elektrischen Signalen und der digitalen Signalrepräsentation zusammenfassen. Ohne Beschränkung der grundlegenden Allgemeinheit der Reversibilität des Prozesses wird der Fall einer Wandlung von der analogen zur digitalen Domäne betrachtet. Die klassische Analog-Digital-Wandlung umfasst die im Folgenden genannten Schritte in dieser Reihenfolge. Dabei sind Randbedingungen einzuhalten, damit kein Informationsverlust im Zuge der Analog-Digital-Wandlung erfolgt.

Zeitquantisierung

Ein reellwertiges Signal $s(t)$ ist periodisch abzutasten mit der Abtastfrequenz f_s . Hierdurch erfolgt die Einteilung des Signals in eine diskrete Folge mit regelmäßigen Zeitintervallen der Länge $T = 1/f_s$.

$$s[k] = \sum_{n=-\infty}^{+\infty} \delta[k - nT] \cdot s(nT) \quad (2.4)$$

Der Abtastprozess ist graphisch in Abbildung 2.3 dargestellt. Für ein Signal, welches spektral den Bereich $f_1 < f < f_2$ belegt, also die Bandbreite $\Delta f_b = f_2 - f_1$ besitzt, ist die für eine korrekte zeitdiskrete Repräsentation minimal nötige Anzahl der Abtastpunkte gemäß der Mindestabtastfrequenz $f_s = 2 \cdot \Delta f_b$ gegeben. Pro 1 Hz Signalbandbreite sind also zwei

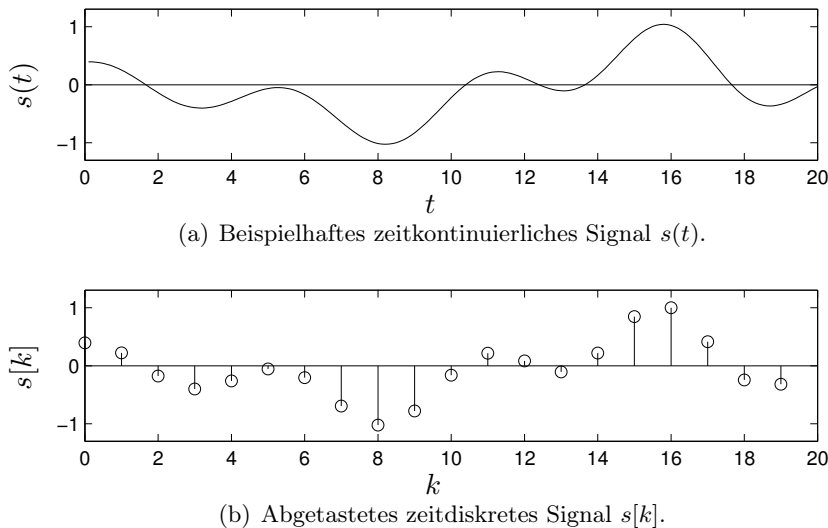


Abbildung 2.3: Prinzip der Signalabtastung mit zeitdiskreter Repräsentation eines beispielhaften Signals.

2 Themenfeldeinordnung und fachliche Grundlagen

reellwertige Abtastpunkte nötig⁹.

Wertdomäne

Bei der Wertequantisierung erfolgt die Eingruppierung eines jeden Abtastwertes in eine diskrete Anzahl meist regelmäßig verteilter Werteintervalle. Die Anzahl der Intervalle ist in der Regel $n = 2^N$, wobei N die Bitbreite des Analog-Digital-Wandlers darstellt. Die pauschale Abbildung eines analogen Signalwertes $s[k]$ in den mittleren Wert eines Quantisierungsintervalls $s_q[k]$ resultiert stets in einem Präzisionsverlust, n stellt somit eine Schranke für die Wandlungsgenauigkeit. Dieser sich ergebende Quantisierungsfehler $e[k]$ kann als additive Störgröße im gewandelten diskreten Signal $s_q[k]$ modelliert werden.

$$s_q[k] = s[k] + e[k] \quad (2.5)$$

Damit eine signalgetreue digitale Repräsentation vorliegt, muss die Leistung des Quantisierungsfehlers $e[k]$ gegenüber der minimal zu erwartenden Nutzsignalleistung vernachlässigbar klein sein.

$$\sum_k \|e[k]\|^2 \ll \sum_k \|s[k]\|^2 \quad (2.6)$$

Aufgrund des additiven Charakters von $e[k]$ ist das Signal- zu Störleistungsverhältnis (engl. Signal-to-Noise Ratio, SNR) einer wertdiskreten Signalrepräsentation $\text{SNR}(s_q[k])$ prinzipbedingt beschränkt und geringer als das des wertkontinuierlichen Eingangssignals $\text{SNR}(s[k])$. Gemäß bekannter Theoreme der Systemtheorie [44] kann für die Schranke des nutzbaren Dynamikbereichs einer Wertdiskretisierung $s_q[k]$ mit Wortbreite von N Bit gezeigt werden:

$$\text{SNR}(s_q[k]) < 4.77\text{dB} + 6.02\text{dB} \cdot N - 20\text{dB} \cdot \log_{10}(\text{PAR}(s[k])) \quad (2.7)$$

So kann aus der Anforderung eines Mindeststörabstandes die notwendige Wortbreite des Wandlers¹⁰ aber auch des digitalen Signalrechen-systems abgeleitet werden. Es ist zu beachten, dass das erzielbare Störleistungsverhältnis neben der Wandlerbitzahl abhängig von der Amplitudenwerteverteilung der Quellsignalform ist. Das Verhältnis des Signalspitzenwertes zum Effektivwert, das heißt der Wurzel des quadratischen Mittels der Signalfolge, ist dazu definiert als der Scheitelfaktor (engl. Peak-to-Average Ratio, PAR).

$$\text{PAR}(s[k]) = \frac{\|\max\{s[k]\}\|}{\sqrt{\frac{1}{l} \sum_{k=1}^l \|s[k]\|^2}} \quad (2.8)$$

⁹Solange dies erfüllt ist, muss die Abtastfrequenz f_s für eine Abtastung ohne Informationsverlust nicht zwingend über der Frequenz der höchsten Signalanteile f_2 liegen. Dies wird als Unterabtastung eines Signals bezeichnet.

¹⁰Die genannte theoretische Größe erfasst jedoch nicht sämtliche Effekte einer realen Wandler-Implementierung. Hierzu zählen beispielsweise Linearitätsfehler im Kollektiv der Abtastintervalle mit resultierenden harmonischen Signalverzerrungen, Störsignaleinkopplungen oder weitere schaltungsinterne Rauschquellen. Deshalb ist in der Praxis für Wandler die Größe des störungsfreien Dynamikbereichs (engl. Spurious Free Dynamic Range, SFDR) anzugeben. Sie setzt die maximale spektrale Leistungsdichte eines Nutzsignals ins Verhältnis zur größten messbaren spektralen Störkomponente nach der Wandlung.

Für Wandler aus dem Bereich der Audiosignalverarbeitung wird typischerweise ein vollausgesteuertes sinusförmiges Quellsignal angenommen, sodass mit $20 \log_{10}(\text{PAR}(s[k]))=3 \text{ dB}$ als Grenze für Störleistungsverhältnis wie allgemein bekannt für das wertquantisierte SNR als Schranke anzugeben ist:

$$\text{SNR}(s_q[k]) < (6.02 \cdot N + 1.76) \text{dB} \quad \text{bei} \quad \text{PAR}(s[k]) \hat{=} 3 \text{dB} \quad (2.9)$$

Jedoch gilt diese Annahme nicht für anspruchsvollere Signalformen. Hierzu gehören speziell diejenigen, welche in der Nachrichtentechnik beispielsweise von den später erläuterten OFDM-Mehrträgerübertragungsverfahren genutzt werden und die sehr hohe Impulsspitzen im Zeitsignalverlauf aufweisen. Bei typischer Approximation eines solchen annähernd zufällig gaußverteilten Zeitsignals auf einen begrenzten Spitzenwert $\text{PAR} \hat{=} 13 \text{dB}$ ergibt sich gegenüber dem Audio-Fall eine erhöhte Anforderung an den Wandler gemäß

$$\text{SNR}(s_q[k]) < (6.02 \cdot N - 8.24) \text{dB} \quad \text{bei} \quad \text{PAR}(s[k]) \hat{=} 13 \text{dB} \quad (2.10)$$

Es ist also zu beachten, dass für das Nachrichtenübertragungssystem mit OFDM-Mehrträgermodulation bei gleichem Ziel-SNR circa 2 bit höhere Wandler- und Rechengenauigkeit vorzuhalten sind als für den Standardfall, in dem ein sinusähnlicher Verlauf der HF-Wellenformen angenommen wird.

2.5.3 Digitale Übertragungsverfahren

Weiterhin soll knapp in die bekannten relevanten Grundlagen digitaler Übertragungssysteme eingeführt werden.

Digitale Modulation und Mapping

Bei digitaler Modulation wird der Trägerschwingungsprozess anhand eines festgelegten Symbolalphabetes \mathcal{M} begrenzter Mächtigkeit variiert. Bei einfachen QAM- und PSK-Modulationen korrespondiert dieses Symbolalphabet im exakten Abtastzeitpunkt der Wandlung des zeitkontinuierlichen Basisbandsignals $\underline{s}(t)$ mit einer Schar festgelegter Punkte in der zeitdiskreten komplexen Ebene von $\underline{s}[k]$.

$$\underline{s}[k] \in \mathcal{M} \quad \forall k \quad (2.11)$$

Dies kann visuell einfach dargestellt werden, Beispiele verschiedener verwendeter Konstellationen \mathcal{M} finden sich in Abbildung 2.4. Die injektive Abbildung der Werte eines Ausschnitts der Sendedatenfolge, je nach Mächtigkeit mehrere Datenbits, auf einen signalisierten Konstellationspunkt wird als Mapping bezeichnet. Die Schrittrate der Modulation ist die Symbolrate, die für das grundlegende QAM- beziehungsweise PSK-System reziprok aus der Symboldauer T_s , dort typischerweise identisch zum Abtasttakt $T=1/f_s$, folgt. Ein konkret übertragener Konstellationspunkt $\underline{s}[k]$ eines Zeitpunktes k wird als Symbol bezeichnet. Die Robustheit des Modulationsverfahrens gegenüber überlagerten Störsignalen ist verknüpft mit der euklidischen Distanz der einzelnen Konstellationspunkte in der komplexen Werteebene.

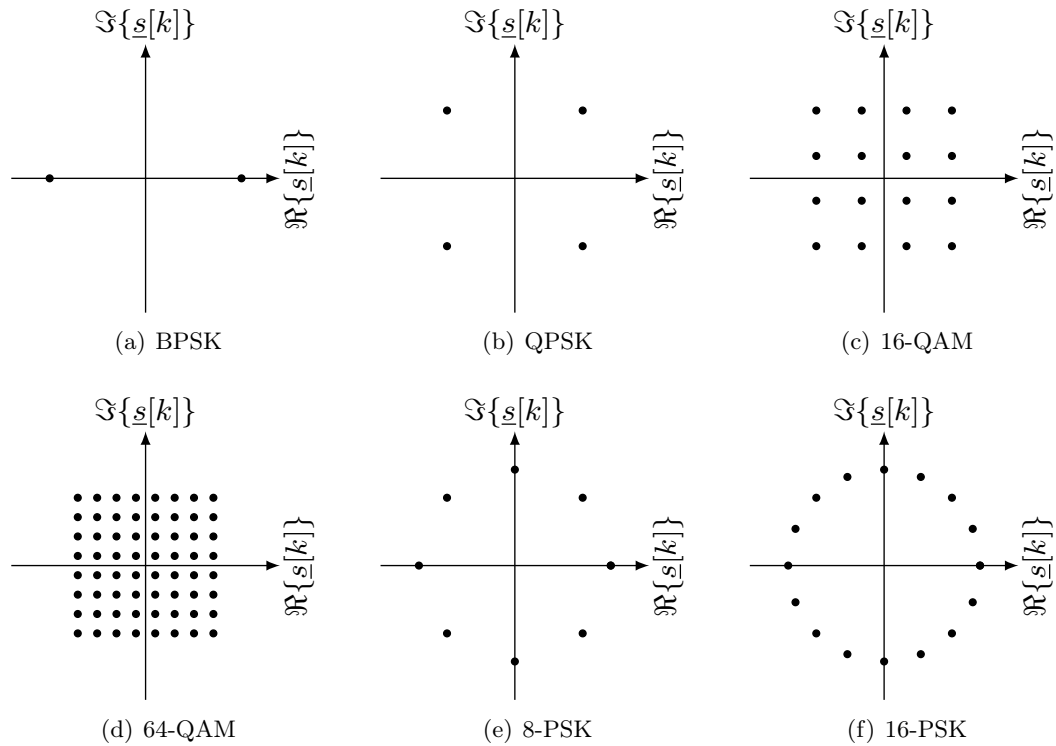


Abbildung 2.4: Beispiele für die Signalkonstellationen in der komplexwertigen Basisbandenebene $\underline{s}[k]$ verschiedener grundlegender QAM-/PSK-Modulationsverfahren bei kohärenter Abtastung im Symboltakt.

Modell der zeitdiskreten Übertragungsstrecke

Am Empfänger kommen die einzelnen gesendeten Symbole beziehungsweise Punkte im Konstellationsdiagramm $\underline{s}[k]$ in gestörter Form an. Dabei ist Rauschen $\underline{n}[k]$ additiv überlagert, zudem verschmiert gegebenenfalls eine in Amplitude als auch Phase dispersiv wirkende Impulsantwort $\underline{h}[k]$ des Übertragungskanals durch eine Faltungsoperation die initial gesendete Datenfolge und somit die Konstellation mehr oder weniger stark.

$$\hat{\underline{s}}[k] = \underline{s}[k] * \underline{h}[k] + \underline{n}[k] \quad (2.12)$$

Werden im Demodulator die Realisierungen von Empfangssymbolen deshalb fälschlicherweise fremden Konstellationspunkten zugeordnet, so treten trotz ausreichendem SNR Bitfehler in der digitalen Übertragung auf. Ist im Funkübertragungskanal Dispersion zu erwarten, sodass relevante Energieanteile die Größenordnung der Symboldauer erreichen oder übersteigen, dann kann durch die Intersymbolinterferenz keine direkte Demodulation des Signals mehr erfolgen. Es muss dann vor dem Demodulator eine Entfaltung mit der Inversen von $\underline{h}[k]$, eine sogenannte Signalentzerrung, durchgeführt werden.

Signaldarstellung im Frequenzbereich und Mehrträgermodulation

Eine elegante Übertragungsmethode mit inhärenter Resistenz gegen Dispersionseffekte bietet die Klasse der Mehrträgermodulationsverfahren. Diese unterteilen eine einzelne hochratige Übertragung in mehrere an sich disjunkte Einzelübertragungen, die gemäß dem Frequenzmultiplexverfahren getrennt sind. Die Eingangsdaten werden anstelle sequentieller Übertragung zu Vektoren $\underline{S} = \{S[0], S[1], \dots\}$ gruppiert und dann parallel auf sämtlichen Teilübertragungsstrecken gesendet. Das Gesamtsendesignal ist die additive Überlagerung der hochfrequenten Schwingungsprozesse sämtlicher modulierter Subträger mit den Indizes K . Jeder individuelle Subträger $\underline{S}[K]$ kann nach einem beliebigen Verfahren moduliert sein. In der Regel werden hierfür PSK- wie auch QAM-Konstellationen verwendet. Gewöhnlich ist der Symboltakt sämtlicher Teilübertragungsstrecken synchron und die Spektralanteile der einzelnen entstehenden Teilübertragungen sind vollständig orthogonal, man spricht dann von Orthogonal Frequency Division Multiplex (OFDM). Das Paket der zeitsynchron parallel übertragenen Symbole wird im Sprachgebrauch als OFDM-Symbol bezeichnet.

Die Synthese eines Summensignals sämtlicher Teilübertragungsstrecken bei gleichzeitiger Sicherstellung der Orthogonalität liefert auf einfache Weise die diskrete Fourier-Transformation. Diese kann effizient mit dem Algorithmus der schnellen Fourier-Transformation (engl. Fast Fourier Transform, FFT) implementiert werden. Typischerweise wird im Modulator die inverse Fourier-Transformation genutzt. Die komplexwertige Ausgangsdatensequenz, welche sich im Zuge der Transformation für den Symbolvektor \underline{S} ergibt, wird anschließend zur Aufprägung auf die Trägerfrequenz einem IQ-Modulator als ein Basisbandsignalabschnitt $\underline{s}[k]$ zugeführt.

$$\underline{s}[k] = \mathfrak{F}^{-1}\{\underline{S}[K]\} \quad (2.13)$$

Im Demodulator am Empfänger erfolgt die Rücktransformation durch die Vorwärts-Fourier-Transformation. Die dispersiv wirkende Impulsantwort der Übertragungsstrecke $\underline{h}[k]$ ist ebenfalls in die Transformation einbezogen und so erscheint am Empfänger jeder Subträger gewichtet mit einem komplexwertigen Faktor $\underline{H}[K]$.

$$\begin{aligned} \hat{\underline{S}}[K] &= \mathfrak{F}\{\underline{s}[k] * \underline{h}[k] + \underline{n}[k]\} \\ &= \mathfrak{F}\{\underline{s}[k]\} \cdot \mathfrak{F}\{\underline{h}[k]\} + \mathfrak{F}\{\underline{n}[k]\} \\ &= \underline{S}[K] \cdot \underline{H}[K] + \underline{N}[K] \end{aligned} \quad (2.14)$$

Jedoch sind die einzelnen Datenwerte im OFDM-Empfangsprodukt $\hat{\underline{S}}[K]$ nicht mehr wie in Gleichung 2.12 untereinander durch Intersymbolinterferenz gestört, sondern trotz Dispersion klar separiert. Die Frequenzselektivität der Impulsantwort $\underline{h}[k]$ kann zu spektralen Leistungseinbrüchen im Signal führen. Entsprechend einer geringen Skalierung durch $\underline{H}[K]$ können bestimmte Unterträger dann aufgrund mangelnder Signalenergie möglicherweise nicht mehr korrekt demoduliert werden. Dieser Effekt wird als Schwund (engl. Fading) bezeichnet und muss durch Redundanz und Fehlerkorrekturverfahren abgefangen werden.

Effekte der Kanaldispersion würden weiterhin die Signalabschnitte an den Übergangsstellen zwischen einzelnen OFDM-Symbolen $\underline{S}_i, \underline{S}_{i+1}$ stören. Um dies zu verhindern wird zwischen die Symbole ein sogenanntes Schutzintervall (engl. Guard Interval, GI) eingefügt, sodass $\underline{h}[k]$ sicher aus- beziehungsweise einschwingt. Es wird als zyklische Verlängerung des Signalabschnittes des OFDM-Symbols, also als Teilkopie, umgesetzt.

Interleaving

Um den Einfluss von Signalstörungen gleichmäßig über sämtliche Bitpositionen einer Informationssequenz zu verteilen und ein lokales, katastrophales Versagen des Fehlerschutzes innerhalb einer gestörten Bitgruppe zu verhindern, kommt häufig ein Verfahren zur Verwürfelung der Anordnung der Bitpositionen zum Einsatz. Dieses Interleaving vermeidet das Auftreten von Bündelstörungen und kann die Effizienz des Fehlerschutzes verbessern. Neben zeitlichem Interleaving, welches sich gegen Impulsstörungen als hilfreich erweist, kann in einem Mehrträgerverfahren auch Frequenzinterleaving gegen schmalbandige Unzulänglichkeiten im Frequenzspektrum des Übertragungskanals angewendet werden.

Fehlerkorrekturverfahren und Kanalcodierung

Fehlerkorrekturverfahren sind nötig, da aufgrund von Störeinflüssen die digitale Nachricht am Empfänger möglicherweise nicht korrekt demoduliert wird. Die Schutzverfahren lassen sich in zwei Klassen einteilen:

- Automatische Wiederholungsanfragen (engl. Automatic Repeat Request, ARQ): Mit Prüfsummenverfahren werden Fehler erkannt, der entsprechende Datensatz wird vom Empfänger durch eine Nachricht an den Datensender aktiv erneut angefordert.
- Vorwärtsfehlerkorrektur (engl. Forward Error Correction, FEC): Im Signal kontinuierlich eingebettete Redundanz wird genutzt, um Übertragungsfehler direkt am Empfänger ohne Kontaktaufnahme zum Sender der Nachricht zu korrigieren.

ARQ-Verfahren finden sich prinzipbedingt nur in Systemen mit Rückkanal und somit nicht klassischen Broadcast-Diensten. Da sie in Protokollschichten oberhalb der physikalischen Schicht (typischerweise in Schichten, welche entsprechend dem OSI-Modell mit Data Link oder Transport Layer korrespondieren) angesiedelt sind, sind sie somit nicht spezifischer Teil einer Betrachtung von Softwareradio-Implementierungen.

Hingegen stellen die FEC-Mechanismen, deren Gebiet auch bezeichnet wird als Kanalcodierung, eine essentielle Komponente einer jeden physikalischen Bitübertragungsschicht dar. Ein Teil der verfügbaren Kanalkapazität wird anstelle zur Übertragung von Nutzinformation zur Übertragung von Redundanz genutzt, welche gemäß einer dem Sender als auch Empfänger bekannten injektiven Codierungsvorschrift bei der Wandlung der Eingangsdatensequenz in die Ausgangsfolge eingebracht wird. Durch diese eingebettete Zusatzinformation lassen sich nach den einschlägigen Regeln der Informationstheorie Übertragungsfehler bis zu einem gewissen Grade erkennen und/oder korrigieren. Eine charakteristische Größe eines Fehlerkorrekturverfahrens stellt die Coderate R_c als Verhältnis der Datenrate von Encodereingang zu -ausgang dar. Je niedriger die Coderate, desto weniger Nutzinformation wird bei gleichzeitig gesteigerter Redundanzinformation und somit gesteigerter Robustheit übertragen. Vorwärtsfehlerkorrekturverfahren können eingeteilt werden in die Gruppen der Block-Codes (beispielsweise darunter Reed Solomon- und LDPC-Codes) und Faltungs-Codes.

Quellencodierung

Die Betrachtung der Methoden zur Quellencodierung ist nicht Gegenstand der vorliegenden Arbeit, da die Verarbeitung der Nutzdaten als Teil der spezifischen Geräteapplikation und nicht der physikalischen Bitübertragungsschicht anzusehen sind, jedoch sind sie essentielle Voraussetzung für digitale Kommunikationssysteme. Vor der eigentlichen Datenübertragung erfolgt in der Regel eine Kompression der digitalen Nutzdaten. Sie hat zum Ziel, die benötigte Datenrate des Übertragungssystems so weit wie möglich abzusenken. Dazu nutzt die Quellencodierung Methoden zur Irrelevanz- und Redundanzreduktion.

Redundanzreduktion nutzt statistische Eigenschaften des Quellsignals. Die Informationsdichte (Entropie) des Signals wird erhöht, wenn durch Prädiktionsverfahren und Entropiecodierer wiederkehrende Muster in den Quelldatensätzen erkannt und eliminiert werden können. Der Prozess kann verlustfrei am Empfänger umgekehrt werden.

Durch Irrelevanzreduktion werden an der Sendestelle diejenigen Datenanteile identifiziert und ausmaskiert, welche zwar Datenrate beanspruchen würden, aber nach der Charakteristik des Nutzers (beispielsweise des menschlichen Ohrs) nicht oder nur unwesentlich zum Gesamteindruck der Nachricht beitragen. Sie können am Empfänger bei der Rekonstruktion ohne negative Auswirkung durch willkürliche Füllinformationen ersetzt werden. Die Irrelevanzreduktion ist verlustbehaftet. Somit ist die empfangene Nachricht keine exakt identische Kopie der gesendeten Nachricht, weshalb die Irrelevanzreduktion im Allgemeinen nicht für Maschine-zu-Maschine-Kommunikation, sondern nur für audio-visuelle, menschenlesbare Nachrichten geeignet ist.

3 Verfügbare Mikroarchitekturen für eingebettete Infotainmentsysteme

3.1	Übersicht der verfügbaren Mikroarchitekturen	40
3.2	Analyse der Mikroarchitekturen	41
3.2.1	Intel Bonell (Intel Atom)	41
3.2.2	AMD Bobcat (AMD Fusion)	43
3.2.3	ARM Cortex-A	44
3.2.4	Texas Instruments TMS320C64x	47
3.2.5	NVidia G9x (Geforce 9)	49
3.2.6	AMD Radeon R800 (AMD Radeon HD)	51
3.2.7	Imagination Technologies SGX (PowerVR)	54
3.3	Einordnung der GPU als Prozessorklasse	56
3.3.1	Betrachtung der Prozessorarchitektur	56
3.3.2	Betrachtung des GPU-Programmierframeworks	57
3.3.3	Softwareoptimierungsprinzipien auf der GPU	59
3.3.4	Fazit zur GPU-Architektur	61
3.4	Architektur-Implementierungsvarianten	62
3.4.1	Betrachtung hinsichtlich theoretischen Maximaldurchsatzes . . .	65

Im vorliegenden Kapitel soll eine Auswahl derjenigen low-cost/embedded-tauglichen Prozessorarchitekturen vorgestellt und vergleichend untersucht werden, welche aussichtsreiche Kandidaten für eine konkrete Hardwarerealisierung einer zukünftigen Automotive Entertainment-Plattform mit potentiell in Software realisierter Signalverarbeitung sind. Die Auswahl umfasst in der Begutachtung auch Grafikprozessoren des Systemverbundes, dabei soll bewusst eine Betrachtung gewählt werden, die stärker an der Schaltungstechnik orientiert ist als in der üblichen, vom Chipdesign abstrahierenden Standardliteratur zur GPU. Diese beschreibt das Themenfeld typischerweise aus der Sicht des die Architektur nutzenden Programmierers. Durch die schaltungstechnik-orientierte Betrachtung wird ein technologischer Vergleich und so eine bessere Einordnung der Prozessorklasse GPU zu den klassischen Prozessoren möglich.

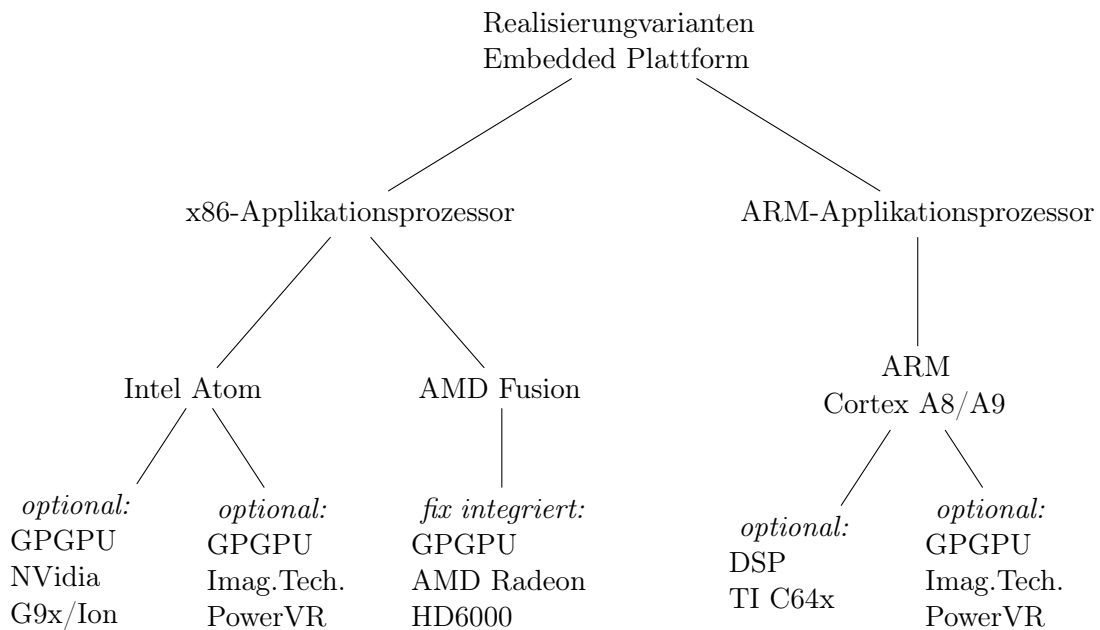


Abbildung 3.1: Übersicht der Plattformarchitekturen und deren Varianten, die zur Implementierung des eingebetteten Infotainment-Zielsystems konkret zur Auswahl stehen.

3.1 Übersicht der verfügbaren Mikroarchitekturen

Abbildung 3.1 zeigt für die Umsetzung einer Infotainment-Zielplattform mögliche Realisierungsvarianten auf. Den Ausgangspunkt stellen die heutigen zwei Hauptgruppen der Applikationsprozessorfamilien x86 und ARM dar. Die x86-Prozessorfamilie, die ihren Ursprung im PC-/Workstation-Markt hat, wird aktuell intensiv durch Systeme für stark kostensensitive Applikationen ergänzt. Diese werden jüngst gezielt auch für den Bereich des mobilen, energiebewussten eingebetteten Computings fortentwickelt. Es finden sich zunehmend Prozessorsysteme, welche bereits on-Chip Grafikeinheiten integrieren oder im Chipsatz durch solche sehr kostengünstig erweitert werden. Sind diese mit frei programmierbaren Prozessorelementen (GPGPU-Fähigkeit) ausgestattet, können sie ebenfalls die Rolle eines weiteren general-purpose Prozessors übernehmen. ARM-basierte Lösungen sind Vertreter der klassischen Domäne eingebetteter Systeme. Typischerweise sind ARM-Exemplare aus dem Leistungssegment in multimedia-orientierten Smartphones und Tablets anzutreffen. Für die Applikationsprozessorklasse ARM sind diverse Integrationsvariationen verfügbar. So finden sich System-on-Chips (SoCs) am Markt, bei denen im Sinne des asymmetrischen Multicore-Computings dem ARM-Kern auf dem Halbleitersubstrat ein klassischer DSP zur Seite gestellt ist. Dieser ist dann speziell zur Behandlung von Performance- und Echtzeitkritischen Signalverarbeitungsalgorithmen vorgesehen. Der im x86-Segment bereits seit längerem etablierte Trend zur Leistungsskalierung durch symmetrische Multicore-CPU's wurde durch die Architekturrevision Cortex A9 auch für ARM-Systeme eröffnet. Ebenfalls sind Grafikprozessoren direkt im SoC kombiniert verfügbar.

3.2 Analyse der Mikroarchitekturen

Die Mikroarchitekturen der möglichen nutzbaren Systemkonzepte sollen im Rahmen dieses Abschnittes eingeführt und hinsichtlich ihrer internen Struktur betrachtet werden. Das Augenmerk liegt dabei auf relevanten Verarbeitungseinheiten für die Arithmetiknutzung.

3.2.1 Intel Bonell (Intel Atom)

Die Familie der Intel Atom x86-Prozessoren zielt auf den Bereich der low-end PC-Produkte, primär für kostengünstigste portable PCs mit sehr niedrigem Leistungsverbrauch. Die Prozessorfamilie erfährt so zunehmenden Einsatz auch in der Domäne der eingebetteten Systeme¹. Die zugrunde liegende Mikroarchitektur wird vom Hersteller Bonell-Architektur genannt.

Generelle Eigenschaften

Details über die Interna des Bonell-Prozessorkerns finden sich in [50] [51]. Er stellt eine vergleichsweise simple x86-Architektur dar. Abbildung 3.2 zeigt schematisch den Aufbau des Prozessorkerns. Das Design ist im Gegensatz zu anderen aktuellen x86 Mikroarchitekturen nicht nach der Designrichtlinie Performance, sondern nach den Merkmalen kostengünstiger Herstellung, also geringer Chipfläche, und geringer elektrischer Leistungsaufnahme ausgelegt. Die Prozessorphipeline besitzt 16 Stufen. Die Architektur ist zweifach superskalar, sodass ein Prozessorkern oft zwei unabhängige Opcodes simultan ausführen kann.

Der Prozessorkern unterstützt die quasi-simultane Ausführung zweier Programmthreads durch hardwaregestützte Kontextwechsel². Hierfür existieren im Prozessorkern separat zwei vollständige Registerbänke, entsprechend auch zwei Instruktionszeiger. Es werden abwechselnd Instruktionen aus dem Programmcode beider Threads in die gemeinsame Ausführungspipeline des Prozessors erteilt. So können effizient Pipeline Stalls und Speicherzugriffslatenzen verdeckt werden: Die Tatsache, dass die Instruktionen aus zwei verschiedenen Programmen oder Programmteilen stammen, garantiert mit hoher Sicherheit Datenuabhängigkeit zwischen den Instruktionen (ILP). Auf diese Weise kann im Umfeld eines Multitaskingbetriebssystems situationsabhängig die Auslastung der Pipeline und somit die systemweite Performance der CPU verbessert werden.

Befehlssatz

Die Mikroarchitektur implementiert die unter dem Namen IA-32 bekannte 32 bit CISC x86-Befehlssatzarchitektur von Intel³. Der Prozessorkern unterstützt des Weiteren die Befehlssatzerweiterung Multimedia Extensions (MMX) für 64 bit SIMD-Operationen und deren direkten Nachfolger Streaming SIMD Extensions (SSE) für 128 bit Vektoren. Letztere Befehlssatzerweiterung wird unterstützt einschließlich der Revisionen SSE3 und SSSE3.

¹Sie übernimmt damit faktisch das Marktsegment der x86-CPUs, welches einst durch die Produkte VIA C3/C7 oder Transmeta Crusoe/Efficeon besetzt war.

²Dies ist bekannt unter dem Markennamen Intel Hyperthreading.

³Prinzipiell ist die Bonell-Mikroarchitektur auch für die Ausführung von Befehlen nach der 64 bit Erweiterung x86-64 geeignet, jedoch ist dies in low-end CPUs der ersten Generation vom Hersteller deaktiviert.

3 Verfügbare Mikroarchitekturen für eingebettete Infotainmentsysteme

SSE führt zusätzliche SIMD-Rechenwerke im Kern ein, diese werden zugleich zur Bedienung der Aufgaben des traditionellen x87-Fließkomma-Coprozessors (engl. Floating Point Unit, FPU) genutzt. SSE ermöglicht Vektoroperationen auf acht jeweils 128 bit breiten Registern. Diese können, ad-hoc je Befehlsopcode, in folgenden SIMD-Konfigurationen für parallele Berechnungen verwendet werden:

- 16x 8bit Festkommaarithmetik.
- 8x 16bit Festkommaarithmetik.
- 4x 32bit Fest- oder Fließkommaarithmetik (“Single Precision”).
- 2x 64bit Fest- oder Fließkommaarithmetik (“Double Precision”).

Somit lässt sich auf einer x86-Singlecore-CPU im Programmfluss eine Nebenläufigkeit beziehungsweise Parallelität von bis zum 16-fachen bei Verarbeitung von 8 bit Festkommawerten erzielen. Zusätzlich können im Idealfall zwei unabhängige dieser 128 bit SSE-Operationen gleichzeitig ausgeführt werden, da für das SSE-Rechenwerk genauso wie für das der gewöhnlichen Befehle eine zweifach superskalare Auslegung vorhanden ist. Die meisten SSE-Operationen werden innerhalb eines Taktzyklus vollständig ausgeführt. Spezialisierte Befehle zur Unterstützung der Domäne der digitalen Signalverarbeitung, beispielsweise gesättigte Arithmetikoperationen, sind vorhanden. Anders als historische x86-Prozessoren machen aktuelle Modelle dieser general-purpose CPUs so im Sinne eines hybriden Ansatzes Architekturaneihen bei DSPs und erscheinen deshalb auch gut für Signalverarbeitungsaufgaben geeignet. Zur Nutzung der SIMD-Fähigkeiten der CPU muss der ausgeführte Programmcode entsprechend entwickelt worden sein.

Speicherzugriffssystem

Im sogenannten Memory Execution Cluster ist sämtliche Logikfunktionalität zum Hauptspeicherzugriff angesiedelt. Die Mikroarchitektur beinhaltet hierbei auch zwei Einheiten zur selbständigen, von der Hauptrecheneinheit unabhängigen Adressberechnung indirekt indizierter Speicherzugriffe (engl. Address Generation Unit, AGU). Pro implementiertem Prozessorkern stehen die folgenden transparenten Cachesysteme zur Verfügung [50]:

- Level 1 Instruktions-Cache, 32 kB, 8-fach assoziativ.
- Level 1 Daten-Cache, 24 kB, 6-fach assoziativ.
- gemeinsamer Level 2 Cache, 512 kB, 8-fach assoziativ.

Die Cacheline-Größe in Byte beträgt stets 64 B. Für effizienten Speicherzugriff muss sich der Datenfluss der Anwendung folglich an 64 B-Grenzen orientieren. Der Datencache unterstützt sowohl spekulatives Prefetching von Daten als auch write-behind/store-forward Techniken. Die Caches einzelner Kerne stellen vollständig getrennte Systeme dar.

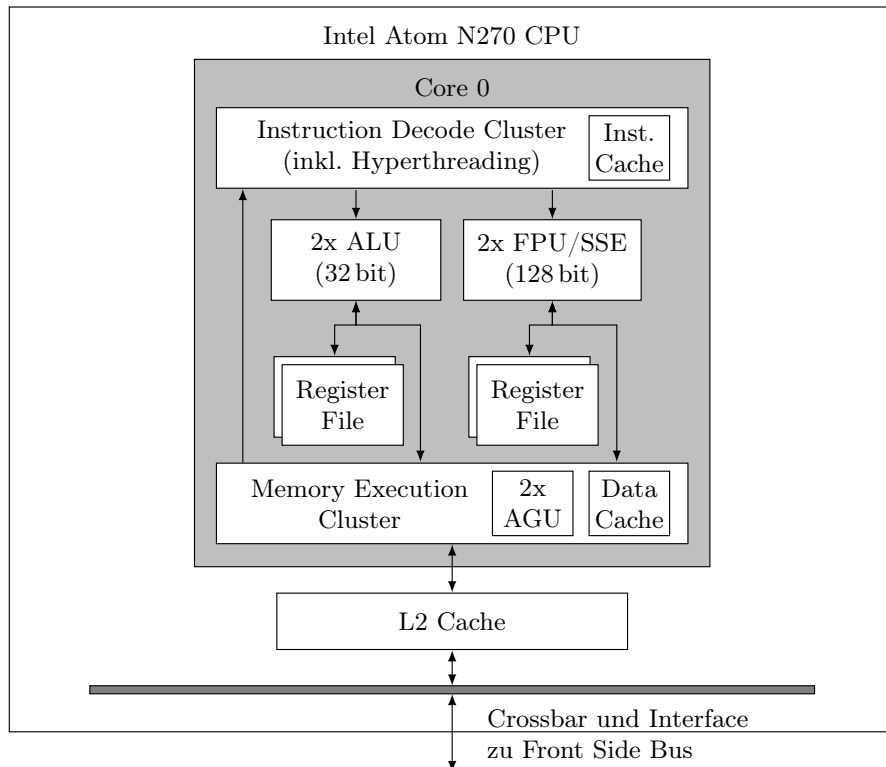


Abbildung 3.2: Aufbau der Intel Bonell-Mikroarchitektur am Beispiel einer Singlecore-CPU Intel Atom N270.

3.2.2 AMD Bobcat (AMD Fusion)

Die AMD Bobcat-Mikroarchitektur für low-end x86-Systeme ist als direkte Konkurrenz zur Intel Atom/Bonell-Mikroarchitektur zu sehen. Sie ist sehr ähnlich aufgebaut. Details finden sich in [53], [54] und [51].

Generelle Eigenschaften

Die Bobcat-Mikroarchitektur implementiert den Intel IA-32 Befehlssatz und unterstützt zusätzlich x86-64 Instruktionen. SIMD-Befehlssatzerweiterungen bis einschließlich SSE4a sind vorhanden. Die Prozessorpipeline ist 13-stufig ausgeführt. Der superskalare CPU-Kern besitzt dual-issue Fähigkeit mit zwei Festkommaarithmetik-ALUs. Die kombinierte FPU/SSE-Einheit ist ebenfalls zweifach superskalar aufgebaut, besitzt aber anders als bei der Intel Bonell-Architektur nur jeweils 64 bit breite Datenpfade. 128 bit FPU/SSE-Operationen werden deshalb im Befehlsdecoder zur Ausführung in zwei getrennte 64 bit Teiloperationen decodiert, entsprechend verhält sich der theoretische Maximaldurchsatz der arithmetischen Operationen.

Zur Sicherstellung einer hohen Auslastung der Ausführungseinheiten verfolgt der AMD Bobcat ein anderes Konzept als der Intel Atom: Pipeline Stalls und Speicherzugriffslatenzen werden verdeckt, indem Instruktionen ohne Datenabhängigkeit aus der Reihenfolge des sequentiellen Programmablaufs vorgezogen werden. Hierzu arbeitet ein dynamischer

3 Verfügbare Mikroarchitekturen für eingebettete Infotainmentsysteme

Scheduler auf einen 56 Mikroops, das heißt decodierte CISC-Befehls(teil)worte, fassenden Instruktionpuffer, dem Reorder Buffer. Für das Vorhandensein dieser Einrichtungen zur Out-of-Order-Execution des Programmcodes sind im Gegenzug und im Unterschied zur Intel Bonnell-Mikroarchitektur keine Mechanismen zum hardwareunterstützten Thread-Kontextwechsel vorgesehen.

Jeder Bobcat-Kern verfügt nach [55] über folgende Cachesysteme:

- Level 1 Instruktions-Cache, 32 kB, 2-fach assoziativ.
- Level 1 Daten-Cache, 32 kB, 8-fach assoziativ.
- Gemeinsamer Level 2 Cache, 512 kB, 16-fach assoziativ.

Im Falle einer Multicore-CPU existieren diese Speicher in unabhängiger Weise pro Kern. Die Cacheline-Größe in Byte beträgt wie beim Intel Atom stets 64 B. Jedoch ist das Raster für bestmöglichen effizienten Speicherzugriff, an dem sich der Datenfluss der Anwendung orientieren muss, nicht an 64 B Grenzen, sondern an engeren 16 B Grenzen orientiert.

3.2.3 ARM Cortex-A

Die ARM-Prozessorfamilie stellt aktuell eine dominante Prozessorarchitektur im Bereich der eingebetteten Systeme dar. Das Produktportfolio leitet sich im Sinne eines Architekturbaustens ab, aus dem je nach Anwendung skalierbar verschiedene Prozessorvarianten zur Realisierung ausgeleitet werden können. Genereller Grundbaustein der Familie ist ein einfach gehaltenes 32 bit RISC-Design mit Fokus auf niedriger elektrischer Leistungsaufnahme.

ARMv7A-Befehlssatzarchitektur

Die aktuelle Revision der Befehlssatzarchitektur ist spezifiziert in [56]. Sie arbeitet auf 16 Registern mit jeweils 32 bit. Hiervon sind 13 Register frei verwendbar, die verbleibenden drei Register übernehmen die Spezialfunktionen des Programm- und des Stackzeigers sowie des Linkregisters, welches zur schnellen Ausführung von Subroutinen genutzt wird. Sämtliche Befehlsörter besitzen prinzipiell die feste Länge⁴ von 32 bit. Seit Revision 6 der Befehlssatzarchitektur wurde diese in Subsets, sogenannte Profile, unterteilt. Für die Revision 7 existieren folgende drei Profile:

- Microcontroller Profile (M-Profile): Basisprofil für typische low-performance Anwendungen aus dem Mikrocontroller-Bereich⁵.
- Real-time Profile (R-Profile): Erweiterung der Systemeigenschaften um Protected Memory, Interrupthandling niedriger Latenz.

⁴Um die Dichte des Programmcodes in Einsatzumgebungen mit geringem Speicher erhöhen zu können kann optional die Implementierung des Thumb-Betriebsmodus in den Prozessor integriert werden. So werden auch 16 bit Befehlsörter unterstützt, welche durch Codetransformation vor dem Befehlsdecoder zur Ausführungszeit für den RISC-Prozessorkern in nativen 32 bit ARM-Code überführt werden.

⁵Zur Gegenüberstellung zur behandelten A-Architektur soll als eine Ausformung des M-Profiles auf den populären ARM Cortex M3-Kern als Beispiel verwiesen werden. Diese CPU besitzt nur eine 3-stufige Pipeline, keinerlei Caches und implementiert nur ein reduziertes Befehlssubset.

- Application Profile (A-Profil): Typischerweise für high-end Embedded-Anwendungen. Unterstützung für geschützte/virtuelle Speicherbereiche (Memory Management Unit, MMU) und Multitasking-Betriebssysteme, hardwaregestützte Interpretierung von Java-Bytecode (Jazelle-Engine).

Als optional implementierbare Fließkommaeinheit existiert die VFP-Teilarchitektur (aktuell in der Revision VFPv3). Für das im Weiteren betrachtete Applikationsprozessor-Profil ARMv7A wurde zusätzlich das Konzept der Advanced SIMD Extensions als Option des Prozessorkerns entworfen, die Implementierung eines solchen Coprozessors ist bekannt unter dem Namen ARM NEON. Dieser Advanced SIMD Media Processing Engine (MPE) stehen 16 Vektorregister mit 128 bit Weite zur Verfügung. Diese können alternativ auch als 32 individuelle 64 bit Register adressiert werden. Sind im Kern gleichzeitig VFP-Fließkommaeinheit und Advanced SIMD Extensions implementiert, so stellt der Registersatz zwischen Fließkommaeinheit wie auch SIMD-Einheit eine geteilte Ressource dar⁶. Das 128 bit Rechenwerk der MPE-Einheit kann je nach Anforderung an Präzision und Parallelität in den folgen SIMD-Modi betrieben werden:

- 2×64 bit, 4×32 bit, 8×16 bit oder 16×8 bit Festkommaarithmetik.
- 4×32 bit oder teilweise 8×16 bit Fließkommaarithmetik.
- 16×8 bit oder 8×16 bit Galois-Feld-Arithmetik.

Der SIMD-Modus der Verarbeitungseinheit wird dabei, wie bei der SSE-Befehlssatzarchitektur der x86-Systeme, je Instruktion ad-hoc festgelegt. Die Load/Store-Einheit unterstützt zusätzlich befehlsinhärentes (De-)Interleaving von Arrays. So können ohne Zusatztaktzyklen direkt beim Lesezugriff beispielsweise im Speicher verschachtelte I- und Q-Komponenten eines komplexwertigen Signals (oder die drei Farbkomponenten eines Bildes) in verschiedene SIMD-Vektorregister entflochten werden. Ebenfalls ist Datentyppropagation, das heißt Typwandlung von Daten zu größeren oder kleineren Wortbreiten, bei vielen Operationen inhärent vor oder nach der eigentlichen Arithmetik ohne weiteren Operationszyklus möglich.

Mikroarchitekturvariante Cortex A8

Der Cortex A8-Kern [57] ist ein Logikdesign, welches die Befehlssatzarchitektur ARMv7A implementiert. Es ist ausgelegt für Taktfrequenzen im Bereich $f_{\text{Clk}}=500 \dots 1000$ MHz. ARM gibt an, dass für eine vollsynthetisierte Implementierungen ohne weitere Optimierungen in einem typischen 65 nm-Halbleiterprozess noch 750 MHz Taktung erreicht werden können. Der Kern besteht nach offizieller Dokumentation aus einer 13-stufigen Pipeline mit superskalaren Ausführungseinheiten⁷. Der Befehlsdecoder verfügt über dual-issue Fähigkeit, kann also zwei Opcodes je Zyklus superskalar auf die vier vorhandenen Ausführungseinheiten verteilen. Zwei der vier Ausführungseinheiten stellen vollsymmetrisch aufgebaute Rechenwerke für skalare Festkommaarithmetik dar, somit besitzt der Cortex

⁶Ähnlich der mit FPU geteilten Registernutzung der x86-Technologie MMX, welche den Vorgänger von SSE darstellt.

⁷In mancher Literatur findet sich die Angabe über 14 Pipeline-Stufen, wenn die Stufe F0 zur Adressgenerierung des Instruction Fetch mitgezählt wird.

3 Verfügbare Mikroarchitekturen für eingebettete Infotainmentsysteme

A8 die Möglichkeit, mit hoher Sicherheit stets zwei gewöhnliche Befehle pro Taktzyklus parallel abzuarbeiten. Des Weiteren ist nebenläufig ein Hardwaremultiplizierer vorhanden sowie die Load/Store-Transfereinheit für Zugriff auf den Speicher. Der Prozessorkern unterstützt jeweils für Daten und Instruktionen Level 1-Caches im Größenbereich 16...32 kB mit 4-facher Assoziativität. Der weiterhin je nach Chipvariante anbindbare Level 2-Cache kann im Bereich 0...1024 kB variieren, die Cacheline-Größe beträgt dabei 64 B.

Der Cortex A8 beinhaltet den NEON Media Coprozessor zur Bereitstellung der Advanced SIMD Extensions und der Fließkommaeinheit. Die Befehle für den Coprozessor werden nach der siebten Stufe am Befehlsdecoder aus der Pipeline des Hauptprozessors ausgeleitet und in die 10-stufige Nebenpipeline des NEON-Subsystems eingeleitet. Der Tausch von Daten zwischen den Registern des Prozessors-Hauptteils und den Registern des NEON-Coprozessors ist zwar prinzipiell vorgesehen, aber nur sehr ineffizient möglich. Dies folgt aus der Gegebenheit, dass ein solcher Transfer als eine Art Synchronisationschranke der beiden ansonsten unabhängigen Rechenwerkssysteme wirkt: Zur Sicherstellung der Datenintegrität werden vor dem Verschieben von Registerinhalten automatisch die beiden Instruktionspipelines des Prozessors geleert. Sie müssen nach dem Datentausch latenzintensiv neu mit Maschinenbefehlen befüllt werden. Es empfiehlt sich also, im Applikationsdesign den Datenfluss direkt zwischen Speicher und der jeweiligen Registerbank der benutzten Verarbeitungseinheit anzulegen. Alle Rechenwerke im NEON-Coprozessor besitzen 64 bit Breite, somit müssen 128 bit Vektoren intern mittels zweier Teiloperationen behandelt werden. Jedoch ist das NEON-Subsystem ebenso wie die Hauptpipeline zweifach superskalar ausgelegt. Zur optimal performanten Nutzung des Speicherinterfaces sind die Vektorzugriffe auf den Hauptspeicher entsprechend 128 bit Wortgrenzen orientiert anzulegen.

Mikroarchitekturvariante Cortex A9

Das Design Cortex A9 [58] implementiert ebenfalls die ARMv7A-Befehlssatzarchitektur. Es ist dem Vorgängerentwurf Cortex A8 im Wesentlichen sehr ähnlich. Die neue Mikroarchitekturrevision des Kerns fügt dem Instruktionsscheduler Fähigkeiten zur Out-of-Order- und Speculative-Execution hinzu. Die maximale Größe der L1-Caches wurde auf 64 kB verdoppelt. Der Cortex A9-Kern ist des Weiteren für symmetrische Multicore-SoC Systeme vorbereitet und kann mit bis zu vier gleichartigen Kernen kooperieren. Somit ist hinsichtlich der Rechenleistung im System eine gute Leistungsskalierbarkeit des Entwurfs gesichert. Ein für den vorgestellten Anwendungsfall wichtiger Unterschied ergibt sich aus der Tatsache, dass die uneingeschränkte dual-issue Fähigkeit der SIMD-Einheit im NEON-Coprozessor des Cortex A9 gegenüber dem Cortex A8 beschnitten wurde. Dies kann bei SIMD-parallelierten Berechnungsroutinen potentiell in Performanceverschlechterungen im Vergleich zum Cortex A8 resultieren. Dafür sind die Logikpfade des Cortex A9-Designs für Betrieb bei höheren Taktfrequenzen (laut Hersteller bei typischem 40 nm-Halbleiterprozess bis zu $f_{\text{clk}}=2.0$ GHz) als der Cortex A8 ausgelegt.

3.2.4 Texas Instruments TMS320C64x

Der C64x-Prozessorkern [60] gehört zur C6000-Serie der von Texas Instruments unter dem Markennamen TMS320 vertriebenen Familie von digitalen Signalprozessoren. Der C64x, initial explizit auch für die Nutzung in drahtloser Infrastruktur entworfen und entsprechend weiterentwickelt [61], findet sich durch die mit dem technischen Fortschritt verknüpfte Preisentwicklung seit mehreren Jahren zunehmend aber auch in mittelpreisigen Anwendungen der Consumer Electronic. Neben dem hier betrachteten Typ C64x existieren die Varianten C67x mit vom Grundprinzip ähnlicher Architektur, welche jedoch in den zwei Datenpfaden Fließkomma- anstelle von Festkommarechenwerken besitzt, sowie die Variante C674x, deren Rechenwerke sowohl Festkomma- als auch Fließkommaberechnungen durchführen können.

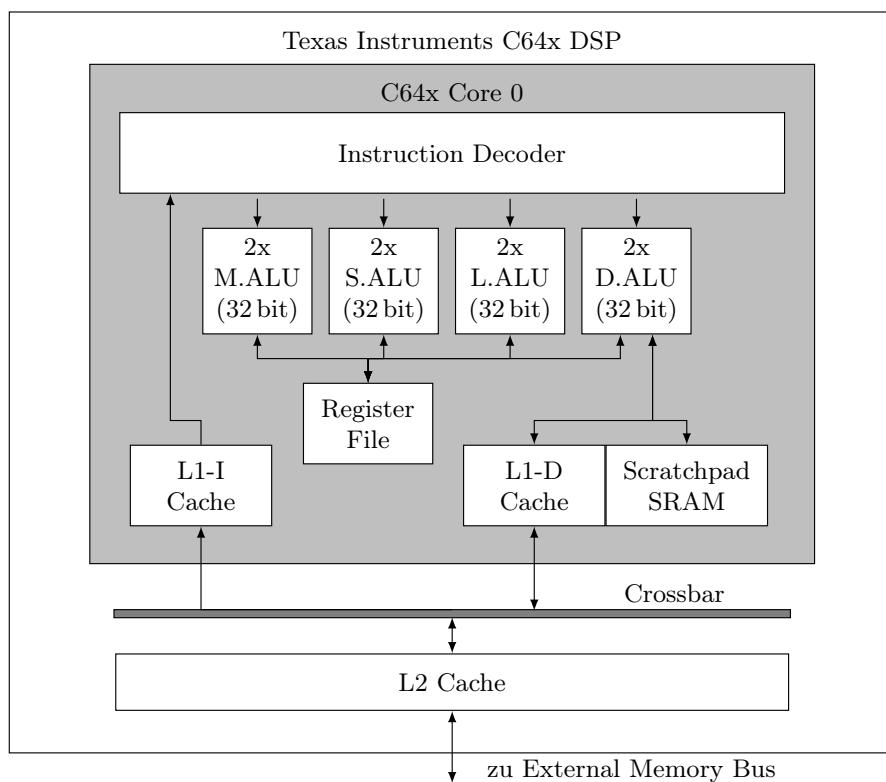


Abbildung 3.3: Mikroarchitektur des DSP-Kerns Texas Instruments C64x.

Rechenwerke

Der Prozessorkern besitzt zwei identische, disjunkte Datenpfade und Registersätze. Neben speziellen Kontrollregistern stehen in Registerbank A und B jeweils 32 Arbeitsregister zu 32 bit zur Verfügung, wovon manche an bestimmte Sonderfunktionen gebunden sind. Jeder Datenpfad 1 und 2 verfügt über vier Ausführungseinheiten mit unterschiedlichen Fähigkeiten:

- *.L1, .L2*: Arithmetisch/logische Operationen, Vergleiche.

3 Verfügbare Mikroarchitekturen für eingebettete Infotainmentsysteme

- *.S1, .S2*: Arithmetisch/logische Operationen (auch gesättigt), Bitmanipulationen.
- *.M1, .M2*: Multiplikationen, Galois-Feld-Operationen.
- *.D1, .D2*: Lade- und Speicheroperationen, Adressgeneration, Additionen mit Sonderfunktion für zyklische Operationen.

Je Taktzyklus greifen die vier Funktionseinheiten mit Endung 1 in der Regel stets auf Bank A zu, diejenigen mit Endung 2 auf Bank B. Zwei Datensonderpfade existieren, um für zwei Funktionseinheiten zusätzlich kreuzweise Registerzugriffe zu ermöglichen.

Durch einen einzelnen C64x-Kern können so bis zu acht datenunabhängige Operationen verschiedener Art gemäß MIMD zeitgleich bearbeitet werden. Dabei kann weiterhin jedes Rechenwerk nach SIMD-Methodik aufgeteilt werden, um pro Zyklus alternativ zwei 16 bit oder vier 8 bit Festkommawerte je 32 bit Register parallel zu verarbeiten. Maximal können also in einem einzelnen C64x-Kern potentiell 32 Datenpunkte parallel bearbeitet werden. Speziell für Anwendungen der Signalverarbeitung können Berechnungen des Weiteren neben der gewöhnlichen Auflösung von 32 bit auch mit einer erhöhten Breite von 40 bit durchgeführt werden. Für diese Art Arithmetikoperationen sind zwei 32 bit Register zu koppeln.

Befehlssatz

Bei klassischen DSPs mit VLIW-Architektur werden stets die Operationen sämtlicher Funktionseinheiten durch ein gemeinsames Codewort bestimmt. Um jedoch die Codegröße kompakt zu halten, speziell im Falle von sequentieller Ausführung, ist im C64x eine Codemorphing-Logik dem Instruktionsdecoder vorgeschaltet: Noch unabhängig von eventueller paralleler Ausführung werden jeweils acht zusammengefasste Befehle als sogenanntes Fetch Packet der Gesamtbreite 256 bit aus dem Programmspeicher geholt. Sie werden anschließend zu Execute Packets umgeformt, indem die darin enthaltenen Befehle auf die zukünftig passenden freien parallelen Ausführungseinheiten verteilt werden. Ein spezielles Steuerflag zu jedem Befehl, welches als p-Bit bezeichnet wird, gibt an, ob der nächste Befehl parallel mit den vorherigen Befehlen ausgeführt werden darf. Falls dies nicht der Fall ist oder bereits alle passenden Ausführungseinheiten besetzt sind, wird ein neues Execute Packet im nächsten Taktzyklus begonnen. Eventuell unbesetzte Rechenwerke pausieren. Die Steuerung des p-Bits erfolgt vorab bei der Codegenerierung durch das Compiler-Framework. Die generierte Befehlsfolge wird bei der Codeerstellung hinsichtlich des Vorhandenseins von Instruction Level Parallelism im Code gezielt untersucht und optimiert.

Wie für DSP-Prozessoren üblich unterstützt der Instruktionsscheduler die hardwareunterstützte Ausführung von Schleifen durch repetitive Befehlsausführung mit Datenindexpropagierung (engl. Zero Overhead Loops).

Speichersubsystem

Der Speicherzugriff des DSPs erfolgt nach dem von modernen general-purpose CPUs gewohnten Modified Harvard-Modell, welches die zwei getrennten Speicherpfade des Prozessorkerns für Instruktionen und Daten über einen vereinheitlichten L2-Cache auf einen

gemeinsamen externen Arbeitsspeicher leitet. Die Chip-internen SRAMs auf L2-Ebene sind teilweise konfigurierbar und können so entweder als L2-Cache oder aber als schnelles Scratchpad-RAM benutzt werden. Je nach konkreter Implementierung des C64x variieren die jeweilig vorhandenen Cache-Größen.

3.2.5 NVidia G9x (Geforce 9)

Neben CPU und DSP sollen auch die Architekturen der Grafikprozessoren vergleichend betrachtet werden. Bei der G90-Serie handelt es sich um die zweite Generation vollständig programmierbarer Grafikchips des Herstellers NVidia.

Befehlssatz und Programmiermodell

NVidia dokumentiert den Befehlssatz der Parallel Thread Execution (PTX) lowlevel-Sprache [65]. Bei diesem generischen Pseudocode handelt es sich jedoch um eine Zwischendarstellung des NVidia CUDA- beziehungsweise OpenCL-Compilers, welche anscheinend nicht direkt mit der tatsächlichen durch den jeweiligen GPU-Chip ausgeführten Maschinsprache identisch ist. Direkte Rückschlüsse hieraus auf die Mikroarchitektur sind deshalb nur begrenzt möglich. Durch den Hersteller ist für GPGPU-Anwendungen ausschließlich die Programmierung in Hochsprache vorgesehen [66]. Hierbei wird vollkommen von der zugrunde liegenden Mikroarchitektur abstrahiert und ein Vektorprogrammiermodell eingeführt, bei dem als zentrales Paradigma ein jedes Vektorelement als virtueller Programmthread⁸ betrachtet wird. Die Abbildung der Workload auf die Konzepte des symmetrischen Multiprocessings und der SIMD-Datenverarbeitung kann durch den Programmierer über die Wahl der sogenannten Gridsize beeinflusst werden, wird aber letztendlich durch das Treiber-/Compiler-Framework durchgeführt.

Rechenwerke und Streaming-Multiprozessor

Die GPU besteht im Sinne eines symmetrischen Multiprozessors je nach Modellvariante aus mehreren, gegebenenfalls aber auch nur aus einem sogenannten Streaming Multiprocessor (SM). Der Aufbau ist in Abbildung 3.4 dargestellt. Jeder SM besitzt die folgenden Rechenwerke:

- Acht via SIMD-Prinzip gekoppelte Rechenwerke, welche pro Zyklus einen Vektor von acht 32 bit Fließkommawerten parallel bearbeiten können. So implementiert der SM eine Gesamtvektorbreite von 256 Bit. Die Berechnungen von Festkommaoperationen werden in den Rechenwerken ebenfalls unterstützt, jedoch kann dies im Vergleich nur langsamer durchgeführt werden [64]. Die Nomenklatur von NVidia bezeichnet jeweils eines dieser acht Rechenwerke als Streaming Processor (SP) oder als "Prozessorkern"⁹.

⁸NVidia bezeichnet jeden Index eines SIMD-Vektors als Thread. Dies weicht vom üblichen allgemeinen Verständnis des Threads in der Informatik ab, bei welcher Threads unabhängig voneinander disjunkte Programmflüsse abarbeiten.

⁹Es ist zu beachten, dass ein SP jedoch entgegen dieser Terminologienutzung nur ein isoliertes SIMD-Teilrechenwerk darstellt – ein SP besitzt nicht den Charakter eines eigenständigen Prozessors.

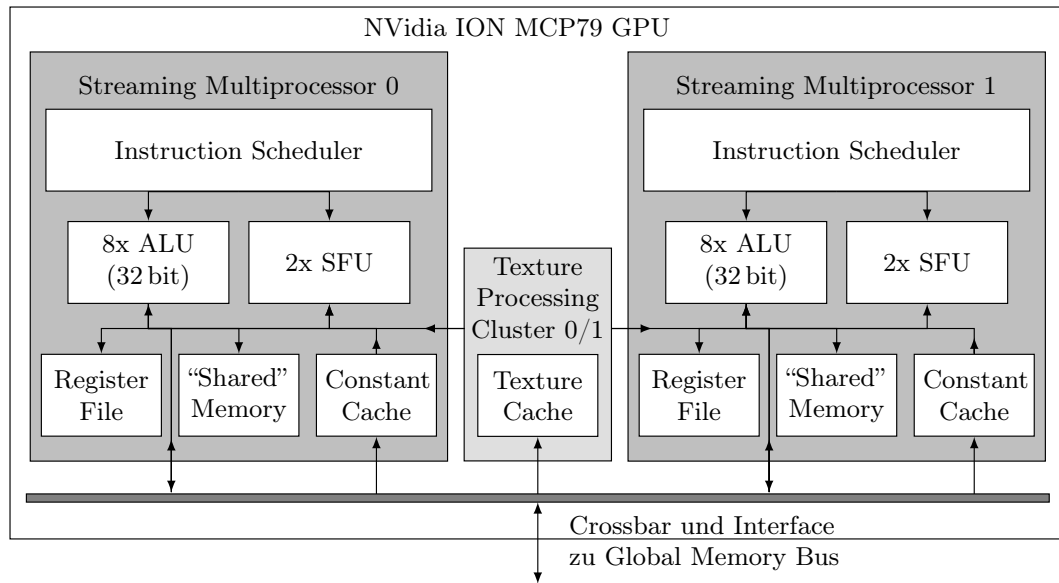


Abbildung 3.4: Mikroarchitektur der G9x-GPUs am Beispiel der NVidia MCP79 Ion.

- Zusätzlich zwei Special Function Units (SFU) für hardwareunterstützte Berechnung transzendentaler Funktionen, beispielsweise trigonometrische Funktionen oder Exponentialfunktionen.

Jeder SM stellt einen vollwertigen Prozessor dar, er besitzt einen eigenen Instruktionenzeiger und arbeitet selbständig einen Ablaufplan ab. Programmflussdivergenzen von GPU-Threads innerhalb eines (Teil-)Vektors löst das SIMD-System gemäß dem in Prozessoren allgemein bekannten Prinzip der Bedingten Ausführung (engl. Conditional Execution) auf: Rechenwerke einzelner Elemente des SIMD-Vektors können aufgrund in vorhergehenden Operationen, typischerweise Vergleichsoperationen, gesetzten Predicate-Flags gezielt für folgende Instruktionen deaktiviert werden. Der Befehlsdecoder des SM besitzt die Fähigkeit, dasselbe Instruktionswort repetitiv zur Ausführung zuzuweisen, wobei Datenspeicher- und Registerindizes für Quell- und Zieloperanden automatisch entsprechend propagiert werden. Dies entspricht im Prinzip den bekannten Mechanismen des Zero Overhead Loopings, das von DSP-Prozessoren bekannt ist. So kann im Zusammenspiel mit dem SDK-Framework dem CUDA-Programmierer eine virtuell vergrößerte SIMD-Breite angeboten und adaptiv statt 8 bis zu 512 breite Datenvektoren pro SM bearbeitet werden. NVidia bezeichnet dies als Single Instruction Multiple Threads (SIMT)-Technologie. Es ist ratsam, diese Möglichkeit zur virtuellen Vektordimensionsvergrößerung durch repetitive Ausführung stets zu nutzen, denn mutmaßlich existiert ein Taktratenunterschied zwischen den schnelleren Ausführungseinheiten und langsameren Subsystemen des Befehlsdecoders¹⁰: Es muss jedes Instruktionswort mehrmals hintereinander ausgeführt werden, um den maximalen Instruktionsdurchsatz der Ausführungseinheiten zu erreichen. Auf dem G9x wurde für performante Ausführung eine nötige Mindestanzahl virtueller Threads $n \geq 64 = 8 \cdot 8$ beobachtet, entsprechend mindestens acht Repetitionszyklen der achtfach SIMD-Einheit. Aus den

¹⁰Dies führt unter anderem zu dem Programmierkonzept der sogenannten Warps der NVidia GPUs.

weiterhin beobachtbaren Latenzen von 24 Taktzyklen bei Register-zu-Register-Zugriffen mit Datenabhängigkeit lässt sich eine Vermutung für die Pipelinetiefe des GPU-Prozessors ableiten – dies gibt auch eine mögliche Erklärung, warum mancher Code die maximale Performance erst bei einer Mindestanzahl von $n \geq 192 = 8 \cdot 24$ Threads erreicht: Hier scheint der Quellcode zu einer zwischen drei benachbarten Instruktions-OpCodes bestehenden Datenabhängigkeit der Operanden zu führen, welche durch den CUDA-Compiler mittels Reorganisation des resultierenden Maschinencodes nicht entflochten werden konnte.

Speicherarchitektur

Zur Bildung von Datenvektoren steht innerhalb des SM eine üppige Bank von 8192 Registern zu je 32 bit (in Summe 32 kB Speicher) zur Verfügung. Aus der SIMD Architektur ergeben sich, wie auch von den SIMD-Einheiten der CPUs bekannt, bei Transfers zum externen RAM-Speicher (bezeichnet bei der GPU als Global Memory) für die Vektoren festgelegte Zugriffsmuster. Diese sind für effizienten Datentransfer bei der Programmierung unbedingt zu beachten (siehe [66], und [67]). Weiterhin steht pro SM intern ein schnelles Scratchpad-RAM der Größe 16 kB bereit. Da hierfür vergleichsweise flexible Zugriffsarten im Prozessor bereitstehen, können mit seiner Hilfe Permutationen des SIMD-Vektors performant vorgenommen werden. Dies entspricht Datenaustausch/Interprozesskommunikation zwischen den virtuellen GPU-Threads innerhalb eines SM, woraus die Bezeichnung des Herstellers “Shared Memory” mutmaßlich resultiert. Kommunikation zwischen unterschiedlichen SM ist über Datenaustausch im externen DRAM umzusetzen. Da dieser Zugriff offensichtlich mit hohen Latenzen verbunden ist, ist die Verteilung einer Workload über mehrere SM nur nutzbringend, falls der aufgespaltene Algorithmus streng datenparallel beziehungsweise nur auf wenig gegenseitigen Informationsaustausch angewiesen ist. Für Lesezugriffe auf den externen DRAM-Speicher besteht die Wahlmöglichkeit, einen Cache (Constant Cache) zu nutzen, welcher allerdings keine Kohärenz zu ins DRAM zurückgespeicherten Daten bietet. Er kann also nur für unveränderte Datensätze genutzt werden. Zwei SM teilen sich eine hardwarebeschleunigt interpolationsfähige Texturdateneinheit, über welche des Weiteren dateninterpolierende Zugriffe auf den externen DRAM erfolgen können. Die Einheit verfügt ebenfalls über einen DRAM-Lesecache (Texture Cache), dessen Cachezeilen-Organisation jedoch gemäß Bildverarbeitungskonzepten speziell für 2D-Datenlokalität ausgelegt ist.

3.2.6 AMD Radeon R800 (AMD Radeon HD)

Die Architektur mit der offiziellen Bezeichnung R800, auch als Codename Evergreen geführt, stellt den Nachfolger der ersten beiden vollprogrammierbaren GPU-Architekturen R600 und R700 von AMD dar. Sie führt die VLIW5 Architektur genannte Befehlssatzarchitektur der R600-Prozessorfamilie fort¹¹. Die Architekturbeschreibung [68] trennt zwischen dem skalaren Kontrollflussprozessor und den eigentlichen Rechenwerkseinheiten. Letztere sind in der Hardwareimplementierung verschachtelt gemäß SIMD- als auch zugleich MIMD-Paradigma vorhanden. Abbildung 3.5 zeigt die R800-Architektur anhand des GPU-Produktes AMD HD6310.

¹¹In der R700-Familie wurde zwischenzeitlich eine VLIW4 genannte, hinsichtlich der Rechenwerkskomplexität reduzierte Architektur eingesetzt.

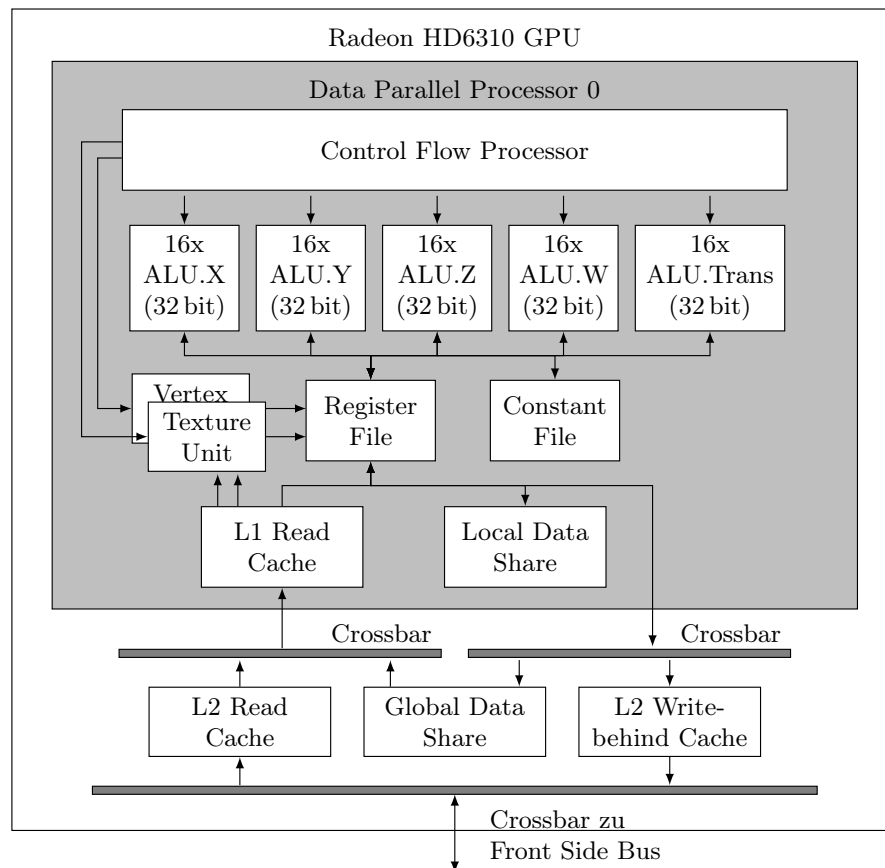


Abbildung 3.5: Mikroarchitektur R800 Evergreen der AMD Radeon HD6310 GPU.

Data Parallel Processor

Stets ein Pool von 16 Arithmetikgruppen wird parallel implementiert und durch einen gemeinsamen Instruktionsdecoder, realisiert durch den sogenannten Kontrollflussprozessor, nach dem SIMD-Paradigma bedient. In einer R800-GPU sind je nach Ausbaustufe ein oder mehrere Data Parallel Processors (DPPs) im Sinne eines symmetrischen Multicore-Verbundes kombiniert.

Arithmetikeinheit

Die Registerbank einer Arithmetikgruppe umfasst 127 general-purpose Register. Die vom Hersteller VLIW5 benannte Architektur stellt je Arithmetikgruppe gemäß MIMD-Konzept fünf nebenläufige Rechenwerke zur Verfügung.

- $ALU.[X, Y, Z, W]$ für gewöhnliche 32 bit Fließkommaoperationen.
- $ALU.Trans$ für gewöhnliche 32 bit Fließkommaoperationen als auch die Berechnung transzendentaler Funktionen.

Die Codierung der Befehle für jede einzelne ALU erfolgt über Worte zu je 64 bit, die Instruction Slots genannt werden. Ein kompletter Befehl (Instruction Group) für einen Zyklus

der Arithmetikeinheit besteht dabei aus mehreren Instruction Slots, welche auf die passenden fünf Rechenwerke verteilt werden. Es erfolgt also fünffache MIMD-Prozessierung. Jede der fünf ALUs erwartet pro Operation bis zu drei Operanden. Sämtliche Quelloperanden werden über ein Multiplexnetz an die Rechenwerke konfiguriert, das eine Vielzahl jedoch nicht frei wählbarer Datenpfade ermöglicht. Für Variablen als Eingangsoperanden sind drei Registerleseports vorgeschaltet, das bedeutet, dass in jeder Instruction Group auf maximal drei unabhängige Eingangsvariablen (Source Operands) zugegriffen werden kann. Folglich bestehen für die simultane Nutzung der fünf Rechenwerke funktional starke Einschränkungen. Hinsichtlich Konstanten sind vier Zugriffe auf ein mit bis zu 512 Festwerten konfiguriertes Constant File möglich (pro Programm definiert), alternativ können der Instruction Group im Befehlsstrom zwei oder vier 32 bit Literale folgen. Ähnlich der Befehlsdecodierung im C64x-DSP schließt ein LAST-Bit im Instruction Slot die Instruction Group ab und triggert den Ausführungszyklus der bis dato zugewiesenen Sequenz von Rechenwerksoperationen. Mitsamt bis zu zwei zusätzlichen Slots zur Definition von Literalen als Operanden kann ein komplettes Befehlswort eine Länge bis zu $(5 + 2) \cdot 64 \text{ bit} = 448 \text{ bit}$ erreichen. Das Array der 16 je DPP kombinierten MIMD-Arithmetikgruppen wird per SIMD-Paradigma vom Kontrollflussprozessor mit Instruktionen bedient. Rechnerisch ergeben sich damit in Summe 80 Rechenwerke pro DPP.

Kontrollflussprozessor

Der Kontrollflussprozessor im DPP besitzt alle Merkmale eines einfachen Mikroprozessors. Im Speziellen kann er Sprünge im Programmablauf durchführen und besitzt einen Stack für Subroutinen. Daneben bereitet der Kontrollflussprozessor auch Speichertransfers zwischen den Arbeitsregistern der Rechenwerke und dem externen RAM-Speicher vor (sogenannte Exports und Imports). Neben dieser kontrollflussorientierten Ausführung kann für bestimmte Codeabschnitte, sogenannte Clauses, in Operationsmodi geschaltet werden, in denen für eine festgelegte Anzahl folgender Befehlsörter diese direkt an ein CoprozessorSubsystem durchgeleitet werden. Speziell für grafische 3D-Anwendung parametrisieren die Vertex-Fetch- und Texture-Fetch-Clauses die zugehörigen Subsysteme der als Hardwareunterstützung vorhandenen Baugruppen. Der Modus der ALU-Clause beinhaltet den Kern eines jeden GPGPU-Berechnungsprogramms: Es werden die folgenden Befehle in das Hauptrechensystem der GPU, das Array aus Arithmetik-Rechenwerken des DPPs, weitergeleitet. Insofern diese Worte direkt in die Ausführungspipeline der Rechenwerke gespeist werden, müssen sie entsprechend der tatsächlichen Abarbeitung stets streng sequentiell aufeinanderfolgen. Sprungverzweigungen sind innerhalb einer Clause nicht möglich. Jedoch ist eine bedingte Befehlsausführung anhand von Predicate-Bits unterstützt. Nach Abarbeitung sämtlicher Befehle der Clause und vollständiger Leerung der ALU-Verarbeitungspipeline übernimmt wieder der Kontrollflussprozessor die weiteren Programminstruktionen und kann beispielsweise Wiederholungsschleifen oder Sprünge durchführen.

Speicherhierarchie

Jeder DPP besitzt 32 kB schnelles Scratchpad-RAM, das Local Data Share (LDS). Dieses ermöglicht analog zum Shared Memory der Nvidia G9x-Architektur mittels flexiblem Zugriff unter anderem das Permutieren der SIMD-Vektoren und somit den Datenaustausch

zwischen den DPP-internen SIMD-Rechenwerken. Des Weiteren steht im Speichersystem des DPP ein read-only Level 1-Cache zur Verfügung. Dieser Cache arbeitet auf einem allen DPPs gemeinsamen Level 2-Lesecache. Daneben ist auf der Level 2-Ebene ein gemeinsamer write-back Cache vorgesehen, in welchen alle Schreibzugriffe sämtlicher DPPs geleitet werden. Genauere Angaben über die implementierten Caches sind nicht bekannt. Zudem existiert ein internes Global Data Share-RAM, auf das alle DPPs Zugriff besitzen, so dass Interprozessorkommunikation gegenüber dem Umweg über den DRAM-Hauptspeicher beschleunigt durchgeführt werden kann.

3.2.7 Imagination Technologies SGX (PowerVR)

Der PowerVR SGX-Kern von Imagination Technologies Ltd., erstmalig erschienen im Jahr 2005, ist seitdem als Marktführer in eingebetteten Grafiksystemen aus dem Smartphone- und Tablet-Bereich vertreten. Das Produkt zielt anstelle höchster Verarbeitungsleistung auf energiesparenden Betrieb. Detaillierte Informationen des Herstellers über das proprietäre Produkt sind nicht in freier Form verfügbar.

Neben den bekannten GPU-Produkten von NVidia und AMD verfügt auch die PowerVR SGX-GPUs über einen Unified Shading-Verarbeitungskern. Diesem Universal Scalable Shader Engine (USSE) genannten Prozessorkern sind nach [69] applikationsspezifische Hardwaresubsysteme für Bildverarbeitung zur Seite gestellt. Diese sind die nicht näher beschriebenen Systeme Tiling Coprocessor, Pixel Coprocessor und Texturing Coprocessor. Die als Vertex Data Master, Pixel Data Master und General Purpose Data Master bezeichneten Systeme stellen vermutlich anwendungsspezifische Lesecaches dar [69]. Mit Chips der Reihe PowerVR SGXMP existieren Realisierungsvarianten, welche zur Gesamtleistungssteigerung bis zu 16 USSE-Kerne in einer symmetrischen Multicore-Anordnung skalieren können.

Unified Shading-Verarbeitungskern USSE

Es werden für die USSE-Verarbeitungskerne folgende Fakten in [70] und [71], Kapitel 5, publiziert: Die Shader der GPU sind vollprogrammierbar, so dass beliebige 2D- oder 3D-Grafikeffekte oder auch Videocodierungsverfahren durch den Entwickler umsetzbar seien. Die Grafikpipeline des PowerVR SGX ist als 64-fach beschrieben, wobei diese auf 16 Threads¹² segmentiert im Zeitmultiplex ausgeführt werden. Dies korrespondiert mit der Angabe von je vier Instruktionen pro Zyklus nach MIMD-/VLIW-Verfahren auf der META-Architektur (siehe folgender Absatz). Diese vier ALUs erlauben 32 bit Fließkommaoperationen, ferner sind sie ähnlich dem C64x-DSP gemäß SIMD aufteilbar für 2-fach 16 bit oder 4-fach 8 bit Festkommaarithmetikoperationen. Lesezugriffe auf das externe DRAM erfolgen über einen Cache, Schreibzugriffe erfolgen ohne Pufferung direkt. Für effiziente Pixelprozessierungs- und Videocodec-Aufgaben sind dedizierte Prozessorbefehle zusätzlich vorhanden.

Weitere Detailinformationen über die im USSE verwendete Mikroarchitektur sind nicht frei verfügbar, jedoch existiert der Hinweis auf prinzipielle Verwandtschaft zur META genannten DSP-Architektur des gleichen Herstellers nach [72]. Es ist dementsprechend sehr

¹²Threads sind hier gemäß der in der allgemeinen Literatur gewohnten Lesart als individuelle Programme zu verstehen, nicht in der Interpretation von NVidia's CUDA Threads.

wahrscheinlich, dass der USSE-Prozessor der SGX-GPU eine entsprechende individualisierte Architekturausleitung darstellt.

Mikroarchitektur META

Die META-Mikroarchitektur wurde gemäß [72] als ein kombiniertes RISC-CPU/VLIW-DSP-Konzept im Jahre 1995 von Imagination Technologies Ltd. initial entwickelt und seitdem in diversen PowerVR Grafikkbeschleunigern, aber interessanterweise auch Digital Audio-Anwendungen des Herstellers genutzt. Zusätzlich wird sie auch an externe Kunden lizenziert¹³. Ein hardwarebasierter präemptiver Echtzeitscheduler ermöglicht der Architektur den nahtlosen Wechsel zwischen verschiedenen Programmthreads ohne wechselbedingten Taktzyklenverlust. Ähnlich Intels Hyperthreading wird das Kontextwechseln mittels multipler, wechselseitig adressierter Registersätze implementiert. Jedoch ist deren Anzahl und entsprechend die mögliche Threadzahl höher, das Augenmerk liegt nicht vorwiegend in der Maskierung von Befehlsausführungslatenzen sondern in der Realisierung hart echtzeitfähiger, hochauflösend-quasisimultaner Multitaskingprogrammausführung. Es existieren zwei Befehlssatztypen. Hierzu gehören DSP-Instruktionen nach VLIW-Konzept und general-purpose RISC-Instruktionen. Gemäß VLIW-Charakter können mehrere vorhandene ALUs direkt angesprochen und vier Instruktionen pro Taktzyklus datenparallel ausgeführt werden.

Fazit zur möglichen Nutzung des PowerVR SGX

Es zeichnet sich ab, dass die PowerVR SGX Architektur von Imagination Technologies Ltd. im Prinzip ebenfalls – genauso wie die Systeme von NVidia und AMD – für GPGPU-Aufgaben fachfremd jenseits der 3D-Grafikberechnung nutzbar ist. Implementierungsversuche an der SGX Architektur konnten jedoch nicht als Teil in die vorliegende Arbeit einfließen: Für das proprietäre System sind die für eine Programmumsetzung nötigen Detailinformationen nicht frei verfügbar, erst seit kurzem (Anfang des Jahres 2013) ist vom Hersteller Imagination Technologies Ltd. eine direkte Programmierung des GPU-Chips offiziell als vorgesehen beziehungsweise unterstützt bekanntgegeben. Der Hersteller stellt dazu unter Auflagen ein SDK-Framework zur Verfügung, welches OpenCL-konform sei, er fordert jedoch hierfür die Unterzeichnung einer Lizenzvereinbarung/eines Dienstleistungsabkommens.

An der Historie des SGX lässt sich die Migration von einem typischen VLIW-DSP-Kern über eine Single-Core- und Low-Performance-GPU hin zu auf mehr Leistung skalierten symmetrischen Multicore-GPU-Systemen erkennen.

¹³Ein Beispiel aus dem Themenkontext der Arbeit stellt der interne DSP der in Kapitel 2 erwähnten DAB-Demodulatorsysteme von Frontier Silicon Ltd. dar.

3.3 Einordnung der GPU als Prozessorklasse

Die Architekturbetrachtung beschäftigte sich intensiv mit Prozessorarchitekturen aus dem Grafikbeschleunigersegment, aber auch mit den klassischen Vertretern der Prozessorsysteme. In der Literatur des CUDA-/OpenCL-Umfeldes wird die programmierbare NVidia- und AMD-GPU in der Regel als eine eigene Prozessorklasse mit neuartiger Mikroarchitektur dargestellt beziehungsweise definiert. Entsprechend wird auch die Programmiermethodik bei Nutzung der GPU für nicht-grafische Anwendungen als neuartig dargestellt. Nach Auffassung des Verfassers trifft dies im Grunde nach der vorliegenden Betrachtung nicht zu. GPUs bestehen aus einem als bekannte Architektur klassifizierbaren Prozessorkern, Elemente wesentlich technisch-neuartigem Charakters existieren hierbei im direkten Vergleich nicht. Diesem Kern sind grafikspezifische Subsysteme als Hardwareimplementierung beiseite gestellt, typischerweise zusätzliche Coprozessor- und Cachesysteme zur 2D-Texturtransformation und zum 3D-Vertex-Tiling.

3.3.1 Betrachtung der Prozessorarchitektur

Für diese These eines Prozessorkerns der GPU mit nicht-neuartiger Architektur sprechen die folgenden Designmerkmale, welche sich für die GPU in den vorigen Abschnitten identifizieren ließen:

Prozessorkern und Multicore-Konzept

Die Mikroarchitekturen der Prozessorkerne der betrachteten GPUs sind, mit Ausnahme applikationsspezifischer Textur- und Vertex-Einheiten, vollständig konform zu bekannten Technologien.

- SIMD-Verarbeitung mit einer moderaten Anzahl (bis zu 16) paralleler Rechenwerke, entsprechend auch dem Grundkonzept von am Markt etablierten CPU-SIMD-Technologien wie SSE und NEON.
- Symmetrisches Multiprocessing, insbesondere zur Leitungsskalierung in unterschiedlichen Chipvarianten einer Familie, ausgehend von einem Grundentwurf. Die Anzahl Subprozessoren bleibt in der Größenordnung derer gewöhnlicher Multicore-CPU's¹⁴.
- MIMD-Verarbeitung im Falle der AMD R800-Architektur (und mutmaßlich auch PowerVR SGX), entsprechend VLIW-ähnlichen Architekturen wie beispielsweise etablierten DSPs. Die in den Arithmetikgruppen der R800-Architektur umgesetzten Konzepte lassen eine starke Ähnlichkeit zu denjenigen des beispielhaft betrachteten C64x DSP-Kerns erkennen.
- NVidia führt die Bezeichnung SIMT für eine flexibel skalierbare, virtuell für den Programmierer vergrößerte SIMD-Breite ein. Diese wird abgebildet auf die tatsächlich in Hardware vorhandene Zahl von acht SIMD-Rechenwerken durch hardwarege-

¹⁴Beispielsweise NVidia Tesla G200 GPU aus dem high-end Segment mit 30 SM-Subprozessoren gegenüber x86-Server-CPU's der Familie AMD Opteron 6300 mit bis zu 16 x86-Kernen. Auch Single-Core GPU's sind existent wie die betrachtete AMD HD6310 oder der NVidia G98 Chip. Die NVidia MCP79 Ion GPU stellt ein Dualcore-System dar.

stützte, repetitive Befehlsausführung mit automatischer Datenadress-/Registerindex-Propagierung. Hardwareunterstützung für repetitive Befehlsausführung ist von DSPs bekannt (Zero Overhead Loops) und dort grundlegende Methodik.

Bezogen auf den vollständigen Chip der GPU bewegt sich die Gesamtanzahl an verfügbaren Rechenwerken nicht in außergewöhnlich höherem Bereich, sondern in ähnlichen Größenordnungen wie für gewöhnliche Multicore-Prozessorsysteme mit jeweils SIMD-verarbeitungsfähigen Kernen üblich.

Speichersystem

Im GPU-System ist eine Hierarchie unterschiedlicher Speichertypen implementiert. Die Speichersysteme der betrachteten GPUs entsprechen allgemein bekannten Prinzipien:

- Der im GPU-Kontext als Global Memory bezeichnete Speicher stellt den in Computersystemen üblichen, zum Prozessorchip externen DRAM dar. Das Local Memory (CUDA) beziehungsweise Private Memory (OpenCL) ist physikalisch identisch, jedoch sind durch das Laufzeitframework in diese Speicherbereiche nur Zugriffe aus einem einzelnen Programmthread gestattet.
- Der Speicherbereich mit der Bezeichnung Shared Memory (CUDA) beziehungsweise Local Memory (OpenCL) entspricht einem schnellen on-Chip Scratchpad-SRAM, beispielsweise bekannt aus der Domäne der DSP-Prozessoren.
- Da mehrfache Speicherzugriffe auf identische Datensätze zu erwarten sind, insbesondere bei der Texturinterpolation oder für die Parametrisierung der 3D-Transformation, sind Lesecachesysteme vorhanden, um repetitive externe Speicherzugriffe zu vermeiden.
- Für die GPU ist die große Registerbank ein Merkmal, jedoch kein Alleinstellungsmerkmal. Der NVidia G9x GPU stehen 8192 Register mit 32 bit pro SM zur Verfügung, der AMD R800 16×128 pro DPP¹⁵.

Die Speicherhierarchie entspricht folglich bekannten Mustern. Diese Muster existieren aufgrund allgemein bekannter Beschränkungen der Halbleiterschaltungstechnik, bei denen Zugriffszeiten gegenüber erzielbarem Speichervolumen abzuwägen sind, in jedem fortgeschrittenen Prozessorsystem. Die Muster in dieser Form sind nicht auf die GPU beschränkt.

3.3.2 Betrachtung des GPU-Programmierframeworks

Der CUDA-/OpenCL-Compiler versucht, eine von der darunterliegenden SIMD-/Multicore-Hardware abstrahierende, ausschließlich auf allgemeine Parallelität bedachte Problembeschreibung zu ermöglichen. Es wird ein Programmierstil des explizit via Syntax gekennzeichneten Parallelismus eingeführt, so dass eine flexible, vollautomatisierte Abbildung auf die jeweilig vorliegende Hardware durch den Compiler möglich wird. Die Sprachkonstrukte sind wie folgt mit der GPU-Hardwarearchitektur verbunden:

¹⁵Die DSP-Architektur der 1986 vorgestellten Motorola 56000er DSP-Familie verfügte bereits über 512 interne Register mit 24 bit Breite [74].

3 Verfügbare Mikroarchitekturen für eingebettete Infotainmentsysteme

- “Thread” (CUDA) beziehungsweise “Work-Item” (OpenCL) korrespondiert mit einem einzelnen Datensatz des datenparallelen Gesamtproblems.
- “Thread Blocks” (CUDA) beziehungsweise “Work-Groups” (OpenCL) stellen symmetrische Teilmengen des datenparallelen Gesamtproblems dar, welche jeweils auf einzelne Prozessorkerne¹⁶ des GPU-Multicore-SoCs zugewiesen werden.
- Bedingt durch deren SIMD-Charakter arbeitet jeder Prozessorkern jeweils eine fixe Gruppe einzelner Threads simultan pro Zeitschritt ab. Diese Gruppengröße ist jedoch im Regelfall kleiner als die Thread-Anzahl des “Thread Blocks” beziehungsweise der “Work-Group”. Deshalb werden von jedem SIMD-Prozessor sequentiell Threadgruppen abgearbeitet, bis der gesamte dem Prozessor zugewiesene “Thread Block” beziehungsweise “Work-Group” abgearbeitet ist¹⁷.
- “Grid” (CUDA) bezeichnet die Zusammenfassung aller einem Multicore-SoC-Device zugewiesener “Thread Blocks”.

Sämtliche der folgend aufgelisteten Eigenschaften gelten als kennzeichnendes Grundkonzept der GPU-Programmiersprachendialekte des CUDA GPU Frameworks als auch des offenen Standards OpenCL:

- “Imperative Style”: Die algorithmische Problembeschreibung erfolgt durch schrittweise imperative Anweisungen, sie ist nicht datenflussorientiert¹⁸.
- “Explicit Parallelism”: Der Programmierer deklariert durch Syntax existente Parallelität für den Compiler.
- “Local View of the Computation”: Ein virtueller skalarer Prozessor ist bei der Bearbeitung mit einem einzelnen elementaren Datenknoten des Gesamtproblems verknüpft.
- “Synchronous Execution of a Single Instruction Stream”: Virtuelle Prozessoren arbeiten identische Befehle im Gleichtakt ab, das heißt in SIMD-Manier.
- “Global Name Space”: Ein virtueller Prozessor kann ebenfalls auf Speicherinhalte anderer virtueller Prozessoren zugreifen. Somit wird Interprozessorkommunikation implementiert.
- Auch wird der Begriff eines “Shared Memory” zwecks Datentausch zwischen Prozessoren benutzt.

Diese vorangegangene Auflistung, wiedergegeben aus der Einleitung von Hatcher et al. [73], fasste im Jahre 1991 die Charakteristiken für die Sprache Dataparallel C, eine Abwandlung von C*, zusammen. Diese C-Spracherweiterung C*, bereits eingeführt im Jahre 1987, beschreibt sich selbst laut Thinking Machines Corp. [75] als Hilfestellung für Programmierer

¹⁶Prozessorkern im ursprünglichen Sinne der IT-Fachsprache (Baugruppe aus Ablaufsteuerung mit Rechenwerken und Registern), nicht in der durch Grafikkartenhersteller proklamierten Nomenklatur (einzelnes isoliertes Rechenwerk).

¹⁷Mit weiteren Hardwarespezifika entsteht so das Konstrukt “Warp” (CUDA).

¹⁸Trotz dieser Tatsache ist die GPU-Programmierung unverständlicherweise oft als “Stream Computing” vermarktet.

massiv paralleler Systeme mit verteiltem Speicher, unter anderem für Vektorprozessoren, und wurde initial für das Großrechensystem CM-2 entworfen. Um ein Beispiel in konkreter Weise anzuführen definiert das “with”-Statement von C* eine explizit parallele, per Index datenknotenbezogene Coderegion für datenparallele Verarbeitung, was vom Prinzip dem Konzept des GPU-Kernels mit virtuellen Threads entspricht. Generell fällt eine Verwandtschaft des datenkollektiv-orientierten GPU-Programmierstils auch zu anderen historischen Großrechnersystemen und den dortigen Ratschlägen der Hersteller für effiziente parallelisierte Programmierung auf¹⁹. Die Abbildung der parallelen Hardwarestruktur auf die syntaktisch/semantische Struktur der verwendeten Spracherweiterung gleicht bei der GPU also derjenigen, welche bereits historische Supercomputer verwendeten.

3.3.3 Softwareoptimierungsprinzipien auf der GPU

Als Folge ist auch der Forschungsgegenstand inklusive der Herangehensweise der Adaptation von Algorithmen bei der GPU nicht vollständig neuartig, die bestehende umfangreiche Literatur aus dem historischen Wissenschaftsumfeld steht im direkten Bezug. Das Einordnen in den Kontext bekannter Architekturkonzepte lässt folglich auf den ersten Blick GPU-spezifische Optimierungstechniken als konform zu bekannter Programmiermethodik der bestehenden Literatur erkennen. Der Hersteller NVidia [49] fasst die Optimierungskonzepte der GPU-Programmierung wie folgt zusammen:

- “Expose sufficient parallelism” / “Find ways to parallelize sequential code”
 - “Adjust kernel launch configuration to maximize device utilization”
- “Avoid different execution paths within the same warp” / “Have coherent execution within warps of threads”
- “Use memory efficiently”
 - “Use shared memory where possible”
 - “Ensure global memory accesses are coalesced” / “Coalesce global memory accesses”
 - “Minimize redundant accesses to global memory”
 - “Minimize data transfers between the host and the device”

In den folgenden Absätzen sollen diese Forderungen in den allgemeinen Softwareentwicklungskontext eingeordnet werden.

Neuformulierung der Algorithmen

Algorithmische Problemstellungen sind entsprechend massiv-paralleler Verarbeitung für die GPU neu zu formulieren.

¹⁹Hierunter zählen neben anderen Cray-1 (1976, Vektorprozessor mit 64 Rechenwerken) [76], Alliant FX/8 (1985, 8 Vektorprozessoren mit 8 Rechenwerken) [77], nCUBE (1985, bis zu 1024 Prozessoren) oder nCUBE-3 (1992, bis zu 65536 Prozessoren) [78].

3 Verfügbare Mikroarchitekturen für eingebettete Infotainmentsysteme

- Das Gesamtproblem zu datenparallelen elementaren Teilproblemen zu fragmentieren entspricht dem Datenverarbeitungsparadigma eines klassischen SIMD-/Vektor-Computers.
- Die Optimierungsanweisung nach der Nutzung sämtlicher verfügbaren Rechenwerke einer Vektor/SIMD-Architektur, soweit es möglich und sinnvoll ist, stellt sich als selbstverständlich für jede Prozessorfamilie dar.
- Das Paradigma der Vermeidung von divergentem Programmfluss betrifft sämtliche Vektor/SIMD-Computersysteme. Entsprechend ihrer Natur ist es logisch, dass SIMD-Architekturen nur den maximalen Durchsatz erreichen können, wenn nicht zur Realisierung von Fallunterscheidungen im Programmfluss ein Anteil der vorhandenen Rechenwerke aufgrund bedingter Ausführung deaktiviert werden muss.
- Das Konzept von Synchronisationspunkten, um über geteilte Speicherbereiche Daten zwischen den ansonsten strikt disjunkt verarbeiteten Teilproblemen gegenseitig auszutauschen, ist von verteilten Berechnungssystemen hinreichend bekannt.

Sämtliche Punkte gelten also nicht explizit für die Programmierung der GPU, sie müssen auch bei effizienter Umsetzung beispielsweise auf den heutigen gewöhnlichen PC-CPU-Architekturen, in der Regel ein SIMD-fähiges x86-Multicoresystem, beachtet werden.

Occupancy

NVidia führt den Begriff der Occupancy (CUDA) zur Bewertung von GPU-Programmroutinen als Gütemaß für den Entwickler neu ein. Diese Metrik soll genauer betrachtet werden.

- Occupancy lässt sich erkennen als derjenige Wert, welcher angibt, wie viel Prozent der gesamten internen Registerbank des Prozessorkerns vom programmierten GPU-Algorithmus genutzt werden.
- Die interne Registerbank wird aufgeteilt zwischen allen Datensatzelementen, welche durch den Compiler zur gemeinsamen Verarbeitung auf einen Prozessorkern (Thread-Block beziehungsweise Work-Group) abgebildet werden.
- Entsprechend folgt trivial, dass die Anzahl der gemeinsam auf einem Prozessorkern verarbeitbaren Datensatzelemente (CUDA-Threads) im implementierten Algorithmus limitiert ist durch die zur Verarbeitung eines einzelnen Datensatzelementes nötige Anzahl von Arbeitsregistern in Zusammenspiel mit der Gesamtzahl verfügbarer Prozessorregister. Im Umkehrschluss muss die Anzahl der vom Algorithmus je Thread allokierten Arbeitsregister reduziert werden, wenn die Threadzahl gesteigert werden soll.
- Das Programmierparadigma, die interne Registerbank des Prozessorkerns so intensiv wie möglich zu nutzen, insbesondere bevor gegenüber dem Prozessor externe, langsamere Speicherzellen einbezogen werden, ist generell allgemein für sämtliche Prozessortypen anerkannt.

Werden nicht sämtliche Register des Prozessors verwendet, ist das definierte Gütemaß Occupancy niedrig. Trotzdem kann ein Algorithmus optimal effizient implementiert sein, auch wenn dabei weniger als sämtliche verfügbaren Arbeitsregister belegt werden. Ein Algorithmus kann hohe Occupancy erreichen und trotzdem ineffizient sein, wenn er pro Verarbeitung eines Datensatzelements viele Arbeitsregister belegt oder die Algorithmenpartitionierung bezüglich der auf einem SM-Prozessorkern ausgeführten GPU-Threadanzahl unnötig in die Höhe getrieben wird.

Speicherzugriff

Beim Speicherzugriff sind auf der GPU reguläre Muster einzuhalten, um einen gemeinsamen (engl. coalesced, nach NVidia) Speichertransfer mehrerer GPU-Threads zu ermöglichen.

- Dies gilt allgemein und in natürlicher Weise für das Laden von Vektoren auf sämtlichen SIMD-Prozessorsystemen. Auch dort würde eine Nichtbeachtung dedizierte zusätzliche Operationen zur Neusortierung der Vektoreinträge erfordern.
- Eine Empfehlung beziehungsweise Forderung nach Ausrichtung der Speicherzugriffe an Wortgrenzen ist generell üblich in Prozessoren²⁰.

Die Hierarchie der Speicherbereiche soll bei der GPU entsprechend ihrer unterschiedlichen Zugriffszeit optimiert genutzt werden:

- Wie bereits für den innersten Speicherbereich, der Registerbank, angesprochen ist es auf jedem Prozessortyp selbstredend, dass schnelle Speicherbereiche soweit als möglich bevorzugt genutzt werden.
- Unnötige mehrfache Speicherzugriffe oder unnötig hohes Transfervolumen, insbesondere diejenigen eines vergleichsweise langsamen Speicherbereichs oder Interprozessorbusses, sind generell in jeder Prozessorarchitektur bei der hardwarenahen Programmierung zu vermeiden.

3.3.4 Fazit zur GPU-Architektur

Nach Auffassung des Verfassers eröffnet die GPU keine eigene neue Prozessorklasse. Die angewendeten Methoden sind im Mikroprozessordesign bekannt, neuartige Prozessordesignkonzepte finden sich bei detaillierter Betrachtung der vorliegenden GPUs nicht. Eine GPU ist vielmehr als ein homogener Multicore-Prozessor mit SIMD-Rechenwerken zu sehen, dessen DSP-ähnliche Mikroarchitektur applikationsspezifisch auf den Anwendungsfall des 3D-Grafikrenderings hin optimiert wurde.

Die Softwaretechniken der GPU-Programmierung sind nicht neu, es treffen im Grunde sämtliche der in der wissenschaftlichen Literatur des historischen Supercomputings etablierten Softwareadaptationstechniken für datenparallele Systeme unverändert auf die GPU-Domäne zu. Entsprechend kann auch erkannt werden, dass sämtliche vom Programmierer

²⁰So sind auch für die CPU-Architekturen x86-SSE und ARM-NEON Speicherzugriffe entsprechend der Wortgrenzen von beispielsweise 128 bit Vektoren auszurichten und die Vektorindizes im Speicher sequentiell beziehungsweise im Rahmen bestimmter Muster abzulegen.

anzuwendenden GPU-Optimierungsempfehlungen genauso wie die jeweils zugrunde liegenden Hardwareelemente bereits vom Prinzip bekannt und somit etablierter Stand der Wissenschaft fortgeschrittener Programmierweisen sind.

3.4 Architektur-Implementierungsvarianten

Für die praktische Evaluierung der betrachteten Mikroarchitekturen in den späteren Kapiteln ist die Auswahl konkret verfügbarer Produktrealisierungen notwendig. Diese erfolgt im Folgenden mit dem Hintergrund einer prototypischen eingebetteten Infotainment-Ziellplattform.

Implementierungsvarianten Intel Bonell (Intel Atom)

Der Intel Bonell-Prozessorkern kommt in verschiedenen Konfigurationen mit der Produktbezeichnung Intel Atom zur Implementierung. Es existieren bislang Konfigurationen als Single- und Dualcore-CPU. Taktraten von bis zu 2.0 GHz werden erreicht. In der ersten Generation der Realisierungsvarianten sind Speichercontroller, Grafikeinheiten oder Peripheriebausteine nicht intern im Prozessorpackage des Atom integriert. Hierfür ist ein externer Companion-Chip nötig, dies können beispielsweise die System Controller Hubs (SCH) der US15-Reihe von Intel oder der im Kapitel später eingeführten NVidia Ion MCP79 sein. Mit der zweiten Implementierungsgeneration sind Varianten verfügbar, welche bei identischem CPU-Kern den bislang externen Speichercontroller und eine Grafikeinheit auf ein gemeinsames Silizium integrieren oder komplette SoC-Lösungen für Smartphones, Tablets und sehr preissensitive low-end Computersysteme darstellen²¹. Die für die Evaluierung der Leistungsfähigkeit der Bonell-Mikroarchitektur ausgewählte Prototypenplattform basiert auf der Intel Atom N270 CPU [52]. Bei dieser Implementierungsvariante der ersten Generation handelt es sich um einen Singlecore-Prozessor. Die Thermal Design Power (TDP) des Prozessorkerns ist in typischem Betrieb mit durchschnittlich 2.5 W elektrischer Verlustleistung angegeben. Der Prozessorkern ist für eine Taktung von 1.6 GHz ausgelegt, der Frontside-Bus zu Speicher und Peripherie wird mit 533 MHz betrieben. Bei $f_{\text{CLK}}=1.6$ GHz Taktung kann für die CPU ein theoretischer Maximaldurchsatz gemäß

$$O_{\text{max, 16 bit}} = 2 \text{ OPS/sHz} \cdot \frac{128 \text{ bit}}{16 \text{ bit}} \cdot 1.6 \text{ GHz} = 25.6 \text{ GOPS/s} \quad (3.1)$$

angenommen werden. Dies gilt hinsichtlich der im Idealfall dual-issue-fähigen SSE-Rechen-einheit bei 16 bit-SIMD-Arithmetik.

Implementierungsvarianten AMD Bonell (AMD Fusion)

Die AMD Bobcat-Mikroarchitektur findet sich als Single- bis Quadcore-Prozessorsystem in verschiedenen Produktimplementierungen der E-Reihe von AMD CPUs [55]. Das SoC AMD E-350 Zacate [54] bietet zwei mit 1.6 GHz getaktete CPU-Kerne. Weiterhin findet

²¹Auch existiert speziell für industrielle Anwendungen die Reihe E600C, bei der dem Prozessorsilizium auf dem Chipträger direkt ein Altera Arria II FPGA zur Seite gestellt ist. Die interne Anbindung des eingebetteten FPGAs an den CPU-Kern erfolgt über PCIe. Varianten mit reduzierter Taktung und erweitertem industriellem Temperaturbereich sind verfügbar.

3.4 Architektur-Implementierungsvarianten

sich auf dem gemeinsamen Silizium ein Grafikprozessor nach der R800 Architektur (siehe entsprechender folgender Abschnitt dieses Kapitels). Das Konzept einer CPU mit direkt integrierter, vollprogrammierbarer GPU wird von AMD als Accelerated Processing Unit (APU) bezeichnet und unter dem Produktnamen AMD Fusion vermarktet. Daneben ist im SoC die Funktionalität des Speichercontrollers integriert. Die Systemplattform wird vervollständigt durch einen zweiten Chip, dem Fusion Controller Hub (FCH) A50M, welcher die peripheren Funktionen im System übernimmt. Die Kommunikation zur APU erfolgt über 4 PCIe-Lanes.

Als erwarteter theoretischer Maximaldurchsatz bei $f_{\text{Clk}}=1.6$ GHz Taktung ergibt sich für einen einzelnen Bobcat CPU-Kern bei 16 bit Arithmetik unter Verwendung von SSE-SIMD-Instruktionen:

$$O_{\text{max, 16 bit}} = 1 \text{ OPS/sHz} \cdot \frac{128 \text{ bit}}{16 \text{ bit}} \cdot 1.6 \text{ GHz} = 12.8 \text{ GOPS/s} \quad (3.2)$$

Bezüglich des Dualcore-SoC AMD E-350 steht insgesamt auf der Plattform entsprechend die doppelte Leistung zur Verfügung.

Implementierungsvarianten ARM Cortex A8/A9 (Texas Instruments DM3730)

ARM Incorp. selbst fertigt keine eigenen Prozessorchips, sondern bietet Halbleiterherstellern die Prozessorarchitektur als Intellectual Property an, hierbei in verschiedenen Formen von Verträgen über Lizenzierung der Befehlssatzarchitektur bis zu fertigen Prozessorkern-Schaltungsdesigns. Für den Cortex A9 ist beispielsweise bekannt, dass ein komplettes Layout (engl. Hard Macro) zur Integration in ein SoC mittels eines 40 nm-Prozesses für ein Dual-Core-System mit bis zu 2 GHz Takt bei 1.9 W Leistungsaufnahme als IP-Block an Halbleiterhersteller angeboten wird. Alternativ gibt es ein verlustleistungs- und chipflächenoptimiertes Makro mit den Grenzdaten 800 MHz und 0.5 W. Unter anderem mit der OMAP Prozessor-Serie bietet der Hersteller Texas Instruments SoC-Systeme mit Produktrealisierungen der Prozessorkerndesigns Cortex A8 und A9 an. Für die spätere Untersuchung wurde aufgrund eines zusätzlichen beiseite gestellten C64x-DSP-Kerns das Cortex A8-basierte SoC mit Bezeichnung DM3730 [62] ausgewählt. Der ARM Cortex A8-Prozessor des SoC kann mit bis zu $f_{\text{Clk}}=1000$ MHz betrieben werden. Der erwartete arithmetische Durchsatz bei dieser maximalen Taktung ergibt sich dann zu

$$O_{\text{max, 16 bit}} = 2 \text{ OPS/sHz} \cdot \frac{64 \text{ bit}}{16 \text{ bit}} \cdot 1 \text{ GHz} = 8 \text{ GOPS/s} \quad (3.3)$$

Implementierungsvarianten Texas Instruments C64x (Texas Instruments DM3730)

Vom C64x-Kern existieren von Texas Instruments diverse Realisierungen in Einzelkonfiguration bis hin zu symmetrischen Multicoresystemen mit sechs Kernen [59]. Auch wird der Kern als SoC-Verbund mit anderen Prozessorfamilien heterogen kombiniert. Im bereits eingeführten und im späteren evaluierten SoC DM3730 [62] von Texas Instruments arbeitet der C64x-Kern wie angesprochen als Subsystem zu einem ARM Cortex A8-Kern. Die Caches am C64x-Kern sind dabei im Einzelnen wie folgt dimensioniert:

- Level 1-Instruktionscache, 32 kB, direkt abgebildet.

3 Verfügbare Mikroarchitekturen für eingebettete Infotainmentsysteme

- Level 1-Datencache, 80 kB, 2-fach assoziativ.
- Bis zu 64 kB gemeinsamer Level 2-Cache, 4-fach assoziativ, aus einem 96 kB fassenden internen SRAM Speicherbereich.

Bei einer Taktung von 600 MHz folgt im Idealfall der theoretisch maximale Arithmetikdurchsatz gemäß

$$O_{\max, 16 \text{ bit}} = 8 \text{ OPS/sHz} \cdot \frac{32 \text{ bit}}{16 \text{ bit}} \cdot 600 \text{ MHz} = 9.6 \text{ GOPS/s} \quad (3.4)$$

Implementierungsvarianten NVidia G9x (NVidia Ion)

Für den Einsatzbereich als separater Grafikkadaper in einem PC-System existieren die Chipvarianten G92 bis G98 als Realisierung der G9x GPU-Architektur. Diese IC-Varianten skalieren die Leistungsfähigkeit über ein bis 16 enthaltene SMs. Als Companion-Chip für die Intel Atom-CPUs wurde der sogenannte Media Communications Processor (MCP) mit dem Codenamen MCP79 entworfen. Das IC integriert neben den für einen Chipsatz essenziellen Funktionen der DRAM- und Peripherie-Controller ebenfalls eine vollprogrammierbare GPU, welche äquivalent zu der stand-alone Lösung mit Codenamen G96 ist. Die GPU greift jedoch nicht auf eigenes RAM zu, sondern bedient sich für den Videospeicher im Sinne einer geteilten Nutzung des DRAMs der Host-CPU. Die MCP79 GPU besteht aus zwei SM-Teilprozessoren. Damit sind insgesamt 16 parallele general-purpose Rechenwerke mit 32 bit Breite verfügbar. Die Taktung des GPU-Teils des MCP79 beträgt $f_{\text{Clk}}=1100 \text{ MHz}$. Der theoretische Maximaldurchsatz für allgemeine arithmetische Berechnungen ergibt sich damit wie folgt:

$$O_{\max} = 2 \text{ OPS/sHz} \cdot 8 \cdot 1.1 \text{ GHz} = 17.6 \text{ GOPS/s} \quad (3.5)$$

Implementierungsvarianten AMD R800 GPU

Die R800-Mikroarchitektur findet sich in den Grafiksystemen der Radeon HD5000-Serie von AMD. Daneben finden sich Vertreter der Architektur in den frühen Exemplaren der HD6000-Serie, insbesondere für den mobilen Einsatzbereich²². Die zugrunde liegenden GPU Chips RV810 bis RV870 skalieren die Leistungsfähigkeit über ein bis zehn integrierte DPPs. Die Bezeichnung AMD Radeon HD6310 stellt den integrierten Grafikcontroller des bereits eingeführten SoC AMD E-350 Zacate dar. Der GPU-Kern wird mit $f_{\text{Clk}}=500 \text{ MHz}$ getaktet. Die Anbindung des externen Speichers erfolgt über ein 64 bit Interface und ist mit einem theoretischen Maximaldurchsatz von 8.5 GB/s spezifiziert. Es ist ein einzelner DPP vorhanden. Die verfügbare Vektor-Breite beträgt folglich 16, jeweils über fünf arithmetische Rechenwerke. Der theoretische Maximaldurchsatz an arithmetischen Operationen kann wie folgt abgeleitet werden:

$$O_{\max} = 1 \text{ OPS/sHz} \cdot 16 \cdot 5 \cdot 492 \text{ MHz} = 39.4 \text{ GOPS/s} \quad (3.6)$$

²²Für spätere Modelle wird eine Folgemikroarchitektur mit dem Namen R900 eingesetzt werden.

Implementierungsvarianten Imagination Technologies PowerVR SGX

Der SGX-Kern ist ebenfalls in dem in der Arbeit untersuchten ARM/DSP-SoC Texas Instruments DM3730 in der Variante SGX530 enthalten, daneben unter anderem in Lizenz in Intel's US15-Chipsätze der ersten Generation für die Intel Atom CPU integriert (SGX535, unter der Bezeichnung Intel GMA500). Weitere Lizenzierungen des SGX- beziehungsweise des SGXMP-Kerns finden sich in diversen smartphoneherstellereigenen SoCs²³. Die vorliegenden Varianten SGX530/SGX535 bestehen symmetrisch aus zwei USSE-Verarbeitungskernen mit $f_{\text{CLK}}=200$ MHz Taktfrequenz. Der theoretische Maximaldurchsatz an arithmetischen Operationen für zwei Kerne, vier nebenläufige Instruktionen pro Taktzyklus und 2-fach SIMD bei 16 bit Festkommatdaten kann wie folgt abgeleitet werden:

$$O_{\text{max}} = 2 \text{ Ops/sHz} \cdot 4 \cdot 2 \cdot 200 \text{ MHz} = 3.2 \text{ GOPS/s} \quad (3.7)$$

3.4.1 Betrachtung hinsichtlich theoretischen Maximaldurchsatzes

Im Vorfeld der tatsächlichen Laufzeitmessergebnisse und entsprechenden Architektureinschätzungen der späteren Fallstudie fasst Tabelle 3.1 die theoretisch für die anvisierte Signalverarbeitungsimpementierung zur Verfügung stehende arithmetische Rechenleistung der einzelnen Mikroarchitekturen zusammen. Es handelt sich dabei um den maximalen Rechenwerksdurchsatz in 16 bit Arithmetikoperationen wie er für den Anwendungsfall Radiosignalverarbeitung später primär benötigt werden wird. Auf den GPU-Architekturen existiert dieser Datentyp nicht beziehungsweise nicht in effizienter Weise, entsprechend stehen die Werte für den dort somit anzuwendenden Fließkommaarithmetiktyp 32 bit. Die Tabelle betrachtet den Operationsdurchsatz²⁴ pro einzelner, isoliertem Prozessorkern. Da jede einzelne gelistete Mikroarchitektur als Mehrkern-System oder Singlecore verfügbar ist, stellt die Normierung mit Fokus auf einen einzelnen Kern keine Beschneidung der allgemeinen Vergleichbarkeit dar. Zur Vollständigkeit sind für Mehrkernsysteme die Leistungswerte des gesamten Chippackages jedoch ebenfalls notiert. Die Werte offenbaren, dass hinsichtlich des maximal möglichen Arithmetikdurchsatzes CPU- und GPU-ICs in derselben Größenordnungen agieren und sich nicht eine Klasse der anderen als vom generellen Ansatz technisch überlegen zeigen kann. Entsprechend dieser abstrakten Voruntersuchung wird für die im Späteren folgenden praktischen Umsetzungen keine außergewöhnlich übermäßige Laufzeitvarianz zwischen den GPU- wie CPU-Implementierungsansätzen erwartet.

Sämtliche betrachteten aktuellen Applikationsprozessorarchitekturen bieten Befehlssatzerweiterungen für beschleunigte Datenverarbeitung mittels SIMD-Vektoroperationen. Hinsichtlich des maximalen Arithmetikdurchsatzes unterscheiden sich die beiden low-cost x86-Architekturen bei gleicher Taktfrequenz in der Theorie um den Faktor zwei. Der Vorsprung

²³Beispielsweise Apple A4 (SGX535, z.B. Apple iPhone 4) bis A6 (SGX543MP3, Tricore, z.B. Apple iPhone 5), Samsung S5P Exynos (z.B. Samsung Galaxy S/Wave/Tab/S4, Google Nexus S/Nexus 10).

²⁴Teilweise wird in Herstellerdokumentation und Literatur zur Bekundung eines gesteigerten Wertes die Interpretationsmöglichkeit angewandt, Prozessorbefehle, welche mehrere arithmetische Elementaroperationen zugleich ausführen, als mehrere Operationen zu werten (bspw. MULADD-Instruktionstypen, eine kombinierte Multiplikation und Addition, als zwei gezählte Operationen). Solche Interpretationsmöglichkeiten sollen beim vorliegenden Vergleich nicht angewendet werden. Ferner existieren diese Kombinationsbefehle auf sämtlichen untersuchten Architekturen. Es existiert somit kein spezifischer Architekturvorteil, so dass sich in der gewählten Betrachtungsweise keine Beschränkung der Objektivität ergibt.

3 Verfügbare Mikroarchitekturen für eingebettete Infotainmentsysteme

Mikroarchitektur	Parallelität ²⁵	Erwarteter Rechendurchsatz
CPU:		
Intel Bonell ↔ <i>Intel Atom N270: 1 Kern</i>	2 × 8	25.6 GOPS/s @ 1.6 GHz
AMD Bobcat ↔ <i>AMD E350: 2 Kerne</i>	8	12.8 GOPS/s @ 1.6 GHz <i>total: 25.6 GOPS</i>
ARM Cortex A8 ↔ <i>Texas Instruments DM3730 SoC: 1 Kern</i>	8	8.0 GOPS/s @ 0.6 GHz
DSP:		
Texas Instruments TMS320C64x ↔ <i>Texas Instruments DM3730 SoC: 1 Kern</i>	8 × 2	9.6 GOPS/s @ 0.6 GHz
GPU:		
NVidia G9x SM ↔ <i>NVidia MCP79 (Ion): 2 Kerne</i>	8	8.8 GOPS/s @ 1.1 GHz <i>total: 17.6 GOPS/s</i>
AMD R800 DPP ↔ <i>AMD E350 (Radeon HD6310): 1 Kern</i>	5 × 16	39.4 GOPS/s @ 0.5 GHz
PowerVR-SGX USSE ↔ <i>Texas Instruments DM3730 (SGX530) und Intel US15W (SGX535): 2 Kerne</i>	4 × 2	3.2 GOPS/s @ 0.2 GHz <i>total: 6.4 GOPS/s</i>

Tabelle 3.1: Vergleich der Mikroarchitekturen bezüglich der theoretischen arithmetischen Leistungsfähigkeit je isoliertem Prozessorkern (darunter: Werte für den Gesamtchip der hier betrachteten Implementierungsvarianten, falls symmetrisches Multicore-Design).

der Intel Bonell-Architektur liegt darin begründet, dass deren zweifache Superskalarität für die volle SIMD-Vektor-Breite von 128 bit gilt. Jedoch, um vorzugreifen, wird sich in Kapitel 5 zeigen, dass dieser Architekturvorteil in der Praxis nicht nutzbringend verwertet werden kann. Die Mikroarchitektur der aus dem Umfeld energieeffizienter eingebetteter Systeme stammenden ARM Cortex A8-CPU verspricht unter Beachtung der wesentlich geringeren Taktung im Vergleich zu den x86-CPU's eine konkurrenzfähige Arithmetikleistung. So kann die ARM-Architektur als dem x86-Ansatz aktueller low-cost PC-Systeme hinsichtlich OPS/Hz als ebenbürtige Systemklasse erachtet werden. Die Architektur der AMD R800/Radeon 6310 GPU weist einen deutlich höheren Wert aus als die NVidia MCP79 Ion GPU. Jedoch sind bezüglich dieses Wertes die dargestellten Mikroarchitekturbeschränkungen der AMD R800-Rechenwerke hinsichtlich der möglichen Operandenwahl im praktischen Betrieb im Gedächtnis zu behalten.

²⁵Soweit möglich für die Nutzung der im Anwendungsfall bevorzugten 16 bit-Arithmetik, ansonsten dem nativen Datentyp der Architektur.

4 Entwicklung einer effizienten Signalverarbeitungskette für DAB-Empfang

4.1	Das DAB-System	68
4.1.1	Eigenschaften und Aufbau des Übertragungsverfahrens	68
4.1.2	Mathematische Formulierung des DAB-Signals	74
4.2	Entwicklung der Empfangsalgorithmen	75
4.2.1	Synchronisation: Signalakquisition	75
4.2.2	Synchronisation: Tracking	87
4.2.3	Demodulation	90
4.3	Nutzdatenhandling	100
4.3.1	Fast Information Channel	100
4.3.2	Main Service Channel	101
4.4	Softwaretechnische Umsetzung des Empfängersystems	101
4.4.1	Entwicklung des Programmcodes	101
4.4.2	Eliminierung von Datentransfers für Zwischenergebnisse	101
4.4.3	Übergang zur Festkommaarithmetik	103
4.4.4	Schnittstellen des Demodulatorsystems	103
4.5	Bewertung der Signalverarbeitungskette	103
4.5.1	Speicheranforderungen	103
4.5.2	Vermessung der Empfangsgüte	104

Für den in der Fallstudie exemplarisch gewählten Radiostandard DAB beschreibt das Kapitel die Herleitung einer Empfänger-Referenzsignalverarbeitungskette, anhand derer später die Evaluierungen der Prozessorplattformen erfolgen sollen. Ziel der Entwicklung ist eine hinsichtlich Ausführungsperformance möglichst optimale Umsetzbarkeit auf Prozessoren. Deshalb liegt der Fokus auf der Auswahl solcher Rechenvorschriften, welche geringen mathematischen Aufwand und minimales Speichertransfervolumen bei Erfüllung der geforderten DAB-standardkonformen Funktionalität benötigen. Die hergeleitete Codebasis

bildet den Ausgangspunkt für die späteren Implementierungsvarianten aus Kapitel 5 und 6 zur Laufzeituntersuchung der Zielmikroarchitekturen¹. Zu Beginn soll ein Überblick über die im Softwaredemodulator umzusetzenden Grundkonzepte des Übertragungsverfahrens DAB gegeben werden. Danach werden die einzelnen Demodulationsalgorithmen im Detail erläutert und die dem erstellten Code zugrunde liegenden Paradigmen dargestellt. Abschließend wird die entwickelte Referenzempfangskette hinsichtlich ihrer nachrichtentechnischen Performance validiert.

4.1 Das DAB-System

Die Standardfamilie Digital Audio Broadcasting (DAB) wurde in den Jahren 1987 bis 2000 durch das Eureka-147 Projekt als moderner, auf digitalen Übertragungstechnologien basierender Nachfolger für die analoge UKW/FM-Radioausstrahlung entwickelt.

4.1.1 Eigenschaften und Aufbau des Übertragungsverfahrens

Für das DAB-Verfahren existieren neben dem Basisverfahren Teilüberarbeitungen beziehungsweise Ergänzungen des Standards:

- Das Basisverfahren ist gemäß ETSI EN 300 401 [2] spezifiziert. Es deklariert die physikalische Übertragungsschicht und die Übertragung von Audiodiensten. Für die Audioquellencodierung wird das Kompressionsverfahren nach MPEG-1 Layer 2 verwendet.
- Der Substandard DAB Plus nach ETSI TS 102 563 [80] fügt Unterstützung für das effizientere Audiokompressionsverfahren MPEG-4 HE-AACv2 hinzu und führt weiterhin eine zweite, zusätzliche Fehlerschutzebene, basierend auf Reed Solomon-Codes, für die Audiodaten ein.
- Mit den als Digital Media Broadcast (DMB) bekannten Systemerweiterungen ETSI TS 102 427 [81] und ETSI TS 102 428 [82] wird die Übertragung von Nutzdaten über einen MPEG-Transportstrom (MPEG-TS) und somit auch von Videoinhalten unterstützt. DMB adaptiert ähnlich DAB Plus ebenfalls eine zweite Fehlerschutzebene anhand von Reed Solomon-Codes. Als Quellencodierverfahren für die Audiodaten wird MPEG-4 HE-AAC oder MPEG-4 BSAC verwendet, für Videodaten kommt H.264/MPEG-4 AVC zum Einsatz.

Alle Teilstandards behalten stets die unterste Schicht des ursprünglichen DAB-Basissystems bei, das heißt das grundlegende Modulationsverfahren, den Multiplex-/Rahmenaufbau und sämtliche Signalisierungsmechanismen. Folglich sind für DAB Plus oder DMB die identischen Kanaldecoder mit gleichen Radiosignalverarbeitungsalgorithmen unverändert verwendbar, eine unterschiedliche Datenbehandlung erfolgt erst nach Demodulation und Multiplextrennung. Durch dies ist auch Rückwärtskompatibilität und Interoperabilität der Sendeinfrastruktur gesichert, es können innerhalb eines Signalmultiplexes zugleich Anteile des Programmbouquets gemäß DAB-, DAB Plus- und DMB-Standard abgestrahlt

¹Des Weiteren diente die hier vorgestellte DAB-Referenzimplementierung als Grundlage für die in [79] durchgeführten Untersuchungen eines kombinierten DAB- und 802.11p-Multistandardempfängers.

werden². Im Vergleich zu anderen Übertragungsstandards können folgende Eigenschaften beziehungsweise Unterscheidungsmerkmale für DAB hinsichtlich seiner Zielgruppe identifiziert werden, beispielsweise in Abgrenzung zu dem im selben Zeitraum in Europa als digitales Übertragungsmedium für Fernsehen ausgelegten DVB-T-Verfahren nach dem Standard ETSI EN 300 744 [3]:

- Auslegung des Modulationsverfahrens primär für hochmobile Teilnehmer (differenziell encodierte Modulation, Empfang bis zu Geschwindigkeiten von $v=200$ km/h).
- Ressourceneffiziente Signalverarbeitung im Sinne günstiger Hardwarekosten sowie geringer Leistungsaufnahme/Batteriebetrieb des Empfängers (Time Slicing, inkohärente Demodulation).
- Dienste mit niedrigerer bis mittlerer Datenrate (Audio oder Handheld-TV).

Wellenform und Modulation

Mehrere DAB-Dienste sind stets in einem Ensemble zu einem Multiplex gruppiert und werden durch einen gemeinsamen Sendemodulator übertragen. Pro Ensemble-Block werden typischerweise sechs bis 16 Audio- und/oder Datenservices zusammengefasst. Ein Block allokiert im elektromagnetischen Spektrum inklusive Schutzabstand stets eine HF-Bandbreite von 1.75 MHz. Dedizierte Bereiche im Spektrum zur Nutzung für DAB wurden im VHF-Band III (174...240 MHz) und im L-Band (1452...1492 MHz) koordiniert. Um die physikalische Übertragungsschicht bestmöglich an die genutzte HF-Funkstrecke anzupassen existieren vier Modi zu deren Parametrisierung. Diese sind angepasst für die Nutzung verschiedener Frequenzbänder und Ausbreitungsszenarien, so dass der Struktur des zugrunde liegenden Sendernetzwerkes Rechnung getragen werden kann. Mode I ist bestimmt für die terrestrische Verbreitung im VHF-Band III. Da Mode I der zum heutigen Datum ausschließlich genutzte DAB-Mode sämtlicher Sendeanlagen-Betreiber in Deutschland ist, soll im weiteren Dokument der Fokus der Betrachtungen für Mode I gelten.

Von Seiten der physikalischen Schicht stellt DAB einen klassischen Vertreter eines auf Orthogonal Frequency Division Multiplex (OFDM)-basierten Übertragungsverfahrens dar³. In Mode I wird ein OFDM-Schema mit 1536 aktiven Unterträgern verwendet. Ein Schutzintervall (engl. Guard Interval, GI) mit zyklischem Präfix (engl. Cyclic Prefix, CP), dessen Länge in etwa circa 25 % der Dauer des eigentlichen OFDM-Symbols beträgt, wird dabei verwendet. Damit ist eine intersymbolinterferenzfreie Übertragung auch bei massiver Mehrwegeausbreitung und entsprechend weitgefächertem Echoprofil im Signal gesichert. Weiterhin wird so auch eine Realisierung von Gleichwellennetzen (engl. Single Frequency Networks, SFN) mit starker Laufzeitdifferenz aufgrund großer örtlicher Ausdehnung ermöglicht. Für die Modulation der einzelnen OFDM-Unterträger kommt eine QPSK-Konstellation zum Einsatz. Um in Szenarios höchster Mobilität des Empfängers

²Auf diese Weise kann beispielsweise ein Multiplex schrittweise von DAB auf DAB Plus migriert werden oder audiovisuelle DMB-Dienste neben DAB (Plus)-Hörfunkprogrammen hinzugefügt werden.

³So können aus den Untersuchungsergebnissen Schätzwerte auch für Übertragungssysteme mit verwandten Technologien abgeleitet werden. Dies soll in Kapitel 7 für das digitale Fernsehübertragungssystem DVB-T durchgeführt werden.

Systemparameter	Wert
Dauer Übertragungsrahmen	$T_F = 96 \text{ ms}$
OFDM-Symbole pro Rahmen	$n_S = 76$
Dauer OFDM-Symbol	$T_S = 1.000 \text{ ms}$
Dauer Schutzintervall	$T_G = 0.246 \text{ ms}$
Dauer Nullsymbol	$T_N = 1.297 \text{ ms}$
OFDM-Subträgerabstand	$1/T_S = 1 \text{ kHz}$
OFDM-Subträger aktiv	$n_K = 1536$
Gesamtbandbreite HF-Kanal	$B_{HF} = 1.75 \text{ MHz}$

Tabelle 4.1: Charakteristika des DAB-Systems im sogenannten Mode I.

den Empfang auf stabile und zugleich einfache Weise zu erlauben, wird differentiell encodiertes Mapping (nach der $\pi/4$ -Shift DQPSK-Methode) verwendet. Es bietet die Möglichkeit zur inkohärenten Demodulation, das heißt am Empfänger kann auf den Einsatz vergleichsweise ressourcenaufwändiger Algorithmen zur Kanalschätzung und einer folgenden Entzerrung verzichtet werden. Die Zuordnung von Datenbits zu OFDM-Unterträgern erfolgt nicht in sequentieller Weise, sondern gemäß einer im Standard festgelegten pseudozufälligen Sequenz. Im Hinblick auf sowohl konstruktiv als auch destruktiv wirkende Einflüsse des Frequenzgangs des Übertragungskanals ist dann die Übertragungsgüte im logischen Datenstrom benachbarter Datenbits im Mittel ähnlich. Dies wird als Frequenzinterleaving bezeichnet. Die einzelnen OFDM-Symbole sind zu logischen Rahmen (engl. Frames) gruppiert. DAB ist ausgelegt auf eine Wellenformsynthese mit einem Grundtakt⁴ zur Zeitdiskretisierung von $f_s=2048 \text{ kSa/s}$. Hiermit kann die analoge Nutzsignalbandbreite von 1.537 MHz durch Abtastprozesse mit ausreichend Spielraum für die Implementierung von Rekonstruktionsfilterflanken gut dargestellt werden. Tabelle 4.1 fasst die wichtigsten Parameter der physikalischen Schicht von DAB in Mode I zusammen.

Logische Multiplex-Organisation

Auf der physikalischen Bitübertragungsschicht der OFDM-Wellenform setzt ein Multiplexmechanismus zur flexiblen Bildung logischer Datensubkanäle auf. Für jeden Multiplex ist die Existenz des Fast Information Channels (FIC) zwingend. Er ist primär dazu bestimmt, die Multiplex Configuration Information (MCI) zu tragen. Die Angaben des MCI werden benötigt, um die weiteren im Multiplexsignal enthaltenen Subdienste, beispielsweise unterschiedliche Musik- und Datendienste, dem Nutzer als vorhanden zu signalisieren als auch um diese im Empfänger aus dem gemeinsamen Multiplexdatenstrom gezielt separieren zu können. Daneben können im FIC in geringem Umfang weitere Datensätze anwendungsspezifischer Art als sogenannte Fast Information Data (FID) gesendet werden. Den eigentlichen Strom der Nutzdaten trägt der Main Service Channel (MSC). Dieser synchrone Datenkanal ist flexibel in bis zu 64 isochrome Subchannels konfigurierbar. Die Subchannels

⁴Bei Anwendung komplexwertiger Samples.

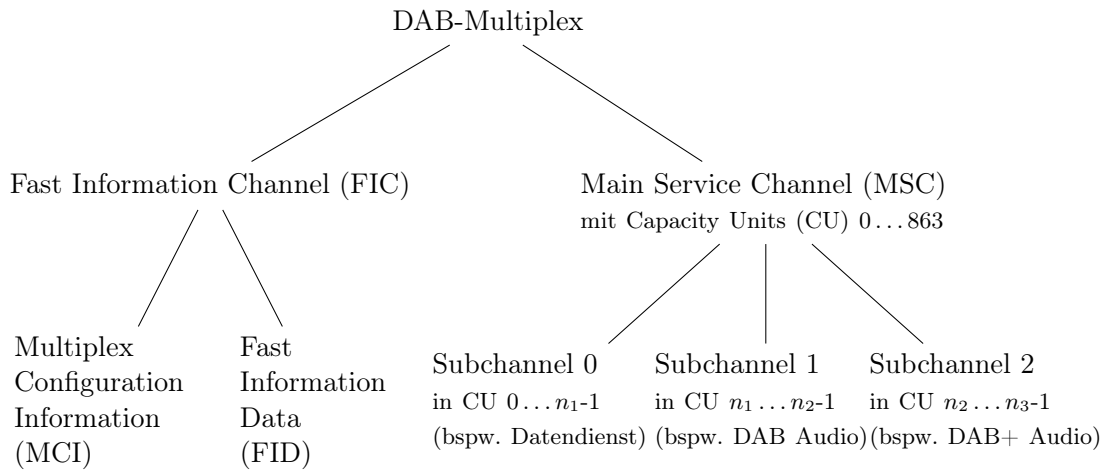


Abbildung 4.1: Multiplexhierarchie des DAB-Datenstroms mit beispielhaft drei Nutzdatenservices unterschiedlichen Formats.

transportieren Audiodatenströme nach MPEG-1 Layer 2 beziehungsweise MPEG AAC im Fall von DAB- und DAB Plus-Hörfunk beziehungsweise einen audiovisuellen MPEG-Transportstrom bei DMB. Auch können für Drittanwendungen wie Datendienste Rohdaten übertragen werden. Für Datendienste ohne feste Datenratenanforderung ist es möglich, diese in gemeinsame Packet Mode-Subchannels zu bündeln. Abbildung 4.1 bildet die Multiplexhierarchie beispielhaft ab.

Organisation des Übertragungsrahmens

Die logische Datenstromorganisation wird in Mode I auf ein 76 OFDM-Symbole zusammenfassendes Rahmenschema mit einer Periodendauer von 96 ms abgebildet (siehe Abbildung 4.2). Um die Empfängersynchronisation zu erleichtern und den gewünschten Gesamtzeitablauf zu erreichen, welcher sich an einer Ableitung aus der studiotypischen Audioabtastrate von 48 kHz orientiert, wurde eine Nullsymbol genannte Pause zwischen jeweils zwei DAB-Rahmen eingeführt. In dieser Phase wird zur Erkennung des Rahmenbeginns die Ausgangsleistung des Senders stark reduziert. Der Beginn eines jeden Rahmens wird als Synchronization Channel (SC) bezeichnet. Zu ihm werden das bereits erwähnte Nullsymbol zur Kennung des Rahmenbeginns sowie das erste tatsächliche OFDM-Symbol des Rahmens gezählt. Dieses Symbol mit der Bezeichnung Phase Reference Symbol (PRS) ist stets eine feste Sequenz. Es wird als Bezugsphase der OFDM-Unterträger für die differentielle Demodulation des verwendeten DQPSK-Schemas benötigt. Daneben bietet es sich als Trainingssequenz während der initialen Signalakquisition an. Nutzdaten im eigentlichen

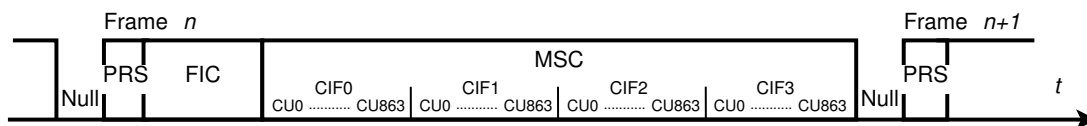


Abbildung 4.2: Zeitliches Schema der Rahmenstruktur in DAB.

4 Entwicklung einer effizienten Signalverarbeitungskette für DAB-Empfang

Sinne enthält der Synchronisation Channel nicht⁵. Die nächsten drei OFDM-Symbole formen den FIC. Der FIC hat eine feste Datenrate von 96 kbit/s uncodiert beziehungsweise 32 kbit/s fehlerschutzcodiert. Im Anschluss folgt der MSC. Dieser ist bei Mode I gleichmäßig in vier kleinere Common Interleaved Frames (CIF), je 18 OFDM-Symbole und 24 ms äquivalenter Dauer, aufgeteilt. Die Datennutzlast eines CIF wird weiter unterteilt in Capacity Units (CU) zu je 64 bit, um Multiplex-Kapazität akkurat und eindeutig zu den einzelnen Teilservices des Ensembles allokiert zu können. Die Zuweisung dieser Datenkapazitäten im Multiplex ist hierdurch fest definiert hinsichtlich der Rahmenposition sowie damit der zeitlichen Position. Der gesamte MSC stellt uncodiert 2.2 Mbit/s Kapazität bereit. Dieser Wert reduziert sich entsprechend des durch die Coderate des ausgewählten Fehlerchutzschemas bestimmten Anteils an hinzugefügter Redundanz.

Zeitliches Interleaving

Für sämtliche Daten des MSC kommt ein Zeitinterleavingmechanismus zur Anwendung. Hierdurch kann auf Kosten einer Latenz die Wirksamkeit der Vorwärtsfehlerkorrektur verbessert werden. Die einzelnen Datenbits werden in Abhängigkeit ihres Positionsindex verzögert und so auf 16 unterschiedliche CIFs verteilt. Das über 16 CIFs durchgeführte Interleaving entspricht einer zeitlichen Interleavingtiefe von 384 ms und somit einer solchen zusätzlichen Prozessierlatenz in Sender und Empfänger. Im Gegensatz zum MSC wird für die Daten des FIC kein zeitliches Interleaving angewendet.

Fehlerkorrektur-Codierung

Jedes 24 ms Fragment eines jeden Dienstes des DAB-Multiplexes formt ein in sich abgeschlossenes, unabhängiges Codewort des Fehlerchutzdecoders. Die Kanalcodierung zum Zwecke der Vorwärtsfehlerkorrektur basiert auf einem einheitlichen Faltungscodiercode der Rate $R_m = 1/4$ (Abbildung 4.3). Aus dem Datenstrom dieses Muttercodes werden im Betrieb durch sogenannte Punktierung weitere Codes höherer Rate abgeleitet. Der seriellen Aus-

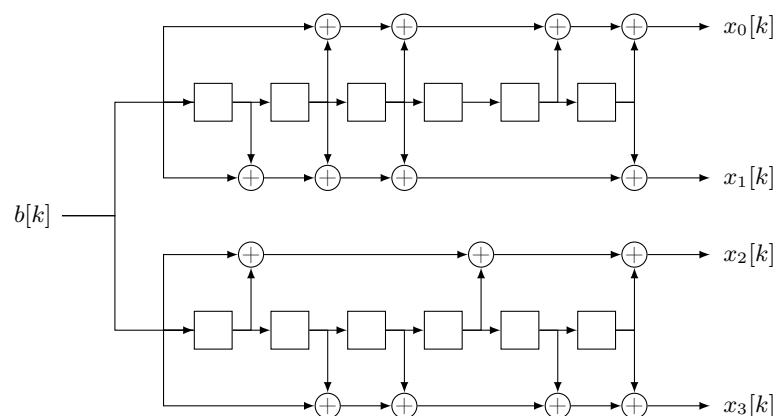


Abbildung 4.3: Blockschaltbild des DAB-Faltungscoders mit der Coderate $R_m=1/4$.

⁵Jedoch besteht nach Standard die Option, eine sendeanlagenspezifische Kennung (engl. Transmitter Identification Information, TII) in die Periode eines jeden zweiten Nullsymbols einzufügen.

gangsfolge des Encoders wird dabei je nach wechselnder momentaner Konfiguration ein Tupeln gemäß $(x_0[k])$, $(x_0[k], x_1[k])$, $(x_0[k], x_1[k], x_2[k])$ oder $(x_0[k], x_1[k], x_2[k], x_3[k])$ hinzugefügt. So wird je nach gewünschter Übertragungsrobustheit pro Einzelservice des Multiplexes eine ratenadaptive Einstellung beliebiger sogenannter Protection Levels mit Raten $R_c > R_m$ möglich. Der Protection Level und somit die Coderate kann selbst innerhalb eines einzelnen Codewortes geändert werden. Das gibt die Möglichkeit, wichtigen Datenbits erhöhten Schutz zu geben. Dies wird beispielsweise für Prüfsummen und die Skalenfaktoren des MPEG-1 Quellencodierverfahrens genutzt, welche bei einer Fehlübertragung direkt zu einem vollständigen Versagen des Audioquellendecoders führen würden.

Energy Dispersal

Ein Scrambling-Mechanismus wird verwendet, um auch im Falle der Übertragung einer Folge aus vielen gleichen Bits oder Bytes eine Datenfolge ohne Gleichanteile oder kurzen periodischen Sequenzen am Eingang des Modulators zu erzeugen. So soll sichergestellt werden, dass die spektrale Leistungsdichte des resultierenden Sendesignals gleichmäßig verteilt und Energiespitzen im Leistungsdichtespektrum verhindert werden (engl. Energy Dispersal). Konkret werden die Nutzdaten des FIC und des MSC vor dem Eingang des Faltungscoders beziehungsweise nach dem Ausgang des Faltungsdecoders wie in Abbildung 4.4 gezeigt einer Exklusiv Oder-Verknüpfung mit einer fixen binären Pseudozufallssequenz (engl. Pseudo Random Binary Sequence, PRBS) unterzogen.

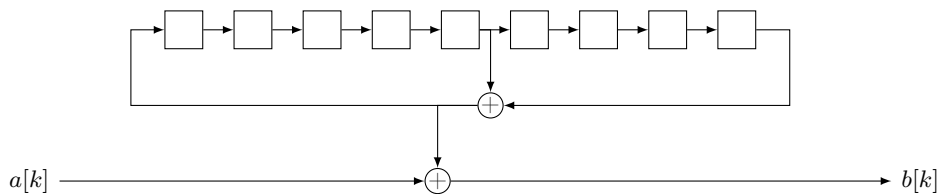


Abbildung 4.4: Der Mechanismus des Energy Dispersal erfolgt durch Verknüpfen der Nutzdaten mit einer von einem linear rückgekoppelten Schieberegister erzeugten PRBS-Folge.

Format der Nutzdaten

Die aus dem FIC gewonnenen Daten bestehen aus einzelnen Fast Information Groups (FIG). Diese Nachrichtenpakete stellen ein DAB-spezifisches Signalisierungsprotokoll dar. Seine Inhalte müssen interpretiert werden, bevor Nutzdaten gezielt aus dem MSC extrahiert werden können. Der MSC-Multiplex selbst liefert am Empfänger bei jedem Eintreffen eines CIF, im Mittel alle 24 ms, für sämtliche enthaltenen Services ein neues Nutzdatensegment. Im Falle von Audiodaten im klassischen DAB-System stellt jedes Segment einen eigenständigen darstellbaren MPEG-1 Audioframe dar und kann direkt an den entsprechenden Audiodecoder übergeben werden, welcher aus dem Datensatz 24 ms PCM-Audio rekonstruieren kann. Bei DAB Plus und DMB erfolgt vor der Decodierung eine Repaketierung mehrerer Segmente zu umfassenderen Superframes.

4.1.2 Mathematische Formulierung des DAB-Signals

Für die Wellenform des OFDM-Signals wird in diesem Abschnitt eine formale mathematische Beschreibung gegeben, um im darauffolgenden Teil des Kapitels die Algorithmen des Empfängersystems ableiten zu können.

Das DAB-Sendesignal

Es liegt eine reine Phasenmodulation der OFDM-Unterträger vor. Entsprechend können die Konstellationsphasoren $\underline{S}_l[K]$ der QPSK-Subträger K eines jeden OFDM-Symbols l in Anlehnung an den Standard [2] wie folgt in Abhängigkeit der datentragenden Phasenlage $f(l, K)$ dargestellt werden (in Mode I gilt $n_K=1536$, $l \in \{1, 2, \dots, 76\}$):

$$\underline{S}_l[K] = \begin{cases} e^{j\frac{\pi}{2}f(l,K)} & \text{für } K \in \{-\frac{n_K}{2}, -\frac{n_K}{2} + 1, \dots, -1, 1, \dots, \frac{n_K}{2} - 1, \frac{n_K}{2}\} \\ 0 & \text{sonst} \end{cases} \quad (4.1)$$

Die Zeitbereichsrepräsentation $\underline{s}_l[k]$ des OFDM-Symbols Nummer l ist dann gegeben durch die Korrespondenz der inversen diskreten Fourier-Transformation mit der Transformationslänge $N=L_S$ (für Mode I gilt $L_S=2048$ bei einem Abtasttakt $f_s=2048$ kSa/s).

$$\underline{s}_l[k] = \mathfrak{F}^{-1}\{\underline{S}_l[K]\} \quad \text{mit } k = 1, 2, \dots, L_S \quad (4.2)$$

Der Hauptbeitrag des einzelnen OFDM-Symbols zum Signalelement lässt sich in einer direkten Äquivalenz auch beschreiben als

$$\underline{s}_l[k] = \sum_{i=1}^{L_S} \underline{s}_l[i] \delta[k - i] \quad (4.3)$$

Ein zyklischer Präfix ergänzt als Schutzintervall das Signalelement des OFDM-Symbols. Die vorhergehende Teilwiederholung der hinteren L_G Abtastwerte (es gilt $L_G=504$ Sa für Mode I) eines jeden Symbols kann dargestellt werden wie folgt.

$$\underline{s}_l^{\text{CP}}[k] = \sum_{i=(-L_G)}^0 \underline{s}_l[i + L_S] \delta[k - i] \quad (4.4)$$

oder durch Substitution auch als

$$\underline{s}_l^{\text{CP}}[k + L_S] = \sum_{i=(L_S-L_G)}^{L_S} \underline{s}_l[i] \delta[k + L_S - i] \quad (4.5)$$

Das Gesamtsignalelement des OFDM-Symbols $\tilde{s}_l[k]$ inklusive zyklischem Präfix folgt zu

$$\tilde{s}_l[k] = \underline{s}_l^{\text{CP}}[k + L_S] + \underline{s}_l[k] \quad (4.6)$$

$$= \sum_{i=(L_S-L_G)}^{L_S} \underline{s}_l[i] \delta[k + L_S - i] + \sum_{i=1}^{L_S} \underline{s}_l[i] \delta[k - i] \quad (4.7)$$

Die Sequenz des Sendesignals $\underline{s}[k]$ sämtlicher OFDM-Symbole l eines Rahmens ergibt sich dann, zur Einfachheit unter Vernachlässigung des Auftretens des DAB-Nullsymbols, mit der Gesamtsymbollänge L_{S+G} (für Mode I mit $L_{S+G}=L_S+L_G=2552$) zu

$$\underline{s}[k] = \sum_l \tilde{s}_l[k - l \cdot L_{S+G}] \quad (4.8)$$

Das DAB-Empfangssignal

Den Empfänger erreicht das ursprüngliche Signal gefaltet mit einer dispersiv wirkenden Kanalimpulsantwort $\underline{h}[k]$. Damit der Demodulator vor Intersymbolinterferenz durch das Schutzintervall sicher geschützt wird, muss im Funkszenario stets für die Einflusslänge der Kanalimpulsantwort $L_h \leq T_\Delta$ gelten. Bedingt durch eine vorhandene Frequenzabweichung Δf_c von Sende- und Empfangsfrequenzoszillatoren⁶ tritt des Weiteren eine Verschiebung im Frequenzbereich auf. Der Frequenzversatz Δf_c korrespondiert nach der Abtastung mit einer diskreten Frequenz $\Delta f_c/f_s$. Zusätzlich ist das empfangene Signal $\hat{\underline{s}}[k]$ gestört durch additives weißes Gauß'sches Rauschen $\underline{n}[k]$.

$$\hat{\underline{s}}[k] = \left(\underline{s}[k] * \underline{h}[k] \right) e^{j2\pi(\Delta f_c/f_s)k} + \underline{n}[k] \quad (4.9)$$

4.2 Entwicklung der Empfangsalgorithmen

Das Set aus Algorithmen eines Empfängers beschränkt sich nicht auf eine simple Inversion der Operationen der Senderkette. Im Speziellen sind durch zusätzliche Funktionsblöcke sämtliche Einflüsse zu schätzen und zu kompensieren, welche auf das Signal während der Übertragung wirken. So lassen sich die nötigen verschiedenen Algorithmen in drei Gruppen einteilen:

- Die initiale Erstakquisition des Signals mit Schätzung sämtlicher Parameter.
- Die fortgesetzte Verfolgung (engl. Tracking) der Signalparameter.
- Die eigentliche, kohärent parametrisierte Datendemodulation des Eingangssignals.

4.2.1 Synchronisation: Signalakquisition

Bevor die eigentliche Demodulation erstmalig beginnen kann, müssen die Zeitbasen des Empfängers auf diejenigen des Senders neu synchronisiert werden. Eine exakte Bestimmung der absoluten Zeit (Synchronisierung auf das Schema des DAB-Rahmenaufbaus) sowie auch die Neuschätzung der Taktfrequenzen im Empfänger ist nötig. Sämtliche frequenzbestimmenden Teile der realen Empfängerschaltung weisen unvermeidlich eine mehr oder weniger hohe Ungenauigkeit und Drift auf. Folglich werden immer unbekannte Differenzen zwischen den internen Takten des Senders und des Empfängers existieren, welche stets neu geschätzt sowie korrigiert werden müssen. Es muss weiterhin während der initialen Synchronisationsphase, auch genannt Akquisition, die Signalverarbeitung des OFDM-Demodulators

⁶Schnell zeitvariante Einflüsse durch Doppler-Frequenzverschiebung aufgrund hoher Mobilität sollen an dieser Stelle vernachlässigt werden.

4 Entwicklung einer effizienten Signalverarbeitungskette für DAB-Empfang

hinsichtlich der im jeweiligen Fall eingesetzten Submodi des Übertragungsverfahrens parametrisiert werden. Die folgenden Größen müssen im Falle des DAB-Empfängers bestimmt beziehungsweise geschätzt werden:

- Der DAB-Übertragungsmodus Mode I, II, III oder IV.
- Der Zeitpunkt t_{Frame} , an dem ein neuer, vollständiger DAB-Rahmen beginnt.
- Die Abweichung Δf_c der Frequenz des lokalen Taktgebers von derjenigen des Oszillators am Sender.

Das Tuning auf einen Nutzdienst innerhalb des Multiplexverbunds, das heißt die Einstellung des Demultiplexers/Time Slicings zu dessen Extraktion aus dem Gesamtstrom sowie die zugehörige Parametrisierung des Fehlerschutzdecoders auf Grundlage von vorliegenden MCI-Information des FIC, gehören nicht zu den Aufgaben der Signalakquisition.

Detektion des DAB-Mode

Zu Beginn wird im digitalen Basisband das Vorhandensein eines validen DAB-Signals geprüft und zugleich der zugehörige DAB-Mode bestimmt. Es existieren prinzipiell zwei Signalcharakteristiken, welche für diese Prüfung genutzt werden können:

- Auswertung der Mode-spezifischen Rahmenlänge L_F durch Messen der Dauer zwischen zwei aufeinanderfolgenden Nullsymbolen.
- Schätzung des verwendeten OFDM Schemas hinsichtlich Symbollänge L_S durch Autokorrelationsverfahren aufgrund der durch den zyklischen Präfix des Schutzintervalls eingeführten Signalselbstähnlichkeit.

Beim naheliegenden ersten Verfahren entsteht aufgrund gleicher Rahmenlänge eine Mehrdeutigkeit zwischen dem Mode II und Mode III des DAB-Systems (siehe Tabelle 4.2). Dies wäre jedoch in der Praxis nicht kritisch, da Mode III zur Direktübertragung via Satellit bislang keine Anwendung findet und dessen Präsenz im Zweifel durch Wahl des HF-Frontends beziehungsweise der Übertragungsfrequenz ohnehin gegeben wäre. Trotzdem wurde für die Schätzung des OFDM-Schemas entschieden, da das Autokorrelationsverfahren die Möglichkeit gibt, Rauschleistung und Interferenzsignale durch zeitliche Integration zu unterdrücken, und somit das Potential für eine weitaus robustere Detektion bietet. Wir nutzen die Statistik des empfangenen DAB-Signals $\hat{s}[k]$. Die Methode der Kreuzkorrelation kann bekanntlich genutzt werden um zwischen zwei separaten Signalen mit verwandten Anteilen deren gegenseitige Verzögerungszeit zu bestimmen, der spezielle Fall der Autokorrelation, um die Verzögerungszeit bezüglich Selbstähnlichkeiten innerhalb eines Signals zu bestimmen.

$$\underline{R}_{\hat{s}\hat{s}}[\tau] = \sum_k \hat{s}[k] \hat{s}^*[k - \tau] \quad (4.10)$$

Im abgetasteten Empfangssignal ist Selbstähnlichkeit mit der Distanz $\tau = L_S$ durch den zyklischen Präfix eines jeden OFDM-Symbols gegeben (Gleichung 4.6). Es ist bekannt, dass

4.2 Entwicklung der Empfangsalgorithmen

DAB-Mode	Symbollänge L_S	Rahmenlänge L_F
I	2048 Sa	196608 Sa
II	512 Sa	49152 Sa
III	256 Sa	49152 Sa
IV	1024 Sa	98304 Sa

Tabelle 4.2: Individuelle Dauer der Nutzperiode eines OFDM-Symbols sowie des Rahmens gemäß [2] in den vier DAB-Übertragungsmodi, angewendet auf die systemtypische Abtastfrequenz $f_s=2048$ kSa/s.

nur Signalanteile, die statistische Ähnlichkeit aufweisen, im Mittel einen Beitrag zur Korrelation liefern (bezeichnet als $\underline{R}'_{\hat{s}\hat{s}}[L_S]$), wohingegen Signalanteile ohne Bezug zueinander im Mittel nicht beitragen (bezeichnet als $\underline{R}''_{\hat{s}\hat{s}}[L_S]$).

$$\begin{aligned}\underline{R}_{\hat{s}\hat{s}}[L_S] &= \underline{R}'_{\hat{s}\hat{s}}[L_S] + \underline{R}''_{\hat{s}\hat{s}}[L_S] \\ &= \underline{R}'_{\hat{s}\hat{s}}[L_S] + 0\end{aligned}\quad (4.11)$$

Bei Betrachtung allein derjenigen Signalstücke der Symbolsequenz $\hat{s}[k]$, für die eine entsprechende Kopie existiert und welche somit in der Lage sind, Korrelationsbeiträge zu generieren, verbleiben selbstverständlich nur diejenigen des zyklischen Präfix. Es ergibt sich am Empfänger nach Gleichung 4.8 in Verbindung mit 4.9

$$\begin{aligned}\hat{s}'[k] &= \sum_l \left(\hat{s}_l^{\text{CP}}[k + L_S] + \hat{s}_l^{\text{CP}}[k] \right) e^{j2\pi k(\Delta f_c/f_s)} * \underline{h}[k] \\ &= \sum_l \left(e^{j2\pi(-L_S)(\Delta f_c/f_s)} \cdot \hat{s}_l^{\text{CP}}[k + L_S] e^{j2\pi(k+L_S)(\Delta f_c/f_s)} + \hat{s}_l^{\text{CP}}[k] e^{j2\pi k(\Delta f_c/f_s)} \right) * \underline{h}[k] \\ &= e^{j2\pi(-L_S)(\Delta f_c/f_s)} \cdot \underline{g}[k + L_S] + \underline{g}[k]\end{aligned}\quad (4.12)$$

wobei Terme entsprechend der folgenden Definition des Hilfsausdruckes $g[k]$ zusammengefasst werden:

$$\underline{g}[k] = \sum_l \left(\hat{s}_l^{\text{CP}}[k] e^{j2\pi k(\Delta f_c/f_s)} \right) * \underline{h}[k]\quad (4.13)$$

Aus Gleichung 4.12 ist offensichtlich, dass ein starker Autokorrelationswert in noch nicht näher untersuchter phasengedrehter Weise durch die Signalkopie des zyklischen Präfix bei Auswertung mit der zugehörigen Symboldauer $\tau=L_S$ auftreten muss. Gemäß den individuellen Längen L_S der OFDM-Symbole in den vier DAB-Betriebsmodi (siehe Tabelle 4.2), wird die Autokorrelationsfunktion des Basisbandsignals $\hat{s}[k]$ im entwickelten Empfänger an den vier konkreten Stellen $\tau \in \{L_{S,\text{Mode I}}, L_{S,\text{Mode II}}, L_{S,\text{Mode III}}, L_{S,\text{Mode IV}}\}$ ausgewertet. Dabei müsste sich nach Gleichung 4.10 in Verbindung mit Gleichung 4.13 im zutreffenden

4 Entwicklung einer effizienten Signalverarbeitungskette für DAB-Empfang

Mode folgendes Ergebnis einstellen:

$$\begin{aligned} \underline{R}_{\hat{s}\hat{s}}[L_S] &= \sum_k \hat{s}[k] \hat{s}^*[k - L_S] \\ &= \sum_k \left(e^{j2\pi(-L_S)(\Delta f_c/f_s)} \cdot \underline{g}[k + L_S] + \underline{g}[k] \right) \left(e^{-j2\pi(-L_S)(\Delta f_c/f_s)} \cdot \underline{g}^*[k] + \underline{g}^*[k - L_S] \right) \end{aligned} \quad (4.14)$$

Nach dem Prinzip aus Gleichung 4.11 verbleibt nach Herausmittlung unkorrelierter Anteile über die Zeit bei der Berechnung der folgende Konvergenzwert:

$$\underline{R}_{\hat{s}\hat{s}}[L_S] = e^{j2\pi L_S(\Delta f_c/f_s)} \sum_k \underline{g}[k] \underline{g}^*[k] = e^{j2\pi L_S(\Delta f_c/f_s)} \sum_k \|\underline{g}[k]\|^2 \quad (4.15)$$

Ein valides DAB-Signal ist detektiert, wenn ein Korrelationsbetrag $\|\underline{R}_{\hat{s}\hat{s}}[\tau]\| > 0$ verbleibt. Im praktischen Betrieb ist dazu eine definierte Energieschwelle zu überschreiten. Im Zweifel wird für denjenigen Mode mit dem betragsmäßig größten Korrelationsergebnis entschieden.

Detektion des Trägerfrequenzoffsets: Schätzung des Feinfrequenzfehlers

Ferner muss am Empfänger die Abweichung des Lokaloszillators von der Trägerfrequenz exakt geschätzt und korrigiert werden. Speziell das Schema der OFDM-Übertragung ist sehr empfindlich gegenüber Frequenzabweichungen: Aufgrund der Störung der Orthogonalität zwischen den einzelnen Subträgern entstehende Interferenzen (engl. Inter Carrier Interference, ICI) führen zur rapiden Degradation der Empfangsgüte. In der vorliegenden Umsetzung wird ein Synchronisationsverfahren ähnlich Sheu et al. [85], Abschnitt 2.1 und 2.3, genutzt, jedoch das Korrelationsprinzip nicht auf den Anteil des Phasenwinkelargumentes, sondern ohne Argumentbildung auf die direkt vorliegende komplexwertige Zahlenfolge angewendet. Quarzoszillatoren aus dem Consumer Electronic-Bereich, wie sie auch im später genutzten Radiofrontend Verwendung finden, weisen typisch eine Abweichung von ± 100 ppm über den gesamten zu erwartenden Temperaturbereich und Lebenszeitraum auf. Für die bei DAB genutzten Trägerfrequenzen f_c gilt circa $f_c < 240$ MHz für das VHF-Band III und $f_c < 1.5$ GHz für das L-Band. Ausgehend von diesen Werten muss eine Frequenzabweichung zwischen Sende- und Empfangsfrequenzoszillator von bis zu $|\Delta f_c| < 24$ kHz und entsprechend $|\Delta f_c| < 150$ kHz erwartet sowie durch die Signalverarbeitung bedient werden können. Wieder kann ein Verfahren basierend auf Autokorrelation hinsichtlich des zyklischen Präfixes des Schutzintervalls zur Schätzung herangezogen werden. Das Argument des bereits berechneten komplexen Wertes $\underline{R}_{\hat{s}\hat{s}}[L_S]$ zeigt nach Gleichung 4.15 in der Ebene des komplexen Basisbandes diejenige Phasendrehung, welche sich durch die Frequenzfehl-anpassung im Zeitraum von L_S Abtastwerten ausgebildet hat.

$$\underline{R}_{\hat{s}\hat{s}}[L_S] = e^{j2\pi L_S(\Delta f_c/f_s)} \cdot C \quad (4.16)$$

Es gilt also der (leider mehrdeutige) Zusammenhang

$$\arg\{\underline{R}_{\hat{s}\hat{s}}[L_S]\} = (2\pi(\Delta f_c/f_s)L_S) \pmod{2\pi} \quad (4.17)$$

Der Ausdruck $1/L_S$ beschreibt beim Abtasttakt f_s eine diskrete Frequenz, welche mit dem Abstand der Unterträger im OFDM-System korrespondiert. Unter diesem Gedanken

4.2 Entwicklung der Empfangsalgorithmen

stellt $\Delta F_c = (\Delta f_c / f_s) / (1/L_S)$ eine auf den Unterträgerabstand normierte diskrete Frequenzabweichung des OFDM-Empfangssignals dar.

$$\frac{1}{2\pi} \arg\{\underline{R}_{\hat{s}\hat{s}}[L_S]\} = \frac{\Delta f_c / f_s}{1/L_S} \pmod{1} = \Delta F_c \pmod{1} \quad (4.18)$$

Der Ausdruck beschreibt eine Mehrdeutigkeit bezüglich des ganzzahligen Anteils $\Delta F_x \in \mathbb{Z}$ dieses Frequenzfehlers.

$$\Delta F_c = \left(\Delta F_x + \frac{1}{2\pi} \arg\{\underline{R}_{\hat{s}\hat{s}}[L_S]\} \right) \quad (4.19)$$

Mit Hilfe des Wertes der Autokorrelation lässt sich also der Nachkommaanteil der auf den Unterträgerabstand normierten Frequenzabweichung ΔF_c schätzen. Die Korrektur dieses Feinfrequenzfehlers ermöglicht die Durchführung der OFDM Demodulation im exakten Raster der Unterträger, somit ohne ICI-Störung und somit ohne Demodulationsverlust. Der noch verbleibende Grobfrequenzfehler führt zu einer ganzzahligen unbekanntenen Indexverschiebung der einzelnen OFDM-Unterträger.

Algorithmus 1: Pseudocode zur Akquisition des Übertragungsmodus und des Feinfrequenzfehleranteils.

```

//  $\hat{s}[\cdot]$ : Basisbandsamples des Empfangssignals

// Initialisierung
 $\underline{R}_{\max} = 0$ 
 $m_{\text{det}} = 0$ 
// Teil I: Für alle DAB-Modes, suche Signal
for  $m = 1 \dots 4$  do
     $\underline{R} = 0$ 
    // Autokorrelation gemäß Symbollänge  $L_{S,m}$  in Modus  $m$ 
    for  $k = 0 \dots L_F$  do
        |  $\underline{R} = \underline{R} + \hat{s}[k]\hat{s}^*[k - L_{S,m}]$ 
    end
    // Entscheidung gemäß Schwellenwert und bisherigem Maximum
    if  $\|\underline{R}\| > \|\underline{R}_{\text{threshold}}\|$  then
        | if  $\|\underline{R}\| > \|\underline{R}_{\max}\|$  then
            | |  $\underline{R}_{\max} = \underline{R}$ 
            | |  $m_{\text{det}} = m$ 
        | end
    end
end
// Teil II: Feinfrequenzfehleranteilbestimmung aus Autokorrelation
 $\Delta F_c = \arg\{\underline{R}_{\max}\} / 2\pi$ 

//  $m_{\text{det}}$ : Detektierter DAB-Mode, oder 0 falls kein DAB-Signal
//  $\Delta F_c$ : Feinfrequenzfehleranteil, normiert auf Trägerabstand

```

Zeitsynchronisation

Die Aufgabe der Zeitsynchronisation liegt zum Einen im Auffinden der zeitlichen Position des DAB-Rahmenbeginns k_{Start} und somit in der Identifikation der einzelnen Ordnungsnummern i der OFDM-Symbole gemäß des DAB-Rahmenschemas. Neben dieser vergleichsweise groben Synchronisation besteht die zweite Aufgabe in der Selektion von L_S konkret zur Symboldemodulation auszuwertenden Abtastwerten aus L_{S+G} möglichen Abtastwerten jedes Symbolintervalls des kontinuierlichen Basisbanddatenstroms $\hat{\underline{g}}[k]$. Für die FFT-Transformation zur OFDM-Demodulation des i -ten Symbols sind ab Rahmenbeginn k_{Start} Abtastwerte wie folgt auszuwählen:

$$\hat{\underline{g}}_i[k] = \hat{\underline{g}}[k_{\text{Start}} + i \cdot (L_{S+G}) + k] \quad (4.20)$$

Die zeitliche Positionierung hinsichtlich des zu verwendenden Schutzintervallanteils der Länge L_G wird bei OFDM als die Wahl des FFT-Fensters bezeichnet. Es besteht dabei die Forderung einer Maximierung der erwünschten Eigensignalleistung $P_{\hat{\underline{g}}_i,i}$ des Symbols i im FFT-Fenster zu der dort durch Intersymbolinterferenz anderer Symbole $j \neq i$ via Dispersion der Kanalimpulsantwort $\underline{h}[k]$ eingebrachten Störleistungen $P_{\hat{\underline{g}}_i,j}$:

$$P_{\hat{\underline{g}}_i,i} / \sum_{j \neq i} P_{\hat{\underline{g}}_i,j} \rightarrow \max_{k_{\text{Start}}} \quad (4.21)$$

Sowohl die Rahmensynchronisation als auch die optimale FFT-Fensterpositionierung lassen sich effizient aus der zeitlichen Lokalisierung des DAB-Nullsymbols ableiten, welches aufeinanderfolgende DAB-Rahmen separiert. Die instantane Leistung des Signals am Sender fällt in diesem Zeitraum, dessen zeitliche Länge in Abtastwerten mit L_N gegeben ist, auf ein lokales Minimum ab (Abbildung 4.5). Um eine Aussage über die Positionierung des FFT-Fensters abzuleiten lässt sich die Idee nutzen, dass in Szenarien mit Mehrwegeausbreitung durch die Signaldispersion das Nullsymbol der Länge L_N wie jedes normale Symbol durch ISI beeinträchtigt wird. Es werden nun durch Dispersion Energieanteile in die eigentlich signalenergiefreie Nullsymbolperiode eingebracht (Abbildung 4.6a). Die Forderung nach optimaler Wahl des FFT-Fensters, beschrieben durch Minimierung des Wertes der störenden ISI-Energieanteile zum Wert der Nutzsignalenergie analog zu Gleichung 4.21, ist für den Nullsymbolabschnitt folglich gleichbedeutend mit der Minimierung jeglichen Leistungseintrages in den Zeitschlitz $i = 0$ des im Optimalfall energiefreien Nullsymbols.

$$\sum_j P_{\hat{\underline{g}}_0,j} \rightarrow \min_{k_{\text{offset}}} \quad (4.22)$$

Gemäß der Einflusslänge der Dispersion des Übertragungskanals $L_{\underline{h}}$, welche im ungünstigsten durch den Empfänger gemäß Standard zu unterstützenden Fall bis zu L_G Abtastwerte beeinflusst, schrumpft die Anzahl der Abtastwerte mit vollständig abgesenkter Signalenergie bis auf $L_N - L_G$ Werte (Abbildung 4.6b). Entsprechend wird es als vorteilhaft angesehen, den Integrationsbereich für die minimale Signalenergiesuche am Empfänger nicht auf die ursprüngliche Länge des Nullsymbols L_N , sondern auf die reduzierte Länge $L_N - L_G$ einzustellen. Die Suche nach einem solchen Bereich schließt somit jegliche ISI im Nullsymbol aus. Weil die nachfolgenden Nutzsymbole zeitlich gekoppelt sind, ist hiermit auch für diese

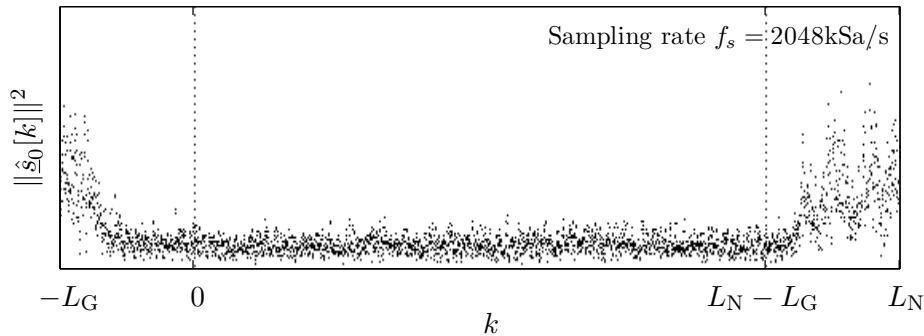
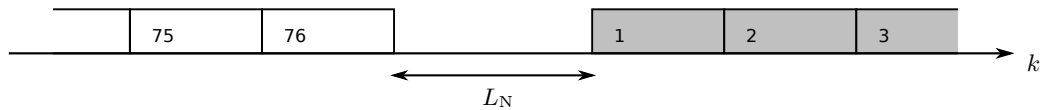
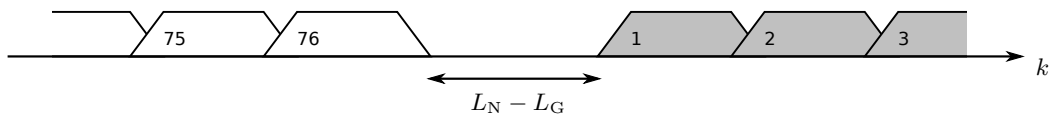


Abbildung 4.5: Gemessenes Empfangssignal $\hat{s}[k]$ im detektierten Bereich des DAB-Nullsymbols. Die reduzierte Signalleistung ist im Zeitbereichssignal deutlich zu erkennen. Zur linken Seite sind Teile des letzten Symbols des vorangegangenen DAB-Rahmens, zur rechten Seite der Beginn des PRS Symbols des nächsten Rahmens erkennbar.

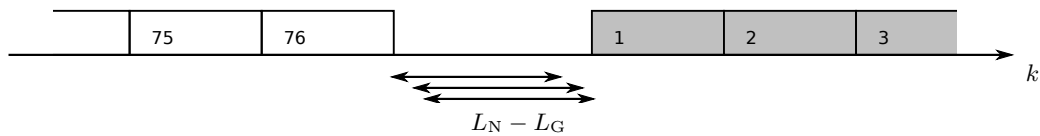
auf einfache Weise ISI-freie Demodulation garantiert. Im Falle ohne beziehungsweise mit wenig Dispersionseinflusslänge $L_h < L_G$ ergibt sich durch den verkürzten Integrationsbereich hinsichtlich der Position des Bereiches minimaler Leistung eine Mehrdeutigkeit (Abbildung 4.6c). Der erläuterte zeitliche Synchronisationsmechanismus hat somit in dieser Situation Freiheiten bei der Wahl einer von mehreren validen OFDM-Fensterpositionen und es ist ein Jitterverhalten des Suchalgorithmus zu beobachten. Jedoch ist dies ohne Auswirkung auf die nachrichtentechnische Funktion. Die Umsetzung einer Suche nach einem Bereich der Länge $L_N - L_G$ mit minimaler Signalleistung $P[k]$ und entsprechend der



(a) Ohne dispersive Kanalimpulsantwort, das heißt $\underline{h}[k] = \underline{C} \cdot \delta[k]$, ist die gesamte Periode des Nullsymbols der Länge L_N ohne Signalenergie.



(b) Im Falle einer Kanalimpulsantwort $\underline{h}[k]$ mit dispersivem Charakter (hier: Maximalszenario mit Dispersionseinflusslänge $L_h = L_G$) wird die Dauer des Bereiches ohne Signalenergie reduziert (hier: auf $L_N - L_G$).



(c) Im nicht-dispersiven Szenario resultiert eine Suche mit reduzierter Fensterlänge der Signalleistungsintegration in einer Mehrdeutigkeit für die Wahl des Fensters.

Abbildung 4.6: Schematische Darstellung der empfangenen OFDM-Symbole und Wahl der Länge des Nullsymbol-Fensters.

4 Entwicklung einer effizienten Signalverarbeitungskette für DAB-Empfang

Position $k=k_{\text{Start}}$ des Nullsymbols am Beginn eines DAB-Rahmens lässt sich formulieren als

$$P[k] = \sum_{i=0}^{L_N - L_G - 1} \|\hat{\underline{s}}[k+i]\|^2 \quad (4.23)$$

und

$$\hat{k}_{\text{Start}} = \arg \min_k \{P[k]\} \quad \text{mit } k \in \mathcal{K} \quad (4.24)$$

Der Suchbereich \mathcal{K} in der Sequenz des Basisbandsignals $\hat{\underline{s}}[k]$ muss etwas größer gewählt werden als die Länge eines kompletten DAB-Rahmens L_F , damit sichergestellt ist, dass mindestens ein Nullsymbol vollständig enthalten ist und detektiert werden kann. Für die effiziente Implementierung der Suche über alle k bietet sich eine rekursive Formeldarstellung an. Mit gleitendem Filterungsfenster (engl. Moving Average-Filter) kann Gleichung 4.23 überführt werden zu

$$P[k+1] = P[k] - \|\hat{\underline{s}}[k]\|^2 + \|\hat{\underline{s}}[k+L_N+L_G]\|^2 \quad (4.25)$$

Nach Gleichung 4.24 ist dieser Ausdruck für einen Bereich $k \in \mathcal{K}$ hinsichtlich k zu minimieren, absolute Werte $P[k]$ sind dann jedoch hierfür nicht von Interesse. Dementsprechend kann der initiale Wert der Rekursion $P[0]$ in der Suche unbestimmt bleiben und im Sinne einer starken Aufwandsreduzierung bei der Berechnung beliebig beziehungsweise zu Null gewählt werden.

Algorithmus 2: Akquisition des Rahmenbeginns

```

//  $\hat{\underline{s}}[\cdot]$ : Basisbandsamples des Empfangssignals

// Initialisierung der Moving Average-Filterung
P = 0;
Pmin = 0;
kmin = 0;
// Suche in Intervall um erwarteten Rahmenbeginn
for k = 0... (LF + LN) do
    // Berechnung der Signalleistung im Integrationsfenster
    P = P -  $\|\hat{\underline{s}}[k]\|^2$ ;
    P = P +  $\|\hat{\underline{s}}[k+L_N-L_G]\|^2$ ;
    // Falls Leistungsminimum: Position speichern
    if P < Pmin then
        | Pmin = P;
        | kmin = k;
    end
end
// Rahmenbeginn liegt an gefundenem Leistungsminimum
kStart = kmin

// kStart: Rahmenstartposition im Basisbandstrom

```

Detektion des Trägerfrequenzoffsets: Schätzung des Grobfrequenzfehlers

Nun ist eine zeitliche Synchronisation auf die Abtastwerte des DAB-Rahmens hergestellt. Die Folge der Abtastwerte des ersten OFDM-Symbols mit Symbolindex $i=0$, des PRS-Symbols $\hat{s}_0[k]$, kann im empfangenen Basisbandsignal lokalisiert und, da die originale Sequenz des PRS-Symbols dem Empfänger genau bekannt ist, zur weiteren Synchronisationsanalyse genutzt werden. Aufgrund der Constant Amplitude Zero Autocorrelation (CAZAC)-Eigenschaft [84] des bei DAB gewählten Pilot-Symbols muss sich bei Faltung mit dem konjugiert-komplexen Spiegelbild, das heißt als Autokorrelationsfunktion, als Folge näherungsweise allein ein isolierter, diskreter Einheitspuls (Abbildung 4.7) ergeben.

$$R_{s_0 s_0}[k] = s_0[k] * s_0^*[-k] \approx \underline{C} \cdot \delta[k] \quad (4.26)$$

Die Eigenschaft ist durch die Anwesenheit einer dispersiv wirkenden Kanalimpulsantwort $\underline{h}[k]$ im Empfangsprodukt $\hat{s}_0[k]$ nicht prinzipiell gestört. Das Ergebnis der Kreuzkorrelation vermittelt dann zwar keinen isolierten Einzelpuls, aber eine Schätzung $\hat{h}[k]$ der vorliegenden Impulsantwort des Übertragungskanals.

$$\hat{s}_0[k] * s_0^*[-k] = (s_0[k] * \underline{h}[k] + \underline{n}[k]) * s_0^*[-k] \approx \underline{h}[k] * \delta[k] \quad (4.27)$$

Die CAZAC-Eigenschaft gilt nicht, falls ein unkorrigierter Frequenzoffset vorliegt. Das Resultat gleicht dann mehr oder weniger einer pseudoräuschartigen Sequenz.

$$(s_0[k] \cdot e^{2\pi(\Delta f_c/f_s)k}) * s_0^*[-k] \not\approx \underline{C} \cdot \delta[k] \quad \text{mit} \quad \Delta f_c \neq 0 \quad (4.28)$$

Im Empfänger liefert also nur bei korrekt geschätztem und anschließend korrigiertem Frequenzoffset eine Kreuzkorrelation des Empfangssignals mit der gespeicherten PRS-Symbolsequenz eine valide Schätzung der Kanalimpulsantwort $\hat{h}[k]$. Der aufgrund arithmetischer Mehrdeutigkeit im vorigen Abschnitt zur Feinfrequenzsynchronisation ΔF_c verbleibende Grobfrequenzfehleranteil ΔF_x kann deshalb durch einen Brute Force-Suchmechanismus bestimmt werden, welcher mit fortgesetztem Adaptieren des möglichen Grobfrequenzoffsets ΔF_x eine Folge der PRS-Kreuzkorrelationsfunktion zu finden versucht, die als eine gültige Schätzkanalimpulsantwort $\hat{h}[k]$ interpretiert werden kann (Abbildung 4.8a). In einem

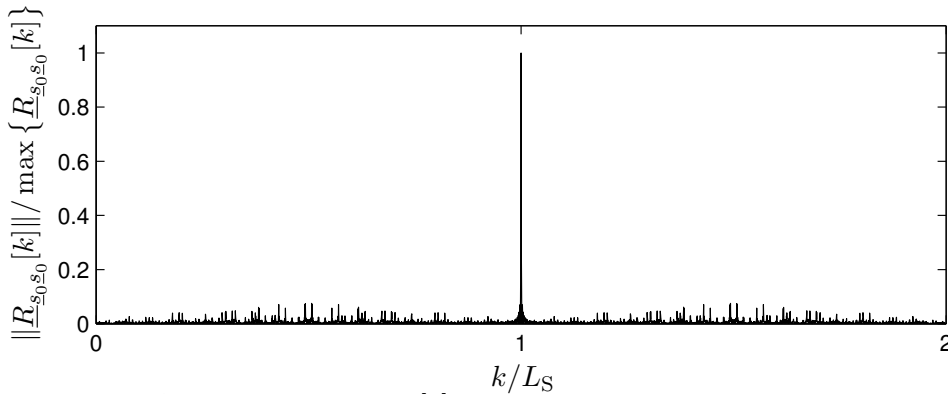


Abbildung 4.7: Autokorrelation $R_{s_0 s_0}[k]$ mit näherungsweise Einheitspulsverlauf, bedingt durch die CAZAC-Eigenschaft der Folge des PRS-Symbols.

4 Entwicklung einer effizienten Signalverarbeitungskette für DAB-Empfang

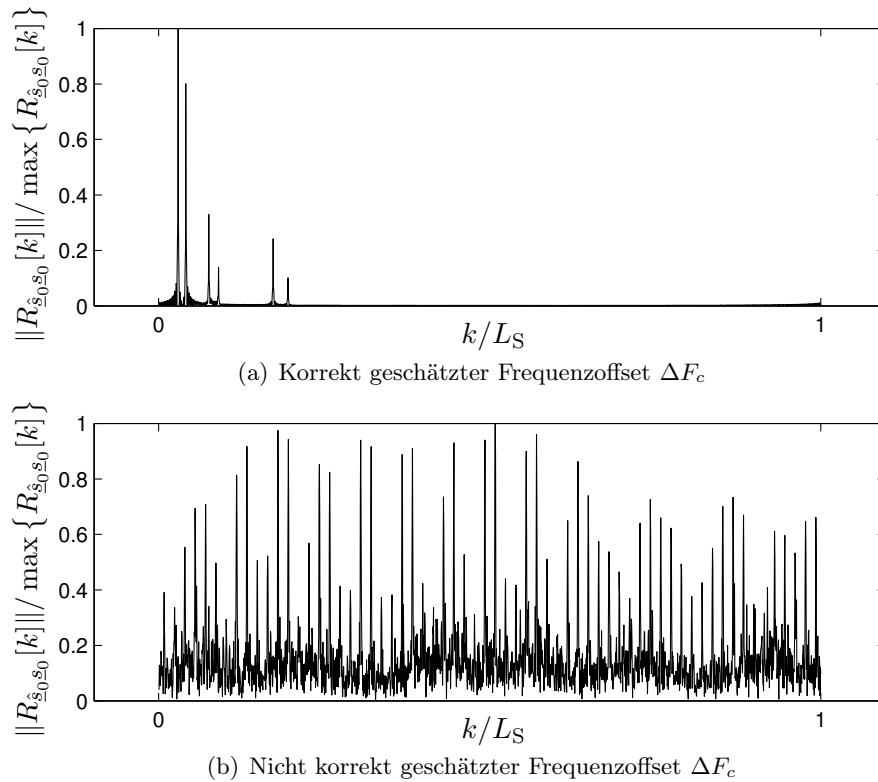


Abbildung 4.8: Beispielhafter Verlauf der Kreuzkorrelation $R_{\hat{\varepsilon}_0, \varepsilon_0}[k]$ in Abhängigkeit der Schätzung des Frequenzoffsets ΔF_c . Die Berechnung über das Verfahren der schnellen Faltung, resultierend in einem zyklisch auf $k = 0 \dots L_S$ beschränkten Indexintervall.

validen Empfangsszenario darf die Länge L_h der tatsächlichen Kanalimpulsantwort $\underline{h}[k]$ niemals größer als die Länge des Schutzintervalls L_G sein, denn andernfalls wäre im Szenario eine korrekte Demodulation des Signals generell nicht möglich⁷. Für nicht zutreffende Schätzung mit ΔF_x stellt sich nach Gleichung 4.28 also ein rauschartiges Ergebnis ein (Abbildung 4.8b) und dies verletzt per se die Charakteristik einer in der Länge auf die des Schutzintervalls begrenzten Impulsantwort. Ist hingegen eine längenbeschränkte Impulssequenz mit nachfolgender Ruhephase im Signal ähnlich Abbildung 4.8a identifiziert, dann liegt eine valide Lösung vor und ist mit Fund des Ganzzahlanteils ΔF_x nach Gleichung 4.19 der vollständige Frequenzoffset ΔF_c korrekt geschätzt. Zur Findung des Ganzzahlanteils ΔF_x folgt, dass bezogen auf das Szenario der für das VHF-Band III hergeleiteten zu erwartenden Frequenzabweichung im Bereich von $-24 \text{ kHz} \dots +24 \text{ kHz}$ bei einem Unterträgerabstand von $1/T_S = 1 \text{ kHz}$ in DAB Mode I bis zu 49 Signalvergleiche durchgeführt werden müssen. Um eine ressourceneffiziente Implementierung des Suchverfahrens abzuleiten wird statt der gewöhnlichen Faltung die zyklische Faltung mittels Frequenzbereichstransformation, auch als Schnelle Faltung bekannt, genutzt. Die Transformationslänge soll auch hier wieder der Länge eines OFDM-Symbols L_S entsprechen. Sämtliche Werte des Ergebnisses mit Indizes außerhalb des Bereiches $0, 1, \dots, (L_S - 1)$ werden bei der zyklischen Faltung

⁷In Mode I des DAB-Systems gilt für diese Länge $L_G = 504 \text{ Sa}$ zu $L_S = 2048 \text{ Sa}$.

4.2 Entwicklung der Empfangsalgorithmen

gemäß einer Modulo-Operation der Indizes additiv in diesen Bereich zurückgefaltet. Mit Hilfe der zyklischen Faltung kann die wiederholte Operation der Faltung (unter Beachtung von Randeffekten der zyklischen Fortsetzung) nach einer Fouriertransformation auf eine aufwandsmäßig überschaubare komponentenweise Vektormultiplikationsoperation reduziert werden.

$$\hat{s}_0[k] * \underline{s}_0^*[-k] \approx \mathfrak{F}^{-1} \{ \mathfrak{F} \{ \hat{s}_0[k] \} \cdot \mathfrak{F} \{ \underline{s}_0^*[-k] \} \} \quad (4.29)$$

Die empfangene Version des PRS-Pilotsymbols wird, nach Korrektur des Feinfrequenzfehlers, einmalig in den Frequenzbereich transformiert als Folge $\hat{S}_0[K]$ über alle OFDM-Subträger K .

$$\hat{S}_0[K] = \mathfrak{F} \{ \hat{s}_0[k] \} \quad (4.30)$$

Im Empfänger kann die originale PRS-Pilotfolge des OFDM-Symbols Nummer $i=0$ bereits in der in den Frequenzbereich transformierten, konjugiert komplexen Variante als konstante Sequenz $\underline{C}_{S_0}[K]$ vorabgespeichert werden.

$$\underline{C}_{S_0}[K] = \underline{S}_0^*[K] = \mathfrak{F} \{ \underline{s}_0^*[-k] \} \quad (4.31)$$

So genügt es, diese Folge für jeden Test hinsichtlich eines potentiellen Grobfrequenzfehlers ΔF_x im Frequenzindex K zu verschieben und multiplikativ mit $\hat{S}_0[K]$ zu verknüpfen. Nach der Fourier-Rücktransformation liegt die gesuchte Kanalimpulsantwortschätzung $\hat{h}_x[k]$ zum potentiellen Grobfrequenzfehler ΔF_x vor und kann der Plausibilitätsprüfung unterzogen werden.

$$\hat{h}_x[k] \approx \mathfrak{F}^{-1} \{ \hat{S}_0[K] \cdot \underline{C}_{S_0}[K - F_x] \} \quad \forall x \quad (4.32)$$

Die Schätzung der Kanalimpulsantwort $\hat{h}_x[k]$ reduziert sich somit pro Test von ΔF_x auf $T_U=2048$ komplexwertige Multiplikation sowie eine aufwandsmäßgebliche 2048-Punkt Fourier-Transformation. Die Bildung einer Metrik für die Plausibilitätsprüfung des geschätzten Impulsverlaufes $\hat{h}_x[k]$ erfolgt nach einem einfachen Schema: Bei einem angenommenen Frequenzversatzfehler ΔF_x wird das Energieintegral des Impulsverlaufes im Bereich $0 < k < L_G$, welcher alleine für eine gültige Impulsantwort in Frage kommen darf, ins Verhältnis zur Energie außerhalb dieses Bereiches gesetzt. In Letzterem dürfen für eine valide Impulsantwort nur marginale Störenergieanteile vorhanden sein. Bei unzutreffendem Schätzwert ΔF_x trifft dies nicht zu, der Wert der Metrik fällt dementsprechend niedrig aus.

$$M_x = \frac{\sum_{k=0}^{L_G-1} \|\hat{h}_x[k]\|^2}{\sum_{k=L_G}^{L_S-1} \|\hat{h}_x[k]\|^2} \quad (4.33)$$

Die Schätzentscheidung erfolgt für denjenigen Wert ΔF_x , welcher M_x maximiert.

$$\Delta F_x = \arg \max \{ M_x \} \quad (4.34)$$

Der nun vollständig ermittelte Gesamtfrequenzfehler $\Delta F'_c$ berechnet sich dann gemäß Gleichung 4.19 als Summe mit dem bereits bestimmten Feinfrequenzfehler.

$$\Delta F'_c = \Delta F_c + \Delta F_x \quad (4.35)$$

Algorithmus 3 fasst das Prinzip der Grobfrequenzfehlersynchronisation zusammen.

Algorithmus 3: Pseudocode der Grobfrequenzfehler-Synchronisation

```

//  $\hat{s}[\cdot]$ : Basisbandsamples des Empfangssignals, Start der Folge an PRS
//  $\Delta F_c$ : Bereits bestimmter Feinfrequenzfehleranteil

// Signalkorrektur gemäß bekanntem Feinfrequenzfehleranteil
for  $k = 0 \dots L_S$  do
  |  $\underline{s}_0[k] = \hat{s}[k] \cdot e^{+j2\pi\Delta F_c k}$ 
end
// Transformation des PRS-Empfangssymbols in Frequenzdomäne
 $\underline{S}_0[\cdot] = \text{FFT}(\underline{s}_0[\cdot])$ 
// Suche über Frequenzbereich
 $F_x = 0$ 
 $M_{\max} = 0$ 
for  $F = -\Delta F_c^{\max} \dots + \Delta F_c^{\max}$  do
  // Schnelle Faltung mit frequenzverschobener Referenz
  for  $K = 0 \dots L_S$  do
    |  $\underline{X}[K] = \underline{S}_0[K] \cdot \underline{C}_{S_0}[K + F]$ 
  end
   $\underline{x}[\cdot] = \text{FFT}^{-1}(\underline{X}[\cdot])$ 
  // Berechnung der Bewertungs-Metrik
   $P_1 = 0$ 
  for  $k = 0 \dots L_G$  do
    |  $P_1 = P_1 + \|x[k]\|^2$ 
  end
   $P_2 = 0$ ;
  for  $k = L_G \dots L_S$  do
    |  $P_2 = P_2 + \|x[k]\|^2$ 
  end
   $M = P_1/P_2$ 
  // Entscheidung für maximale Metrik
  if  $M > M_{\max}$  then
    |  $M_{\max} = M$ 
    |  $F_x = F$ 
  end
end
end
// Fortschreiben zu Gesamtfrequenzfehler
 $\Delta F'_c = \Delta F_c + F_x$ 

//  $\Delta F'_c$ : Gesamtfrequenzfehler, normiert auf Trägerabstand

```

4.2.2 Synchronisation: Tracking

Die geschätzten Synchronisationsparameter stellen keine konstanten Werte dar, sondern müssen über die Zeit kontinuierlich adaptiert werden. Gründe liegen vor allem in der Abhängigkeit der elektrischen Bauteile von Umgebungseinflüssen, beispielsweise Temperaturveränderungen, und als Folge einer Drift der Zeitbasen im Empfänger, aber auch in den per se fehlerbehafteten initialen Schätzwerten, welche ein langfristig stabiles Fortschreiben dieser Werte unmöglich machen. Der über die Zeit unausweichliche Verlust der Kohärenz am Empfänger würde sich in steigenden Bitfehlerraten und schließlich in einem kompletten Empfangsversagen aufgrund Synchronisationsverlustes äußern. Es zeigt sich bei Betrachtung der Algorithmen, dass diese zur Synchronisationsnachführung mit weitaus weniger Ressourcenaufwand realisiert werden können als bei der initialen Akquisition. Im Folgenden werden die resultierenden effizienten Verfahren für das Tracking der Zeit- und Frequenzabweichungen beschrieben.

Rahmen- und OFDM-Fenster-Synchronisation

Bei der initialen Synchronisation des Rahmenbeginns beziehungsweise OFDM-Fensters musste ein Bereich von mindestens der Länge eines kompletten DAB-Rahmens hinsichtlich eines Minimums der instantanen Signalleistung untersucht werden. Im Fall des Trackings ist die zeitliche Position des nächsten Nullsymbols $k'_{\text{start}} = k_{\text{start}} + L_{\text{F}} + \Delta k_{\text{start}}$ bereits direkt vorhersehbar bis auf den möglichen Abweichungswert Δk_{start} . Dieser ist durch Fehler der Zeitbasis der Empfängerschaltung motiviert, gegebenenfalls auch durch Änderungen im Echoprofil des Übertragungskanal. Folglich muss lediglich ein kleiner Schwankungsbereich

Algorithmus 4: Pseudocode zum Tracking des Rahmenbeginns

```

//  $\hat{s}[\cdot]$ : Basisbandsamples des Empfangssignals, Start der Folge an PRS

// Initialisierung der Moving Average-Filterung
P = 0
Pmin = 0
kmin = 0
// Suche in Intervall um erwarteten Rahmenbeginn (Annahme: bei k = 0)
for k = -LG...LG do
    // Berechnung der Signalleistung im Suchfenster
    P = P -  $\|\hat{s}[k]\|^2$ 
    P = P +  $\|\hat{s}[k + L_{\text{N}} - L_{\text{G}}]\|^2$ 
    // Falls Leistungsminimum: Position speichern
    if P < Pmin then
        | Pmin = P
        | kmin = k
    end
end
// Korrigierter Rahmenbeginn liegt an gefundenem Leistungsminimum
 $\Delta k_{\text{start}} = k_{\text{min}}$ 

```

4 Entwicklung einer effizienten Signalverarbeitungskette für DAB-Empfang

um die erwartete Position des Nullsymbols anstelle eines kompletten Rahmens aktiv beobachtet werden. Damit der neue Synchronisationspunkt sicher enthalten ist und gefunden wird, wird gemäß der definierten maximalen zu erwartenden Länge des Echoprofils naheliegend $\Delta k_{\text{start}} \in \{-L_G, \dots, +L_G\}$ als verringerter zu prüfender Suchbereich gewählt. Die nun drastische Komplexitätsreduktion der zu verarbeitenden Abtastwerte im Vergleich zum Verfahren der Akquisition folgt zu

$$\frac{2L_G}{L_F} = \frac{2 \cdot 504 \text{ Sa}}{196608 \text{ Sa}} = 0.51 \% \quad (4.36)$$

Erneut ist die initiale Summe der Moving Average-Filterung irrelevant hinsichtlich der Minimierungsoperation. Das Konzept ist zusammengefasst in Algorithmus 4.

Frequenzoffset-Synchronisation

Der Trägerfrequenzfehler Δf_c entwickelt sich kontinuierlich über die Zeit, sein Verlauf ist dabei stetig. Um die Drift des Trägerfrequenzfehlers zu verfolgen wird in Anlehnung an die Schätzung des Feinfrequenzfehlers der initialen Akquisition die Autokorrelation genutzt. Es

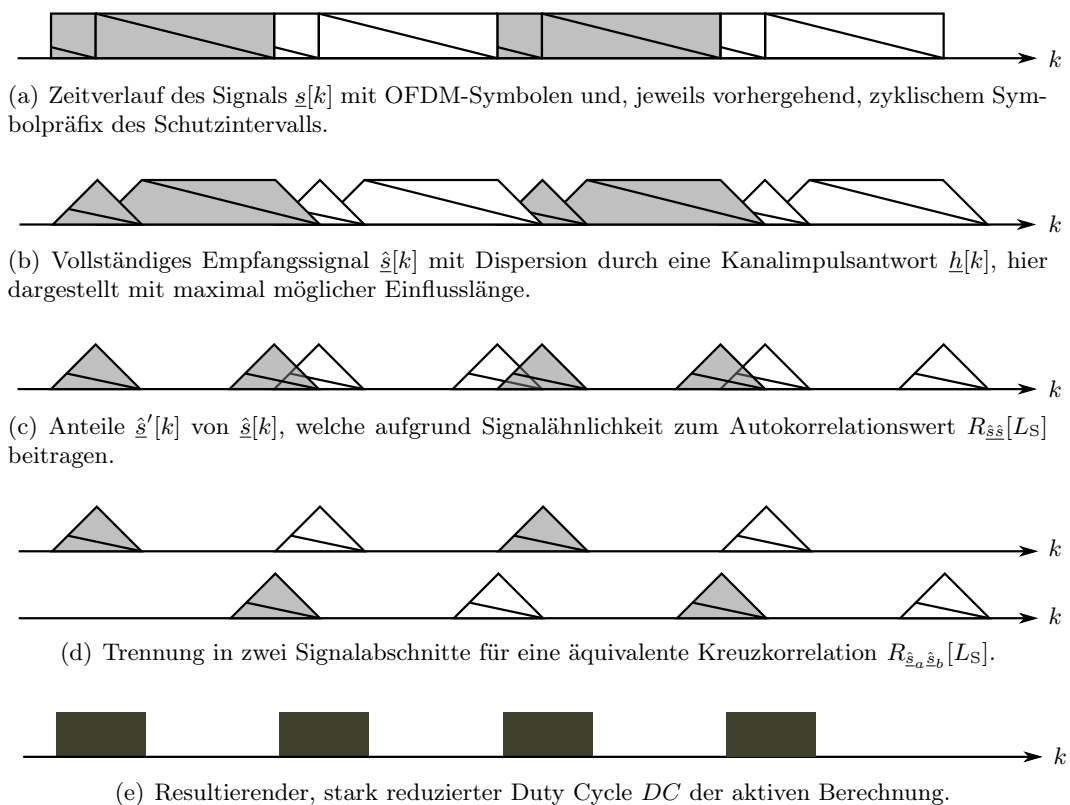


Abbildung 4.9: Schematische Darstellung der Aufwandsreduktion der für das Tracking des Frequenzfehlers nötigen Autokorrelationsberechnung über a priori-Wissen bezüglich der Position des zyklischen Symbolpräfix.

ist nun jedoch möglich, anstelle der vollständigen Autokorrelationssumme nur diejenigen Signalanteile in die Berechnung einzubeziehen, welche aktiv zum Schätzwert beitragen. Die zeitlichen Positionen, an welchen sich die Selbstähnlichkeiten im OFDM-Signal befinden müssen, sind nun genau bekannt. Dies sind der zyklische Präfix des Schutzintervalls und der korrespondierende Anteil des Symbols selbst. Abbildung 4.9 zeigt das Vorgehen zur Aufwandsreduktion. Es müssen L_G Abtastwerte des Schutzintervalls zusammen mit der durch die Dispersion der Kanalimpulsantwort verursachten Echos berücksichtigt werden. Letztere können im ungünstigsten Mehrwegeausbreitungsszenario weitere L_G Samples beitragen. Zum Autokorrelationswert tragen also pro OFDM-Symbol bis zu $2L_G$ Abtastwerte aktiv bei. Dies reduziert die Anzahl an benötigten komplexen Additionen und Multiplikationen für Mode I gegenüber der ursprünglichen Berechnungsmethode ohne Wissen über die zeitliche Synchronität auf circa 40 %:

$$\frac{2L_G}{L_S + L_G} = \frac{2 \cdot 504 \text{ Sa}}{2048 \text{ Sa} + 504 \text{ Sa}} = \frac{1008}{2552} \approx 40\% \quad (4.37)$$

Außerdem wird die Qualität der Schätzung verbessert, da diejenigen Abtastwerte, welche keinen konstruktiven Beitrag zur Schätzung liefern, sondern als sich ausmittelnde Rauschquelle auftreten, von vornherein aus der Berechnung ausgeschlossen werden. Um die Komplexität weiterhin zu reduzieren genügt es, nur jedes zweite Symbol in die Berechnung miteinzubeziehen. Praktische Tests hierfür zeigen, dass die Schätzung von ΔF_c auch dann noch in Szenarien schlechter Signalqualität und somit im Grenzbetrieb des Datendemodulators genau ist. Dies resultiert in einer Lastreduktion auf 20 % im Tracking gegenüber dem ursprünglichen Korrelationsberechnungsverfahren der Akquisition zur Identifizierung des Feinfrequenzfehleranteils.

Als weitere Vereinfachung gegenüber dem Synchronisationsverfahren der initialen Signalakquisition kann aufgrund des stetigen Werteverlaufs angenommen werden, dass der neue

Algorithmus 5: Pseudocode zum Tracking des Trägerfrequenzfehlers

```

//  $\hat{s}[\cdot]$ : Basisbandsamples des Empfangssignals, Start der Folge an PRS
//  $\Delta F_c$ : Bereits bestimmter Feinfrequenzfehleranteil

// Initialisierung kumulativer Autokorrelationsswert
 $\underline{R} = 0$ 
// Für alle relevanten Abtastwerte sämtlicher OFDM-Symbole
for  $l = 0 \dots 75$  do
  for  $k = -L_G \dots L_G$  do
    // Korrelation OFDM-Symbol mit zugeh. zykl. Präfix
     $\underline{R} = \underline{R} + \hat{s}[l \cdot L_{S+G} + k] \cdot \hat{s}^*[l \cdot L_{S+G} + k + L_S]$ 
  end
end
// Fortschreiben des neu detektierten Frequenzfehleranteils
 $\Delta F_c = \Delta F_c + \arg\{\underline{R}\}/2\pi$ 

//  $\Delta F_c$ : Neuer Gesamtfrequenzfehler, normiert auf Trägerabstand

```

4 Entwicklung einer effizienten Signalverarbeitungskette für DAB-Empfang

Wert des Trägerfrequenzfehlers sich in der unmittelbaren Umgebung des alten Schätzwertes befinden muss. Eine aufwändige Brute Force-Suche zur Auflösung von Mehrdeutigkeiten hinsichtlich eines willkürlichen Grobfrequenzfehleranteils ist dann nicht mehr von Nöten. Betrachtet in der zeitkontinuierlichen Domäne darf sich für Eindeutigkeit der zeitvariante Frequenzfehler Δf_c über den Betrachtungszeitraum, hier der Zeitdauer eines DAB-Rahmens T_F , maximal in einen Frequenzbereich der Ausmaße des Frequenzabstandes der OFDM-Subträger $1/T_U$ verändert haben. Entsprechend folgt die Anforderung an die zulässige Driftrate:

$$\left| \frac{d}{dt} \Delta f_c \right| < \frac{0.5 \cdot 1/T_S}{T_F} \quad (4.38)$$

Diese kurzzeitige Driftratenbedingung kann von frequenzbestimmenden Oszillatorbauelementen des Frontends leicht eingehalten werden: Für Mode I folgt eine maximal erlaubte Driftrate der Oszillatorbaugruppe von $\pm 0.5 \cdot 1 \text{ kHz}/96 \text{ ms} = \pm 5.2 \text{ kHz/s}$. Deshalb kann im laufenden Betrieb alleine aus der relativen Änderung des Frequenzfehlers der Wert des absoluten Fehlers zweifelsfrei fortgeschrieben werden. Algorithmus 5 fasst das Vorgehen des aufwandsreduzierten Frequenzfehler-Trackings zusammen.

4.2.3 Demodulation

Nachdem die Synchronisation des Empfängers hinsichtlich Zeit und Frequenz gesichert ist, können die eigentlichen Datendemodulationsalgorithmen auf das Empfangssignal angewendet werden.

Stationsname	Bitrate r_b	Coderate R_c	Multiplex- anteil in Capacity Units	Anteil tatsächlich relevanter OFDM-Symbole
FIC-Daten	32 kbit/s	1/3	-	3/3
BR Verkehr	48 kbit/s	1/2	35/864	2/18
B5 plus	96 kbit/s	1/2	70/864	3/18
on3-radio	128 kbit/s	1/2	96/864	2/18
BAYERN plus	128 kbit/s	1/2	96/864	2/18
Radio Galaxy	160 kbit/s	1/2	116/864	3/18
Bayern 2 Süd	160 kbit/s	1/2	116/864	3/18
Bayern 3	160 kbit/s	1/2	116/864	4/18
BR-KLASSIK	192 kbit/s	1/2	140/864	3/18
Rock Antenne	192 kbit/s	1/2	140/864	4/18

Tabelle 4.3: Konfiguration einiger realer, beispielhaft ausgewählter DAB-Services bei Angabe des zugehörigen Anteils der im OFDM-Signal tatsächlich betroffenen Anteile.

Time Slicing-Betrieb

Im DAB-Rahmenaufbau ist die Übertragung der einzelnen Nutzdatservices stets vollständig separiert und ihre Position im kontinuierlichen Datenstrom genau bestimmt. Folglich muss nicht zwingend die Gesamtheit aller empfangenen Daten durch den Demodulator zur Auswertung eines einzelnen Services bearbeitet werden⁸. Um den Rechenaufwand minimal zu halten werden gemäß des Time Slicing-Konzeptes die Demodulationsmethoden nur auf diejenigen Signalanteile angewendet, deren MSC-Nutzdatensegmente, das heißt Capacity Units (CU), anschließend in der Datensenke verwertet werden. Als Beispiel soll für eine DAB-Übertragung im Mode I ein selektierter Service des Multiplexes dienen, welcher in den CIFs des MSC die CU der Menge $\{c_1, \dots, c_2\}$ belegt. Die Menge der im Rahmen beteiligten OFDM-Symbole mit Ordnungsnummern i_{MSC} , gruppiert gemäß der vier CIFs eines Mode I-Rahmens, kann dann identifiziert werden zu

$$\begin{aligned}
 i_{\text{MSC}} \in \left\{ \left(4 + \left\lfloor \frac{c_1}{864} \right\rfloor \right) \dots \left(4 + \left\lceil \frac{c_2}{864} \right\rceil \right), \right. \\
 \left. \left(4 + 18 + \left\lfloor \frac{c_1}{864} \right\rfloor \right) \dots \left(4 + 18 + \left\lceil \frac{c_2}{864} \right\rceil \right), \right. \\
 \left. \left(4 + 36 + \left\lfloor \frac{c_1}{864} \right\rfloor \right) \dots \left(4 + 36 + \left\lceil \frac{c_2}{864} \right\rceil \right), \right. \\
 \left. \left(4 + 54 + \left\lfloor \frac{c_1}{864} \right\rfloor \right) \dots \left(4 + 54 + \left\lceil \frac{c_2}{864} \right\rceil \right) \right\}
 \end{aligned} \tag{4.39}$$

Hinsichtlich des OFDM-Schemas müssen entsprechend nicht zwingend sämtliche Symbole i des Rahmens, $i \in \{0 \dots 71\}$ für Mode I, prozessiert werden. Pro CIF sind von 18 Symbolen nur n_{CIF} zu bearbeiten.

$$n_{\text{CIF}} = \left\lceil \frac{c_2}{864} \right\rceil - \left\lfloor \frac{c_1}{864} \right\rfloor \tag{4.40}$$

Neben dem gewünschten MSC-Anteil ist der die Signalisierungsinformationen beinhaltende FIC stets vollständig zu demodulieren, entsprechend der OFDM-Symbole i_{FIC} gemäß

$$i_{\text{FIC}} \in \{1, 2, 3\} \tag{4.41}$$

Es ist zu beachten, dass stets vorab ein zusätzliches OFDM-Symbol bearbeitet werden muss. Dies ist darin begründet, dass die differentielle Demodulation der Unterträger sich stets an der Phasendrehung zwischen zwei aufeinanderfolgenden Symbolen orientiert. Es ergibt sich für den gesamten Empfänger einschließlich der obligatorischen $n_{\text{FIC}}=3$ Symbole des FIC das Time Slicing-Gesamtastverhältnis DC , gemäß dessen der Time Slicing-Betrieb des OFDM-Demodulators direkt den Prozessor entlastend wirkt:

$$DC = \frac{(1 + n_{\text{FIC}}) + 4(1 + n_{\text{CIF}})}{76} \tag{4.42}$$

Tabelle 4.3 zeigt exemplarisch empfangbare DAB-Services im Raum Bayern und gibt an, welche CU-Anteile beziehungsweise wie viele von insgesamt 18 OFDM-Symbolen eines

⁸Beispielsweise bei DVB-T muss im Unterschied stets der gesamte Transportstrom aller übertragenen Nutzdatservices demoduliert werden. Erst im Anschluss daran können im demodulierten Transportstrom die einzelnen Nutzdatserviceanteile verschiedener Services identifiziert, voneinander separiert und nicht benötigte Datenanteile verworfen werden.

jeden CIF bei Selektion dieses Services tatsächlich betroffen sind und durch den Demodulator aktiv bearbeitet werden müssen. Für die maximal im Feldbetrieb auftretende Bitrate $r_b=192$ kbit/s ist zu erkennen, dass dies stets $n_{\text{CIF}} \leq 4$ Symbole sind, entsprechend einer notwendigen Aktivität des Demodulators gemäß einem Duty Cycle $DC \leq 31.6\%$. Die anhängige Prozessorlast des OFDM-Demodulators kann folglich relevant gesenkt werden.

Korrektur des Trägerfrequenzoffsets

Generell existieren zwei Methoden, um den detektierten Trägerfrequenzfehler zu korrigieren:

- Die Nachregelung des lokalen Empfängeroszillators der Hardware durch “Ziehen” der frequenzbestimmenden Bauteile. In der Schaltungstechnik des HF-Frontends muss hierzu eine entsprechende Stelleinrichtung vorgesehen sein.
- Die mathematische Korrektur innerhalb der Domain der digitalen Signalverarbeitung. Der Lokaloszillator wird vollständig freilaufend betrieben. Für entsprechend vereinfachte Frontend-Hardware ist jedoch in der Signalverarbeitung zusätzlicher Rechenaufwand aufzubringen.

Es wird im Weiteren das flexiblere zweite Konzept verfolgt. Das Vorgehen ist in Algorithmus 6 dargestellt. Es handelt sich im Grunde um eine simple Signalremodulation, welche anhand des detektierten Trägerfrequenzversatzes ΔF_c das Signal durch eine Verschiebung im Frequenzbereich auf die korrekte Zwischenfrequenzlage transformiert. Die Remodulation wird auf die gemäß des Time Slicing-Verfahrens als relevant eingestuften Symbolabschnitte des Basisbandsignals angewandt. Im Zuge der Frequenzkorrektur übernimmt der Funktionsblock zugleich die Aufgabe, die Nutzperiode der benötigten OFDM-Symbole gemäß dem FFT-Fenster aus dem kontinuierlichen Signalstrom der Basisband-Abtastwerte auszuschneiden und dabei die Perioden des Schutzintervalls zu verwerfen. Die Abtastwerte werden in einem neuen Speicherbereich entsprechend den Anforderungen des nachfolgenden Funktionsblocks der Fourier-Transformation passend einsortiert.

Algorithmus 6: Pseudocode zur Korrektur des Frequenzfehlers

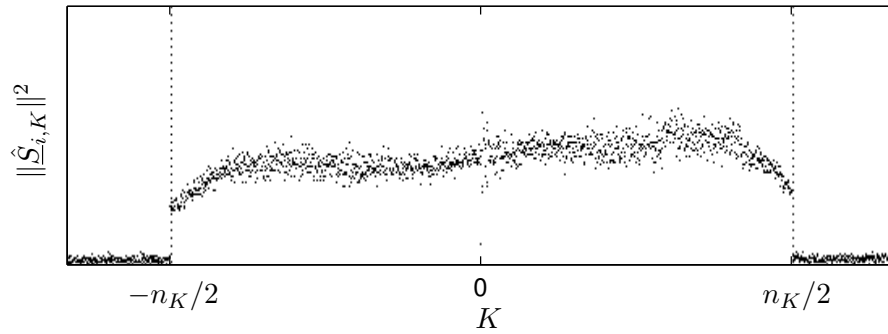
```

//  $\hat{s}[\cdot]$ : Basisbandsamples des Empfangssignals, Beginn an Symbol
//  $\Delta F_c$ : Bestimmter Frequenzfehler

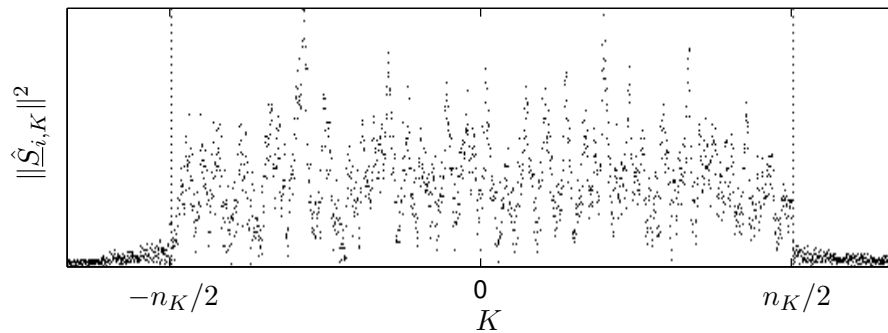
// Für alle angeforderten OFDM-Symbole
for  $i = 0 \dots (n_i + 1)$  do
    for  $k = 0 \dots L_S$  do
        // Remodulation und Speicherung in FFT-Buffer
         $\underline{s}_i[k] = \hat{s}[i \cdot L_{S+G} + k] \cdot e^{-j2\pi\Delta F_c(i \cdot L_{S+G} + k)/L_S}$ 
    end
end

//  $s_i[\cdot]$ : Extrahierte und frequenzkorrigierte Nutzsignalsegmente für FFT

```



(a) Szenario 1: Starkes Empfangssignal, ein direkter Pfad (Nähe Olympiapark, München, Deutschland).



(b) Szenario 2: Starke Reflektionen, vier beitragende SFN-Sendeanlagen (ein tiefes Tal in Unterfranken, Deutschland).

Abbildung 4.10: Betragsspektrum nach der Fourier-Transformation im OFDM-Demodulator. Im Falle von Signalreflektionen ergeben sich stark variierende Pegel für die einzelnen Subträger des OFDM-Symbols.

OFDM-Demodulation

Die OFDM-Demodulation der Teilsequenzen $\hat{s}_i[k]$ eines jeden Empfangssymbols i erfolgt durch eine diskrete Fourier-Transformation. Diese ist als schnelle Fourier-Transformation (engl. Fast Fourier Transform, FFT) implementiert. Im Modus I des DAB-Systems ist, entsprechend $L_S=2048$ Sa, eine Transformation mit $N=2048$ Punkten gewählt.

$$\hat{\underline{S}}_i[K] \circ \bullet \hat{s}_i[k] \quad \text{mit} \quad N = 2048 \quad (4.43)$$

Nach der Durchführung dieser Transformation ergibt sich eine spektrale Darstellung des Signalelementes (siehe Abbildung 4.10). Es werden diejenigen FFT-Bins der Seitenbereiche, welche mit keinen aktiven OFDM-Subträgern des Sendesignals korrespondieren, verworfen. Die Fourier-Transformation erfüllt so zugleich die Aufgabe eines steilflankigen Kanalfilters.

DQPSK-Demapping

Jeder OFDM-Unterträger ist gemäß dem $\pi/4$ -DQPSK-Verfahren moduliert und stellt zwei Bit Information dar. Dass sämtliche, original als $\underline{S}_i[K]$ gesendeten OFDM-Unterträger mit Index K den Empfänger durch die Einflüsse des dispersiven Übertragungskanals individuell

4 Entwicklung einer effizienten Signalverarbeitungskette für DAB-Empfang

im Pegel skaliert und phasengedreht erreichen, wird modelliert durch den komplexwertigen Faktor $\underline{H}_i[K]$. Zusätzlich ist jedem Symbol i additiv weißes Gauß'sches Rauschen (engl. Additive White Gaussian Noise, AWGN) $\underline{N}_i[K]$ überlagert.

$$\hat{\underline{S}}_i[K] = \underline{H}_i[K] \cdot \underline{S}_i[K] + \underline{N}_i[K] \quad (4.44)$$

Da sich die Kanaleinflüsse zeitlich relativ zum Symboltakt langsam ändern, gilt für die Zeitspanne von zwei Symbol dauern $2T_S + T_G \approx 2 \mu s$ auch bei hoher Dynamik des Übertragungskanals in guter Näherung

$$\underline{H}_{i-1}[K] \approx \underline{H}_i[K] \quad (4.45)$$

Die als Phasenwinkelunterschied zum vorausgehenden Symbol codierte Information der differentiellen $\pi/4$ -DQPSK-Modulation kann zwischen zwei aufeinanderfolgenden OFDM-Symbolen mittels einer komplexen Multiplikation der Werte der zugehörigen FFT-Bins als inkohärente Demodulation erlangt werden. Dabei gilt:

$$\begin{aligned} \hat{\underline{S}}_i[K] \hat{\underline{S}}_{i-1}[K]^* &= (\underline{H}_i[K] \underline{S}_i[K] + \underline{N}_i[K]) (\underline{H}_{i-1}[K] \underline{S}_{i-1}[K] + \underline{N}_{i-1}[K])^* \\ &\approx \|\underline{H}_i[K]\|^2 (\underline{S}_i[K] \underline{S}_{i-1}[K]^*) + \tilde{\underline{N}}_i[K] \end{aligned} \quad (4.46)$$

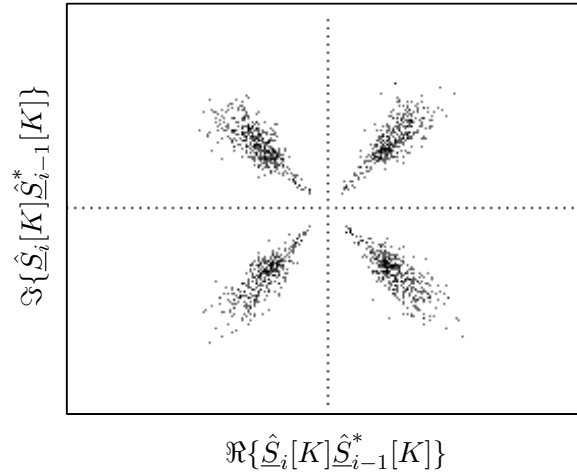
Sämtliche sich beim Ausmultiplizieren ergebenden Störanteile sind zur Übersichtlichkeit zusammengefasst in $\tilde{\underline{N}}_i[K]$. Je weniger Energie für den Unterträger mit Index K gemäß Übertragungsfunktion $\underline{H}_i[K]$ den frequenzselektiven Übertragungskanal passiert, desto höher ist relativ gesehen der Anteil der Leistung der konstant additiv überlagerten Störung $\underline{N}_i[K]$. Entsprechend weniger zuverlässig ist die Information mit dann geringer Kreuzleistung $\hat{\underline{S}}_i[K] \hat{\underline{S}}_{i-1}[K]^*$ zu gewichten. Das Abbilden der n_k Träger aller betrachteten OFDM-Symbole zu einer stringenten Datenbitsequenz folgt nach

$$b[q_{re}] + j \cdot b[q_{im}] = \hat{\underline{S}}_i[K] \hat{\underline{S}}_{i-1}[K]^* \quad (4.47)$$

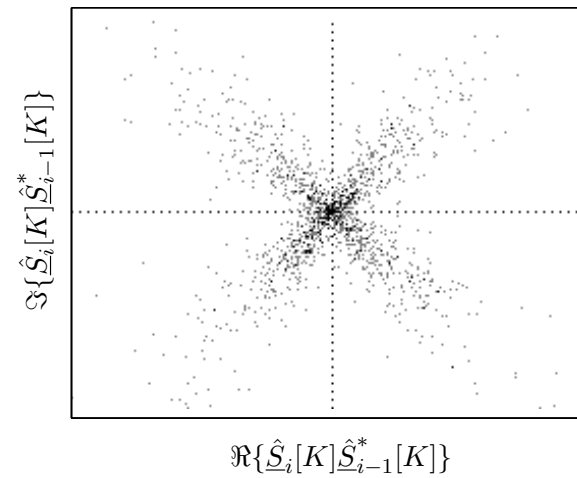
Die Wertefolge der aktiven Subträger wird in Gleichung 4.47 hinsichtlich der Indizes K und q_{re} beziehungsweise q_{im} gemäß einer im Standard definierten Funktion permutiert beziehungsweise neu angeordnet. Nach DAB-Standard gilt dabei Gleichung 4.48, es ist im Funktionsblock zugleich die Aufgabe des nach DAB-Standard geforderten Frequenzinterleavings $f_{\text{Interl.}}(K)$ zu behandeln.

$$q_{re} = 2(i-1)n_k + f_{\text{Interl.}}(K), \quad q_{im} = (2(i-1)+1)n_k + f_{\text{Interl.}}(K) \quad (4.48)$$

Die resultierenden Real- und Imaginärteile $b[q_{re}]$ und $b[q_{im}]$ korrespondieren direkt zu den Zuständen zweier gesendeter, binärer Datenbits. Je negativer beziehungsweise positiver die Komponentenwerte $b[\cdot]$ sind, mit desto höherer Wahrscheinlichkeit wurde ein Bit des Wertes 1 beziehungsweise 0 empfangen. Die Entscheidung für einen Bitzustand im Sinne harten, binären Demappings wird jedoch hier nicht getroffen. Vielmehr erfolgt durch den DQPSK-Demapper eine Weitergabe der vollständigen Zuverlässigkeitsinformationen, im entwickelten Programmcode quantisiert mit 8-bit Wortbreite, an nachgelagerte Funktionsblöcke. So können Sensitivitätsgewinne in der nachfolgenden statistischen Analyse des Fehlerschutzdecoders erlangt und damit die Empfindlichkeit des Empfangssystems erhöht werden. Dies ist bekannt als Soft Decision-Konzept [86]. Das gesamte Funktionsprinzip ist qualitativ in Algorithmus 7 zusammengefasst.



(a) Szenario 1: Starkes Empfangssignal, ein direkter Pfad (Nähe Olympiapark, München, Deutschland).



(b) Szenario 2: Starke Reflexionen, vier beitragende SFN-Sendeanlagen (ein tiefes Tal in Unterfranken, Deutschland).

Abbildung 4.11: Signalkonstellation in der komplexen IQ-Ebene des inkohärenten $\pi/4$ -DQPSK-Demodulators für ein Funkzenario ohne nennenswerte beziehungsweise mit starken Mehrwegereflektionen.

Algorithmus 7: Pseudocode zum Demapping der DQPSK-Konstellation

```

//  $\hat{S}_i[\cdot]$ : Frequenztransformierte OFDM-Signalsegmente

// Für alle angeforderten OFDM-Symbole
for  $i = 1 \dots n_i$  do
    // Für alle informationstragenden OFDM-Unterträger
    for  $q = 0 \dots (n_K - 1)$  do
        // Wandlung Bitposition in Unterträgerindex
         $K = \text{IndexBit2Carrier}(q)$ 
        // Differentielle Demodulation
         $b[2(i - 1) \cdot n_K + B] = \text{Re}\{\hat{S}_i[K] \cdot \hat{S}_{i-1}^*[K]\}$ 
         $b[(2(i - 1) + 1) \cdot n_K + B] = \text{Im}\{\hat{S}_i[K] \cdot \hat{S}_{i-1}^*[K]\}$ 
    end
end

//  $b[\cdot]$ : Demodulierte Datenfolge als Soft Bit-Repräsentation

```

Zeitdeinterleaving

Für die Datenfolge des MSC muss ein Zeitdeinterleaving über eine Dauer von 16 CIF-Blöcken, bei einer CIF-Taktung von 24 ms entsprechend einer Realzeit von 384 ms angewendet werden (Algorithmus 8). Durch das Prinzip des Forney-Interleavers [87] erfährt jedes den Interleaver durchlaufende Soft Bit in Abhängigkeit vom Bitpositionsindex eine individuell nach DAB-Standard festgelegte Verzögerungszeit.

Algorithmus 8: Pseudocode des Zeit-Deinterleavers

```

//  $b[\cdot]$ : Demodulierte Datenfolge als Soft Bit-Repräsentation
//  $u_i[\cdot]$ : Deinterleaving-Speicher
//  $n_{\text{CIF}}$ : Fortlaufender Zähler des CIF-Index

for  $k = 0 \dots n - 1$  do
     $i = \text{BitReverse}(k \text{ AND } 15)$ 
     $i = (i + n_{\text{CIF}}) \text{ AND } 15$ 
     $u_i[k] = b[k]$ 
end

 $c[\cdot] = u_{(n_{\text{CIF}} \text{ AND } 15)}[\cdot]$ 

//  $c[\cdot]$ : Soft Bit-Codeword, deinterleaved

```

Fehlerschutz-Decodierung

Der nun vorgestellte Funktionsblock erfüllt gemeinsam die folgenden im DAB-System geforderten Aufgaben:

- Depunktierung der Eingangsdatenfolge.
- Viterbi-Algorithmus zur Faltungscodierung:
 - Berechnung der Zustands-Metriken.
 - Metrik-Akkumulation, Vergleich und Selektion eines Trellis-Pfades (bezeichnet als engl. Accumulate Compare Select, ACS).
 - Traceback der Trellis-Matrix.
- PRBS-Descrambling der Ausgangsdatenfolge.

Bei der Depunktierung der Eingangsdatenfolge handelt es sich um einen rein sequentiell auf die Eingangsfolge arbeitenden Teilalgorithmus. Gemäß der Idee der Protection Level des DAB-Systems variiert im Übertragungssystem fortdauernd die Codierungsrate nach den im Standard festgelegten Tabellen von Nutzdatenbit zu Nutzdatenbit [2], entsprechend den vier möglichen Momentanraten $R_c \in \{1/4, 1/3, 1/2, 1/1\}$. Deshalb sind für ein Nutzdatenbit stets ein Tupel variabler Länge aus ein bis vier Bits dem Eingangsdatenstrom des Decoders $c[\cdot]$ zu entnehmen. Sämtliche Tupel müssen sodann jedoch mit für den Decoder entscheidungsneutralen Datenpunkten des Wertes 0 auf die volle Tupellänge vier aufgefüllt werden, da der Kern des Fehlerschutzdecoders mit der festen Rate $R_m=1/4$ des Muttercodes arbeitet und deshalb stets mit vier Datenpunkten zur Berechnung einer einzelnen Decoder-Metrik gespeist werden muss. Um die Zahl der Speichertransferoperationen zu minimieren und ein erneutes Schreiben und Lesen aus dem Arbeitsspeicher zu vermeiden, wird das Depunktieren direkt in den Code zum Laden der Eingangsdaten des späteren Viterbi-Fehlerschutzdecoders eingebettet.

Der zum Fehlerschutz eingesetzte Faltungscodierung kann als ein Hidden Markov-Modell betrachtet werden. Die zu decodierenden Datentupel stellen die Emissionen am Ausgang eines Faltungscoders dar, dessen unbekannte innere Zustandsvariablen, und somit dessen Eingangsdatensequenz, sind aufgrund der Beobachtung der empfangenen Datentupel zu schätzen. Diese Decodierung kann rechnerisch effizient durch eine Implementierung des bekannten Viterbi-Algorithmus durchgeführt werden [83]. Mit den beiden möglichen Datenbitzuständen 0 und 1 am Eingang des Hidden Markov-Modells kann jeder Zustand in exakt zwei Folgezustände übergehen. Der von DAB zum Fehlerschutz verwendete Faltungscodierung besitzt 64 mögliche innere Zustände. So sind zum Vergleich der bestmöglichen Passung sämtlicher möglichen Signalelemente \vec{E}_j aller Zustände j mit der konkret empfangenen Emission 64 Vergleichsmetriken je Bit der Nutzdatensequenz zu berechnen. In Berücksichtigung der Historie potentieller Vorgängerzustände durch Metrikakkumulation und Maximierung dieser wird für jeden der 64 Zustände auf ein mögliches Datenbit 0 oder 1 am Eingang des unbekanntes Encoders und damit einen der beiden möglichen Übergänge zu einem Folgezustand dieses Encoders geschlossen. Diese Entscheidung wird für jeden Schritt in einer Spalte der sogenannten Trellis-Matrix \mathbf{T} vermerkt.

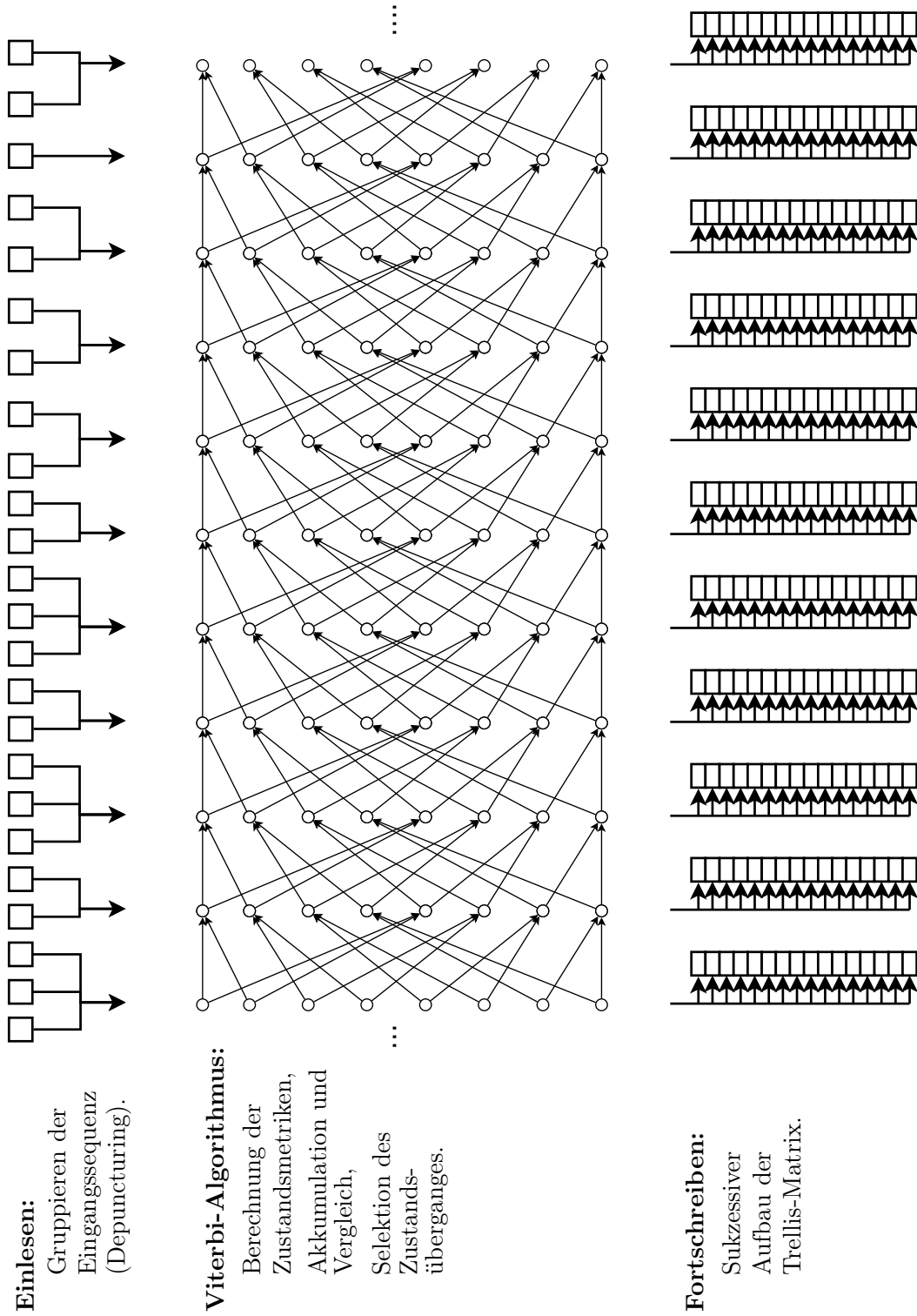


Abbildung 4.12: Schematische Darstellung des Datenflusses des Fehlerschutzdecoders, ohne Backtrace als zweitem Teil des Viterbi-Algorithmus. Zur Übersichtlichkeit sind nicht sämtliche 64 Zustände des Viterbi-Trellis dargestellt.

Algorithmus 9: Pseudocode des Fehlerschutz-Decoders

```

// c[·]: Zu decodierende Datenfolge als Soft Bit-Repräsentation

// Initialisierung
 $\vec{M} = \vec{0}$ 
 $i_c = 0$ 
// Teil 1: Aufbau des Trellis
for  $k = 0 \dots (l_{MSC} - 1)$  do
  // Depunktieren der Eingangssequenz
  for  $i = 0 \dots 3$  do
    if PunctureScheme[ $4k + i$ ] then
       $s_i = c[i_c]$ 
       $i_c = i_c + 1$ 
    else
       $s_i = 0$ 
    end
  end
  end
  // Für alle 64 Zustände
  for  $j = 0 \dots 63$  do
    // Metrik-Berechnung
     $m = (s_0, s_1, s_2, s_3) \cdot \vec{E}_j$ 
    // Viterbi's Add, Compare, Select
     $m_1 = \vec{M}[\text{StateTransitionTable}[j, 0]] + m$ 
     $m_2 = \vec{M}[\text{StateTransitionTable}[j, 1]] - m$ 
    if  $m_1 > m_2$  then
       $\vec{M}[j] = m_1$ 
       $\mathbf{T}[j][k] = 0$ 
    else
       $\vec{M}[j] = m_2$ 
       $\mathbf{T}[j][k] = 1$ 
    end
  end
  end
   $\vec{M}[\cdot] = \vec{M}'[\cdot]$ 
end
// Teil 2: Backtrace des Trellis
 $j = 0$ 
for  $k = (l_{MSC} - 1) \dots 0$  do
  // Datenablage und PRBS-Descrambling
   $d[k] = \mathbf{T}[j][k] \mathbf{XOR} \text{PRBS}[k]$ 
   $j = \text{StateTransitionTable}[j, d[k]]$ 
end

// d[·]: Decodierte binäre Datenfolge

```

4 Entwicklung einer effizienten Signalverarbeitungskette für DAB-Empfang

Nachdem alle Eingangsdaten verarbeitet wurden und der Aufbau der Trellis-Matrix \mathbf{T} komplett erfolgt ist, beginnt der zweite Abschnitt des Funktionsblockes. Der Algorithmus des sogenannten Viterbi-Backtrace erarbeitet einen Pfad mit minimaler Metrik durch die von der Trellis-Matrix \mathbf{T} aufgespannte Fläche und decodiert so die ursprüngliche Nachrichtensequenz. Dabei ist es eine Operation mit rein sequentiellm Ablauf. Die Trellis-Matrix wird spaltenweise in entgegengesetzter Richtung zu ihrem Aufbau durchlaufen. Beim Abspeichern der decodierten Nutzdaten ist hier zugleich eine Exklusiv Oder-Verknüpfung implementiert mit der vorab gespeicherten, gemäß Standard benötigten DAB-spezifischen Pseudozufallsfolge des PRBS Energy-Dispersals.

Abbildung 4.12 zeigt in schematischer Form mit Fokus auf dem entstehenden Datenfluss noch einmal die im ersten Abschnitt beschriebenen Schritte der anzahlvariablen Gruppierung von Eingangsbits, der akkumulierenden Metrikberechnung mit Maximierung hinsichtlich jeweils zweier möglicher Zustandsübergänge sowie den Aufbau der Trellis-Matrix \mathbf{T} durch sukzessives Wegschreiben dieser Entscheidungsergebnisse. Algorithmus 9 bildet den vollständigen Funktionsblock des Fehlerschutzdecoders ab.

4.3 Nutzdatenhandling

Mit der Energy Dispersal-Operation endet die Algorithmenkette der digitalen Radiosignalverarbeitung. Die digitalen Nutzdaten $d[\cdot]$ des MSC- beziehungsweise FIC-Stroms stehen bereit zur Auswertung. Weitere Elemente des Empfängers sind somit im eigentlichen Sinne keine Besonderheit des Untersuchungsgegenstandes einer softwarebasierten Signalverarbeitung. Trotzdem sind die im Folgenden dargestellten Elemente beziehungsweise Schnittstellen zwingend notwendig, um ein praxistaugliches, lauffähiges Empfängersystem vollständig darstellen zu können.

4.3.1 Fast Information Channel

Die im FIC beinhalteten Steuerungsinformationen werden entsprechend dem im DAB-Standard [2] spezifizierten Paketprotokoll geparkt und verarbeitet. Im Speziellen werden aus den beinhalteten Informationen die folgenden vier zum Betrieb notwendigen Listen aufgebaut:

- **Sub Channels:** Position- und Codierungsinformationen zur Extraktion einzelner logischer Subkanäle aus dem Multiplex durch den Faltungsdecoder.
- **Audio Services:** Liste sämtlicher ID-Kennungen der im Multiplex vorhandenen Audioservices und deren Datenstromkonfiguration.
- **Data Services:** Liste sämtlicher ID-Kennungen der im Multiplex vorhandenen Datenservices und deren Datenstromkonfiguration.
- **Service Labels:** Menschenlesbare Namen für jede ID-Kennung eines jeden angebotenen Dienstes.

4.4 Softwaretechnische Umsetzung des Empfängersystems

Zur Steuerung des Empfängersystems und dem Abruf der Ensemblelisten durch den Nutzer werden entsprechende Interface-Routinen durch das Softwaremodul exportiert. Die Routinen akzeptieren als Parameter in einfacher Weise Stationsnamen in Klartext und interpretieren diese über die geführten Listen selbständig in ID-Kennungen und Subkanäle.

4.3.2 Main Service Channel

Die aus dem MSC extrahierten Nutzdatenanteile der Services werden jeweils an eine zu registrierende Handler-Funktion übergeben. Diese Rückruffunktion besitzt die Aufgabe, eine Schnittstelle zu einem externen Applikationsprozess herzustellen und diesem die Daten zur Verfügung zu stellen. Die dortige Verarbeitung der Inhalte an sich, beispielsweise die Decodierung und Wiedergabe quellencodierter (Audio-)Daten, ist nicht Bestandteil der vorliegenden Untersuchungen.

4.4 Softwaretechnische Umsetzung des Empfängersystems

Im vorliegenden Abschnitt soll auf die softwaretechnischen Aspekte der Umsetzung der Signalverarbeitungskette eingegangen werden. Einen schematischen Überblick über den logischen Datenfluss des Softwarestacks zeigt Abbildung 4.13.

4.4.1 Entwicklung des Programmcodes

Vor der Formulierung in der Hochsprache C werden die im vorigen Kapitel beschriebenen Demodulationsalgorithmen im Zuge einer ersten Implementierungsstufe in Matlab entwickelt und innerhalb dieser Simulationsumgebung auf zufriedenstellende Ergebnisse und korrekte Empfängerfunktion hin validiert. Nach erfolgreicher Prüfung des algorithmischen Gesamtsystems erfolgt die Portierung der Signalverarbeitung in die Hochsprache C, um aus der rein simulativ einsetzbaren Matlab-Software eine produkttaugliche, weil ressourceneffiziente und deshalb auf einem Prozessor mit beschränkter Leistung echtzeitfähige Implementierung zu erhalten. Der Kern der Referenz-Signalverarbeitung in der Hochsprache C resultiert in circa 7000 Zeilen Programmcode. Der erstellte Quellcode ist konform zum ANSI C-Standard. Es bestehen keinerlei externe Abhängigkeiten zu Bibliotheken außer der C-Standardbibliothek. So ist eine hohe Portierbarkeit hinsichtlich sämtlicher zu untersuchender Zielplattformen sichergestellt.

4.4.2 Eliminierung von Datentransfers für Zwischenergebnisse

Ein signifikanter Performancegewinn für die Ausführungsgeschwindigkeit auf Prozessoren kann durch die gezielte Verschmelzung benachbarter Funktionsblöcke einer logisch partitionierten Signalverarbeitungskette erreicht werden. Damit dies möglich ist, müssen für die Ausgangs- beziehungsweise Eingangsdaten der Algorithmen identische Zugriffsmuster und Zugriffsabfolgen konstruierbar sein. Durch Vermeidung des Zwangs zur temporären Speicherung der Zwischenergebnisse wird die Anzahl der durch einen Prozessor benötigten Speichertransferoperationen stark reduziert. Entsprechend konnten die folgenden Algorithmen des Empfängerstacks, welche in der an der Nachrichtentheorie orientierten Betrachtung

4 Entwicklung einer effizienten Signalverarbeitungskette für DAB-Empfang

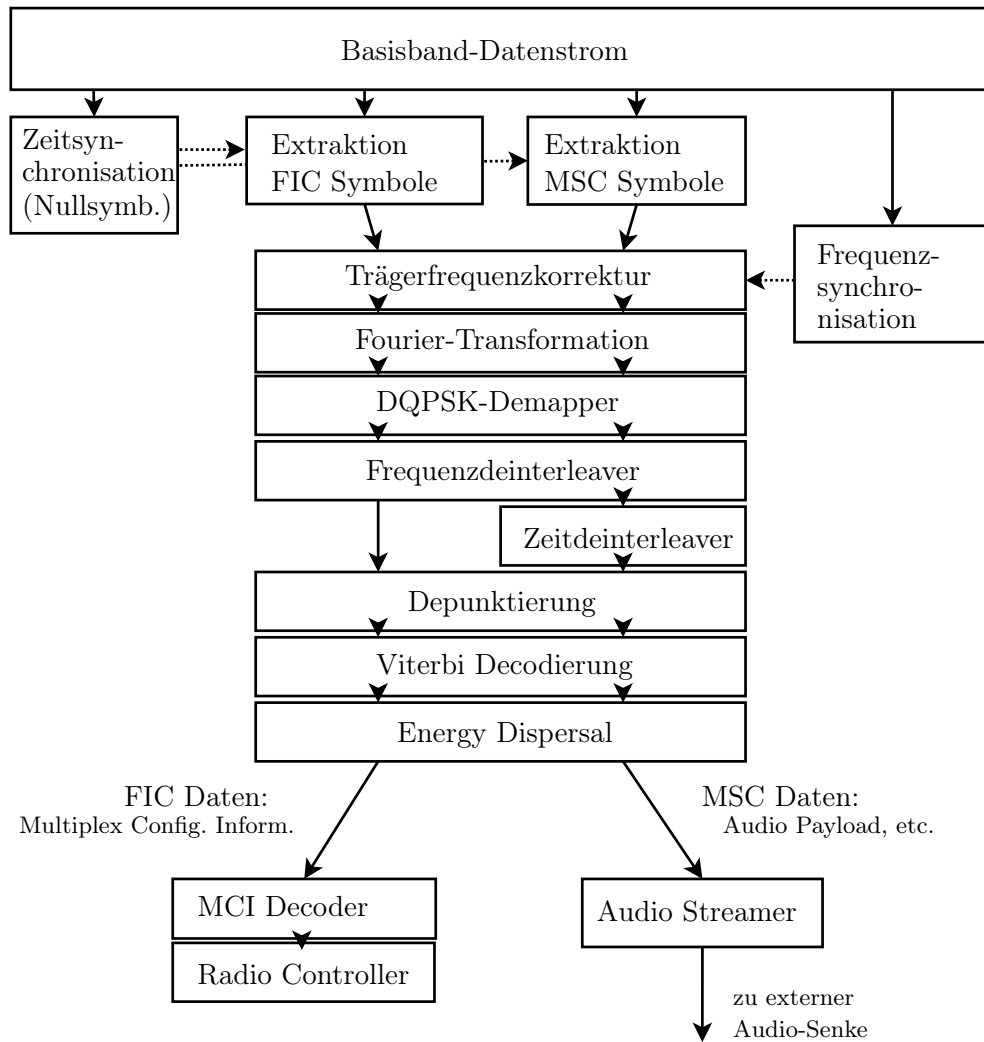


Abbildung 4.13: Schematische Übersicht beziehungsweise Visualisierung des Datenflusses im entwickelten Empfängerstack.

Die Funktionsblöcke sind logisch separiert dargestellt, wie bereits beschrieben zu gemeinsamen Funktionsblöcken zusammengefasst werden:

- Depuncturing, Viterbi-Decoder und Reverse Energy Dispersal.
- Trägerfrequenzkorrektur und OFDM-Symboleextraktion.
- Fourier-Transformation und Frequenzdeinterleaver, dabei für letzteren Zusammenfassung mit Bit Reversal-Stufe der FFT.

4.4.3 Übergang zur Festkommaarithmetik

Auf vielen Prozessortypen erfordern Berechnungen in Festkommaarithmetik weniger Zeitressourcen als Fließkommaoperationen. Des Weiteren erlaubt die Festkommaarithmetik Datentypen mit geringerer Wortbreite. Die Datentransfermenge des Prozessors kann so weiter gesenkt werden durch Reduktion numerisch unnötig hoher Wortbreiten. Das zugrunde liegende Modell der Matlab-Umgebungen basierte auf Fließkommaarithmetik als Standarddatentyp. Für die C-Realisierung als Festkommasignalverarbeitungskette werden nun anstelle von Fließkommawerten mit minimal 32 bit Wortbreite Integervariablen mit einer numerischen Präzision von typisch 16 bit verwendet. Mittels Datentypen von 16 bit Wortbreite ist nach Gleichung 2.7 für OFDM-Modulation theoretisch ein ausreichender Dynamikumfang von bis zu 88 dB darstellbar. Der Wert reduziert sich in der Praxis durch fortgesetzte Rundungsfehler bei jeder durchgeführten Rechenoperation im Verlauf der Signalverarbeitungskette, jedoch erweist sich in der Implementierung die Qualität von 16 bit Datentypen bis hin zur Domaine des Demappings als mehr als ausreichend. Zwischen dem Ausgang des Demappers und dem ressourcenaufwändigen Viterbi-Algorithmus kann sogar eine Repräsentation mit nur 8 bit Wortbreite für die Soft Bit-Werte verwendet werden.

4.4.4 Schnittstellen des Demodulatorsystems

Als Eingangssequenz werden blockweise angelieferte IQ-Basisbanddaten (komplexwertig, 2×16 bit, Abtastrate $f_s = 2048$ kSa/s) erwartet. Dem Nutzer des Empfängerstack-Programmcodes werden über exportierte Funktionen der Listenabruf der verfügbaren Audio- beziehungsweise Datenservices ermöglicht. Über weitere Funktionen können gewünschte Services selektiert und entsprechend die Signalverarbeitungskette zum Decodieren der zugehörigen MSC-Anteile angewiesen werden. Ausgangsseitig stellt die Implementierung via Callback-Funktion die decodierten digitalen (Audio)Daten im 24 ms-Takt bereit.

4.5 Bewertung der Signalverarbeitungskette

Im Folgenden sollen die Kenngrößen der entwickelten Signalverarbeitungskette dargestellt werden, welche von der später eingesetzten Zielpattform unabhängig sind. Von der jeweilig genutzten Rechenplattform abhängige Größen sind dann im Speziellen die Laufzeitwerte, die in späteren Kapiteln angegeben werden.

4.5.1 Speicheranforderungen

Bis auf den Puffer des Basisbandsignals, welcher im Minimum einen kompletten DAB-Rahmen der Realzeitdauer 96 ms aufnehmen können muss, skalieren sämtliche anderen Puffer linear in Abhängigkeit von der gewünschten maximal durch den Empfänger zu unterstützenden Datenrate der zu demodulierenden Services. Eine Dimensionierung des Systems für Unterstützung einer als absolutes Maximum angenommenen DAB-Service-datenrate von 384 kbit/s resultiert zur Laufzeit in einer gesamten Arbeitsspeicherallokation von knapp unter 3 MB (Tabelle 4.4). Im entwickelten Programmcode erfolgt ausschließlich die Nutzung statischer Speicherallokation, sämtliche dynamische Speicherverwaltung ist explizit vermieden. Im Laufzeitbetrieb des Demodulators werden entsprechend keinerlei

Speicherbereich	Volumen	Begründet durch
Basisbandsignal bei $T_{\text{Buffer}}=256$ ms	2048 kB	$256 \text{ ms} \cdot 2048 \text{ kSa/s} \cdot 4 \text{ B/Sa}$
Zeit-Deinterleaver bei $r_b \leq 384$ kbit/s	640 kB	$(16 \text{ CIF} \cdot 1 \text{ B/b} \cdot 24 \text{ ms/CIF} \cdot r_b)$
FFT-Scratchpad bei $r_b \leq 384$ kbit/s, damit $n_{\text{S,FFT}} \leq 10$ Sym.	80 kB	$(10 \text{ Sym.} \cdot 2048 \text{ kSa/Sym.} \cdot 4 \text{ B/Sa})$
Viterbi-Trellis bei $r_b \leq 384$ kbit/s	72 kB	$(64 \cdot (1/8) \text{ B/b} \cdot 24 \text{ ms} \cdot r_b)$

Tabelle 4.4: Auflistung der wesentlichen benötigten Speicherressourcen des DAB-Decoders.

Speicherallokationen mehr vorgenommen. Dies führt zu einer erhöhten Softwarestabilität und ist außerdem effizient im Sinne einer hohen Verarbeitungsgeschwindigkeit [88]. Der Verzicht auf Methoden der dynamischen Speicherverwaltung geschieht ohne jegliche Beschneidung der Flexibilität, da sämtliche für die digitale Signalverarbeitung benötigten Speicherressourcen bereits vor der Laufzeit eindeutig vorherbestimmbar und deshalb im System ohnehin konstant vorzuhalten sind.

4.5.2 Vermessung der Empfangsgüte

Zur nachrichtentechnischen Verifikation der Signalverarbeitungskette wurde die Demodulationsgüte des entwickelten, mit effizienzoptimiert reduzierten Wortbreiten auf Festkommaarithmetik basierenden C-Codes vermessen. Abbildung 4.14 zeigt die gemessene Bitfehlerrate (engl. Bit Error Rate, BER) am resultierenden Empfänger in Abhängigkeit vom Signal-zu-Rauschleistungsverhältnis (engl. Signal-to-Noise Ratio, SNR) des Demodulatoreingangssignals. Die Rauschleistung N ist hierbei bezogen auf die OFDM-Systembandbreite von 1.536 MHz. Hinsichtlich des theoretischen nachrichtentechnischen Optimums liegt der Implementierungsverlust der vorgestellten Signalverarbeitungskette bei circa 0.3 dB, im Vergleich zu dem für DAB vorgeschlagenen Referenzempfangssystem der dritten Generation nach dem Implementierungsleitfaden ETSI TR 101 496-3 konnte eine um 0.4 dB verbesserte Demodulationsperformance erzielt werden [89]. Die hergeleitete Signalverarbeitungskette erfüllt somit mit einem sehr guten Ergebnis den Anspruch an eine qualitativ repräsentative und industrietaugliche Implementierung. Die nachrichtentechnische Performance der vorgestellten Software-Signalverarbeitungskette ist vor allem motiviert durch die großzügige Wahl einer 8 bit-Repräsentation der Soft Bit-Werte zwischen Demapper und Faltungsdecoder. In einer typischen Implementierung einer IC-Logikschaltung kann hier durch Wahl einer geringeren Bitbreite die Schaltungskomplexität im Decoder auf Kosten der Empfangsgüte gesenkt werden. Abgesehen vom isolierten Einzelbit stellt für eine Softwarelösung auf einem Prozessor typischerweise das Byte mit acht Bit die kleinste zu verarbeitende numerische Einheit dar (sogenannte Bytemaschine). Eine reduzierte Diskretisierung unterhalb von 8 bit, beispielsweise auf 3 bit, wäre auf einem Byte-orientierten Prozessor mit keinem Effizienzgewinn hinsichtlich der Berechnungsdauer verbunden. Die Validität der vorgestellten

4.5 Bewertung der Signalverarbeitungskette

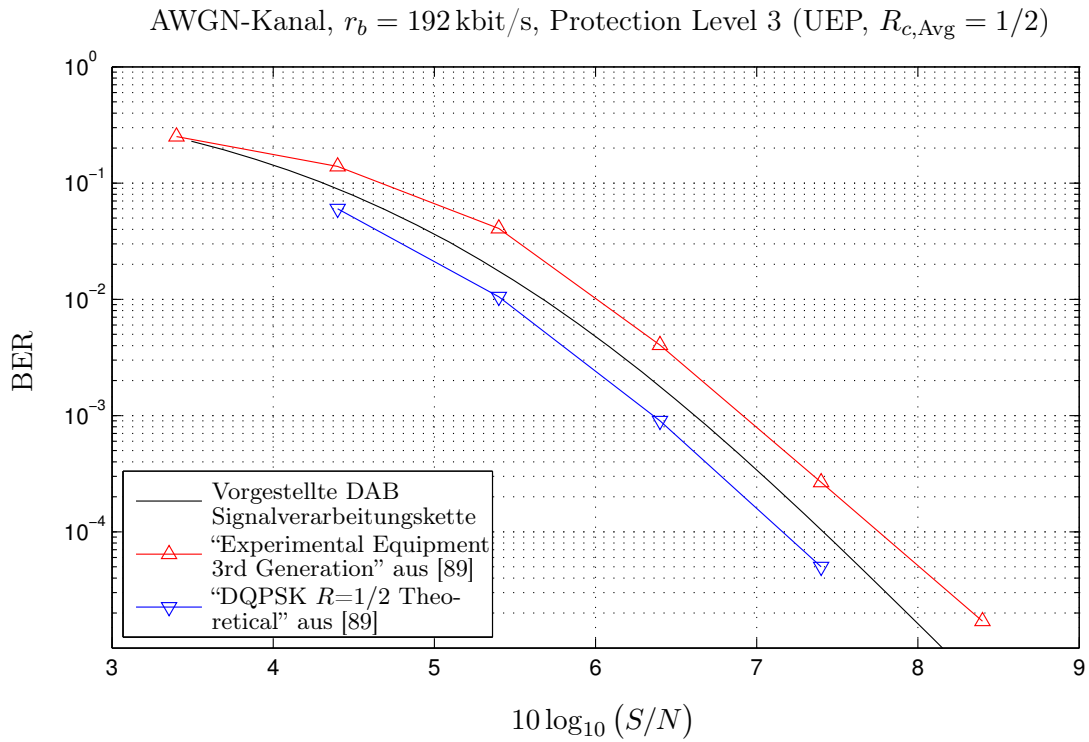


Abbildung 4.14: Vergleich der vorgestellten DAB-Signalverarbeitungskette mit Testequipment nach ETSI TR 101 496-3 [89] und der theoretischen Grenze für DQPSK-Modulation mit Faltungscodes für eine Coderate $R_c = 1/2$.

ten Signalverarbeitung konnte durch Messung mit Testsignalen und als Feldversuch in den bayerischen Gleichwellennetzen des Bayerischen Rundfunks, der Bayern Digitalradio GmbH und der Media Broadcast GmbH mit mobilem Empfängersystem für Fahrtgeschwindigkeiten von $v=200 \text{ km/h}$ erfolgreich nachgewiesen werden.

5 Evaluierung des Laufzeitverhaltens der x86-/GPU-Architekturen

5.1	Methodik der Laufzeitanalyse	108
5.1.1	Instrumentierung/Event-based Profiling	108
5.1.2	Statistisches Profiling	108
5.1.3	Profiling der CPU	109
5.1.4	Profiling der GPU	110
5.1.5	Auswahl relevanter Regionen für die Codeoptimierung	110
5.2	Intel Atom x86-/NVidia Ion GPU-Plattform	111
5.2.1	Hardwarearchitektur	112
5.2.2	Software und Betriebssystem	113
5.2.3	Evaluierung der Referenz-Hochsprachenimplementierung	116
5.2.4	Evaluierung der Variante mit SIMD-optimiertem SSE-Code	118
5.2.5	Evaluierung der Nutzung des GPU-Prozessors	125
5.3	AMD Fusion x86-/GPU-Plattform	135
5.3.1	Systemsoftware-Unterschiede zur Intel Atom-Plattform	136
5.3.2	Evaluierung der Hochsprachen-Referenzimplementierung	136
5.3.3	Evaluierung der Variante mit SIMD-optimiertem SSE-Code	138
5.3.4	Evaluierung der Nutzung des GPU-Prozessors	138
5.4	Zusammenfassung und Diskussion der x86-Plattformen	141
5.4.1	x86-/GPU-Plattform Intel Atom/NVidia Ion	141
5.4.2	x86-/GPU-Plattform AMD Fusion	141
5.4.3	Implementierungsempfehlung und erwarteter Ressourcenverbrauch	144

Die erarbeitete Verarbeitungskette aus DAB-Empfangsalgorithmen wird im vorliegenden Kapitel auf die Prozessormikroarchitekturen der x86-/GPU-Zielplattformen abgebildet. Es werden die folgenden, für ein eingebettetes Infotainmentsystem als geeignet eingestuft low-cost Zielplattformen betrachtet:

- Intel Atom-Systemplattform mit Zusatzoption NVidia Ion (chipsatzintegrierte GPU).

- AMD Fusion-Systemplattform (kombiniertes CPU/GPU-SoC).

Neben der trivialen Übertragung eines vom Prinzip portablen Hochsprachenmodells soll der Code intensiv architektur- und plattformspezifisch angepasst werden. Im Speziellen ist dies zum einen die Nutzung von Maschinensprache/SSE-SIMD-Befehlssatzerweiterungen der x86-CPU's sowie zum anderen die Verwendung der auf den Testplattformen vorhandenen freiprogrammierbaren Grafikbeschleuniger als Hilfssystem zur Signalverarbeitung. Ferner wird der Signalverarbeitungskern um Komponenten ergänzt, welche Basisbanddaten in Echtzeit von einem HF-Frontend empfangen und demodulierte Radiodaten via Interprozesskommunikation einer als externe Audio-/Datensenke agierenden Anwendung im System zur Verfügung stellen. Somit wird realitätsnah ein vollständiges Infotainment-Gesamtsystem dargestellt. Entsprechend fließen bei der Analyse auch Aspekte neben dem reinen Signalverarbeitungskern, beispielsweise die Last des Datentransports des Basisbandsignals, ein. Vorab soll für die anzuwendenden Techniken einer Laufzeitanalyse der theoretische Hintergrund geklärt werden. Anschließend erfolgt die Betrachtung der Intel Atom-Plattform mit Option einer integrierten NVidia Ion-GPU. Für die AMD Fusion-Plattform werden die Prozessorauslastungsuntersuchungen analog durchgeführt.

5.1 Methodik der Laufzeitanalyse

Als Ziel der Laufzeituntersuchung gilt die Angabe der für die verschiedenen Implementierungsvarianten und Elemente eines softwarebasierten DAB-Demodulators okkupierten Rechenzeitressourcen, das heißt der damit einhergehenden Auslastung der Prozessorarchitektur. Zur Analyse des Laufzeitverhaltens eines Prozessors stehen prinzipiell zwei Ansätze zur Wahl.

5.1.1 Instrumentierung/Event-based Profiling

Die erste Methode ist Instrumentierung beziehungsweise Event-based Profiling. Bei bestimmten Aktionen, wie beispielsweise dem Sprung in eine oder aus einer Funktionsroutine, sorgt speziell eingefügter Code dafür, dass Messdaten aufgezeichnet werden. Diese Messdaten sind in der Regel die Zeitstempelwerte einer Echtzeituhr mit hoher Auflösung.

5.1.2 Statistisches Profiling

Alternativ existiert die Methode der statistischen Sampling-Analyse. Während der Untersuchung wird in regelmäßigen Zeitabständen gemäß einer festgelegten Zeitbasis der Wert des Prozessor-Instruktionszeigers ausgelesen und diese Adresse abgespeichert. Nach der Sample Collection kann durch Nachbereitung der Daten jeder Instruktionszeigeradresse und somit jedem Zeitschlitz eine zugehörige Coderegion zugerechnet werden. Als Ergebnis einer Häufigkeitsanalyse entsteht eine Liste der systemweiten anteiligen Prozessorlast pro Programmcoderegion. Die erlangten Ergebnisse sind statistischer Natur, um ihre Qualität zu sichern soll eine Betrachtung hinsichtlich der theoretischen Hintergründe vorangehen. Betrachtet man jede zu untersuchende Coderegion für sich, so stellt jedes durch den Profiler gespeicherte Sample des Prozessorbefehlszeigers ein Bernoulli-Experiment dar: Die Speicheradresse des Befehlszeigers gehört entweder zum untersuchten Codefragment oder nicht.

Alle Experimente sämtlicher Zeitschlitze sind in ihrer Durchführung prinzipiell voneinander statistisch unabhängig. Für eine sehr häufige Durchführung $n \rightarrow \infty$ dieses Experimentes stellt die mittlere Erfolgswahrscheinlichkeit p dann die prozentuale, anteilige Prozessorlast der Coderegion dar. Zur Bestimmung der Schätzgüte der Realisierung von p kann folgende Überlegung angewendet werden: Der der Zufallsgröße p zugrunde liegende Zufallsprozess kann für eine Serie von Bernoulli-Experimenten mit der Annahme häufiger Versuchsdurchführung $n \rightarrow \infty$ und kleiner Erfolgswahrscheinlichkeit $p \ll 1$ bekanntermaßen mit Hilfe der Poisson-Verteilung angenähert werden (siehe beispielsweise in [90]).

$$\lim_{n \rightarrow \infty} \mathcal{B}_{n,p}(k) \approx \mathcal{P}_\lambda(k) \quad (5.1)$$

Diese lässt sich wiederum für häufige erfolgreiche Realisierungen $\lambda \gg 1$ durch eine Normalverteilung approximieren.

$$\mathcal{P}_\lambda(k) \approx \mathcal{N}(\mu = \lambda, \sigma^2 = \lambda) \quad (5.2)$$

Nach bekanntem Verfahren aus der Theorie der statistischen Tests lässt sich nun auf einfache Art und Weise bei einer gewünschten Sicherheit, der Wahrscheinlichkeit α , zu einem Schätzwert p und einer Versuchsdurchführungszahl n eine Fehlerschranke ϵ angeben.

$$\epsilon = \Phi(1 + \alpha/2) \sqrt{\frac{p(1-p)}{n}} \quad (5.3)$$

Dabei stellt $\Phi(\cdot)$ das Gauß-Integral dar. Bei Lösung des quadratischen Gleichungsproblems lässt sich nun für die gewünschte Sicherheit α bekanntermaßen ein Konfidenzintervall I der Schätzung von p angeben.

$$p_{1,2} = p \pm \epsilon, \quad I = [p_1; p_2] \quad (5.4)$$

Umgekehrt lässt sich hieraus für den gegebenen Anwendungsfall zu einer maximalen tolerierten Fehlerschranke die minimal nötige Anzahl an Versuchsdurchführungen, das heißt Sampling-Punkten n_{\min} , errechnen, die der statistische Profiler sammeln muss.

$$n_{\min} = \Phi^2(1 + \alpha/2) \frac{p(1-p)}{\epsilon^2} \quad (5.5)$$

Die zur Abschätzung des Laufzeitverhaltens für die Untersuchung einer Software minimal nötige zu überwachende Programmlaufzeit folgt in trivialer Weise umgehend aus dem verwendeten Samplingintervall T_{sampling} des Profilers.

$$T_{\min} = n_{\min} \cdot T_{\text{sampling}} \quad (5.6)$$

5.1.3 Profiling der CPU

Bei den untersuchten CPU-Systemen können Laufzeitmesswerte nicht in einfacher Weise durch Instrumentierung gewonnen werden, indem Codefragmente zur Speicherung des Performance Timer-Zeitstempels zu Beginn und Ende der zu untersuchenden Coderegion eingefügt werden und die ermittelte Zeitdifferenz anschließend ausgewertet wird. Zum einen kann die untersuchte Coderegion jederzeit ohne Notifizierung der Applikation durch das übergeordnet agierende Betriebssystem unterbrochen, pausiert und erst zu späterem Zeitpunkt fortgesetzt werden, was die Differenzzeitmessung verfälscht. Gründe hierfür liegen

in möglichen Unterbrechungsanforderungen (Interrupt Requests, IRQs) der Hardwareperipherie an die CPU sowie in der prinzipiellen Gegebenheit eines Multitasking-Systems, in dem verschiedene Tasks im Zeitschlitzbetrieb präemptiv geschedult werden. Ein zweites Argument resultiert aus dem Anspruch einer systemweiten Performanceanalyse, welche zum Ziel hat, neben der Applikation ebenfalls relevante fremde Codeanteile der Systemsoftware zu messen und auch im Zuge der Untersuchung unter Umständen neu als relevant zu identifizieren: In fremde binäre Programmmodule kann kein Code zur Instrumentierung für eine zeitstempelbasierte Messung eingebracht werden.

Als Methode der Wahl erscheint deshalb das statistische Profiling, da es nicht-intrusiv angewandt werden kann und bereits vom Prinzip die Gesamtheit des auf einem Prozessor ausgeführten Programmcodes erfasst. Um eine Basis für die spätere Versuchsdurchführung zu erlangen werden nun konkrete Werte festgeschrieben. Sie sollen in der weiteren Arbeit als untere Schranken für die Qualität der Messungen dienen:

Die Schätzung der Prozessorauslastung \hat{p} eines Programmteiles, welcher tatsächlich eine Last von $p = 1\%$ verbraucht, soll mit einer Sicherheit $\alpha = 99\%$ auf $\epsilon = \pm 0.1\%$ genau sein¹.

Für dieses Konfidenzmaß ergibt sich mit Gleichungen 5.5 und 5.6, dass für die Anzahl der Samplingpunkte des Profilers n stets die folgende untere Schranke zu erfüllen ist:

$$n > n_{\min} = \Phi^2(1 + 0.99/2) \frac{0.01(1 - 0.01)}{0.001^2} \approx 61875 \quad (5.7)$$

5.1.4 Profiling der GPU

Auf der GPU werden die Rechenkernel hingegen exklusiv ausgeführt. Es existieren keine Unterbrechungsanforderungen während der Laufzeit des Kernels. So kann einfaches Eventbasiertes Profiling über Performance Counter eingesetzt werden. Hierzu bieten die genutzten Entwicklungsframeworks entsprechend bereits native Unterstützung. Die Zeitmessungen des GPU-Profilers basieren auf durch den GPU-Treiber überwachten Zeitstempeln, welche den Bearbeitungsanfang und das signalisierte Bearbeitungsende des GPU-Kernels bestimmen.

5.1.5 Auswahl relevanter Regionen für die Codeoptimierung

Während der GPU-Code im entsprechenden C-Sprachdialekt CUDA beziehungsweise OpenCL plattformspezifisch angepasst wird, ist für die CPU zur Nutzung der SIMD-Eigenschaften ein Eingriff auf Maschinenbefehlsebene sinnvoll. Entsprechend der im Rahmen der vorliegenden Untersuchung beschränkten menschlichen Ressourcen wird der Schritt der arbeitsintensiven händischen CPU-Maschinencodeoptimierung nur für die in der Referenzimplementierung als laufzeitdominant identifizierten Codeanteile durchgeführt, andere Anteile des Quelltextes werden in Hochsprache C belassen.

Im Sprachgebrauch der Codeanalyse stellen solche sogenannten Hot Spots Programmanteile dar, denen ein signifikant überdurchschnittliches Maß an akkumulierter Prozessorzeit zuzurechnen ist. Meist sind dies Codepassagen, welche in hoher Frequenz wiederholt

¹So werden im Weiteren sämtliche in Prozent angegebenen Laufzeitmessergebnisse auf eine Nachkommastelle gerundet angegeben.

ausgeführt werden. Sie werden durch einen abwechselnden Entwicklungs- und Laufzeitanalyseprozess identifiziert und optimiert. Aufgrund des resultierenden starken Effektes einer Laufzeitreduktion solcher Coderegionen auf das Laufzeitverhalten der Gesamtapplikation sind sie die primären Kandidaten für die Durchführung von Codeoptimierungen. Die Optimierung weiterer Codeanteile birgt stets ein wirtschaftlich ungünstigeres Verhältnis des Nutzens der Optimierung zu eingesetzten Ressourcen des Programmierers, beziehungsweise sie können den gewünschten Effekt nahezu vollständig vermissen lassen [38].

5.2 Intel Atom x86-/Nvidia Ion GPU-Plattform

Die erste für die Evaluierung der Leistungsfähigkeit ausgewählte Prototypenplattform des low-cost x86-Segments basiert auf der Intel Atom N270 CPU, getaktet mit 1.6 GHz. Als Grafikkontroller ist die freiprogrammierbare integrierte GPU des Chipsets Nvidia Ion verbaut. Die Testplattform ist mit einer Bank DDR2-800 RAM ausgestattet, welche eine theoretische maximale Datentransferrate von 6400 MB/s bereitstellt. Diese teilen sich CPU und GPU². Tabelle 5.1 beschreibt im Detail die Charakteristika der vorliegenden Hardware.

Testplattform 1	
CPU: Intel Atom N270	
Mikroarchitektur	Intel Bonnell, x86/IA-32
Prozessorkerne	1
Kerntakt	1600 MHz
FSB-Takt	533 MHz
Betriebssystem	Microsoft Windows XP Service Pack 3
Code-Generierung	Microsoft C/C++ Optimizing Compiler 16.00
Laufzeitanalyse	Intel VTune Amplifier XE 2011
GPU: Nvidia MCP79 Ion	
Kerne (Streaming Multiprocessors)	2
Kerntakt	1100 MHz
Global Memory	host-mapped
Treiber	Nvidia Driver 286.19
Code-Generierung	Nvidia CUDA Toolkit 4.1
Laufzeitanalyse	Nvidia Visual Profiler 4.1
DRAM Arbeitsspeicher:	
Typ und Timing	DDR2-800, 5CL
Memory Channels	1 ("Single Channel")

Tabelle 5.1: Die im Dokument zur Evaluierung genutzte low-cost x86/GPU Plattform auf Intel Atom/Nvidia Ion-Basis.

²Als Spitzenwert der Speicherbandbreite, welche der chipatzintegrierten Nvidia Ion GPU real als Maximum zur Verfügung steht, wurde mittels Testkopieroperationen ein Wert 5420 MB/s gemessen.

5.2.1 Hardwarearchitektur

Abbildung 5.1 zeigt die allgemeinen Basiselemente einer x86-Systemhardware sowie auf welche Weise sich die GPU in die Systemarchitektur eines x86-Systems einfügt.

Baugruppen eines x86-Systems

An der Seite der eigentlichen CPU existieren weitere Funktionsblöcke, welche in den umgangssprachlich als Chipsatz bezeichneten Bausteinen zu finden sind. Die Kommunikation zwischen CPU und Arbeitsspeicher (RAM) wird vom Memory Controller Hub (MCH) geleitet, der zur CPU via Front Side Bus (FSB) verbunden ist. Der IO Controller Hub (ICH) konzentriert die verbleibenden langsameren Subsysteme. Dies sind beispielsweise die Host Controller für Massenspeichermedien (beispielsweise SATA) und USB-Peripherie. Daneben agiert er als Erweiterungsbus-Hub (typischerweise PCIe) zur Anbindung beliebiger systemerweiternder Hardware. MCH und ICH bieten Direct Memory Access (DMA) Einheiten, sodass Datentransfers von Peripheriegeräten in den oder aus dem Arbeitsspeicher ohne Belastung der CPU abgearbeitet werden können.

Rolle der GPU

Die GPU ist ein weiteres Datenverarbeitungssystem neben der CPU, welches über schnellen Zugriff zum Systembus verfügt. Sie hat ebenfalls direkten Zugriff zum Speichercontroller MCH des x86-Systems. Dies ist heute typischerweise über eine Anbindung mittels einer oder mehrerer elektrischer Verbindungen nach PCIe-Busstandard realisiert. Ein Integrated Graphics Processor (IGP) ist in die ICs des Chipsatzes oder der CPU vollständig integriert. Da low-end IGP-GPUs kein eigenes Framebuffer-/Video-RAM besitzen, nutzen sie hierfür über den MCH den Hauptarbeitsspeicher mit und teilen diesen mit der CPU³. Die IGP-GPU stiehlt bei Speicherzugriffen somit Anteile der der CPU verfügbaren Speicherbandbreite.

Der Grafikprozessor eines GPGPU-tauglichen x86-Systems scheint so aus Sicht der Architekturanbindung tatsächlich als im System potentiell der CPU ebenbürtig. Weiterhin erscheint das Konzept als eine mögliche Alternative zu bekannten Strukturen, in denen einem Applikationsprozessor zur Signalverarbeitung ein Basisbandprozessor-Subsystem separat zur Seite gestellt wird.

Anbindung eines HF-Frontends

Das klassische x86-System stellt eine reine Rechenplattform dar und muss um eine elektrische Antennensignalanbindung erweitert werden, damit ein vollständiges Radioempfangssystem dargestellt werden kann. Dies ist auch bei der bewertenden Vorabuntersuchung der auf der Plattform ausgeführten Signalverarbeitungskette notwendig, da beim realen

³Verschiedene Varianten der Chipintegration der genannten logischen Einheiten existieren. Die Funktionalität des MCH und des ICH, ursprünglich stets in getrennte ICs gesplittet, ist auch bekannt unter den traditionellen Namen North- und Southbridge. Für Kombinationen aus MCH und IGP wird oft der Begriff Graphics and Memory Controller Hub (GMCH) genutzt. Diese Kombinationsvariante implementiert der im Kapitel untersuchte NVidia Ion Chipsatz. Jüngst bilden MCH als auch IGP bereits ein gemeinsames SoC mit einem oder mehreren CPU-Kernen, wie bei dem später vorgestellten AMD Fusion-System.

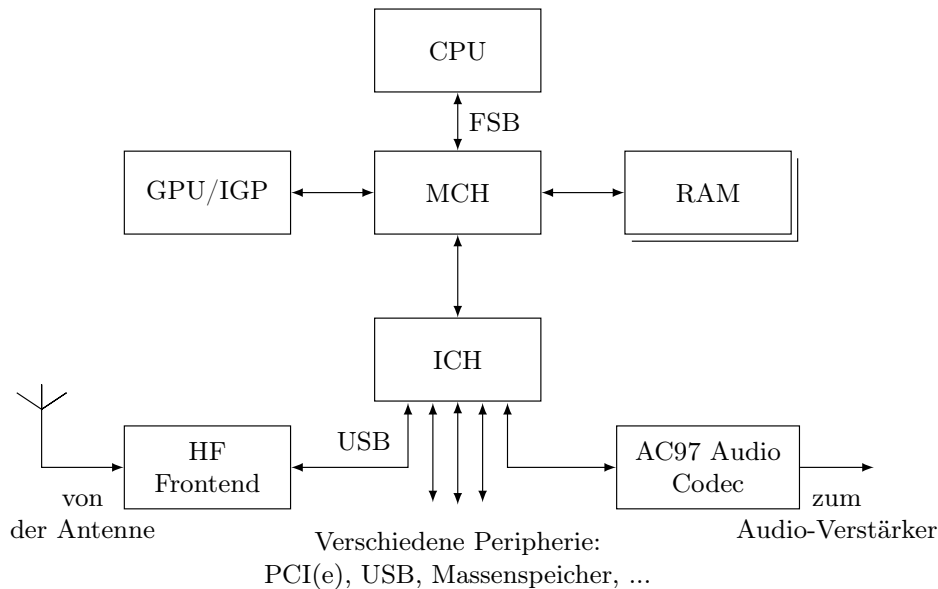


Abbildung 5.1: Systemarchitektur einer x86-Plattform mit integrierter GPU. Spezifisch für das vorliegende Anwendungsszenario ist ein Datenpfad von einem Antennensignal-Subsystem zur Audioausgabe hinzugefügt.

Betrieb auch reine Speichertransfers des digitalisierten Antennensignals die Hardwareressourcen des Mikrocomputergesamtsystems beanspruchen. Dies würde bei einer rein simulativen Evaluierung nicht realitätsnah berücksichtigt. Das für DAB notwendige Frontend liefert eine Basisbandrepräsentation des selektierten Empfangssignals in der Samplingrate $f_s = 2048 \text{ kSa/s}$. Die Inphase- und Quadratur-Anteile (IQ) der komplexwertigen Abtastwerte sind jeweils mit 16 bit quantisiert, so dass in Echtzeit kontinuierlich ein Datenstrom der Gesamtdatenrate von 64 Mbit/s an der Verarbeitungsplattform angeliefert wird. Für die Anbindung der x86-Plattform an eine Frontend-Baugruppe auf der gleichen Leiterplatte bzw. im selben Steuergerät bieten sich die Bussysteme PCIe und USB an. Aus Gründen der Flexibilität wird in der vorliegenden Untersuchung für eine exemplarische Anbindung über das USB-Subsystem entschieden und ein entsprechendes Wandlersystem genutzt.

5.2.2 Software und Betriebssystem

Die Intel-Plattform verwendet eine schlanke Installation von Microsoft Windows XP, Service Pack 3, als Betriebssystem. Zur Übersetzung des Programmcodes wurde der Microsoft Optimizing C/C++ Compiler 16.00 verwendet. Der Compiler wurde dabei stets parametrisiert zur Generierung von Code mit maximaler Ausführungsgeschwindigkeit (Option /O2). Für GPGPU-Unterstützung sind der NVIDIA Treiber Version 286.19 und das CUDA Toolkit 4.1 SDK [34] installiert.

Laufzeitanalyse

Als statistische Profilersoftware kommt für die Intel Atom CPU das herstellereigene Tool Intel VTune Amplifier XE 2011 zur Anwendung. Die Software bietet zur Analyse des Lauf-

5 Evaluierung des Laufzeitverhaltens der x86-/GPU-Architekturen

zeitverhaltens Zugriff auf interne Performance Monitoring Units (PMUs) der Intel CPU. Das benutzte Samplingintervall des Profilers beträgt $T_{\text{sampling}} = 1 \text{ ms}$. Für diesen Wert lässt sich mittels Gleichung 5.7 die minimal nötige Beobachtungs- und damit Messdauer umgehend angeben⁴.

$$T_{\alpha=0.99\%} = n_{\text{min}} \cdot T_{\text{sampling}} = 61875 \cdot 1 \text{ ms} \approx 62\text{s} \quad (5.8)$$

Wie bereits ausgeführt besitzt der Bonell-Prozessorkern des Intel Atom die hardwaregestützte Fähigkeit zur alternierenden Ausführung zweier Threads auf einem Prozessorkern, um die Rechenwerke auch in Zeiten auftretender Pipeline Stalls und Speicherzugriffslatenzen effizient zu nutzen. Jedoch ist es bei eingeschaltetem Hyperthreading unmöglich, in einer Messung CPU-Last akkurat einem der beiden quasisimultan ausgeführten Threads zuzuweisen. Vielmehr beeinflussen sich die beiden stets zugleich aktiven Threads gegenseitig in ihrem Laufzeitverhalten, weil die gemeinsam genutzte Prozessorressource je nach Befehlsstromtyp der einen oder der anderen Anwendung dynamisch (und somit stets unterschiedlich) vom Instruktionsscheduler zugewiesen wird. Aus diesem Grunde wird für die durchgeführten Messungen Hyperthreading auf dem Intel Atom deaktiviert. Da Hyperthreading jedoch den Instruktionsthroughsatz der CPU und somit die Performance nur steigert, stellen die präsentierten Ergebnisse hinsichtlich der Gesamtleistungsfähigkeit der CPU eine garantierte untere Schranke dar, welche im Normalbetrieb je nach Art einer zweiten parallel auszuführenden Multitaskingapplikation situationsabhängig überboten werden kann.

Coderegionen der Betriebssystem-Module

Um sowohl belastbare als auch faire Aussagen über die benötigte Auslastung der CPU-Ressourcen zu erhalten, zeigte es sich als nicht ausreichend, nur die eigenen Code-Anteile in das Laufzeitprofiling miteinzubeziehen. Neben einer eigentlichen Testanwendung mit ihren Signalverarbeitungsalgorithmen existieren weitere notwendigerweise auszuführende dritte Softwaremodule im realen Anwendungsszenario einer Mikrocomputerplattform. Dies sind insbesondere Teile des Betriebssystems. Zur Einordnung der vorgestellten Ergebnisse des systemweiten Codeprofilings wird im vorliegenden Abschnitt auf die im System ausgeführten fremden Codeanteile beziehungsweise Interna des verwendeten Host-Betriebssystems eingegangen werden, welche durch den Profiler im Späteren als relevant identifiziert wurden.

Das auf der Testplattform verwendete Betriebssystem Windows XP⁵ gehört zur Klasse der Systeme mit Microsoft Windows NT-basiertem Kern. Einen hierarchischen Überblick über die für die Untersuchung wesentliche Software-Architektur auf der NT-basierten Plattform gibt Abbildung 5.2. Die Teilkomponenten des Betriebssystems beziehungsweise dessen Erweiterungsmodule (Details finden sich beispielsweise in [91]) können in vier Untergruppen separiert werden:

⁴Diese Mindestanforderungen soll im Weiteren für alle durch statistisches Profiling gewonnenen Messungen erfüllt werden, indem eine weit längere Beobachtungsdauer $T=300\text{s}$ gewählt ist.

⁵Wie auch Windows 7 sowie die Varianten für eingebettete x86-Systeme Windows XP Embedded und Windows 7 Embedded.

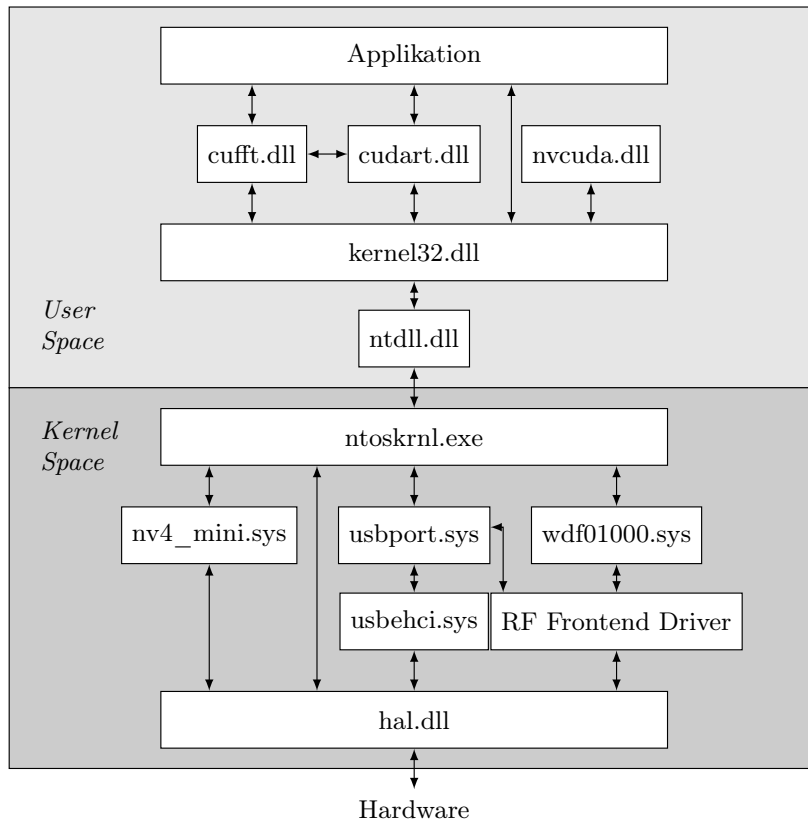


Abbildung 5.2: Überblick über die Hierarchie der für die Laufzeitprofilanalyse relevanten Codemodule von Betriebssystem und Gerätetreibern für den untersuchten Anwendungsfall CUDA-basierter GPGPU-Computings.

- Der Hardware Abstraction Layer (HAL) des im Kernel-Space ausgeführten Moduls `hal.dll` bietet ein einheitliches Interface für den Zugriff auf allgemeine lowest-level Funktionalität der Computerplattform, welche durch die jeweilig verbauten MCH- und ICH-Chipsätze zur Verfügung steht. Dies beinhaltet das Annehmen von Interrupt-Requests angeschlossener Geräte und die Zustellung dieser an das jeweilig zuständige Modul der Gerätetreibersoftware. Ebenfalls sind grundlegende Bustransaktionsmechanismen, beispielsweise für PCIe, implementiert.
- Der Programmcode des eigentlichen Windows Microkernels findet sich im Modul `ntoskrnl.dll`. Gemäß dem Konzept des Microkernels konzentriert es nur die zentralen, essentiellsten Funktionen des Betriebssystems, wohingegen sämtliche weitere Funktionalität in externe, nachladbare Module ausgelagert ist. Diese sogenannten Windows Executive Services beinhalten beispielsweise das Speichermanagement und den Prozess-Scheduler.
- Das Modul `ntdll.dll`, welches im nicht-privilegierten User Space ausgeführt wird, stellt eine Schnittstelle zu diesen Kernel-Services her. Diese Schnittstelle ist bekannt als die Windows Native API.

5 Evaluierung des Laufzeitverhaltens der x86-/GPU-Architekturen

- Ebenfalls bereits im User Space läuft das Modul `kernel32`. Es stellt die gewöhnliche API-Programmierschnittstelle bereit, welche typischerweise Windows Applikationsentwicklern bekannt ist⁶.

Zur Nutzung einer GPU samt ihrer GPGPU-Eigenschaft sind zugehörige Treiber notwendig. Sie sind kein originaler Bestandteil des NT-Betriebssystemkerns, sondern erweitern diesen.

- Der Grafiktreiber der im Späteren verwendeten GPU-Familie des Herstellers NVidia ist `nv4_mini.sys`.
- Wenn die Grafikkarte im GPGPU-Modus betrieben wird, ist zu beobachten, dass zusätzlich die folgenden Module zum Windows NT System nachgeladen werden: `nvcuda.dll`, `cudart.dll`.
- Falls von der Applikation genutzt stellt `cufft.dll` eine herstellereigene Bibliothek für die Fourier-Transformation dar.

Die folgenden Module sind spezifisch für den hier zu untersuchenden Anwendungsfall der Anbindung eines Radiofrontends über das USB-Subsystem.

- Um das genutzte USB-Radiofrontend in das System einzubinden ist ein entsprechender, eigener Gerätetreiber notwendig, der die Funktionalität des Frontends für die Plattform verfügbar macht. Der vorliegende Gerätetreiber baut auf die Architektur auf, welche durch das Windows Driver Framework `wdf01000.sys` bereitgestellt wird.
- Der Protokollstack für den zugrunde liegenden USB 2.0-Bus und der Port Driver des USB-Interfacechips sind in den Windows-Modulen `usbehci.sys` und `usbport.sys` implementiert.

5.2.3 Evaluierung der Referenz-Hochsprachenimplementierung

Bei der ersten Implementierungsvariante für die vorliegende Plattform wird als Signalverarbeitungskern der in Kapitel 4 vorgestellte C-Programmcode direkt compiliert. Wie dort beschrieben hat der Signalverarbeitungskern effizienzoptimierte mathematische Algorithmen, jedoch sind im Programmcode noch keinerlei architekturenspezifische Codemuster beinhaltet. Dieser Signalverarbeitungskern wird zur Zielapplikation gelinkt, welche die plattformspezifische Anbindung des Empfangsfrontends inklusive der Datenweiterleitung zum Signalverarbeitungscode, ein Steuerungsinterface sowie die Weiterleitung der demodulierten Audiodaten an einen applikationsexternen MPEG-Decoder realisiert.

⁶Die Einführung einer zweiten Abstraktionsschicht zwischen innerstem Betriebssystemkern und Anwendung, eines Kompatibilitäts-Layers, liegt in der Fähigkeit zur Betriebssystem-Virtualisierung begründet. Es können Applikationen mit unterschiedlichem Windows-Binärformat (16 bit, 32 bit, ggf. 64 bit) nebeneinander ausgeführt werden und andere Betriebssysteme emuliert werden (beispielsweise DOS oder OS/2).

DAB-Demodulation(FIC $r_b=32$ kbit/s, $R_c=\frac{1}{3}$ und MSC $r_b=192$ kbit/s, $R_c=\frac{1}{2}$)











Code-Bereich	Anteilige Prozessorauslastung
<i>CPU Intel Atom N270 ($f_{clk}=1.6$ GHz), C Referenzmodell:</i>	
Schätzung Δt -Fehler	0.1 % 
Schätzung Δf_c -Fehler	0.9 % 
Korrektur Δf_c -Fehler	2.3 % 
Fourier-Transformation	16.2 % 
DQPSK-Demapper	1.1 % 
Zeit-Deinterleaver	0.8 % 
Viterbi-Fehlerschutzdecoder	17.4 % 
in Summe: 38.6 %	
Treibermodule HF-Frontend	3.8 % 
Betriebssystem, HAL	1.2 % 
Betriebssystem, Sonstiges	3.4 % 
in Summe: 47.1 %	

Tabelle 5.2: Last des Intel Atom CPU-Kerns bei Betrieb der DAB-Empfängerapplikation ohne plattformspezifische Optimierung.**Bewertung**

Bereits ohne plattformspezifische Optimierungen des Signalverarbeitungskerns ist dieser in Realzeit auf der vorliegenden Plattform mit Intel Atom Prozessor lauffähig. Bei der evaluierenden Messung wurde der Empfang mit einer im typischen Betrieb als maximal angenommenen Datenrate des Audioservices von $r_b=192$ kbit/s durchgeführt (siehe Tabelle 4.3). Dabei wird die CPU annähernd zur Hälfte (47.1 %) ausgelastet⁷. Tabelle 5.2 zeigt in der ersten Hälfte die Aufschlüsselung der Prozessorauslastung nach einzelnen Teilalgorithmen der DAB-Signalverarbeitungskette (in Summe 38.6 %). Im zweiten Teil weist sie weiterhin die systemweiten Rechenlastanteile aus, welche durch den DAB-Empfang motiviert und so ebenfalls diesem zugerechnet werden müssen. Als Verwaltungs-Overhead des Betriebssystems, beispielsweise zum Garantieren des Zeitablaufs inklusive Scheduling, werden circa 3 % Last generiert. Die Behandlung des Transports der Basisbanddaten vom über USB angeschlossenen HF-Frontend in den Arbeitsspeicher belastet das System mit circa 4 % in den Treibermodulen des USB-Subsystems sowie circa 1 % im Bereich des HAL des Betriebssystems.

⁷Jedoch ist bei einer Bewertung einer Prozessorlast zu bedenken, dass in Multitaskingsystemen bereits weit unterhalb von 100 % Auslastung eine Reduzierung der Agilität beobachtbar wird und mit Verlust der Echtzeitfähigkeit zu rechnen ist. So ist beispielsweise für das Multitasking-Verfahren Rate Monotonic Scheduling eine Garantie für Echtzeitbetrieb nur bis zu circa 70 % Prozessor-Auslastung möglich [92].

Laufzeitdominante Coderegionen

Als laufzeitdominant offenbaren sich die folgenden Funktionsblöcke

- Fehlerschutzdecoder (Depuncturing, Viterbi Decoder, PRBS-Energy Dispersal),
- Schnelle Fouriertransformation FFT,

sowie, jedoch bereits weit weniger anspruchsvoll als die beiden erstgenannten Funktionsblöcke,

- Korrektur des Δf_c -Fehlers.

Besonderheiten der Mikroarchitektur wie die speziellen SSE-SIMD-Funktionseinheiten der x86-CPU finden durch triviale Übergabe des in C verfassten Grundmodells an den Compiler noch keine ausreichende Berücksichtigung. Entsprechend kann die gewonnene Implementierung alleine nicht als faires Maß für die Bewertung der Leistungsfähigkeit der CPU herangezogen werden, beispielsweise im Vergleich zur GPU. Die als Hot Spots identifizierten Teilalgorithmen müssen einer intensiven, architekturenspezifischen Optimierung unterzogen und anschließend erneut hinsichtlich ihres Laufzeitverhaltens untersucht werden.

5.2.4 Evaluierung der Variante mit SIMD-optimiertem SSE-Code

Gemäß der vollständigen Nutzung der anvisierten Mikroarchitektur Intel Bonell nutzt die nun vorgestellte Implementierungsvariante die Opcodes der SSE-Befehlssatzerweiterungen bis einschließlich der Revision SSE3. Es wurden die im vorigen Abschnitt identifizierten drei Hot Spots neu umgesetzt⁸. Bei der Durchführung der Codeoptimierung wurde der Datenfluss für die Nutzung dieser SIMD-Instruktionen angepasst beziehungsweise neu organisiert und durch händische Programmierung in Assembler eine annähernd optimale Wahl von Prozessorinstruktionen in den zeitkritischen Abschnitten sichergestellt. Dabei wurde auf einen möglichst hohen Anteil an SIMD-Vektoroperationen im kritischen Programmcodeabschnitt geachtet [38, 39, 51], um so dem nach dem Amdahl'schen Gesetz möglichen Geschwindigkeitsgewinn durch Parallelisierung bestmöglichst nahezukommen.

Optimierte Umsetzung des Fehlerschutz-Decoders

Wie in Kapitel 4 vorgestellt wurde, wird zu Beginn des Viterbi-Algorithmus in einem jeden Iterationsschritt ein Vektor von vier 8 bit Soft Bit-Werten $\vec{a} = (x_1, x_2, x_3, x_4)$ gebildet. Da es sich beim Auslesen des Quelldatenstroms im Zusammenhang mit dem unregelmäßig zusätzliche Datenpunkte einfügenden Depuncturing-Vorgang um eine sequentielle Operation handelt, erfolgt dies noch ohne Nutzung der SSE-Vektoreinheit. Das gewonnene 4-Tupel wird nun aus dem traditionellen 32 bit x86-Registersatz in denjenigen der SSE-Verarbeitungseinheit transferiert und dort in die niederwertigen 32 bit eines 128 bit Vektorregisters platziert. Gemäß der Algorithmenvorschrift werden nun sämtliche Linear kombinationen der Soft Bit-Werte zur Ermittlung der Pfadübergangsmetriken als Vektor

⁸Unabhängig von den SSE-Optimierungen der Signalverarbeitungsalgorithmen konnten zusätzliche kleinere Laufzeitverbesserungen gegenüber der ursprünglichen Implementierung bei den Blocktransfers des USB-Frontends erreicht werden.

gebildet. Sie dienen als Teilterme in den weiteren Berechnungen. Hinsichtlich $2^4 = 16$ möglicher Kombinationen können unter Berücksichtigung betragsidentischer negierter Paare acht unterschiedliche Teilterme identifiziert werden und so sind von Vektor \vec{b} nur acht Werte zu berechnen.

$$\vec{b} = \{a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 \quad \forall \quad a_1, a_2, a_3 \in \{-1, +1\}, a_4 \in \{-1\}\} \quad (5.9)$$

Die Bitbreite eines SSE-Registers wird durch einen Vektor $\vec{b} = \{b_1, b_2, \dots, b_8\}$ mit acht Elementen zu 16 bit Präzision vollständig genutzt. In diesem Schritt werden neben der trivialen arithmetischen Vektoradditionsoperation im Speziellen die Möglichkeiten der SIMD-Einheit zur Propagation der Bit-Präzision von Vektorelementen sowie der Permutation dergleichen verwendet.

Nun folgt der Kern des Viterbi-Algorithmus, die Akkumulierungs-, Vergleichs- und Maximierungs-Operationsfolge (ACS). Es werden die einzelnen Pfadübergangsmetriken aus \vec{b} genutzt, um für sämtliche 64 Faltungscodestände die zugehörigen Vektorelemente der akkumulierten Metriken $\vec{c} = \{c_1, c_2, \dots, c_{64}\}$ in jedem Decoderschritt zu propagieren. Diese werden jeweils durch 16 bit-Werte dargestellt, somit würden zur vollständigen Ablage der 64 Werte acht SSE-Register zu 128 bit benötigt. Da die SSE-Architektur jedoch insgesamt nur acht Arbeitsregister zur Verfügung stellt und auch Platz für Hilfsvariablen benötigt wird, ist ein Ausweichen auf Plätze im externen Speicher (beziehungsweise im Cache) bei der programmiertechnischen Umsetzung notwendig. Der bei der Berechnung nötige Zugriff auf die entsprechenden Indizes der Vektoren \vec{b} und \vec{c} ist stark verschränkt, er erfolgt als gegebene Funktionen aus der konkret verwendeten Faltungscodenvorschrift. Um datenparallele Arithmetikoperationen anzuwenden müssen in den Quellregistern Vektoren mit entsprechender Datensatzpassung arrangiert werden. Zur Bildung dieser jeweilig passend arrangierten Vektoren aus dem ursprünglichen Vektor \vec{b} und \vec{c} werden wieder intensiv Permutationsoperationen für die durch das SIMD-Register dargestellten Vektorinhalte genutzt. Die Zuordnungsfunktionen f_1, f_2, f_3, f_4 nach der Faltungscodenvorschrift sind durch die verwendeten SSE-Vektorpermutationsoperationen hart encodiert im Programmcode abgebildet. Nach vollständiger Rearrangierung sämtlicher Quelloperanden in den Registern beginnt der eigentliche datenparallele Propagierungsschritt von \vec{c} nach \vec{c}' . Die Arithmetikberechnung erfolgt mit zwei separaten Vektoradditionsoperationen, auch der komponentenweise Vergleich beider Vektoren und die Selektion des Maximums erfolgt in Vektorprozessierung.

$$c'_i = \max \{c_{f_1(i)} \pm b_{f_2(i)}, c_{f_3(i)} \pm b_{f_4(i)}\} \quad (5.10)$$

Für welchen Teilterm bei der Maximierungsoperation entschieden wurde, muss weiterhin in der Trellis-Matrix \mathbf{T} abgespeichert werden. Einer binären Vergleichsentscheidung entspricht eine Informationstiefe von nicht mehr als 1 bit. Wieder erfolgt die Nutzung von SSE-Instruktionen zur Datentyppropagation, um zuerst die Bitbreite aller 64 Vektorelemente von 16 bit auf 8 bit zu reduzieren und schließlich weiter auf eine Bitmaske von nur insgesamt 64 bit abzubilden. Somit wird das Ergebnis des Viterbi-Schrittes, eine Spalte von \mathbf{T} , effizient mit einem einzelnen Hauptspeichertransfer von nur 64 bit abgelegt. Neben der reduzierten Speicherbandbreite wird durch die kompakte Darstellung im Speicher auch das Potential hinsichtlich der Nutzung des volumenlimitierten Caches optimiert, im

Idealfall kann das gesamte Trellis-Zwischenergebnis sämtlicher Iterationsdurchläufe bis zur vollständigen Abarbeitung der Eingangssequenz komplett im Cache gehalten werden.

Dieses Zwischenergebnis \mathbf{T} wird sodann abschließend vom Viterbi-Backtrace ausgewertet. Da es sich wie in Kapitel 4 beschrieben um einen rein sequentiellen Vorgang handelt, können zu dessen Beschleunigung keine SSE-Vektoroperationen eingesetzt werden. Vielmehr wird auf eine effiziente Umsetzung in traditioneller Maschinensprache geachtet. Die Kette der im vorherigen Teilalgorithmus geschriebenen 64 bit-Worte wird vom Ende her eingelesen und aus diesem Wort jeweils eine einzelne Bitposition, entsprechend der relevanten Trellis-Entscheidung, ausgewertet. Die daraus resultierende Nutzdatensequenz wird Bit für Bit über ein Schieberegister zu vollständigen Datenbytes zusammengesetzt und byteweise abgespeichert. Dabei erfolgt zugleich die Exklusiv Oder-Verknüpfung mit der vorabgespeicherten PRBS-Folge der Energy Dispersal-Operation.

Optimierung des FFT-/Frequenzinterleaving-Blockes

Wie in Kapitel 4 erläutert, ist, bedingt durch das Frequenzinterleaving des DAB-Standards, nach der Transformation der Signalfragmente des Basisbandstromes in den Frequenzbereich eine Permutation der Informationsfolge durchzuführen. Deshalb wird für die Entwicklung einer schnellen Fourier Transformation entschieden, welche ihre Bit Reversal-Umsortierung am Ende durchführt, um diese und somit die resultierenden Speicheroperationen mit der Frequenzinterleaverpermutation verschmelzen zu können. Mit der Gegebenheit, dass die Eingangssequenz eine Folge im Zeitbereich darstellt, ergibt sich also eine FFT-Variante der sogenannten Decimation in Frequency (DIF)-Klasse. Im Referenzmodell der Signalverarbeitung war die schnelle Fourier-Transformation mit $N=2048$ Punkten als Standardvariante nach Cooley und Tukey, auch Radix 2-FFT-Algorithmus genannt, realisiert. Die Benennung Radix 2 deutet dabei darauf hin, dass der Gesamtalgorithmus so partitioniert ist, dass in jedem Berechnungsschritt stets zwei Datenpunkte miteinander verarbeitet werden. Detaillierte Abhandlungen über mathematisch effizienzoptimierte Berechnungsalternativen der Fourier-Transformation finden sich beispielsweise in [93] und [94]. Zur Effizienzsteigerung der Implementierung des Algorithmus bietet sich nun an, pro Berechnungsschritt eine höhere Anzahl $n>2$ von Datenpunkten zu Berechnungstermen zusammenzufassen:

- Hinsichtlich der mathematischen Gesamtkomplexität resultiert eine solche Radix n -Modifikation des ursprünglichen FFT-Algorithmus mit $n>2$ in einer leicht reduzierten Gesamtanzahl von Arithmetikoperationen. So benötigt beispielsweise ein Radix 4-Berechnungsschritt drei Multiplikations- und acht Additionsoperationen. Er ist äquivalent zu vier Radix 2-Schritten mit insgesamt vier Multiplikations- und acht Additionsoperationen. Somit senkt die Partitionierung des FFT-Algorithmus mit $n=4$ die Last durch Multiplikationsoperationen auf 75 %, während diejenige der Additionsoperationen unverändert bei 100 % bleibt [93].
- Wichtiger erscheint jedoch für die Praxis, dass auf einem Prozessor die Datenlokalität stark erhöht werden kann: Wird bei einem Radix 4-Berechnungsschritt das gesamte Datenset einmal in die Prozessorregister gelesen und als Zwischenergebnis wieder zurück in den vergleichsweise langsamen Hauptspeicher geschrieben, so muss bei der äquivalenten Variante unter Verwendung zweimal zweier Radix 2-Schritte doppelt so

oft das komplette Datenset im langsamen Arbeitsspeicher durchlaufen werden (siehe Abbildung 5.3).

FFTs mit Radix n -Elementen höheren Grades, wobei also N mit $n > 4$ faktorisiert wird (beispielsweise $n=16$), würden in diesem Sinne nach der Theorie weitere Gewinne zulassen. Entsprechend der anwachsenden Komplexität der Berechnungsterme steigt aber auch der Bedarf an internen Registern des Prozessors zum Halten der Zwischenergebnisse. Effizienzgewinne durch Einsatz einer höheren Radix n -Algorithmenvariante sind deshalb nur möglich, solange für die Radix n -Berechnung sämtliche Zwischenergebnisse vollständig in internen Registern des CPU-Kerns abgelegt werden können und nicht externe, langsame RAM-Speicherplätze miteinbezogen werden müssen. Dies kann auf der vorliegenden SSE-Befehlssatzarchitektur mit acht Arbeitsregistern, wie später erläutert, nur für eine Implementierung $n \leq 4$ erfüllt werden.

Der Algorithmus der vorliegenden FFT mit $N = 2^{11} = 2048$ Punkten wird dementsprechend als Split-Radix Verfahren mit der heterogenen Faktorisierung $N = 4^5 \cdot 2^1$ ausgedrückt:

- Es können fünf Radix 4-Stufen gebildet werden, welche so viele Paare aus ursprünglichen Radix 2-Stufen wie möglich zu Radix 4-Varianten zusammenfassen.
- Eine einzelne Stufe muss in gewohnter Weise als Radix 2 verbleiben. Da die letzte Radix 2-Stufe der Decimation in Frequency-FFT als Besonderheit ohne Multiplikationen auskommt, das heißt aus reinen Additionsoperationen besteht, wird diese ohnehin einen Sonderfall darstellende Stufe hierfür ausgewählt.

Im nativen FFT-Schema sind in sämtlichen Stufen der FFT in den Berechnungselementen, sogenannten FFT-Butterflies, unterschiedliche Datenpunkte zu verknüpfen. Durch geeignete Permutation der Indizes der Zwischenspeicherplätze lässt sich eine über sämtliche

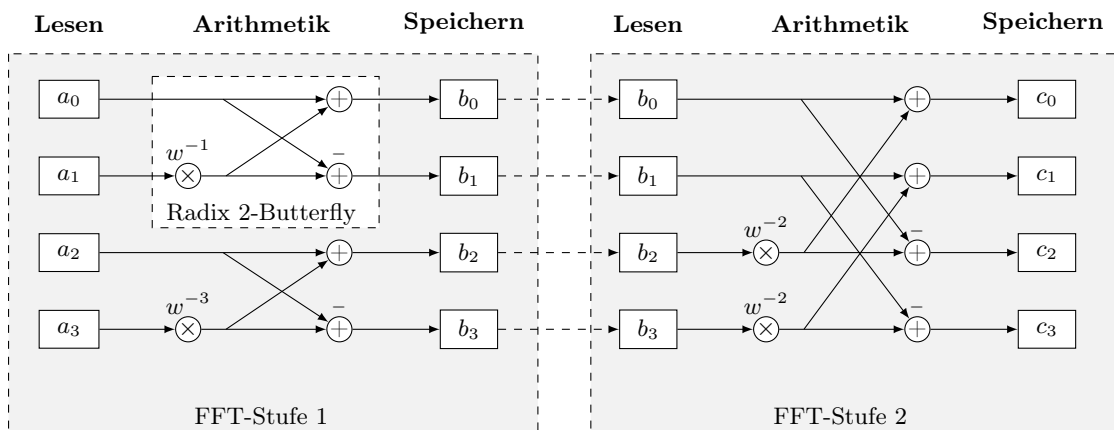


Abbildung 5.3: Insgesamt sind 11 Stufen zu 1024 strukturidentischen Radix 2-Berechnungselementen für die geforderte 2048 Punkt-FFT notwendig. Durch adäquate Gruppierung und Zusammenfassung bei der Berechnung kann die Datenlokalität benachbarter Radix 2-Elemente genutzt und Zugriffe auf den Hauptspeicher zum Speichern und Lesen von Zwischenergebnissen reduziert werden. Das Bild zeigt als Ausschnitt den Datenfluss zweier FFT-Stufen mit je zwei Radix 2-Elementen.

Stufen reguläre Struktur finden (Constant Geometry-FFT nach M.C. Pease [95]). Abbildung 5.4 zeigt schematisch das Prinzip für zwei Stufen einer Radix 4-FFT. Die im Lade- und Speichervorgang regularisierte Struktur bietet für die vektorisierte Berechnung Vorteile, wie später erläutert wird. Für die konkrete Umsetzung ist die regularisierte Constant Geometry-Struktur der Radix 4-Butterfly-Kerne nach Abbildung 5.4b als Blockmakro implementiert, daneben existiert Code zu dessen jeweiliger Parametrisierung sowie der Code der speziellen finalen Radix 2-Stufe. Zur Ausnutzung der vorhandenen Parallelität der 128 bit breiten SSE-Verarbeitungseinheit behandelt das Makro wie in Abbildung 5.4b stets vier unabhängige, benachbarte Radix 4-Terme parallel.

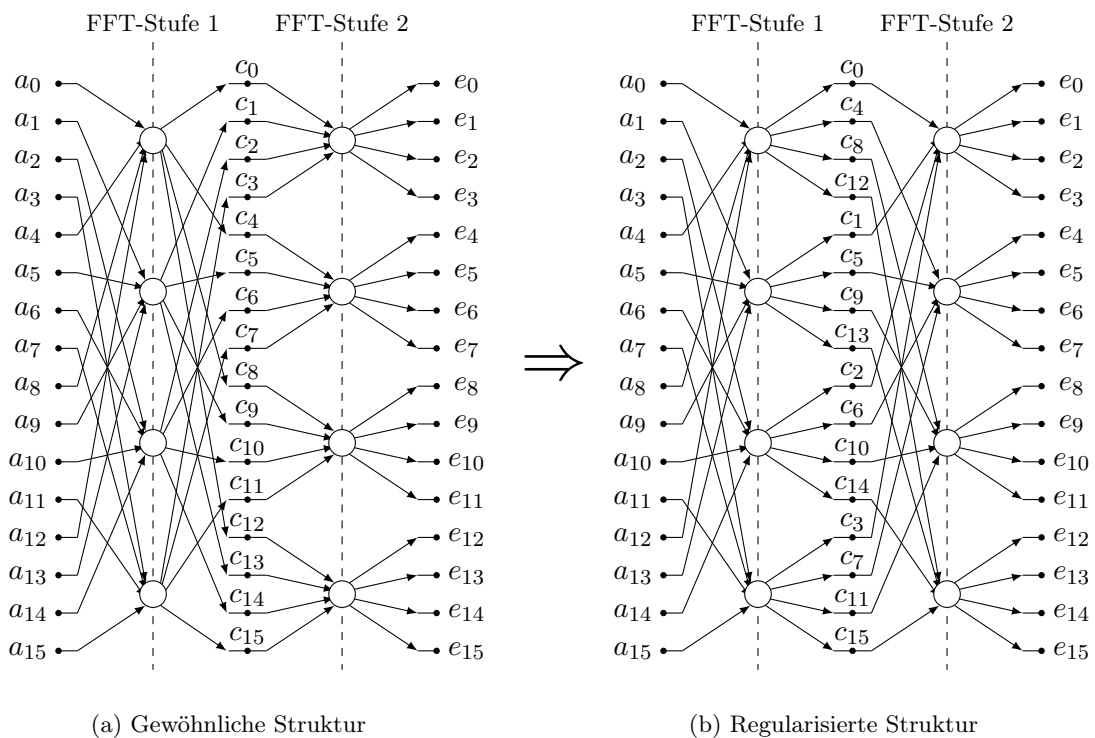


Abbildung 5.4: Durch Entflechten der Indexpositionen der Zwischenergebnisspeicherplätze lässt sich der Datenfluss der einzelnen FFT-Stufen vereinheitlichen.

Die vier Radix 4-Butterflies des Makros stellen vier mal vier Eingangs- und auch Ausgangsvariablen dar. Pro Rechenschritt werden also vier mal acht Skalare per SIMD-Vektoren nebenläufig verarbeitet. Die Hälfte des acht Register umfassenden SSE-Registersatzes der CPU ist so bereits durch Quell- und Zieloperanden belegt, es werden vier 128 bit Arbeitsregister benötigt. Viermal werden entsprechend der natürlichen Anordnung im Hauptspeicher vier komplexe Zahlenwerte mit je zwei verschachtelten 16 bit Real- und Imaginärteilkom-

ponenten nebeneinander in jeweils ein 128 bit SSE-Register (XMM0 bis XMM3) geladen.

$$\begin{aligned}
\vec{XMM0} &= \{\Re\{a_0\}, \Im\{a_0\}, \Re\{a_1\}, \Im\{a_1\}, \Re\{a_2\}, \Im\{a_2\}, \Re\{a_3\}, \Im\{a_3\}\} \\
\vec{XMM1} &= \{\Re\{a_4\}, \Im\{a_4\}, \Re\{a_5\}, \Im\{a_5\}, \Re\{a_6\}, \Im\{a_6\}, \Re\{a_7\}, \Im\{a_7\}\} \\
\vec{XMM2} &= \{\Re\{a_8\}, \Im\{a_8\}, \Re\{a_9\}, \Im\{a_9\}, \Re\{a_{10}\}, \Im\{a_{10}\}, \Re\{a_{11}\}, \Im\{a_{11}\}\} \\
\vec{XMM3} &= \{\Re\{a_{12}\}, \Im\{a_{12}\}, \Re\{a_{13}\}, \Im\{a_{13}\}, \Re\{a_{14}\}, \Im\{a_{14}\}, \Re\{a_{15}\}, \Im\{a_{15}\}\}
\end{aligned} \tag{5.11}$$

Dann erfolgt die eigentliche datenparallele Berechnung der vier Radix 4-FFT-Butterflies mittels SSE-Vektorarithmetik. Das erste Butterfly arbeitet nun stets auf den Vektorindizes 0 und 1 der SSE-Register, das zweite parallel berechnete Butterfly auf den SSE-Registerindizes 2 und 3, etc. Die zweite Hälfte des Registersatzes (XMM4 bis XMM7) wird innerhalb der Berechnung zwingend zum Fassen der Konstanten des Fourier-Drehzeigers (die sogenannten Twiddle-Faktoren) sowie von Zwischenergebnissen benötigt. Sie wird auch speziell dazu genutzt, unabhängige Teilinstruktionen nebeneinander abzubilden, um durch die dual-superskalar ausgelegte SSE-Einheit eine parallele Abarbeitung zu ermöglichen. Während der Berechnung werden aus einer vorausberechneten Tabelle in der benötigten Reihenfolge die Twiddle-Faktoren der FFT sequentiell eingelesen. Dabei kann für die vier benachbarten und parallel berechneten Radix 4-Butterflies näherungsweise stets der identische Twiddlefaktor eingesetzt werden. Der Speicherzugriff auf die Twiddlefaktortabelle erfolgt mit 32 bit und wird im 128 bit Register dupliziert. So wird die Größe der Tabelle sowie die nötige Datentransferrate um den Faktor vier reduziert. Nun liegen die Ergebnisse der Berechnungsstufe wie folgt in den Registern vor:

$$\begin{aligned}
\vec{XMM0}' &= \{\Re\{c_0\}, \Im\{c_0\}, \Re\{c_1\}, \Im\{c_1\}, \Re\{c_2\}, \Im\{c_2\}, \Re\{c_3\}, \Im\{c_3\}\} \\
\vec{XMM1}' &= \{\Re\{c_4\}, \Im\{c_4\}, \Re\{c_5\}, \Im\{c_5\}, \Re\{c_6\}, \Im\{c_6\}, \Re\{c_7\}, \Im\{c_7\}\} \\
\vec{XMM2}' &= \{\Re\{c_8\}, \Im\{c_8\}, \Re\{c_9\}, \Im\{c_9\}, \Re\{c_{10}\}, \Im\{c_{10}\}, \Re\{c_{11}\}, \Im\{c_{11}\}\} \\
\vec{XMM3}' &= \{\Re\{c_{12}\}, \Im\{c_{12}\}, \Re\{c_{13}\}, \Im\{c_{13}\}, \Re\{c_{14}\}, \Im\{c_{14}\}, \Re\{c_{15}\}, \Im\{c_{15}\}\}
\end{aligned} \tag{5.12}$$

Durch SSE-Instruktionen zur Datenpermutation werden die Registerinhalte passend zum verschachtelten Datenfluss am Eingang des Constant Geometry-Butterfly einer jeden FFT-Stufe (siehe Abbildung 5.4b) neu angeordnet. Durch temporäre Behandlung der Vektorregisterinhalte als Datenelemente zu 32 bit bleiben die Paare aus 16 bit Real- und Imaginärteilen während der Permutation benachbart.

$$\begin{aligned}
\vec{XMM0}'' &= \{(\Re\{c_0\}, \Im\{c_0\}), (\Re\{c_4\}, \Im\{c_4\}), (\Re\{c_8\}, \Im\{c_8\}), (\Re\{c_{12}\}, \Im\{c_{12}\})\} \\
\vec{XMM1}'' &= \{(\Re\{c_1\}, \Im\{c_1\}), (\Re\{c_5\}, \Im\{c_5\}), (\Re\{c_9\}, \Im\{c_9\}), (\Re\{c_{13}\}, \Im\{c_{13}\})\} \\
\vec{XMM2}'' &= \{(\Re\{c_2\}, \Im\{c_2\}), (\Re\{c_6\}, \Im\{c_6\}), (\Re\{c_{10}\}, \Im\{c_{10}\}), (\Re\{c_{14}\}, \Im\{c_{14}\})\} \\
\vec{XMM3}'' &= \{(\Re\{c_3\}, \Im\{c_3\}), (\Re\{c_7\}, \Im\{c_7\}), (\Re\{c_{11}\}, \Im\{c_{11}\}), (\Re\{c_{15}\}, \Im\{c_{15}\})\}
\end{aligned} \tag{5.13}$$

Dann kann das Ergebnis, in passender Datenanordnung für die Register der nächsten FFT-Berechnungsstufe, über Vektorspeichertransfers im Arbeitsspeicher abgelegt werden.

Korrektur des Trägerfrequenzfehlers

Die Remodulation zur Frequenzfehlerkorrektur basiert ähnlich dem Grundelement der FFT-Routine als Basisfunktion auf komplexwertiger Multiplikation mit einem Drehzeiger. Die Werte des komplexen Drehzeigers $e^{-jx} = e^{-j2\pi F_c \cdot k / L_S}$ aus Algorithmus 6 werden wie auch bereits bei der Referenzversion in C-Hochsprache aus einer vorgeschichteten Sinus/Cosinus-Tabelle als Funktion des Phasenwinkelarguments $F_c \cdot k$ entnommen. Da jedoch von Abtastwert zu Abtastwert die Schrittweite Δx innerhalb dieser LUT-Tabelle abhängig vom momentanen Frequenzfehler F_c ist und somit die benötigten Tabellenindizes nicht direkt benachbart sind, ist ein paralleles Laden als Vektor mit einem SIMD-Speicherzugriff nicht realisierbar. Entsprechend wird die Zusammenstellung des SIMD-Vektors aus den Drehzeiger-Tabelleneinträgern sequentiell ausgeführt.

Pro Symbol sind 2048 Abtastwerte im kontinuierlichen Basisbandstrom zu extrahieren und nach der Korrektur in die Eingangspuffer der FFT zu schreiben. Die Auswahl dieser Basisbandabtastwerte und der zugehörige Speicherzugriff ist eigentlich durch die von der Zeitsynchronisation ermittelte optimale Position des OFDM-FFT-Fensters genau vorgegeben. Der Prozessor kann jedoch nur Vektorspeicherzugriffe, welche sich im Arbeitsspeicher in ihrer Ausrichtung an 128 bit-Wortgrenzen orientieren, effizient ausführen. Deshalb wird eine bewusste marginale Fehlpositionierung des FFT-Fensters herbeigeführt, um in jedem Fall schnellstmögliche, an Wortgrenzen ausgerichtete Datentransfers zu ermöglichen. Der Zugriff auf die Folge aus komplexwertigen Zweiertupeln mit 2×16 bit erfolgt somit mit einem Fehler des Array-Index von ± 2 . Dies entspricht bezogen auf den DAB-Abtasttakt einem Zeitfehler im kontinuierlichen Basisbandstrom von äquivalent circa $\pm 1 \mu\text{s}$. Aufgrund des zyklischen Schutzintervalls $T_G = 0.246 \text{ ms}$ der OFDM-Symbole bleibt dieser bewusste Positionierungsfehler im nachrichtentechnischen Bezug ohne Folge.

Resultate und Bewertung der Codeoptimierungen

Die Untersuchung der Laufzeit des optimierten Codes mit Nutzung des SSE-Subsystems der CPU zeigt eine gegenüber dem nicht architekturenspezifisch optimierten C-Referenzmodell stark verbesserte Performance (Tabelle 5.3). Dies ist vor allen Dingen auf den nun maximal achtfachen internen Parallelitätsgrad der Datenverarbeitung zurückzuführen. Für den Funktionsblock des Viterbi-Decoders (Reduktion der CPU-Last von 17.36 % auf 2.71 %) und der Trägerfrequenzfehler-Korrektur (von 2.32 % auf 0.35 %) werden Beschleunigungsfaktoren von circa 6 bis 7 erreicht. Für die Berechnung der Fourier-Transformation konnte gar die zurechenbare CPU-Auslastung um den Faktor 12 (von 16.18 % auf 1.35 %) gesenkt werden. Hier zeigen sich neben der Parallelisierung des Programmcodes vor allem die Änderungen hinsichtlich der Speicherlokalität als sehr wichtiges Optimierungselement. Bezüglich der vollständigen DAB-Signalverarbeitungskette konnte mittels der vorgestellten Optimierungen eine Beschleunigung der Berechnung um mehr als den Faktor fünf (von einer CPU-Auslastung 38.6 % auf 7.3 %) durch die manuelle plattformspezifische Optimierung gegenüber dem direkt durch den Compiler aus dem ursprünglichen Hochsprachenreferenzmodell generierten Programmcode erzielt werden. Bezogen auf das komplette Digitalradio-Demodulationssystem inklusive des obligatorischen Basisbanddatentransports und des Betriebssystemoverheads verbleibt immer noch eine Reduktion der CPU-Last um nahezu den Faktor vier (47.1 % auf 12.1 %).

DAB-Demodulation(FIC $r_b=32$ kbit/s, $R_c=\frac{1}{3}$ und MSC $r_b=192$ kbit/s, $R_c=\frac{1}{2}$)











Code-Bereich	Anteilige Prozessorauslastung
<i>CPU Intel Atom N270 ($f_{clk}=1.6$ GHz), mit SSE-Assembler:</i>	
Schätzung Δt -Fehler	0.1 % 
Schätzung Δf_c -Fehler	0.9 % 
Korrektur Δf_c -Fehler *)	0.4 % 
Fourier-Transformation *)	1.4 % 
DQPSK-Demapper	1.3 % 
Zeit-Deinterleaver	0.7 % 
Viterbi-Fehlerschutzdecoder *)	2.7 % 
in Summe: 7.3 %	
Treibermodule HF-Frontend	2.7 % 
Betriebssystem, HAL	0.4 % 
Betriebssystem, Sonstiges	1.7 % 
in Summe: 12.1 %	
*) Optimiert unter Nutzung von SSE3-Assembler.	

Tabelle 5.3: Last des Intel Atom CPU-Kerns bei Betrieb der DAB-Empfängerapplikation mit Nutzung des SSE-Coprozessors.**5.2.5 Evaluierung der Nutzung des GPU-Prozessors**

In diesem Abschnitt soll nun die Einbeziehung der GPU der kombinierten Intel Atom/NVidia Ion-Plattform in das Signalverarbeitungssystem untersucht werden. Zur Nutzung der Rechenleistungsressourcen der GPU wird das NVidia CUDA-Framework genutzt. Die Entscheidung für CUDA C anstelle von OpenCL als Sprachdialekt der Umsetzung ist darin begründet, dass Voruntersuchungen zeigten, dass das CUDA C-SDK zum Zeitpunkt der Untersuchung das besser laufzeitoptimierte und besser an die Hardwarearchitektur angepasste Framework für die NVidia GPU darstellte.

Portierung der Signalverarbeitungskette auf die GPU

Einzelne GPU-Codefragmente, die sogenannten Kernels, können über das CUDA-Framework in die GPU geladen werden. Ihre Ausführung wird gemäß der Technik eines gewöhnlichen Remote Procedure Calls (RPC) von der Host-CPU initiiert. Die Signalverarbeitungsalgorithmen des C-Referenzmodells werden unter expliziter Deklaration paralleler Abschnitte mit dem Sprachdialekt CUDA C als GPU-Kernels neu formuliert. Um die Signalverarbeitung auf der GPU des Systems auszuführen sind neben der Anpassung der

Signalverarbeitung an sich eine Reihe von konzeptionellen Änderungen der Basissoftware nötig. Dies sind insbesondere die folgenden Modifikationen:

- Ein Ringpuffer im Hauptspeicher nimmt wie gewohnt die vom DMA erlangten Basisbanddaten des USB-Frontends entgegen. Dieser Ringpuffer, welcher sich im gemeinsam von CPU und GPU erreichbaren Hauptspeicher befindet, wird über das GPU-Framework als gemäß CUDA-Nomenklatur Host Pinned genannter Speicherbereich allokiert. Der physikalische Speicherbereich wird dadurch vom Paging-Mechanismus der virtuellen Speicherverwaltung des CPU-Betriebssystems ausgeschlossen⁹. Auf diese Weise kann die GPU beziehungsweise die auf ihr ausgeführten Kernel im Weiteren direkt und ohne vorherige Absprache mit der CPU auf die kontinuierlich gestreamten Basisbanddaten zugreifen, ein unnötiger weiterer Kopiervorgang zwischen Speicherbereichen entfällt.

In gleicher Weise wird derjenige Pufferspeicherbereich allokiert, welcher dafür vorgesehen ist, die vom GPU-Demodulatorsystem decodierten Nutzdaten der CPU zurückzuführen¹⁰.

- Der Referenzcode nutzt zur Signalverarbeitung Festkommaarithmetik, vorwiegend mit 16 bit Datentypen. Jedoch zeigte sich erwartungsgemäß, dass diese Datentypen nicht der Mikroarchitektur der initial als Fließkommaprozessor entworfenen GPU gerecht werden. Da Fließkommaoperationen performanter ausgeführt werden, nutzt die GPU-Portierung zur Signalverarbeitung gemäß dieses nativen Datentyps der GPU nun Fließkommawerte der kleinstmöglichen Präzision 32 bit.
- Die CPU agiert als Master, welcher den Aufruf der GPU-Signalverarbeitungskette initiiert und diese steuert. Sämtliche Rechenalgorithmen sind von der CPU auf die GPU ausgelagert, allein der Algorithmus zur Detektion des DAB-Rahmenbeginns im Basisbandstrom verbleibt auf der CPU¹¹. Der Algorithmus benötigt nur wenig Ressourcen und ist zudem aufgrund des eingesetzten Moving Average-Algorithmus mit sequentiell-iterativem Charakter nicht für die GPU in eine geeignete datenparallele Struktur umformbar. Die durch die GPU prozessierten Daten korrespondieren jeweils hinsichtlich des Signals mit einer Realzeit von 96 ms, einem kompletten DAB-Rahmen entsprechend.
- Um den Setup-Overhead der GPU-Kernelaufrufe gering zu halten sind Funktionsblöcke soweit wie möglich in gemeinsamen GPU-Kernels kombiniert. Die Separierung

⁹Das hinter CUDA Host Pinned Memory stehende Prinzip ist identisch zu der bekannten Art und Weise, gemäß derer Pufferspeicher bei Gerätetreibermodulen für gewöhnliche Peripherie-Controller (wie in Festplatteninterfaces oder Soundkarten) umgesetzt werden: Speicherbereiche sind im physikalischen Speicheradressraum vor Relokation durch die virtuelle Speicherverwaltung zu schützen, solange der Direct Memory Access (DMA) durch die Peripheriehardware andauert.

¹⁰Das gesamte Ausmaß an Pinned Memory und somit hart gesperrten Speicherseiten des Hauptarbeitsspeichers beträgt bei der Implementierung weniger als 3 MB. So sind im Gesamtsystem keine adversen Effekte auf die RAM-Verwaltung des Betriebssystems in Folge einer exzessiven Zahl für dynamische Speicherverwaltungszwecke gesperrter Speicherseiten zu erwarten.

¹¹Auch die Algorithmen zur initialen Akquisition werden nicht auf die GPU portiert. Sie werden nur ein einziges Mal bei Start oder Sendekanalwechsel ausgeführt und zeigen sich dementsprechend und erwartungsgemäß als im Gesamtbild der Signalverarbeitungskette nicht laufzeitmaßgeblich.

des Gesamtalgorithmus zu einzelnen GPU-Kernels erfolgt nur dort, wo die Struktur der DAB-Signalverarbeitungskette es erfordert. Dies sind diejenigen Stellen, an denen sich der individuelle Parallelitätsgrad und somit die Dimension der GPU-Workload ändert.

Aufbau und Berechnungsmethodik der GPU-Kernel

Abbildung 5.5 gibt eine Übersicht über die hier vorgestellte, die GPU nutzende Variante der Signalverarbeitungskette. Die einzelnen GPU-Kernel werden im Folgenden erläutert.

- **Schätzung des Trägerfrequenzfehlers:** Das fortgesetzte Schätzen und Nachführen des Fehlers Δf_c zwischen der Oszillatorfrequenz des Frontends und der tatsächlichen Empfangsfrequenz basiert, wie bereits in Kapitel 4 beschrieben, auf einer Autokorrelation des Basisbandsignals. Die hierzu nötige Summe über komplexwertige Multiplikationsoperationen ist aufgrund der vollständigen Datenunabhängigkeit der Einzelterme gut geeignet für eine parallelisierte Berechnung. Um für die finale Summenbildung den Vektor zu einem skalaren Wert zu reduzieren wird das klassische Schema der Parallel Sum Reduction, siehe [67], angewendet.

- **Korrektur des Trägerfrequenzfehlers:** Auch die Remodulation zur Korrektur des geschätzten Trägerfrequenzfehlers basiert, wie in Kapitel 4 dargelegt, im Kern auf der Anwendung von vollständig disjunkten, komplexwertigen Multiplikationsoperationen der Signalabtastwerte mit Drehzeigern. Dies kann leicht parallel in CUDA C ausgedrückt werden für alle Symbole $i \in I$ und dort jeweils alle Abtastwerte eines Symbols $k = 0, 1, \dots, L_{\text{Sym}} - 1$. Jeweils eine Operation ohne jegliche weitere Datenabhängigkeit wird pro Basisband-Abtastwert angewendet. Durch Test und Analyse auf der genutzten GPU zeigt sich die Verarbeitung der Workload aller $k=2048$ Abtastwerte eines OFDM-Symbols am effizientesten bei der Aufteilung pro SM in 256 parallele GPU-Threads, welche dann jeweils 8 Abtastwerte sequentiell bearbeiten.

Ähnlich der vorgestellten SSE-Implementierung wird auch bei der GPU-Implementierung die ursprünglich geschätzte Position des OFDM-FFT-Fensters im Basisbandstrom so angenähert beziehungsweise modifiziert, dass die Zugriffe auf den Basisband-Pufferspeicher an Wortgrenzen ausgerichtet sind und der GPU-Hardware effiziente Vektorspeichertransfers ermöglicht werden.

- **Fourier-Transformation:** Für die benötigte Umsetzung einer diskreten Fourier-Transformation mit $N=2048$ Punkten wird auf die CUFFT-Bibliothek [96] zurückgegriffen, die der GPU-Hersteller Nvidia selbst bereitstellt und welche somit als quasioptimale FFT-Implementierung auf der vorgesehenen Hardware angesehen wird.
- **DQPSK-Demapper:** Die Demodulation der einzelnen OFDM-Subträger aller Indizes K nach Gleichung 4.47 erfolgt unabhängig voneinander. Sie kann somit effizient auf der GPU für alle K parallelisiert dargestellt werden. Jedoch muss in diesem Verarbeitungsschritt weiterhin das im DAB-Standard [2] angewandte Frequenzinterleaving der Subträgerindizes erfolgen und das Datenset entsprechend einer Permutation unterzogen werden. Dies wird durchgeführt, während die demodulierten Daten von

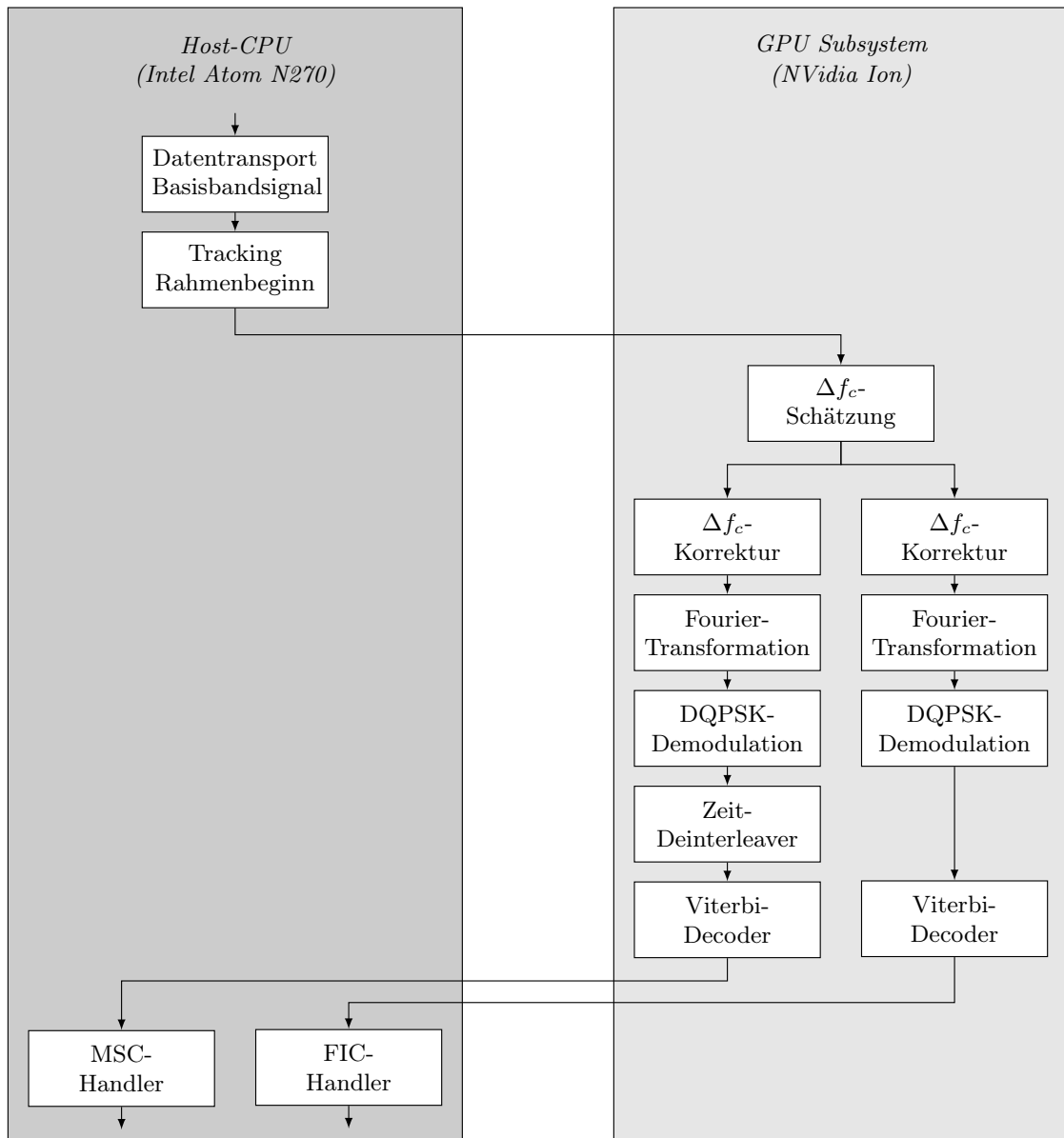


Abbildung 5.5: Funktionsschema der Variante der DAB-Empfangskette mit Auslagerung der Signalverarbeitungskern von der Host-CPU auf die GPU.

der GPU zurück in den Speicher geschrieben werden. Die Permutationsvorschrift des Scattering-Schemas ist entsprechend ihrer nachrichtentechnischen Aufgabe extrem irregulär konstruiert und es zeigt sich, dass dies nur schwer in Einklang zu den nativen parallelen Speicherzugriffsmustern der GPU [67] zu bringen ist.

Die Workload-Aufteilung erfolgt so, dass pro SM ein Symbol i bearbeitet wird, die 1536 datentragenden OFDM-Unterträger K jedes Symbols i werden dabei auf 64 parallele GPU-Threads aufgeteilt. Auf der verwendeten GPU-Mikroarchitektur zeigte sich in Laufzeittests des Kernels keine verbesserte Performance, wenn die Aufteilung der Workload eine Anzahl Threads höher als 64 pro SM nutzt.

- **Zeit-Deinterleaver:** Für denjenigen Teil der DAB-Wellenform, welche mit dem MSC-Segment des Rahmens korrespondiert, ist ein zweiter Descrambling-Mechanismus anzuwenden, der die demodulierten Datenbits zeitlich verschachtelt.

Obwohl auch hier die Speicherzugriffe des zugehörigen Scattering-Algorithmus vom Prinzip strikt datenparallel sind, müssen erneut die von der GPU-Hardware favorisierten Speicherzugriffsmuster durch die pseudozufälligen Permutationsregeln des DAB-Standards verletzt werden. Bei der Programmumsetzung sind entsprechend während des Dateneinlesens, bei der die Permutation erfolgt, Speicherzugriffslatenzen nicht zu vermeiden. Beim Zurückschreiben in den Speicher kann ein effizientes paralleles, vollständig kombiniertes Ablegen der Datenvektoren erfolgen. 64 Threads pro SM zeigen sich auf der vorliegenden GPU-Hardware erneut als eine gute Aufteilung der Workload hinsichtlich der Laufzeitperformance des Kernels.

- **Viterbi-Fehlerschutzdecoder:** Hinsichtlich des Faltungscodes beinhaltet ein DAB-Rahmen des Mode I stets vier in sich abgeschlossene Codewörter derselben Struktur, jeweils für den FIC und jeden Subchannel des MSC. Es bestehen keine Datenabhängigkeiten und somit müssen bei der getrennten Decodierung keine Informationen ausgetauscht werden. So bietet sich für die Partitionierung der Workload die Decodierung eines einzelnen Codeworts pro SM der GPU an. Die Gruppen aus jeweils vier Codewörtern identischer Struktur werden dem GPU-Framework für parallele Stapelverarbeitung auf den SMs präsentiert¹².

Der Faltungscod zur Fehlerkorrektur besitzt 64 Zustände und entsprechend müssen pro Schritt 64 Zustandsmetriken des Hidden Markov-Modells propagiert werden. Zur parallelen Berechnung dieser fällt die Entscheidung zwangsweise für eine Dimensionierung von 64 GPU-Threads pro SM. Einem Thread fällt eine spezielle Funktion zu: Er muss sämtliche nicht-parallelisierbare Aufgaben des Decoders übernehmen. Während dieser Zeit müssen die weiteren Threads des SM warten. Dies korrespondiert auf der der GPU zugrunde liegende SIMD-Prozessorarchitektur damit, dass dann nur eines der Rechenwerke des Vektorprozessors aktiv arbeitet.

Das Einlesen der Quelldaten in jedem Arbeitsschritt des Decoders beginnt, indem dieser eine Thread das zwangsweise sequentielle Depunktieren der Eingangssequenz durchführt. Die prozessierten Eingangsdaten werden im Anschluss über das Shared Memory der GPU direkt den anderen Threads, dies meint im Hardwarebezug

¹²Die vorliegende GPU mit zwei SMs ist so sicher hinsichtlich Parallelitätsgrad stets vollständig ausgelastet, ebenso eine GPU-Hardware, welche die doppelte Anzahl SM implementieren würde.

die Elemente des SIMD-Rechenwerkes, als gemeinsame Datenbasis zur Verfügung gestellt. Gemäß des nachrichtentechnischen Konzeptes der Transinformation muss der Faltungsdecoder nun die bislang in vorigen Schritten gewonnenen Metrikinformationen zwischen sämtlichen bislang getrennten Berechnungs-Threads des SM austauschen. Dieser Speicherzugriff ist von seiner Struktur nicht regulär, sondern stark verschränkt (siehe auch Gleichung 5.10) und dementsprechend mit Latenzen im GPU-Verarbeitungsablauf verbunden. Um hiernach sicherzustellen, dass die aktualisierte Information zwischen allen parallelläufigen GPU-Threads getauscht ist, muss im CUDA-Code ein Synchronisationsbarrierenbefehl (engl. Sync Barrier) eingefügt werden. Nach der anschließenden, nebenläufigen Berechnung der propagierten Metrikwerte werden diese über einen gemeinsamen parallelen Schreibvorgang für den nächsten Iterationsschritt im Shared Memory der GPU abgelegt.

Nachdem sämtliche Eingangsdaten prozessiert sind und die Matrix des Viterbi-Trellis komplett erstellt wurde, endet der parallelisierbare Teil des Fehlerschutzdecoders. Der notwendige Backtrace, welcher allerdings vergleichsweise wenig arithmetische Berechnungen erfordert, wird erneut durch den speziellen Einzelthread zur Behandlung der sequentiellen Aufgaben bewältigt.

Analyse der Ausführung der GPU Kernels

Zuerst werden die Berechnungsroutinen aus der für sie gewöhnlichen Perspektive des GPU-Profilings untersucht. Gegenstand der Laufzeitmessung ist die Verarbeitung eines DAB-Rahmens der Realzeitdauer 96 ms. Tabelle 5.4 zeigt im Detail die Kernel-Laufzeiten auf der GPU. Daneben sind die Parallelitätsdimensionen¹³ sowie die vom Profiler ermittelte Speicherdurchsatzrate aufgezeigt. Mehrere Algorithmen sind zweifach gelistet, da sie jeweils einmal zur Prozessierung des FIC-Datenblocks und einmal zur Prozessierung des MSC-Datenanteils vom DAB-Decoder verwendet werden. Die chronologische Abfolge der Kernelaufufe zeigt Tabelle 5.5. Die Zeitleiste startet mit Initiierung des ersten GPU-Kernelaufufs durch die Host-CPU und endet, sobald das demodulierte Ergebnis der CPU als verfügbar signalisiert ist. Somit sind in der Zeitleiste die durch den Gerätetreiber für das Setup der GPU aufzuwendenden Latenzen enthalten. Die Gesamtzeitdauer von Initiierung bis zur Signalisierung des Ergebnisses beträgt $\Delta t=28.1$ ms. Dabei führt die GPU während 25.9 ms aktiv Berechnungen durch. Bezogen auf die Dauer eines DAB-Rahmens in Realzeit $T_F=96$ ms ist der GPU-Kern somit zu 26.9% ausgelastet.

Erwartungsgemäß ihres ursprünglichen Einsatzgebietes¹⁴ zeigt sich auch im vorliegenden Anwendungsfall die GPU-Mikroarchitektur mit einem sehr geringen Laufzeitmesswert als besonders gut geeignet für die Berechnung des FFT-Algorithmus. Der Viterbi-Fehlerschutzdecoder sticht in den Messwerten durch lange Laufzeit hervor, er ist der bei weitem dominante Berechnungskernel der GPU-Implementierungsvariante. Aufgrund des Sammelns und Verteilens von Transinformation im Algorithmus des Viterbi-Decoders entstehen für die strikt auf Datenparallelität entwickelte GPU-Mikroarchitektur ungünstig

¹³Dies drückt nochmals im Detail aus, wie die Workload auf die individuellen SM-Prozessoren (Inter-SM Parallelitätsgrad/Grid Size) und innerhalb eines SMs auf SIMD-Threads (Intra-SM Parallelitätsgrad/Block Size) partitioniert wurde.

¹⁴Entsprechend der Nutzung von 2D-FFT-Operationen zur Texturfilterung im Umfeld der 3D-Grafikberechnung.

DAB-Demodulation

(FIC $r_b=32$ kbit/s, $R_c=\frac{1}{3}$ und MSC $r_b=192$ kbit/s, $R_c=\frac{1}{2}$)

GPU Compute Kernel Nvidia MCP79 Ion	Laufzeit (in μ s)	Inter-SM Parallelität (Grid Size)	Intra-SM Parallelität (Block Size)	Lese- durchsatz (in MB/s)	Schreib- durchsatz (in MB/s)
Schätzung Δf_c -Fehler	213.5	1	256	1120	0.14
Fast Information Channel:					
Korrektur Δf_c -Fehler	63.34	4	256	600.7	965.2
Fourier-Transformation	114.6	4	128	527.8	527.8
DQPSK-Demapper	213.7	3	64	1390	532.7
Viterbi-Fehlenschutzdecoder	3827	4	64	97.1	193.2
Main Service Channel (hier $n=4$):					
Korrektur Δf_c Fehler	303.9	$4 \times (n+1)=20$	256	733.7	1155
Fourier-Transformation	484.4	$4 \times (n+1)=20$	128	626.3	626.3
DQPSK-Demapper	1201	$4 \times n=16$	64	1069	540.9
Zeit-Deinterleaver	1175	4	64	131.0	1079
Viterbi-Fehlenschutzdecoder	18280	4	64	89.3	243.0

n : Anzahl zu extrahierender OFDM-Symbole je CIF-Rahmen des DAB-Multiplexsignals
(gemäß Time Slicing-Betrieb abhängig von Service-Bitrate r_b und Codierungsrate R_c)

Tabelle 5.4: Profiling der DAB-spezifischen Signalverarbeitungskernel für die Demodulation eines 96 ms DAB-Rahmens auf der x86/GPU-Plattform Intel Atom N270/NVidia MCP79 Ion.

DAB-Demodulation

(FIC $r_b=32\text{ kbit/s}$, $R_c=\frac{1}{3}$ und MSC $r_b=192\text{ kbit/s}$, $R_c=\frac{1}{2}$)

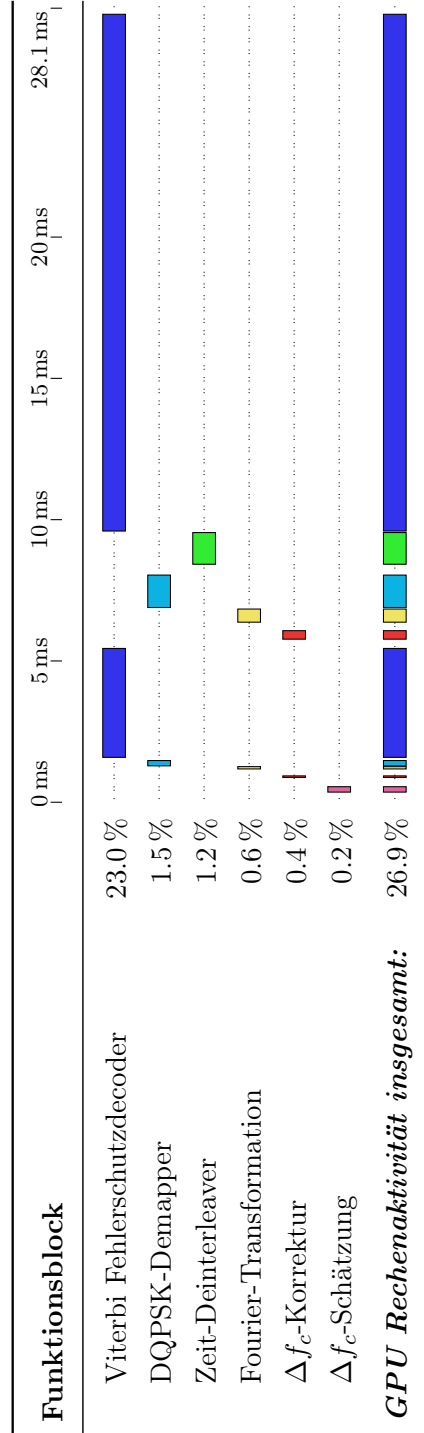


Tabelle 5.5: Gemessene GPU-Rechenaktivität der Compute-Kernel im chronologischen Ablauf für die Demodulation eines 96 ms DAB-Rahmens auf der NVIDIA Ion-Plattform.

verschränkte Speicherzugriffe, auch sind die bit-orientierten Logikoperationen nur umständlich umsetzbar. Weiterhin sind die Teiloperationen des Depunktierens nach DAB-Standard und des Trellis-Backtrace schlicht Teilalgorithmen rein sequentieller Natur. Folglich besteht hier keinerlei Möglichkeit, das nach Theorie vorliegende Leistungspotential der für datenparallele Algorithmen ausgelegten GPU-Mikroarchitektur erreichen zu können.

Analyse der Auswirkungen auf die Host-CPU

Neben der reinen Untersuchung der Ausführungscharakteristika der GPU-Kernel innerhalb des GPU-Subsystems selbst sollen zusätzlich die Auswirkungen der GPU-Nutzung auf die Host-CPU untersucht werden. Erwartungsgemäß sollte die Auslagerung der Algorithmen weg von der CPU hin auf die GPU eine günstige Entlastung auf Seite der CPU bewirken. Die anteilige GPU-Prozessorlast ist in der ersten Hälfte der Tabelle 5.6 nach Funktionsblock summiert abgebildet, die Resultate der Untersuchung der CPU-Auslastung während deren Ausführung auf der GPU zeigt die zweite Hälfte der Tabelle 5.6.

DAB-Demodulation

(FIC $r_b=32$ kbit/s, $R_c=\frac{1}{3}$ und MSC $r_b=192$ kbit/s, $R_c=\frac{1}{2}$)












Code-Bereich	Anteilige Prozessorauslastung
SDR-Prozessor NVidia Ion MCP79 ($f_{CLK}=1.1$ GHz):	
Δf_c -Schätzung	0.2 % 
Δf_c -Korrektur	0.4 % 
Fourier-Transformation	0.6 % 
DQPSK-Demapper	1.5 % 
Zeit-Deinterleaver	1.2 % 
Viterbi-Fehlerschutzdecoder	23.0 % 
in Summe: 26.9 %	
Host-CPU Intel Atom N270 ($f_{CLK}=1.6$ GHz):	
DAB-SDR-Applikation	0.3 % 
Treibermodule & Laufzeitumgebung GPU	6.2 % 
Treibermodule HF-Frontend	3.0 % 
Betriebssystem, HAL	12.5 % 
Betriebssystem, Sonstiges	7.5 % 
in Summe: 29.6 %	

Tabelle 5.6: Vergleich der Lastanteile der DAB-Softwaredemodulation für die Variante unter Nutzung der GPU der NVidia Ion-Plattform.

5 Evaluierung des Laufzeitverhaltens der x86-/GPU-Architekturen

Wie auch bei der Untersuchung der Code-Varianten, welche allein auf der CPU ausgeführt werden, lässt sich auch eine Systemlast durch Codeanteile extern zur Hauptanwendung erkennen. Dabei muss nun dem durch GPU-Betrieb generierten Overhead spezielle Beachtung geschenkt werden.

- ***DAB-SDR-Applikation***: Dies ist die Code-Region der eigenen DAB-Testapplikation. In den Fällen der Signalverarbeitung ohne GPU-Unterstützung wurden hierfür 38.1% an anteiliger CPU-Last für die Referenzimplementierung sowie 7.4% für die SSE-optimierte Implementierung ausgewiesen. Es kann eindeutig beobachtet werden, dass durch Ausführung der Signalverarbeitung auf der GPU die Last der eigentlichen Arithmetikroutinen erfolgreich fort von der CPU ausgelagert wurde. Die anteilige CPU-Last der DAB-Applikation schrumpft auf marginale 0.3%. Bei alleiniger Betrachtung dieser eigenen Codebasis scheint die These einer Entlastung der CPU durch Auslagerung von Algorithmen auf die GPU vorerst bestätigt.
- ***Treibermodule und Laufzeitumgebung GPU***: Mit Einführung der GPU-Beschleunigung verbringt die CPU nun zusätzlich 6.24% der Zeit in Coderegionen des NVidia GPU Software Frameworks (externe Module `nvcuda.dll`, `cudaart.dll`, `cufft.dll`, `nv4_mini.sys`).
- ***Treibermodule HF-Frontend***: Erwartungsgemäß ist die CPU-Auslastung, welche eindeutig allein der Funktionalität des prototypischen HF-Frontends zurechenbar ist und die Datenakquisition von diesem via USB-Bus behandelt, unverändert stabil bei circa 3.5% (`usbport.sys`, `usbhci.sys`, `wdf01000.sys` sowie das der Frontend-Hardware direkt zugehörige Treibermodul).
- ***Betriebssystem, Hardware Abstraction Layer (HAL)***: Ausgenommen von der Betrachtung sonstiger Betriebssystemelemente wird das Modul des Hardware Abstraction Layers `hal.dll`. Ohne Nutzung der GPU waren 1.17% CPU-Last zurechenbar, motiviert hauptsächlich durch die Hardwaretransaktionen des HF-Frontends am USB-Buscontroller. Dem HAL-Modul wird besondere Beachtung zuteil, da sein CPU-Lastanteil sich mit Nutzung der GPU um den Faktor zehn erhöht (12.47%).
- ***Betriebssystem, Sonstiges***: Verschiedener Overhead, bedingt durch das Betriebssystem (Module `ntoskrnl.exe`, `ntdll.dll`, `kernel32.dll`), benötigte vormals 2.79% für die C-Referenzimplementierung und 1.66% CPU-Last für die SSE-Variante. Mit GPU-Nutzung steigt die im Betriebssystem-Code verbrachte anteilige Prozessorzeit nun auf 7.51%.

In Summe ergibt sich auf der untersuchten low-cost CPU/GPU-Plattform eine totale CPU-Auslastung bei Nutzung der GPU von 29.6%. Diese Beobachtung überrascht insbesondere vor dem Hintergrund, dass keinerlei Rechenoperationen der Signalverarbeitung mehr auf der CPU ausgeführt werden und selbst die optimierte, alleine die CPU nutzende SSE-Variante mit 12.1% bedeutend weniger CPU-Gesamtsystemlast verursachte.

5.3 AMD Fusion x86-/GPU-Plattform

Im Anschluss wird die auf dem AMD E-350 Zacate basierende CPU/GPU-Plattform untersucht. Auf der Seite der CPU besitzt sie zwei AMD Bobcat x86-Kerne. Sie stellt somit in der Produktklasse das direkte Konkurrenzprodukt zur Intel Atom-Familie dar, weist jedoch wie bereits in Kapitel 3 dargelegt in der internen Mikroarchitektur einige Unterschiede auf. Zugleich bietet sie in direkter SoC-Integration eine programmierbare Grafikeinheit nach der AMD R800-Architektur. Zur Bewertung werden die entwickelten Varianten der Signalverarbeitung des DAB-Softwareradios nun auf diesem System hinsichtlich ihres Laufzeitverhaltens untersucht. Die Messdurchführungen erfolgen in Analogie zu denjenigen auf der Intel Atom-Plattform. Die Angaben zur Prozessorauslastung sind stets, auch im Folgenden, zum Zwecke der besseren Vergleichbarkeit auf einen einzelnen CPU-Kern bezogen. Für die vorliegenden Dualcore-CPU E-350 entsprechen 100% CPU-Last somit der Volllast eines der beiden Prozessorkerne beziehungsweise einer halben Auslastung des gesamten CPU-SoC.

Die AMD-Plattform wird mit einer schlanken Windows 7-Installation betrieben. Für

Testplattform 2	
CPU: AMD Fusion E-350	
Mikroarchitektur	AMD Bobcat (x86/IA-32 als auch x86-64)
Prozessorkerne	2
Kerntakt	1600 MHz
FSB-Takt	nicht anwendbar, MCH in SoC integriert
Betriebssystem	Microsoft Windows 7 Professional SP1
Code-Generierung	Microsoft C/C++ Optimizing Compiler 16.00
Laufzeitanalyse	AMD CodeXL 1.1.2885
GPU: AMD Radeon HD 6310, integriert in AMD E-350 SoC	
Mikroarchitektur	AMD R800
Kerne (DPPs)	1
Kerntakt	500 MHz
Global Memory	host-mapped
Treiber	AMD Radeon HD Driver 9.12.0.0
Code-Generierung	AMD Accelerated Parallel Processing SDK 2.5
Laufzeitanalyse	AMD CodeXL 1.1.2885
DRAM Arbeitsspeicher:	
Typ und Timing	DDR3-800, 6CL
Memory Channels	1 ("Single Channel")

Tabelle 5.7: Die zweite evaluierte Systemplattform auf Basis des kombinierten x86/GPU AMD Fusion-Konzeptes.

die GPGPU-Nutzung sind AMD-Treiber 9.12.0.0 und das OpenCL 1.2-konforme Framework des AMD Accelerated Parallel Processing SDK 2.5 [35] vorhanden. Zur CPU-Laufzeitanalyse wird die Software namens CodeXL des Chipherstellers AMD in der Version 1.1.2885 eingesetzt, welche äquivalente Funktionalität zu Intels VTune bietet. Das angewendete Profilersamplingintervall beträgt ebenfalls 1 ms. Das Tool bietet des Weiteren die Möglichkeit der Untersuchung der Laufzeit des Codes der GPU und wird hierzu im Folgenden genutzt. Tabelle 5.7 fasst die Hauptcharakteristika hinsichtlich Hardware und der für diese eingesetzten Software zusammen.

5.3.1 Systemsoftware-Unterschiede zur Intel Atom-Plattform

In Revisionen des Betriebssystems bis Windows XP wurde die jeweils zur vorliegenden Hardwareausprägung der Plattform gehörende Datei des HAL stets bei der Installation einheitlich in `hal.dll` umbenannt. In Windows 7 wird diese Umbenennung nicht durchgeführt, das jeweilige funktional äquivalente Modul wird direkt verwendet und trägt unterschiedliche Namensvariationen. Deshalb ist in der später genutzten AMD-Testplattform mit Windows 7 das Modul `halmacpi.dll` als HAL beobachtbar. Es findet sich auch statt `ntoskrnl.exe` im Profileroutput `ntkrnlpa.exe` als genutzter Microkernel. Dieses Binary stellt eine interfacekompatible Variante des Windows-Betriebssystemkerns dar, welche von Microsoft mit Support für Physical Address Extension (PAE)¹⁵ kompiliert wurde.

Der Grafiktreiber der im Späteren verwendeten AMD/ATI GPU der Familie Radeon R800 ist das kernel-mode Modul `atikmdag.sys`. Für den Betrieb im GPGPU-Modus wird er unterstützt durch die nachgeladenen Codeanteile `amdocl.dll` für die OpenCL-Schnittstellenfunktionalität sowie `dxgmms1.sys`, `dxgkrnl.sys` und `atidxx32.dll` für den Zugriff auf die Grafikkarte über das Framework des Multimediainterfaces DirectX. Mit marginalem Anteil an CPU-Last (<0.1%) sind außerdem die Module `atimpag.sys` und `atigktxx.dll` zu beobachten.

Mit Unterstützung von USB Version 3.0 durch das Betriebssystem wird der Bus und somit das angeschlossene HF-Frontend anstelle von `usbehci.sys` durch das Treibermodulsystem `nusb3xhc.sys` bedient.





















5.3.2 Evaluierung der Hochsprachen-Referenzimplementierung

In Anlehnung an die Untersuchung der Intel Atom-Plattform soll zuerst erneut die allein durch den Compiler aus dem C-Referenzmodell abgeleitete Implementierung ohne x86-spezifische Quellcodeoptimierungen untersucht werden. Tabelle 5.8 zeigt in der oberen Hälfte hierzu die anteilige Prozessorauslastung nach Algorithmen der Signalverarbeitung aufgeschlüsselt. Zusätzlich sind die Lastanteile für Betriebssystem und Betrieb des Radiofrontends angegeben.

Der AMD Bobcat-Kern wird in Summe zu 33% ausgelastet. Damit setzt sich bei identischer Software des C-Modells ohne x86-spezifische Optimierung die AMD-Mikroarchitektur gegenüber der Intel Bonell-Mikroarchitektur (CPU-Last 47%) positiv ab.

¹⁵Unterstützung für x86-Systeme mit der Fähigkeit zur Nutzung von mehr als 4 GB Arbeitsspeicher.

DAB-Demodulation(FIC $r_b=32$ kbit/s, $R_c=\frac{1}{3}$ und MSC $r_b=192$ kbit/s, $R_c=\frac{1}{2}$)

Code-Bereich	Anteilige Prozessorauslastung
CPU AMD E-350 ($f_{clk}=1.6$ GHz, ein Kern), C Referenzmodell:	
Schätzung Δt -Fehler	0.1 % 
Schätzung Δf_c -Fehler	0.4 % 
Korrektur Δf_c -Fehler	1.4 % 
Fourier-Transformation	10.1 % 
DQPSK-Demapper	1.2 % 
Zeit-Deinterleaver	0.6 % 
Viterbi-Fehlerschutzdecoder	13.6 % 
in Summe: 27.4 %	
Treibermodule HF-Frontend	3.1 % 
Betriebssystem, HAL	0.2 % 
Betriebssystem, Sonstiges	2.3 % 
in Summe: 33.0 %	
CPU: AMD E-350 ($f_{clk}=1.6$ GHz, ein Kern), mit SSE Assembler:	
Schätzung Δt -Fehler	0.1 % 
Schätzung Δf_c -Fehler	0.5 % 
Korrektur Δf_c -Fehler *)	0.2 % 
Fourier-Transformation *)	1.1 % 
DQPSK-Demapper	0.3 % 
Zeit-Deinterleaver	0.2 % 
Viterbi-Fehlerschutzdecoder *)	3.8 % 
in Summe: 6.1 %	
Treibermodule HF-Frontend	1.5 % 
Betriebssystem, HAL	0.1 % 
Betriebssystem, Sonstige	1.8 % 
in Summe: 9.5 %	

*) Optimiert unter Nutzung von SSE3-Assembler.

Tabelle 5.8: Gemessene Prozessorauslastung eines x86-Kerns des AMD E-350 SoC bei Ausführung des reinen C-Codes des Referenzmodells, das heißt ohne die Nutzung der SIMD-Fähigkeiten der SSE-Einheit als auch des speziell SSE-optimierten Codes.

5.3.3 Evaluierung der Variante mit SIMD-optimiertem SSE-Code

Anschließend erfolgt die Untersuchung der hinsichtlich der Verwendung von SSE-Vektoroperationen speziell architektureoptimierten Codevariante (siehe Tabelle 5.8, unterer Teil). Aufgrund der kompatiblen Befehlssatzarchitektur kann direkt die für die Intel x86-Plattform erarbeitete Implementierung übernommen werden. In der optimierten Variante lastet der DAB-Demodulationsvorgang den CPU-Kern zu 9.5% aus. Die AMD Bobcat-Mikroarchitektur schneidet somit bei gleicher Taktung erneut besser ab als die Intel Bonell-CPU (CPU-Last 12%), jedoch fällt der Unterschied im Verhältnis etwas geringer aus.

5.3.4 Evaluierung der Nutzung des GPU-Prozessors

Anders als für die beiden vorangegangenen Untersuchungsgegenstände des Systemteils der x86-CPU kann für die Evaluierung der GPU nicht direkt auf die Softwareimplementierungen aus den Voruntersuchungen zur Intel Atom/NVidia Ion-Plattform zurückgegriffen werden. Es wird eine neue, syntaktisch gezielt auf das AMD Framework hin angepasste Softwarevariante abgeleitet. Die GPU-Kernel sind implementiert mit Hilfe der Beschreibungssprache OpenCL. Semantisch und algorithmisch folgt die Codevariante trotzdem im Wesentlichen allen Parallelisierungsparadigmen wie sie auch für die CUDA Implementierung auf der NVidia GPU galten. Für die FFT wird nun auf die plattformspezifische Bibliothek *clAmdFft* des GPU-Herstellers AMD zurückgegriffen [36].

Laufzeitanalyse der GPU-Kernel

In äquivalenter Form der Angaben zur NVidia GPU zeigt Tabelle 5.9 die detaillierten Informationen und Messungen zu den einzelnen GPU-Kernels auf der SoC-integrierten AMD Radeon HD6310-GPU auf. Die Implementierung zeigt ähnliches Verhalten wie das System der NVidia GPU: Der Fehlerschutzdecoder erweist sich als laufzeitlimitierendes Glied der Signalverarbeitungskette, 28% von 31% GPU-Auslastung werden konsumiert durch den DAB-standardkonformen Viterbi-Algorithmus mit Eingangssequenzdepunktierung. Die restlichen, aber nicht für die Gesamtlaufzeit maßgeblichen Funktionsblöcke haben nach den vom AMD CodeXL-Profiler gelieferten Werten im Vergleich sehr gute Performanceergebnisse (siehe Tabelle 5.10, oberer Teil).

Laufzeitanalyse der Host-CPU

Erneut wird auch das Verhalten der Host-CPU während der Ausführung der GPU-Routinen untersucht. Die Ergebnisse sind in Tabelle 5.10, unterer Teil, zusammengefasst. Die Analyse erfolgt in direkter Analogie zur Methodik der Intel Atom/NVidia Ion-Plattform mit entsprechender Interpretation der Profiler-Ausgaben unter Berücksichtigung der Software-Unterschiede gemäß Abschnitt 5.3.1.

Interessanterweise zeigt die Host-CPU des AMD E350-SoC im GPGPU-Betrieb eine vergleichsweise geringe Last von 10.6% im Kontrast zur Intel Atom mit NVidia Ion GPU (29.6%). Der unsymmetrische x86-/GPU-Multicore-Systemverbund der AMD-Plattform arbeitet also weit effektiver zusammen. Ob die Ursache in Details der Hardwareauslegung oder den proprietären Softwaremodulen der zugehörigen Treibersysteme der Hersteller begründet ist, muss ungeklärt bleiben. Trotzdem bleibt festzuhalten, dass die CPU-Last bei

DAB-Demodulation

(FIC $r_b=32\text{ kbit/s}$, $R_c=\frac{1}{3}$ und MSC $r_b=192\text{ kbit/s}$, $R_c=\frac{1}{2}$)

GPU Compute Kernel	Laufzeit (in μs)	Inter-SM Parallelität (Grid Size)	Intra-SM Parallelität (Block Size)	Lese- durchsatz (in MB/s)	Schreib- durchsatz (in MB/s)
AMD Radeon HD6310					
Δf_c -Schätzung	259.4	1	256	921.7	0.12
Fast Information Channel:					
Δf_c -Korrektur	158.5	4	64	240.1	385.7
Fourier-Transformation	114.0	4	256	530.6	530.6
DQPSK-Demapper	157.2	3	64	1892.6	725.1
Viterbi-Fehlerschutzdecoder	4748.4	4	64	78.26	155.7
Main Service Channel (hier $n=4$):					
Δf_c -Korrektur	496.6	$4 \times (n+1)$	64	449.0	706.7
Fourier-Transformation	504.4	$4 \times (n+1)$	256	601.5	601.5
DQPSK-Demapper	655.4	$4 \times n$	64	2567.3	989.7
Zeit-Deinterleaver	519.3	4	64	296.4	2442.6
Viterbi-Fehlerschutzdecoder	22306.7	4	64	89.3	243.0

n : Anzahl zu extrahierender OFDM-Symbole je CIF-Rahmen des DAB-Multiplexsignals
(gemäß Time Slicing-Betrieb abhängig von Service-Bitrate r_b und Codierungsrate R_c)

Tabelle 5.9: Profiling der DAB-spezifischen Signalverarbeitungskernel für die Demodulation eines 96 ms DAB-Rahmens auf der x86/GPU-Plattform AMD E-350 Fusion.

DAB-Demodulation(FIC $r_b=32$ kbit/s, $R_c=\frac{1}{3}$ und MSC $r_b=192$ kbit/s, $R_c=\frac{1}{2}$)












Code-Bereich	Anteilige Prozessorauslastung
SDR-Prozessor AMD Radeon HD6310 ($f_{clk}=500$ MHz):	
Schätzung Δf_c -Fehler	0.3 % 
Korrektur Δf_c -Fehler	0.7 % 
Fourier-Transformation	0.6 % 
DQPSK-Demapper	0.8 % 
Zeit-Deinterleaver	0.5 % 
Viterbi-Fehlerschutzdecoder	28.2 % 
in Summe: 31.1 %	
Host-CPU AMD E-350 ($f_{clk}=1.6$ GHz, ein Kern):	
DAB-SDR-Applikation	5.6 % 
Treibermodule & Laufzeitumgebung GPU	2.1 % 
Treibermodule HF-Frontend	1.1 % 
Betriebssystem, HAL	0.3 % 
Betriebssystem, Sonstiges	1.5 % 
in Summe: 10.6 %	

Tabelle 5.10: Prozessorauslastung der GPU des AMD E-350 SoC sowie die verbleibende Last des AMD E-350 Bobcat-Kerns als die GPU überwachende Host-CPU während der Demodulation eines DAB-Signals.

Auslagerung der Berechnung auf die GPU auch bei dem AMD E350-System für den gegebenen Anwendungsfall der Radiosignalverarbeitung kritisch zu sehen ist: Die Systemlast der Host-CPU bei Ausführung der SSE-/SIMD-Codevariante betrug 9.5 %. Auch bei der AMD E350-Plattform führt also die Auslagerung der SDR-Algorithmen auf die GPU aufgrund des für sie nötigen Betriebsoverheads folglich nicht zum erhofften Zweck einer Entlastung der CPU.

Die Aufteilung des Ressourcenverbrauchs zeigt, dass der Anteil der verbrauchten CPU-Ressourcen in Laufzeit- und Treibersystem-Modulen der GPU sowie des HAL im Unterschied zur NVidia Ion-GPU gering ausfällt. Es verbleibt jedoch ein nennenswerter Lastanteil im eigenen Applikationscode der SDR-Anwendung, obwohl in diesem durch die CPU keine mathematischen Berechnungen durchgeführt werden. Mutmaßlich werden GPU-Housekeeping-Aufgaben, welche beim NVidia-Framework in Prozessen außerhalb der Nutzanwendung getätigt werden, vom AMD-SDK bei der Softwareerstellung direkt zum Anwendungsprogramm gelinkt.

5.4 Zusammenfassung und Diskussion der x86-Plattformen

Abbildung 5.6 fasst die Ergebnisse hinsichtlich der Auslastungen auf den für Signalverarbeitung genutzten Prozessoren (CPU oder GPU) zusammen. Abbildung 5.7 stellt die systemweite Auslastung der zugehörigen (Host-)CPUs gegenüber. Es wurden jeweils drei Implementierungsvarianten des DAB-Softwareradios für zwei x86-Plattformen mit verschiedenen Mikroarchitekturen vorgestellt.

5.4.1 x86-/GPU-Plattform Intel Atom/NVidia Ion

Bei direkter Ausführung des C-Codes des Referenzmodells auf der Bonell-Mikroarchitektur der Intel Atom Plattform wurden die Laufzeitressourcen der CPU zu 46.7% beansprucht. Durch architektur-spezifische Optimierungen, speziell durch Nutzung der Vektoroperationen der SSE-SIMD-Verarbeitungseinheit der x86-Architektur, konnte diese Last auf 11.9% gegenüber dem initial durch den Compiler aus dem C-Modell generierten Ergebnis gesenkt werden. Des Weiteren wurde als dritte Code-Variante untersucht, die nach G9x-Architektur programmierbare NVidia MCP79 Ion-GPU der Plattform zur Signalverarbeitung einzusetzen. Die eigentliche Last der arithmetischen Berechnungen konnte erfolgreich von der CPU fort auf die GPU ausgelagert werden. Sie beansprucht den vorliegenden GPU-Prozessor zu 26.9%. Jedoch ist bei genauer Analyse zu erkennen, dass im Vergleich zu der SSE-optimierten CPU-Implementierung der Einsatz der "GPU Beschleunigung" die Intel Atom Host-CPU nicht entlastet. So wird im Gegenteil bei systemweiter Betrachtung, das heißt bei Betrachtung auch außerhalb der eigenen Codebasis, offensichtlich, dass sich die Last der Host-CPU überraschenderweise durch die Auslagerung der Berechnungen auf das GPU-Subsystem signifikant von 11.9% auf 29.6% erhöht – der im heterogenen Multicore-System mit der Nutzung des GPU-Subsystems verbundene, neu eingeführte Verwaltungsaufwand ist größer als der Rechenaufwand, welcher auf dieses Subsystem ausgelagert werden kann.

5.4.2 x86-/GPU-Plattform AMD Fusion

Für die CPU-Implementierungen auf der AMD Bobcat-Mikroarchitektur des SoC AMD E-350 Fusion liegt das direkt compilierte C-Modell bei 33.3% Prozessorlast auf einem CPU-Kern. Die Bobcat-Architektur scheint somit leicht leistungsfähiger bezüglich der Ausführung des Codes. Vermutlich liegt der Grund in der komplexeren Befehlsdecodereinheit der CPU, welche beispielsweise Out-of-Order-Mechanismen bietet. Hinsichtlich der Ausführung der händisch assembleroptimierten Variante mit SSE-SIMD-Nutzung kann die CPU-Last auf 9.53% gesenkt werden. Während die Architektur-betrachtung hinsichtlich maximaler theoretischer Zahl an Arithmetikoperation pro Sekunde mittels SSE-Einheit für einen AMD Bobcat CPU-Kern einen weitaus geringeren Wert auswies als für den Intel Bonell-Kern (siehe Tabelle 3.1), sind die beiden CPU-Architekturen im tatsächlich erreichbaren Durchsatz annähernd äquivalent, sogar ist der AMD Bobcat-Kern leicht performanter. Im Detail sind Unterschiede erkennbar, welche sich im Mittel jedoch fast vollständig ausgleichen: So ist die Laufzeit des Viterbi-Algorithmus länger (circa 40%), jedoch führen andere Algorithmen andererseits leicht schneller aus. Ursachen könnten neben der Innenstruktur des Prozessorkerns auch in unterschiedlicher Auslegung der Speicheranbindungen begründet sein.

DAB Demodulation

(FIC $r_b=32$ kbit/s, $R_c=1/3$ und MSC $r_b=192$ kbit/s, $R_c=1/2$)

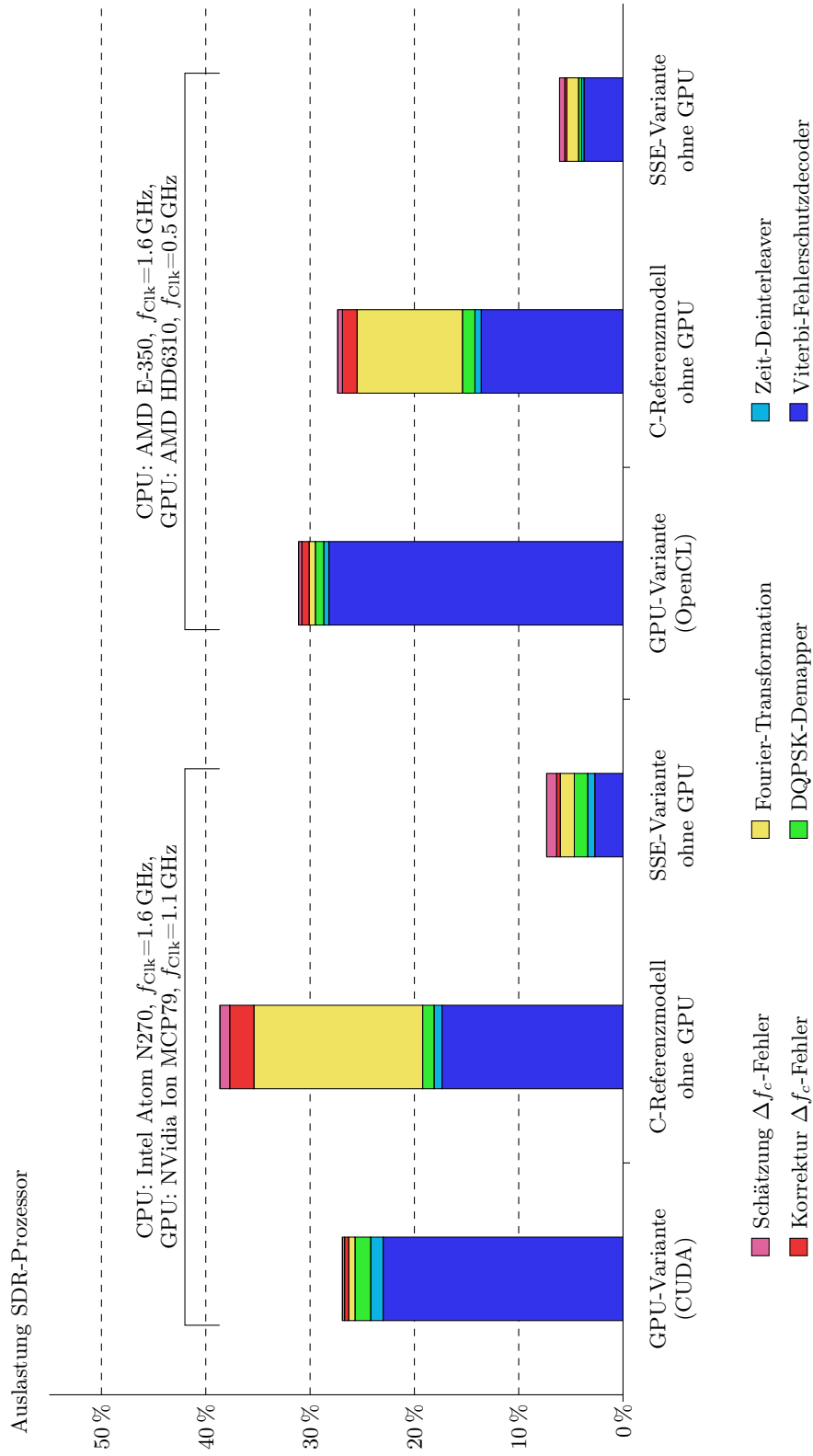


Abbildung 5.6: Vergleich der Prozessorauslastung der DAB-Softwaredemodulationsroutinen für die drei Implementierungsvarianten auf dem jeweils genutzten Compute-Device.

DAB Demodulation

(FIC $r_b=32$ kbit/s, $R_c=1/3$ und MSC $r_b=192$ kbit/s, $R_c=1/2$)

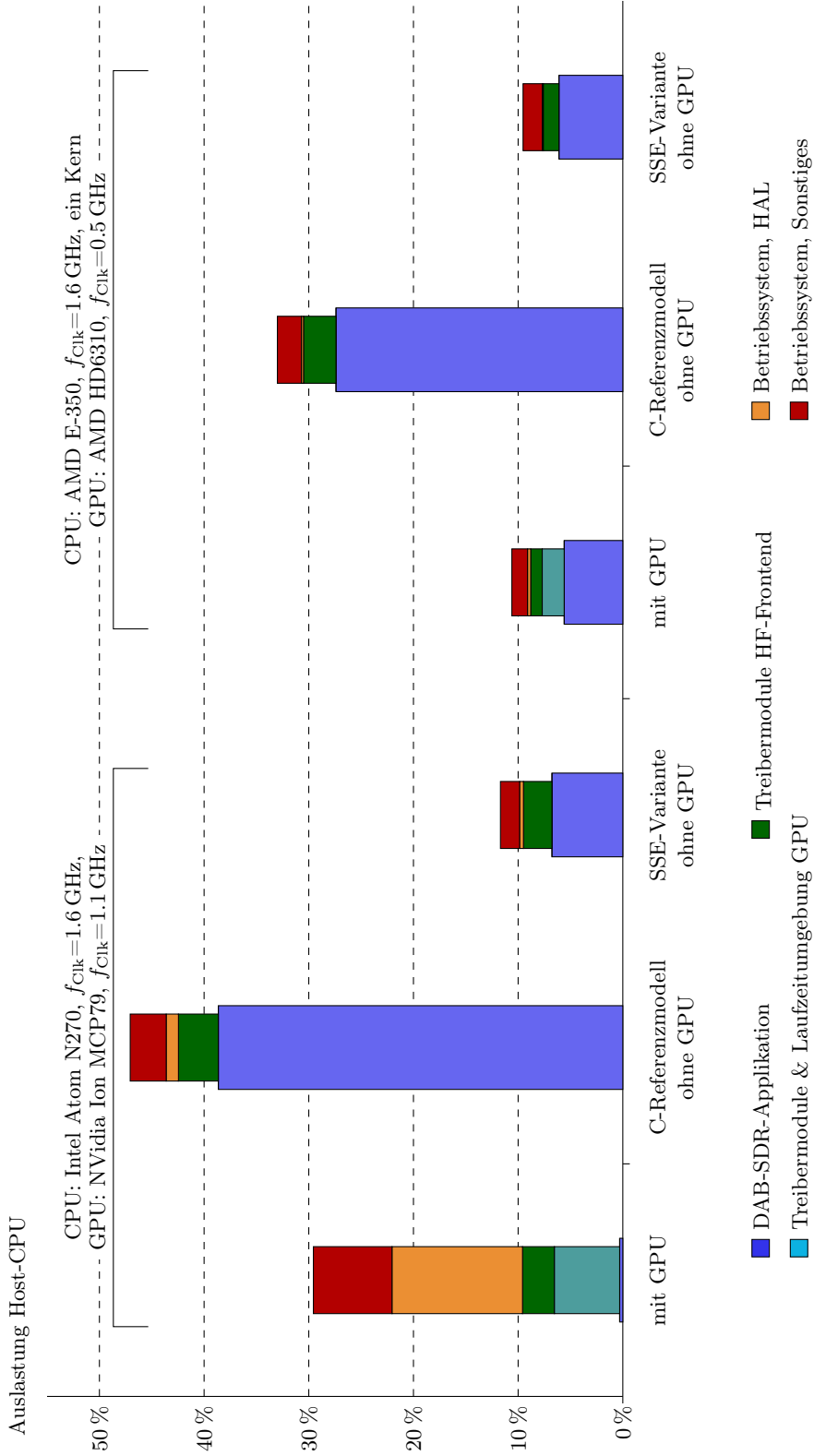


Abbildung 5.7: Vergleich der Prozessorauslastung der x86-Host-CPU während live/on-air Empfang ohne und mit zusätzlicher Nutzung des GPU-Prozessors.

5 Evaluierung des Laufzeitverhaltens der x86-/GPU-Architekturen

Des Weiteren wurde die im SoC AMD E-350 Fusion integrierte Radeon HD6310 GPU mit R800-Mikroarchitektur evaluiert. Hinsichtlich der GPGPU-nutzenden DAB-Variante liegt die Auslastung der R800-GPU bei Nenntaktfrequenz für die exemplarische Demodulationskette bei 31.1 %, somit bewegt sich der Ressourcenverbrauch in der Größenordnung wie auf der NVidia Ion-GPU. Auch hier zeigt sich, dass die auf theoretischem Maximaldurchsatz an Arithmetikoperationen beruhende Klassifizierung nicht aussagekräftig wäre, da die Radeon HD6310 GPU der NVidia Ion GPU danach stark überlegen sein müsste. Im GPU-Betrieb ergibt sich auf dem AMD Fusion-SoC eine wesentlich geringere Belastung der Host-CPU durch GPU-Verwaltungs-overhead. Die systemweite CPU-Last während des GPU-Betriebes ist mit 10.6 % bedeutend geringer als im Intel/NVidia-Systemverbund (26.9 %). Trotzdem bedeutet dies im Vergleich zur SSE-optimierten Variante, welche allein eine x86-CPU des SoC mit 9.5 % systemweiter CPU-Auslastung nutzt, dass die GPU als zusätzliches Subsystem es ebenso wie die NVidia Ion-GPU aufgrund des eingeführten Verwaltungs-overheads nicht vermag, die Host-CPU zu entlasten.

5.4.3 Implementierungsempfehlung und erwarteter Ressourcenverbrauch

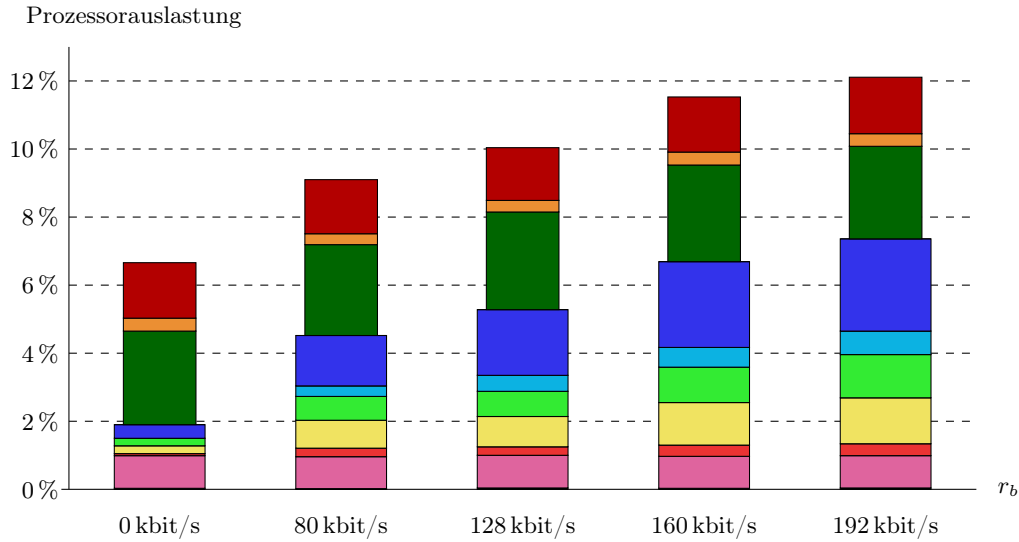
Als Ergebnis kann somit festgehalten werden, dass für den vorliegenden Anwendungsfall des softwarebasierten DAB-Radiodemodulators auf den untersuchten low-cost x86-Plattformen der Einsatz einer GPU zur Signalverarbeitung nicht zu empfehlen ist. Der Betrieb des heterogenen CPU/GPU-Multicore-Systems scheint ressourcenaufwändiger als das Durchführen der optimierten Berechnung der Signalverarbeitung alleine auf der Host-CPU.

Abschließend kann aufgrund der Ergebnisse eine mittels intensiver SIMD-Optimierung und Nutzung der SSE-Fähigkeiten gestaltete Implementierungsvariante auf der CPU zur Umsetzung von Radioalgorithmen empfohlen werden. Eine eventuell vorhandene GPU im System sollte nicht dazu verwendet werden und gegebenenfalls ruhen. Abbildung 5.8 zeigt für den Implementierungsvorschlag die gemessenen CPU-Ressourcen, welche für verschiedene Service-Bitraten r_b des DAB-Dienstes auf den Plattformen in der empfohlenen Implementierungsvariante verbraucht werden.

5.4 Zusammenfassung und Diskussion der x86-Plattformen

DAB Demodulation Intel Atom N270, 1600 MHz, mit SSE

(FIC $r_b=32$ kbit/s, $R_c=1/3$ und MSC r_b variabel, $R_c=1/2$)



DAB Demodulation AMD E-350, 1600 MHz, mit SSE

(FIC $r_b=32$ kbit/s, $R_c=1/3$ und MSC r_b variabel, $R_c=1/2$)

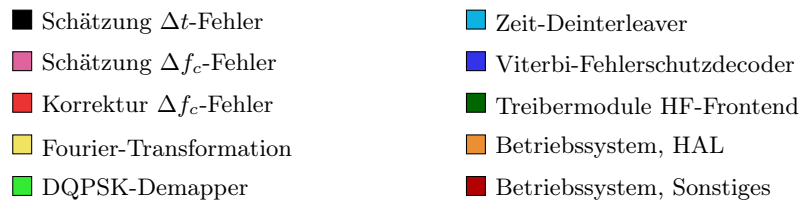
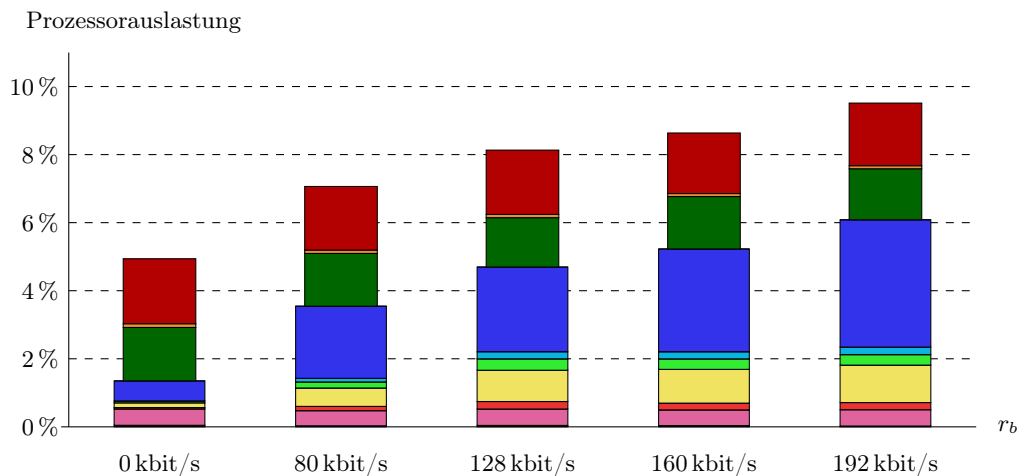


Abbildung 5.8: Messung der okkupierten CPU-Ressourcen auf den beiden x86-Plattformen bei live-/on-air-Empfang mit verschiedenen Service Bitraten r_b .

6 Evaluierung des Laufzeitverhaltens der ARM-/DSP-Architekturen

6.1	Evaluierung der ARM-Architektur	147
6.1.1	Software und Betriebssystem des ARM-Systems	147
6.1.2	Evaluierung der C-Referenzvariante	149
6.1.3	Evaluierung der Variante mit NEON-optimiertem Code	150
6.2	Nutzung des DSP-Subsystems	153
6.2.1	Interprozessorkommunikation im heterogenen ARM/DSP-System	153
6.2.2	Anpassung der Signalverarbeitungskette an die DSP-Architektur	153
6.2.3	Laufzeitanalyse des DSPs	156
6.3	Diskussion der Ergebnisse	157

Im folgenden Kapitel soll das Potential aktueller Mikroarchitekturen der ARM CPU-Familie für den Anwendungsfall der Radiosignalverarbeitung untersucht werden. Für die Untersuchung ist ein Evaluationsboard auf Basis des Texas Instruments DM3730 ausgewählt. Dieses System-on-Chip (SoC) besitzt eine ARM Cortex A8-CPU, zusätzlich ist ein DSP-Kern des Typs C64x+ verbaut. So erweist sich die Plattform als sehr gut geeignet, um zum einen die Leistungsfähigkeit eines aktuellen ARM-Applikationsprozessors zu evaluieren und zugleich das Ergebnis einer DSP-Implementierung gegenüberzustellen. Tabelle 6.1 listet die Details der eingesetzten Hard- und Software auf.

6.1 Evaluierung der ARM-Architektur

Zuerst soll alleine der Teil des SoC evaluiert werden, welcher die ARM-Mikroarchitektur darstellt. Dies entspricht einer gewöhnlichen Applikationsprozessorplattform ohne zusätzlichem DSP-Subsystem.

6.1.1 Software und Betriebssystem des ARM-Systems

Auf dem ARM-Applikationsprozessor wird als Betriebssystem ein an das System angepasstes schlankes Embedded Linux der Angstrom-Distribution verwendet. Es basiert auf dem

Testplattform 3	
(SoC Texas Instruments DM3730)	
CPU:	
Mikroarchitektur	ARM Cortex A8 mit NEON SIMD-Coprozessor (in-order, 2-fach superskalar)
Anzahl Kerne	1
Kerntakt	bis zu 1000 MHz (hier: 600 MHz)
L1-Cache	32 kB Data + 32 kB Instr.
L2-Cache	256 kB
Software	
Betriebssystemkernel	Linux 2.6.39
Codegenerierung	GCC 4.6.3
Laufzeitanalyse	OProfile 0.9.6
DSP:	
Mikroarchitektur	TMS320C64x (VLIW)
Anzahl Kerne	1
Kerntakt	bis zu 660 MHz (hier: 600 MHz)
L1-Cache/-SRAM	32 kB Instr. + 80 kB Data
L2-Cache/-SRAM	96 kB
Software	
Betriebssystemkernel	DSP/BIOS 5.41
Codegenerierung	TI C6000 Code Generation Tools 7.04
DRAM:	
Typ und Taktung	LP-DDR1, 200 MHz

Tabelle 6.1: Daten der zur Untersuchung genutzten kombinierten ARM-/DSP-Plattform.

Linux Kernel 2.6.39. Zur systemweiten Laufzeitanalyse des ARM-Systems wird das Tool OProfile in der Version 0.9.6 genutzt. Hierbei handelt es sich um einen statistischen Profiler. Das standardmäßige Samplingintervall von OProfile beträgt 100000 Prozessorzyklen. Entsprechend beispielsweise einem typischen Prozessortakt von 600 MHz ergibt sich ein Samplingintervall der Länge $T_s=0.167$ ms und somit nach Gleichung 5.7 eine minimal geforderte Messdauer von $T_{\min}=10.3$ s bei der Laufzeituntersuchung¹. Zum Übersetzen des Quellcodes wird die GCC-Compilersuite in Version 4.6.3 verwendet. Der Compiler wurde für Codeerstellung gemäß geringster Laufzeit parametrisiert (Option *-Ofast*).

¹Es wird wie in Kapitel 5 stets im Folgenden eine großzügigere Messdauer $T=300$ s verwendet werden.

6.1.2 Evaluierung der C-Referenzvariante

Im ersten Schritt wird erneut eine rein C-basierte Implementierungsvariante mit den Algorithmen aus Kapitel 4 für die Signalverarbeitung eingesetzt. Diese sind für das ARM-Linux-System in eine neu entworfene Rahmenapplikation eingebettet. Analog zur Rahmenapplikation des x86-Systems übernimmt diese die plattformabhängige Ansteuerung des genutzten USB-HF-Frontends und den Transport der von diesem digitalisierten Empfangsbasisbanddaten sowie auf der Seite des Systemausgangs auch die Anbindung an die externe Nutzdatensenke für die Audio- und Datenservices.

Die Messungen in Abbildung 6.1 zeigen, dass mit dem unoptimierten Code auf einem mit 600 MHz getakteten ARM Cortex A8 bereits eine Ausführung der Signalverarbeitungskette in Echtzeit möglich ist, jedoch unter starker CPU-Last. Bei der als maximal betrachteten Service-Bitrate von $r_b=192$ kbit/s werden die Prozessorressourcen zu 65.1 % ausgelastet.

DAB-Demodulation ARM Cortex A8, 600 MHz

(FIC $r_b=32$ kbit/s, $R_c=1/3$ und MSC r_b variabel, $R_c=1/2$)

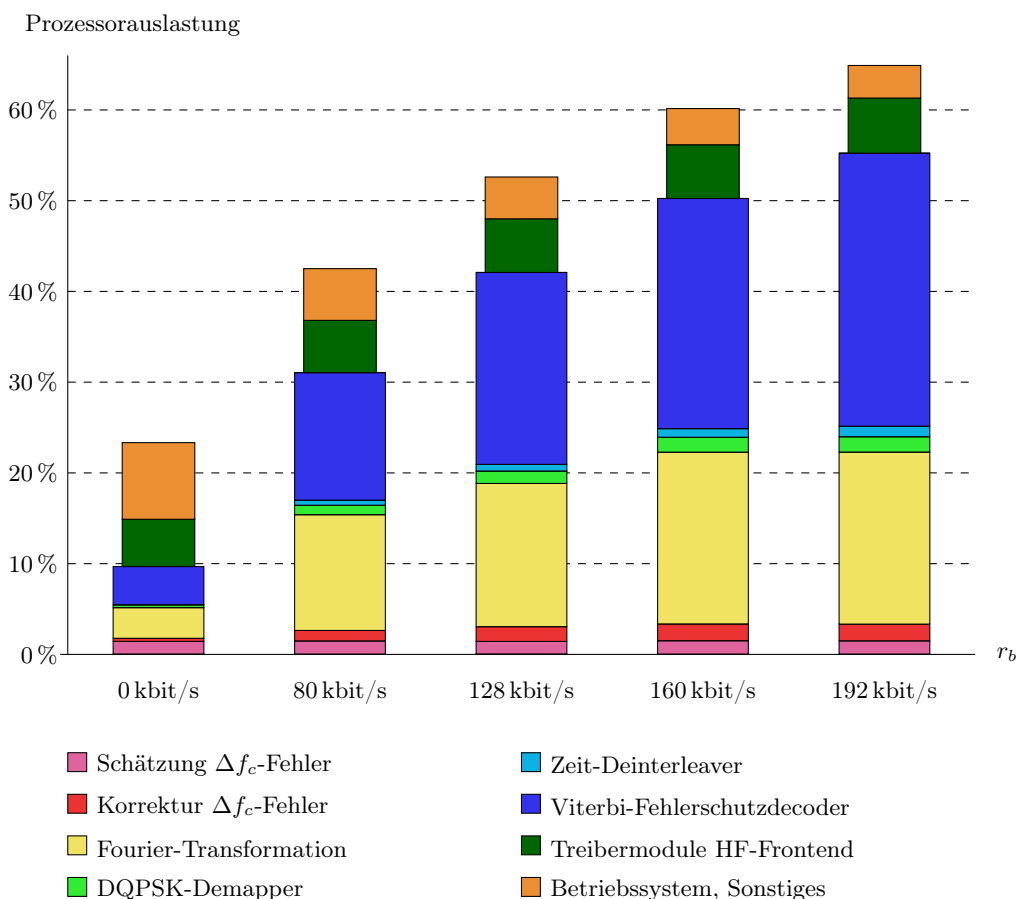


Abbildung 6.1: Last des ARM Cortex A8 CPU-Kerns bei Betrieb der DAB-Empfängerapplikation ohne Nutzung des NEON SIMD-Coprozessors in Abhängigkeit der DAB-Service-Bitrate r_b .

Die mathematischen Signalverarbeitungsalgorithmen stellen dabei 55.4% Prozessorlast dar. Als laufzeitdominant sind analog zu Implementierungsvarianten aus Kapitel 5 die Funktionsblöcke der Fourier-Transformation (19.0%) und des Viterbi-Decoders (30.1%) erkennbar. Derjenige Zeitanteil an Prozessorlast, welcher weiterhin zur Signalverarbeitung für die Darstellung der vollständigen Anwendung benötigt wird (Betriebssystemroutinen und Handling des USB-Frontends), trägt zusätzlich 9.7% bei.

6.1.3 Evaluierung der Variante mit NEON-optimiertem Code

Im zweiten Schritt werden die primären zwei Hotspots der Signalverarbeitung, Fourier-Transformation und Viterbi-Fehlerschutzdecoder, einer intensiven Optimierung unterzogen. Der Fokus liegt dabei erneut auf der Nutzung von Vektoroperationen, welche in diesem Falle in dem im ARM-Kern vorhandenen NEON-Coprozessor realisiert sind. Da die Befehlssatzerweiterung Advanced SIMD Extensions der NEON-Funktionseinheit als adäquates Pendant zur SSE-Technologie der x86-Systeme anzusehen ist, können die Algorithmen meist durch vom Prinzip ähnliche Programmstrukturen implementiert werden. Im Weiteren wird deshalb hinsichtlich der optimierten Funktionsblöcke nur der Unterschied zum Aufbaukonzept der SSE-optimierten Funktionsblöcke dargestellt.

Viterbi-Fehlerschutzdecoder

Die durch den Befehlssatz gebotenen Möglichkeiten zur Vektorpermutation sowie Variation der Bitbreite innerhalb des SIMD-Vektors ermöglichen auf dem ARM NEON-System wie bei x86/SSE eine effiziente Umsetzung des inhomogenen Datenflusses. Anders als bei der SSE-Einheit des x86-Prozessors stehen dem NEON-Coprozessor mit sechzehn 128 bit breiten Registern doppelt so viele Arbeitsregister zur Verfügung. Hieraus folgt in der Implementierung ein wesentlicher Unterschied zum Viterbi-Decoder der x86/SSE-Implementierung: Es können sämtliche akkumulierten Metriken der 64 Zustände des Faltungscodes direkt in den Arbeitsregistern des Prozessors belassen werden. Ein ständiger Zugriff auf einen langsameren, prozessorkernexternen RAM- oder Cache-Speicher ist somit für die Propagierung der Metriken nicht notwendig.

Fourier-Transformation

Auch im Konzept der ARM NEON FFT-Implementierung äußert sich die gegenüber der Intel SSE-Architektur erhöhte Speichertiefe der Registerbank. Wie bereits in Kapitel 5 beschrieben könnte bei der Berechnung der FFT durch die größere Registerbank mehr Lokalität im Speicherzugriff mittels Erhöhung des Radix-Grades genutzt werden. Jedoch wird in der vorliegenden FFT statt eines Radix 8-Butterfly immer noch ein Radix 4-Butterfly als Kernelement genutzt. Es wird die erhöhte Registertiefe verwendet, um statt vier nun acht Radix 4-Butterflies pro Iterationsschritt parallel zu berechnen. Hiermit wird die Möglichkeit der NEON-Mikroarchitektur optimal genutzt, zwei 128 bit SIMD-Register gleichzeitig über einen Speicherzugriff von 256 bit zu laden und zugleich die Inhalte wortweise zu verzahnen². So kann die FFT-Implementierung einen kompletten Vektor aus acht Datenpunkten,

²Das hierzu gehörige Schlagwort der Befehlssatzarchitektur ARMv7a lautet: Vector Load/Store of N-Element Structure.

DAB-Demodulation ARM Cortex A8 NEON, 600 MHz

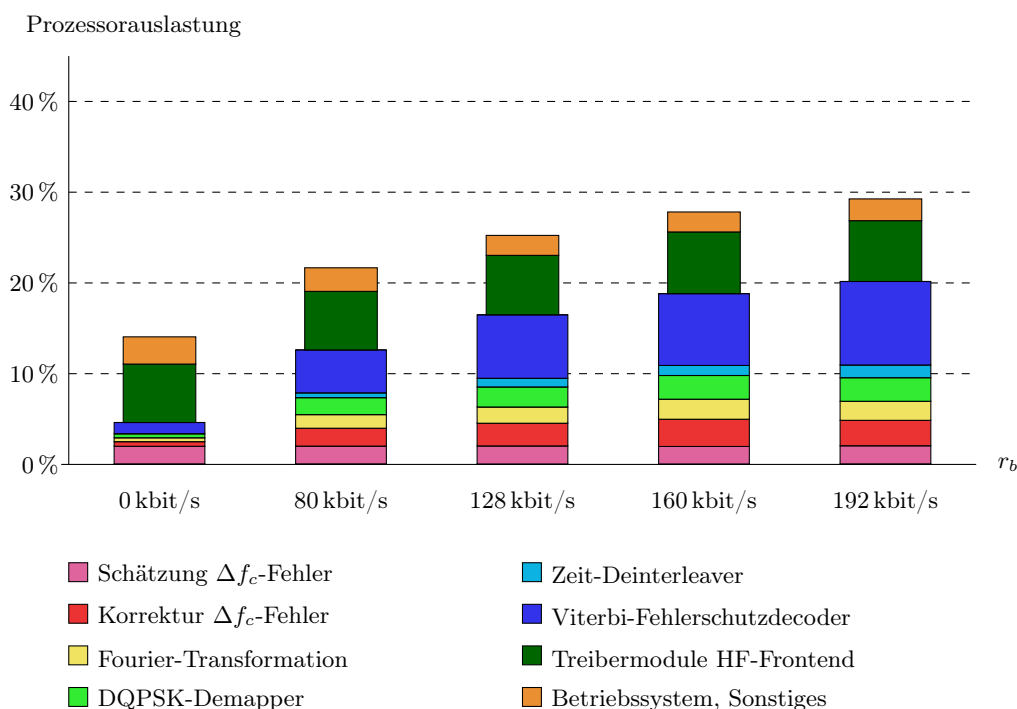
(FIC $r_b=32$ kbit/s, $R_c=1/3$ und MSC r_b variabel, $R_c=1/2$)

Abbildung 6.2: Last des ARM Cortex A8 CPU-Kerns bei Betrieb der DAB-Empfängerapplikation unter Nutzung des NEON SIMD-Coprozessors in Abhängigkeit der DAB-Service-Bitrate r_b .



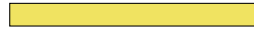













basierend je auf 16 bit-Anteilen für Real- und Imaginärteil, mit bestmöglicher Effizienz aus dem externen RAM-Speicher laden beziehungsweise dorthin zurückspeichern.

Messung der optimierten Signalverarbeitungskette

Abbildung 6.2 zeigt die Messungen der Anwendung mit optimiertem FFT- und Viterbi-Funktionsblock auf dem ARM Cortex A8 bei 600 MHz Taktung. In Abhängigkeit von der Bitrate des DAB-Services ist bei Nutzung des NEON-Coprozessors für die Signalverarbeitung eine Prozessorlast von bis zu 21.6% (ohne NEON-Nutzung: 55.4%) bei der maximal angenommenen Datenrate $r_b=192$ kbit/s nötig. Der Funktionsblock des Viterbi-Decoders stellt dabei mit 9.2% stets noch den dominanten Anteil an Prozessorlast dar. Unter Berücksichtigung der vollständigen DAB-Empfangsanwendung ergibt sich inklusive Basisbandtransport und Betriebssystem für das gesamtheitliche System eine Prozessorlast von bis zu 29.3% (vormals 65.1% ohne NEON-Nutzung).

DAB-Demodulation

(FIC $r_b=32$ kbit/s, $R_c=\frac{1}{3}$ und MSC $r_b=192$ kbit/s, $R_c=\frac{1}{2}$)

Code-Bereich	Anteilige Prozessorlast
ARM Cortex A8 ($f_{Ck}=600$ MHz), C-Referenzmodell:	
Schätzung Δt -Fehler	0.1 %
Schätzung Δf_c -Fehler	1.5 % 
Korrektur Δf_c -Fehler	1.8 % 
Fourier-Transformation	19.0 % 
DQPSK-Demapper	1.7 % 
Zeit-Deinterleaver	1.2 % 
Viterbi-Fehlerschutzdecoder	30.1 % 
	in Summe: 55.4 %
Treibermodule HF-Frontend	6.1 % 
Betriebssystem, Sonstiges	3.6 % 
	in Summe: 65.1 %
ARM Cortex A8 ($f_{Ck}=600$ MHz), mit NEON-Assembler:	
Schätzung Δt -Fehler	0.1 %
Schätzung Δf_c -Fehler	2.0 % 
Korrektur Δf_c -Fehler	2.8 % 
Fourier-Transformation *)	2.1 % 
DQPSK-Demapper	2.6 % 
Zeit-Deinterleaver	1.4 % 
Viterbi-Fehlerschutzdecoder *)	9.2 % 
	in Summe: 20.2 %
Treibermodule HF-Frontend	6.7 % 
Betriebssystem, Sonstiges	2.4 % 
	in Summe: 29.3 %

*) Optimiert unter Verwendung von NEON-Assembler.

Tabelle 6.2: Last des CPU-Kerns ARM Cortex A8 bei Betrieb der DAB-Empfängerapplikation ohne und mit Nutzung des NEON SIMD-Coprozessors.

6.2 Nutzung des DSP-Subsystems

Als nächstes soll der Implementierung auf dem general-purpose ARM-Prozessor, aber auch denjenigen der x86/GPU-Plattformen, die Demodulation mit speziell für Signalverarbeitung ausgelegter Mikroarchitektur gegenübergestellt werden³. Hierzu wird der ebenfalls in der DM3730-Plattform vorhandene C64x-DSP verwendet. Die DSP-Implementierung wurde umgesetzt mit den TI C6000 Code Generation Tools 7.0.4 und DSP/BIOS Version 5.41.

6.2.1 Interprozessorkommunikation im heterogenen ARM/DSP-System

Auf dem DSP dient der DSP/BIOS-Kernel von Texas Instruments als schlankes Laufzeitsystem. Berechnungsmodule werden in dessen Kontext als sogenannte Nodes bezeichnet und besitzen ein standardisiertes Codeformat, welches Einsprungspunkte für Initialisierung, Ausführung (das heißt aktive Berechnung) und Deinitialisierung festlegt [98]. Auf dem Linux-Betriebssystem des im Systemverbund übergeordneten ARM-Applikationsprozessors ist zur Steuerung des DSP ein Linux-Kernelmodul, genannt DSP/BIOS Bridge, zu nutzen. Nach dem Reset und Einspielen des Programmabbildes in den DSP können die enthaltenen Nodes vom ARM-Host auf Anfrage gestartet werden. Dessen genutzte Hauptspeicherbereiche können ebenfalls in den virtuellen Adressraum des DSPs eingeblendet und so geteilt werden. Neben der direkten Kommunikation über geteilte Speicherbereiche werden durch DSP/BIOS mit Streams und Message-Queues auch Mechanismen auf höherer, abstrahierter Schicht zur Interprozessorkommunikation bereitgestellt.

Dieses Konzept ist demjenigen der GPU-Computing Frameworks sehr ähnlich. Es zeigt sich, dass die dort verwendeten Konzepte der GPU-Kernels vom Prinzip etablierten Methoden der geteilten Datenverarbeitung und Interprozessorkommunikation zwischen einem Host-Applikationsprozessor und untergeordneten DSP-Subsystemen entsprechen.

6.2.2 Anpassung der Signalverarbeitungskette an die DSP-Architektur

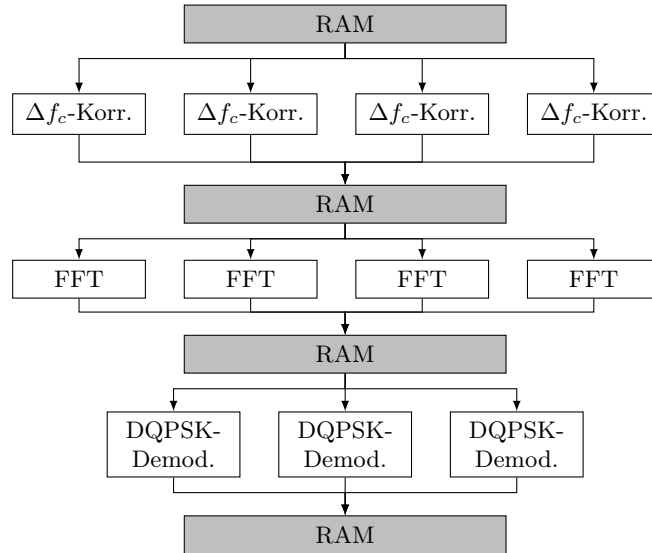
Gemäß der Anwendungsempfehlung des Herstellers erfolgt die Programmierung in Hochsprache und dabei die Codeübersetzung durch den herstellereigenen C-Compiler. Ein Codereword mit Einsatz von Maschinensprache wurde nicht durchgeführt, entsprechend der Auslegung als Nicht-Vektorprozessor war eine spezielle maschinensprachenorientierte manuelle Anpassung des Codes an solche SIMD-Strukturen nicht notwendig. Jedoch wurde die Architektur der Software dahingehend angepasst, dass zur Bildung besserer Cachelokalität bestehende, ursprünglich für datenparallele Verarbeitungsstrukturen vorbereitete Datenflüsse aufgelöst und für den DSP in sequentielle, datenlokale Verarbeitung überführt wurden.

Sequentielle anstelle paralleler Datenverarbeitung

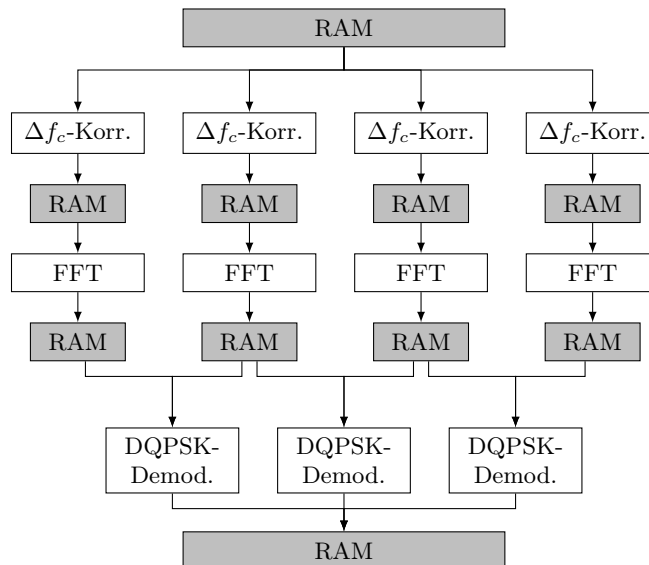
Die Architektur des DSPs, im Gegensatz besonders zur GPU, ist nicht auf massiv parallele Datenverarbeitung hin entworfen. Die bei 16 bit Wortbreite vorhandene zweifache SIMD-

³Die Portierung des Codes auf die DSP-Plattform erfolgte in Zusammenarbeit mit der im Rahmen der Dissertation betreuten Diplomarbeit aus [97].

6 Evaluierung des Laufzeitverhaltens der ARM-/DSP-Architekturen



(a) Datenflussarchitektur mit Fokus auf massiv-paralleler Datensatzverarbeitung.



(b) Separate Abarbeitung von im Speichervolumen reduzierten Teildatensätzen.

Abbildung 6.3: Anpassung der Datenflussarchitektur der Signalverarbeitungskette an das volumenbegrenzte Cachesystem des DSPs.

Fähigkeit je Register kann gut für komplexwertige Datenverarbeitung genutzt werden. Eine Auslegung der Software auf höher datenparallele Strukturen ist aber ohne Nutzen. Die DSP-Architektur ist dagegen sehr gut geeignet, vorwiegend sequentielle Algorithmen mit Folgen unterschiedlicher Befehle schnell zu bearbeiten. Hier kann der Compiler mittels VLIW-Worten MIMD-Befehlsstrukturen abbilden. Der Softwareentwurf auf der vorliegenden DSP-Architektur basiert also auf dem exakt gegensätzlichen Designparadigma zur GPU: Für die Implementierungsvarianten der GPU-Prozessoren wurden stets möglichst viele gleichartige Datensätze im Speicher aggregiert und zur mehrfachen, nebenläufigen Ausführung durch einen Algorithmus geplant, um die symmetrische Parallelität des Multicore/SIMD-Systems auszunutzen.

Volumenbegrenztheit des Cache-Speichersystems

Die im SoC DM3730 verbaute Variante des C64x-DSPs bietet interne SRAM-Speicher, welche entweder als transparenter Cache bei externen Speicherzugriffen oder explizit als schnelles Daten-SRAM konfiguriert werden können. Jedoch sind diese vergleichsweise klein, es sind nur maximal 64 kB als L2-Cache verfügbar⁴. Es können statt eines gesamten DAB-Rahmens der Länge 96 ms nur 32 ms der zu verarbeitenden Basisbanddaten gleichzeitig im schnellen Speicher des L2-Caches gehalten werden. Entsprechend ist bei unverändertem Vorgehen ein vorzeitiges Überlaufen des Caches zu erwarten, neue Daten würden ältere, aber stets noch benötigte Daten verdrängen. Dies würde den Cache wirkungslos machen. Gleiches gilt potentiell für sämtliche weiteren Teilergebnisse zwischen den Einzelalgorithmen der Signalverarbeitungskette.

Rückbau der massiv-parallelen Software-Auslegung

Die Signalverarbeitungskette war vormals auf das Ziel der datenparallelen Vereinheitlichung von Algorithmen ausgelegt, das heißt gleiche Algorithmen wurden quasisimultan auf verschiedene Datenfragmente des Basisbandsignals zur Ausführung angeordnet (Abbildung 6.3a). Nun wird die Struktur umpartitioniert, sodass einzelne Datensets durch so viele verschiedene Algorithmen wie möglich in Folge bearbeitet sind, bevor ein neues Datenset begonnen wird. So wird mittels kleiner Quell- und Zieldatensätze eine Erhöhung der Datenlokalität und dementsprechend hinsichtlich des Basisbandsignals eine Vermeidung von Cache-Verdrängung und in Folge von ungepufferten Zugriffen auf den langsamen externen DRAM-Speicher erzielt.

In der Implementierung wird für eine Schleife über alle OFDM-Symbole die Kette aus dem Algorithmus der Frequenzoffsetkorrektur, der Fourier-Transformation und des DQPSK-Demappers in direkter sequentieller Abfolge ausgeführt (siehe Abbildung 6.3b). Es findet sich in den weiteren Laufzeitangaben deshalb ein die vormaligen Einzelalgorithmen kombinierender Funktionsblock. Generell werden die Ratschläge zur bestmöglichen Nutzung des Cachesystems gemäß [63] beachtet.

⁴Der auf demselben System-on-Chip DM3730 vorhandene, im vorigen Teilkapitel behandelte ARM Cortex A8 verfügt mit 256 kB über die vierfache Menge L2-Cache.

6.2.3 Laufzeitanalyse des DSPs

Anders als auf dem ARM-Prozessor mit Multitasking-Betriebssystem wird auf dem DSP kein statistisches Profiling zur Analyse eingesetzt. Ähnlich der GPU kann durch exklusive Ausführung des eigenen Codes eine Unterbrechungen des Programmablaufs durch andere Prozesse ausgeschlossen werden. Deshalb kann nach Herstellerempfehlung [99] zur Laufzeit- und somit Performance-Messung direkt ein hochauflösender, interner Timer eingesetzt werden, der an den Takt des Prozessorkerns gekoppelt ist. Zu Beginn und Ende der untersuchten Coderegionen wird der Wert dieses Timers ausgelesen und die entsprechende Algorithmenlaufzeit in Prozessorzyklen ermittelt.

Insgesamt ergibt sich für die DAB-Signalverarbeitungskette eine Auslastung des DSPs bei 600 MHz Taktung von 43.1 % bei Demodulation eines DAB-Services mit $r_b=192$ kbit/s. Für den Gesamtfunktionsblock aus Korrektur des Trägerfrequenzfehlers, der Fourier-Transformation und des DQPSK-Demappers konnte eine anteilige Prozessorlast des DSPs von 5.0 % erreicht werden. Ohne das vorgestellte systemspezifische Umstrukturierungskonzept zur Erhöhung der Datenlokalität betrug diese 13.2 %.

Der Funktionsblock des Viterbi-Decoders stellt auch bei dieser Mikroarchitektur den dominanten Algorithmus dar. Jedoch zeigt sich auf dem vorliegenden DSP eine anteilig betrachtet vergleichsweise große Prozessorlast von 30 %. Ähnlich der GPU mag auf dem vornehmlich für Arithmetik optimiertem DSP der zusätzlich stark auf logische Operationen angewiesene Viterbi-Algorithmus weniger effizient zu implementieren sein als reine Arithmetik-Routinen.

DAB-Demodulation Texas Instruments C64x DSP, 600 MHz

(FIC $r_b=32$ kbit/s, $R_c=1/3$ und MSC r_b variabel, $R_c=1/2$)

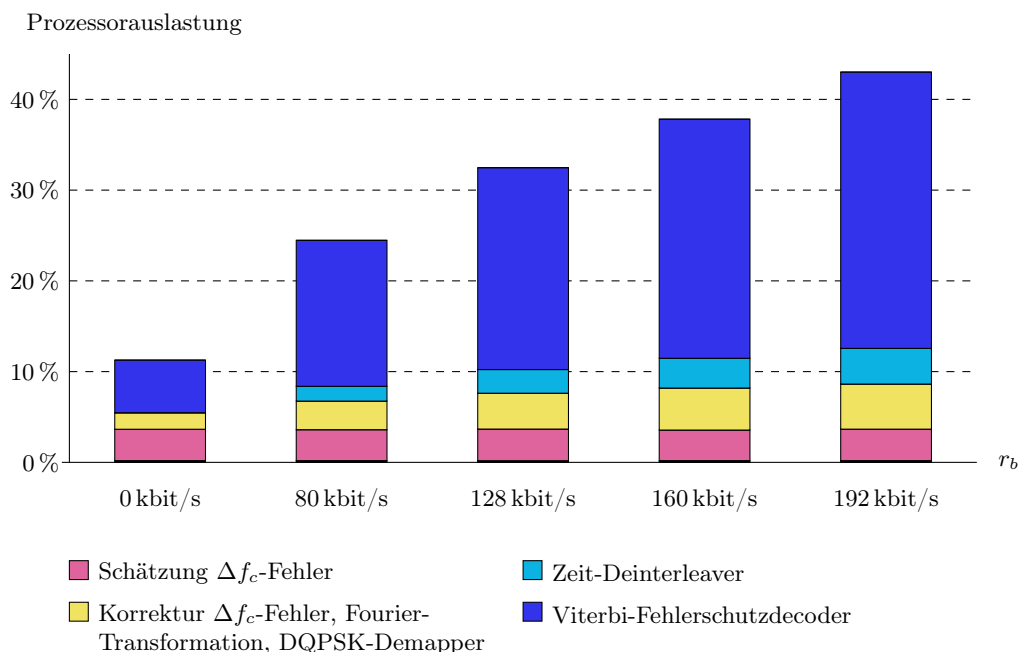


Abbildung 6.4: Last des TI C64x DSP-Kerns bei Betrieb der DAB-Signalverarbeitung, in Abhängigkeit der DAB-Service-Bitrate r_b .

6.3 Diskussion der Ergebnisse

Im Folgenden sollen die auf der ARM-/DSP-Plattform gesammelten Messwerte diskutiert werden.

Einfluss der SIMD-Optimierungen auf der ARM-Architektur

Bei der Implementierung der Fourier-Transformation konnte die achtfache SIMD-Parallelität der vorliegenden ARM NEON-Mikroarchitektur erfolgreich in direkten Geschwindigkeitsgewinn umgesetzt werden. Gegenüber der vom C-Compiler direkt generierten Variante wurde der FFT-Funktionsblock durch Assembleroptimierung mit expliziter Einbindung von ARM NEON-Instruktionen um mehr als den Faktor neun beschleunigt (von 19 % auf 2.0 % Prozessorlast bei 600 MHz Taktung). Beim Viterbi-Algorithmus fällt der Geschwindigkeitsgewinn, welcher erzielt werden konnte, weniger extrem aus. Dennoch konnte der rechenintensivste Funktionsblock um den Faktor drei beschleunigt werden (von 30 % auf 10.5 % Prozessorlast bei 600 MHz Taktung) und so die Gesamtanforderung an die Ressourcen der Plattform drastisch gesenkt werden. Bezogen auf die gesamte Signalverarbeitungskette konnte durch diese Codeoptimierungen die Zahl der nötigen Prozessorzyklen auf fast ein Drittel (Senkung der Prozessorlast von 55.4 % auf 21.6 %) reduziert werden. Abbildung 6.5 fasst dies gegenüberstellend zusammen.

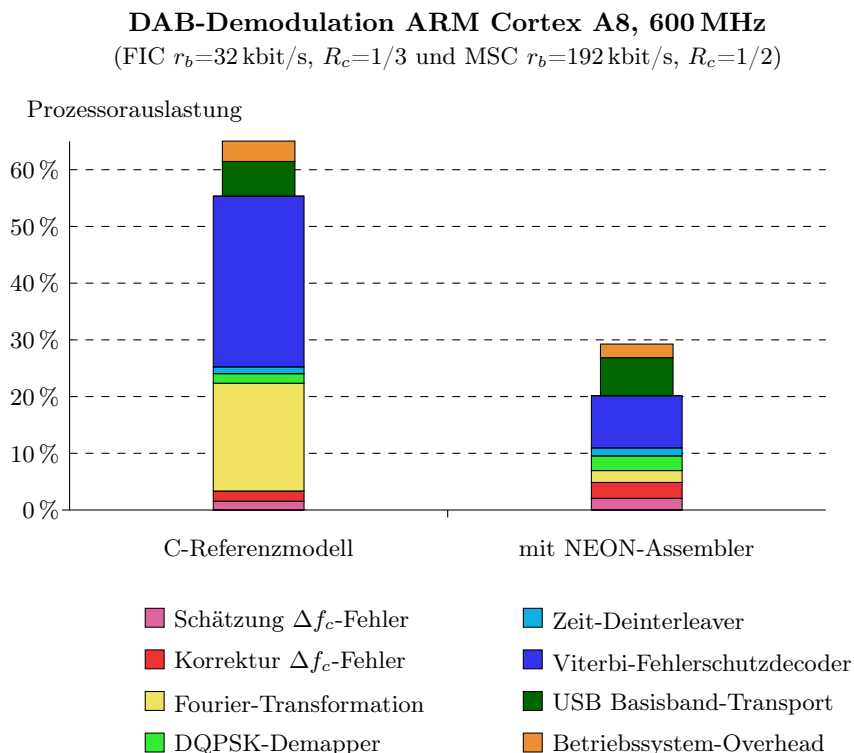


Abbildung 6.5: Gegenüberstellung der gemessenen Last des ARM Cortex A8 CPU-Kerns bei Betrieb der DAB-Empfängerapplikation ohne und mit Nutzung der NEON SIMD-Instruktionen.

Abhängigkeit von der ARM-Prozessorkern-Taktfrequenz

Zur besseren Vergleichbarkeit der Architektur wurde der ARM-Prozessor bislang identisch zum DSP mit 600 MHz getaktet. In Abbildung 6.6 sind die Laufzeitanalysen für sämtliche verfügbaren Taktungsvarianten des vorliegenden Prozessorkerns aufgetragen. Die Laufzeitwerte skalieren unter Berücksichtigung des konstant bleibenden Taktes des Systemspeichers ($f_{\text{Mem}}=200$ MHz) gut entsprechend der Erwartung in Abhängigkeit der variablen Taktung des Kerns. Es zeigt sich, dass bei Verwendung des NEON-Coprozessors das Empfänger-system problemlos auch bei einer reduzierten Taktung von 300 MHz in Echtzeit lauffähig ist. Hier werden im Szenario bei $r_b=192$ kbit Nutzdatenrate 51.1% der Ressourcen des ARM Cortex A8 belegt, davon 36.9% für die Signalverarbeitung. Für den hochvolumigen Transport der rohen Basisbanddaten vom Frontend in den Speicher ist im USB-Subsystem des verwendeten Linux-Betriebssystems bei Taktung mit 600 MHz eine Systemlast von circa 6.5% belegt. Bei niedrigerem Prozessortakt steigt sie in guter Näherung in rezipro-kem Maße. Bei Erhöhung des Prozessortaktes scheinen jedoch andere Komponenten des vorliegenden Systems limitierend zu sein (mutmaßlich Speicher, Peripherie-Controller),

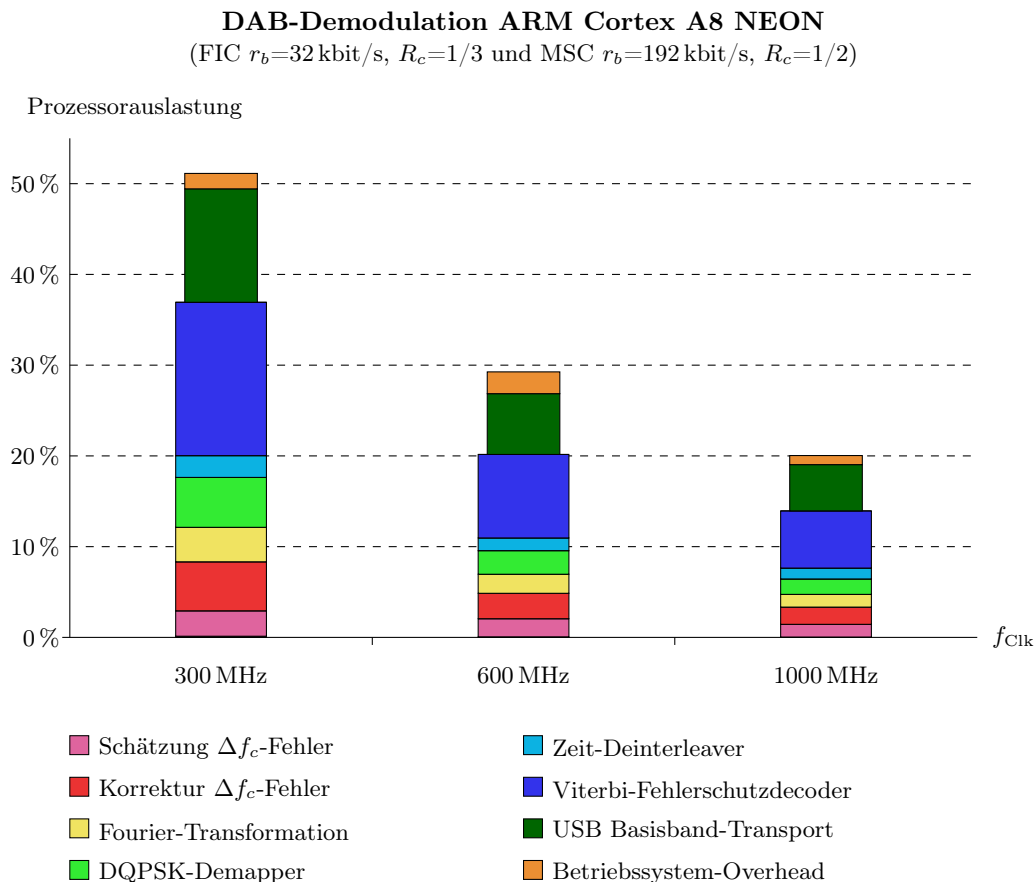


Abbildung 6.6: Last des ARM Cortex A8 CPU-Kerns bei Betrieb der DAB-Empfängerapplikation unter Nutzung des NEON SIMD-Coprozessors in Abhängigkeit der Prozessortaktfrequenz f_{Clk} .

entsprechend verändert sich die beobachtbare Prozessorlast nur wenig. Die Prozessorlast, welche bei der Ausführung der DAB-Applikation weiterhin vom verwendeten Linux-Betriebssystem für sonstige Hilfsaufgaben (beispielsweise Multithreading-Koordinierung) aufzubringen ist, bleibt in allen Fällen unter 2.5 %.

Zusammenfassend ist für einen ARM Cortex A8-Prozessor mit NEON-Coprozessorsystem in Näherung ein äquivalenter Prozessortakt von 150...200 MHz zur Realisierung eines DAB-Empfängersystems nach Abbildung 6.6 einzuplanen. Ohne NEON-Coprozessor sind nach dem Wert für die maximal zu erwartende Servicebitrate aus Abbildung 6.1 als Taktungsäquivalent grob circa 400 MHz vorzuhalten.

Einordnung des ARM mit NEON Media Coprozessor zum klassischen DSP

Tabelle 6.3 vergleicht bei identischer Taktung des Prozessorkerns die Laufzeitanalyse der ARM NEON-Implementierung mit derjenigen für den TI C64x-DSP. Wie bereits angesprochen zeigt der implementierte Funktionsblock des Viterbi-Decoders auf dem untersuchten DSP bei relativer Betrachtung unterdurchschnittliche Laufzeitleistung. Werden die restlichen Funktionsblöcke ohne den besagten Viterbi-Algorithmus betrachtet, so ist die Prozessorauslastung mit 12.6 % in Summe auf dem C64x-DSP beziehungsweise 11.0 % auf









DAB-Demodulation	
(FIC $r_b=32$ kbit/s, $R_c=\frac{1}{3}$ und MSC $r_b=192$ kbit/s, $R_c=\frac{1}{2}$)	
Code-Bereich	Anteilige Prozessorlast
TI C64x ($f_{clk}=600$ MHz):	
Schätzung Δt -Fehler	0.2 %
Schätzung Δf_c -Fehler	3.5 % 
Korr. Δf_c -Fehler, FFT, DQPSK-Dem.	5.0 % 
Zeit-Deinterleaver	3.9 % 
Viterbi-Fehlerschutzdecoder	30.5 % 
in Summe: 43.1 %	
ARM Cortex A8 ($f_{clk}=600$ MHz), mit NEON-Assembler:	
Schätzung Δt -Fehler	0.1 %
Schätzung Δf_c -Fehler	2.0 % 
Korr. Δf_c -Fehler, FFT, DQPSK-Dem.	7.5 % 
Zeit-Deinterleaver	1.4 % 
Viterbi-Fehlerschutzdecoder	9.2 % 
in Summe: 20.2 %	

Tabelle 6.3: Vergleich der Prozessorlast des DAB-Demodulatorkerns der Implementierungsvariante für den TI C64x-DSPs mit derjenigen der ARM NEON-Architektur bei gleicher Taktung des Prozessorkerns.

dem ARM Cortex A8 sehr ähnlich. Abschließend kann die Frage, ob der ARM-Prozessor mit NEON-Coprozessor für den untersuchten Anwendungsfall funktional als Alternative zu klassischen DSPs zu klassifizieren ist, eindeutig bejaht werden.

Performance der DSP-Architektur und Betrachtung des Viterbi-Funktionsblocks

Für die numerischen Routinen der Trägerfrequenzfehlerkorrektur, der Fourier-Transformation und des DQPSK-Demappers zeigte sich der C64x-DSP als performante Architektur. Jedoch ergab sich für den Viterbi-Algorithmus eine verhältnismäßig hohe anteilige Prozessorlast von 30 %. Diese Laufzeitauffälligkeit des DSP deckt sich interessanterweise vom Prinzip mit den entsprechenden Beobachtungen beim Vergleich der Messwerte der beiden GPU-Mikroarchitekturen zu den x86-CPUs (siehe Tabellen 5.3 und 5.6 sowie 5.8 und 5.10). Als Ursache wird die stark bit-logikorientierte Arbeitsweise des Algorithmus vermutet, welche mit bevorzugt auf Arithmetik ausgelegten Mikroarchitekturen keine ideale Passung ermöglicht.

Zur Einordnung und Prüfung dieser vorliegenden Implementierung des Viterbi-Algorithmus soll auf eine verfügbare Applikationsnotiz [100] referenziert werden, welche über ein ähnliches System berichtet. Es wird eine Realisierung eines Viterbi-Decoders für den GSM-Standard auf der verwandten DSP-Mikroarchitektur C55x [101] vorgestellt. Dort konnte ein Rahmen von 189 bit bei angegebenen $0.58 \cdot 10^6$ Taktzyklen decodiert werden. Extrapoliert auf die hier vermessene DAB-Gesamtdatenrate $r_b = 192 \text{ kbit/s} + 32 \text{ kbit/s} = 224 \text{ kbit/s}$ für MSC und FIC ergäbe sich rechnerisch ein Bedarf von $678 \cdot 10^6$ Taktzyklen pro Sekunde auf der C55x-Architektur. Unter Berücksichtigung der Systemunterschiede von C55x zu C64x kann das eigene auffällige Messergebnis von der Größenordnung zu demjenigen aus der vorgenannten Abschätzung gut eingeordnet werden:

- Der VLIW-DSP C64x besitzt eine leistungsfähigere Mikroarchitekturauslegung als der C55x-DSP. Die maximal mögliche SIMD/MIMD-Parallelität pro Instruktionszyklus ist, vorbehaltlich deren voller Nutzbarkeit, um den Faktor 8 erhöht (32 bit statt 16 bit DSP, vierfache Anzahl an nebenläufigen Rechenwerken).
- Jedoch existieren Sonderbefehle des C55x speziell für den Viterbi-Algorithmus auf dem C64x nicht⁵.
- Weiterhin sind die stark angewachsene Framegröße und somit mutmaßliche Effizienzverluste durch das begrenzte Fassungsvermögen des Cachespeichers zu bedenken.

Der hier genutzte C64x-DSP wurde bei 600 MHz zu 30.5 % belegt, es werden also äquivalent circa $183 \cdot 10^6$ Taktzyklen pro Sekunde für den Viterbi-Algorithmus konsumiert. Das Verhältnis der beiden Taktzyklenwerte 678/183 ist 3.7. Der Faktor 3.7 liegt unter Berücksichtigung der obig angesprochenen, die Leistung des C64x im Vergleich zum C55x steigender Architekturmerkmale im konkludent nachvollziehbaren Bereich und spricht somit

⁵Die DSP-Familie C5000/C54x/C55x des gleichen Herstellers bietet spezielle Prozessorbefehle, die explizit zur beschleunigten Umsetzung des Viterbi-Algorithmus vorgesehen sind. Mittels einer Compare Select Store Unit (CSSU) können diese Prozessoren in einem Taktzyklus Pfadmetriken vergleichen, den Maximalwert propagieren sowie das Entscheidungsergebnis als isoliertes Bit abspeichern. Diese Sonderbefehle existieren beim Produkt C64x nicht.

trotz des anfangs auffälligen Messwertes für eine valide Testimplementierung des vorliegenden Fehlerschutzdecoders. Die eingangs dargelegte Einschätzung der DSP-Architektur hinsichtlich deren Verarbeitungsfähigkeit für den Viterbi-Algorithmus wird also durch die nun durchgeführte Validitätsprüfung der Funktionsimplementierung gestützt⁶.

⁶Es sind Derivate des C64x-Basisprozessors speziell für nachrichtentechnische Anwendungen mit einem als VCP bezeichneten eigenständigen Hardwarecoprocessor für den Viterbi-Algorithmus ausgerüstet worden [102]. Solche auf einen Algorithmus festgelegten Beschleunigersysteme entsprechen jedoch nicht mehr dem hier zu untersuchenden Konzept des flexiblen Software Defined Radio. Sie sind für Übertragungsverfahren mit anderer als der vorab festgelegten Codestruktur unbrauchbar. Die Beurteilung der Leistungsfähigkeit der C64x-Mikroarchitektur erfolgte deshalb ohne Betrachtung eventueller, in manchen Spezialvarianten prozessorkernextern hinzugefügten, anwendungsfallsspezifisch festgelegter Viterbi-Zusatzeinheiten.

7 Weitere Untersuchungen im Umfeld der entwickelten Signalverarbeitungskette

7.1	Betrachtung der elektrischen Leistungsaufnahme	164
7.1.1	Beschreibung des Messaufbaus	164
7.1.2	Untersuchung der low-cost x86/GPU-Plattform	164
7.1.3	Untersuchung der ARM-basierten Plattform	166
7.1.4	Fazit zur elektrischen Leistungsaufnahme	166
7.2	Einordnung der low-cost x86-CPU zu Workstation x86-CPU's 166	166
7.2.1	Die Intel Sandy Bridge-Plattform	166
7.2.2	Messungen und Laufzeitvergleich	167
7.3	Vergleich des prozessorbasierten Lösungsansatzes zu einem FPGA-basierten System	169
7.3.1	Einordnung eines FPGA-Entwurfs	169
7.3.2	Resümee zu einer möglichen Präferenz zwischen FPGA und Pro- zessor	171
7.4	Abschätzung eines Softwaredemodulators für DVB-T	171
7.4.1	Eigenschaften von DVB-T	172
7.4.2	Komplexität eines gewählten Übertragungsszenarios	173
7.4.3	Ergebnisse der Abschätzung	173
7.5	Prototypische Integration des Empfängers in ein Versuchs- fahrzeug	176
7.5.1	Automotive Ethernet/IP	176
7.5.2	SDR-Demodulator und Audioquelle	177
7.5.3	Audiosenke und Bedieneinheit	178
7.5.4	IP-basiertes Audio Streaming	179
7.5.5	IP-basierte Middleware	181
7.5.6	Resümee zum Demonstrator-Gesamtfahrzeug	182

Im folgenden Kapitel sollen die weiterhin zum Themenbereich der Arbeit durchgeführten Untersuchungen zusammengefasst werden. Zum einen erfolgt die Einordnung des hybriden x86-/GPU-Signalverarbeitungssystems hinsichtlich der elektrischen Verlustleistung.

Es wird daneben die Anforderung der Laufzeitressourcen betrachtet, wenn für die Algorithmenkette anstelle eines low-cost Systems ein aktuelles Workstation-Computersystem eingesetzt wird. Auch wird ein Vergleich der prozessorbasierten Signalverarbeitung zu einem FPGA-basierten Ansatz durchgeführt. Eine auf die Ergebnisse des Hörfunksystems gestützte Extrapolation soll weiterhin die Ressourcenanforderungen bei softwarebasierter Demodulation von Fernsehübertragungssystemen abschätzen. Abschließend wird das im Rahmen der Arbeit ausgerüstete Versuchsfahrzeug vorgestellt, in welches das entwickelte Softwareradio mittels eines experimentellen Ethernet-basierten Bordnetzinterfaces zu integrieren war.

7.1 Betrachtung der elektrischen Leistungsaufnahme

Zur Einordnung der Designvarianten wird als eine weitere Metrik die nötige Energieaufnahme der Plattformen x86/GPU und ARM betrachtet. Auf der x86-/GPU-Architektur (Intel Atom/NVidia Ion) ist zu untersuchen, welchen Einfluss die aktive Nutzung der GPU auf den zur Bewältigung des gestellten Anwendungszweckes nötigen elektrischen Gesamtenergieverbrauch hat.

7.1.1 Beschreibung des Messaufbaus

Zur Leistungsmessung wird als Messgröße die Stromaufnahme der vollständigen Plattform bei konstanter Betriebsspannung ermittelt. Die Stromaufnahme wird mittels eines Shunt-Widerstandes am zugänglichen zentralen Versorgungsspannungseingang der jeweiligen Testplattform durchgeführt. So sind neben dem Prozessor/den Prozessoren auch die im realen Betrieb der Systemplattform notwendigen Subsysteme wie die des Arbeitsspeichers und der Schnittstellencontroller erfasst¹. Als Festspannungsquelle dient ein Labornetzteil, für dessen Spannungsstabilität über den gesamten Arbeitsbereich eine lastabhängige Regelabweichung kleiner 15 mV bei 5...20V Ausgangsspannung spezifiziert ist. Da die Stromaufnahme der Computersysteme großen Unregelmäßigkeiten unterliegt beziehungsweise durch starke Impulsformen gekennzeichnet ist, wird das Spannungssignal des Shunt-Stromwandlers vor der Auswertung zur Mittelwertbildung durch ein RC-Tiefpassfilter geführt. Dieses bildet eine großzügige Zeitkonstante von $T_{RC} \geq 20$ s. Zur weiteren Signalauswertung wird ein Speicheroszilloskop genutzt.

7.1.2 Untersuchung der low-cost x86/GPU-Plattform

Für die Intel Atom-Plattform ist in Bereitschaft eine mittlere Leistungsaufnahme von 9.6 W (Tabelle 7.1) zu beobachten. Im aktiven Rechenbetrieb zeigt sich die SIMD-optimierte Implementierung mit Nutzung von SSE-Maschinensprache, welche auch die berechnungseffizienteste darstellt, erwartungsgemäß als diejenige, die am wenigsten Energie zur Erfüllung der gestellten Aufgabe verbraucht. Jedoch ist der Unterschied zur weit weniger leistungsfähigen C-Referenzimplementierung gering. Er beträgt nur 440 mW, was bezogen auf die Gesamtleistungsaufnahme der Plattform 3% entspricht. Für beide Varianten wird das

¹Komponenten außerhalb der Kernplattform wie Massenspeichergeräte oder Displays sind nicht in die Leistungsaufnahmemessung einbezogen.

7.1 Betrachtung der elektrischen Leistungsaufnahme

Plattform Intel Atom N270/NVidia MCP79 Ion

(DAB Demodulation mit FIC $r=32$ kbit/s, $R_c=\frac{1}{3}$ und MSC $r=192$ kbit/s, $R_c=\frac{1}{2}$)






System-Zustand	Elektrische Leistungsaufnahme
System in Bereitschaft:	
HF-Frontend standby	9.56 W 
HF-Frontend aktiv	10.44 W 
Ausführung der Signalverarbeitung:	
CPU ohne SSE-Nutzung	11.10 W 
CPU mit SSE-Nutzung	10.66 W 
GPU-Nutzung	12.48 W 

Tabelle 7.1: Leistungsaufnahme der Intel Atom/NVidia Ion-Testplattform bei Ausführung des DAB-Digitalradioszenarios.

GPU-Subsystem nicht zu arithmetischen Berechnungen genutzt, es ist jedoch aktiv und generiert ein gewöhnliches Videoausgangssignal.

Für die Implementierungsvariante, in der die GPU neben der gewöhnlichen Grafikausgabe zur Berechnung der Signalverarbeitung hinzugezogen wird, steigert sich die elektrische Leistungsaufnahme des Systems um 1.4 W beziehungsweise 1.8 W im Vergleich zu den Berechnungsvarianten nur auf der CPU. Für die Bewältigung der gleichen Aufgabe verbraucht das Gesamtsystem bei Einsatz der GPU im Vergleich zur optimierten CPU-Variante also 13 % mehr Energie.

Plattform ARM Cortex A8 (TI DM3730)

(DAB Demodulation mit FIC $r=32$ kbit/s, $R_c=\frac{1}{3}$ und MSC $r=192$ kbit/s, $R_c=\frac{1}{2}$)





System-Zustand	Elektrische Leistungsaufnahme
System in Bereitschaft:	
HF-Frontend standby	5.12 W 
HF-Frontend aktiv	5.95 W 
Ausführung der Signalverarbeitung:	
Ohne NEON-Nutzung	6.19 W 
Mit NEON-Nutzung	6.17 W 

Tabelle 7.2: Leistungsaufnahme der ARM Cortex A8-Testplattform bei Ausführung des DAB-Digitalradioszenarios.

7.1.3 Untersuchung der ARM-basierten Plattform

Die Plattform mit ARM Cortex A8 wird bei einer typischen Taktung von 600 MHz betrachtet (Tabelle 7.2). Mit aktivem HF-Frontend beträgt die Leistungsaufnahme des Gesamtsystems knapp 6.0 W. Der Start der Signalverarbeitungsroutinen bedingt einen Anstieg der Leistungsaufnahme um circa 200 mW auf knapp 6.2 W. Zwischen der Ausführung der ursprünglichen C-Implementierung der Signalverarbeitung und der weniger anteiligen Prozessorzeit konsumierenden, laufzeitoptimierten Variante mit Nutzung der SIMD-Prozessierung durch die NEON Vektoreinheit ist auf der vorliegenden Plattform de facto kein Unterschied in der Leistungsaufnahme feststellbar.

7.1.4 Fazit zur elektrischen Leistungsaufnahme

Die Messwerte in Tabelle 7.1 und 7.2 zeigen, dass die elektrische Leistungsaufnahme der Plattformen nur in verhältnismäßig geringem Maße von durchgeführten Berechnungsaktivitäten abhängig ist. Vielmehr hängt die elektrische Leistungsaufnahme davon ab, welche Systemkomponenten aktiv sind beziehungsweise sich in wachem Zustand befinden. Zu diesen Systemen gehören neben CPU auch Chipsatz, RAM sowie sämtliche andere Hilfsbaugruppen der Plattformen, und so eben auch die GPU. Deren aktive Nutzung zur Berechnung schafft ein weiteres aktives Element. Die Host-CPU muss während der GPU-Berechnung weiterhin aktiv bleiben (siehe auch Laufzeitmessungen aus Kapitel 5) und so erhöht die Nutzung der GPU die elektrische Leistungsaufnahme des Gesamtsystems. Aus Energieverbrauchssicht ist die Nutzung der GPU als Hilfsprozessor im untersuchten Anwendungsfall also kontraproduktiv (+13% relativ, +1.8 W absolut). Die ARM SoC-Plattform TI DM3730 verbraucht in absoluten Zahlen weit weniger Energie als die x86 Intel Atom/Nvidia Ion-Plattform (6.2 W statt 10.7 W). Bei Beachtung und Einbeziehung der unterschiedlichen Systemtaktungen (600 MHz statt 1600 MHz) beziehungsweise des total verfügbaren Rechenpotentials relativiert sich dies jedoch, so dass die gemessene höhere Leistungsaufnahme der vorliegenden Intel Atom-Plattform dieser nicht als grober Nachteil ausgelegt werden kann.

7.2 Einordnung der low-cost x86-CPU zu Workstation x86-CPU

In Kapitel 5 wurde eine optimierte Basisbandverarbeitung eines DAB-Empfängers für die x86-Architektur vorgestellt. Die Implementierung ist jedoch nicht auf die Anwendung im Kontext der vorgestellten embedded-tauglichen CPU-Exemplare der x86-Prozessorfamilie beschränkt. Zusätzlich zum primären Untersuchungsgegenstand der Arbeit, der Untersuchung von low-cost Plattformen, soll deshalb in diesem Abschnitt der Arbeit knapp auch eine Betrachtung der Implementierung bei Anwendung auf ein typisches vollwertiges Notebook/Desktop x86-System erfolgen.

7.2.1 Die Intel Sandy Bridge-Plattform

Hierzu wurde ein Testsystem mit einer CPU der Intel Sandy Bridge-Mikroarchitektur ausgewählt. Dessen Intel Core i3-2310M Dualcore-CPU ist mit moderater Energieaufnahme

Systemplattform	Intel Atom N270	Intel Core i3-2310M
CPU-Kern(e)	1x Bonnell	2x Sandy Bridge
Mikroarchitektur	in-order, 2-fach superskalar	out-of-order, 6-fach superskalar
Kerntakt	1600 MHz	2100 MHz
SIMD-Coprozessor	128 bit (SSE)	256 bit (AVX)
L1 Daten-Cache	24 kB	je Kern 32 kB
L1 Instr.-Cache	32 kB	je Kern 32 kB
L2 Cache	512 kB	je Kern 256 kB
L3 Cache	nicht vorhanden	3 MB
Thermal Design Power	2.5 W	35 W
Chipsatz	NVidia MCP79 Ion	Intel HM65
DRAM Typ und Timing	DDR2-800, 5CL	DDR3-1333, 9CL

Tabelle 7.3: Vergleich der in Kapitel 5 evaluierten low-cost Plattform zu einem nun betrachteten typischen Workstation-System.

für performante mobile Notebook-PCs entwickelt. Als Designparadigma steht trotzdem hohe Rechenleistung vor niedriger elektrischer Leistungsaufnahme². Die Mikroarchitektur der Sandy Bridge CPU-Familie ist entsprechend komplexer angelegt als die Bonnell-Architektur des Intel Atoms. Einen wesentlichen Architekturunterschied stellt das Vorhandensein einer dritten Cache-Ebene dar. Dieser L3-Cache ist mit mehreren MB Kapazität großzügig bemessen. Als Antizipation für weitergehende SoC- und Manycore-Konzepte (bis zu acht Cores pro Chip) ist der L3-Cache anders als bei früheren Intel Multicore-Architekturen nicht über eine Crossbar an die Kerne, sondern über eine leistungsfähige 256 bit-Ringbusstruktur sowohl an diese als auch den Memory Controller Hub angekoppelt. Der Cache wie auch Ringbus kann eine maximale Datenrate von 96 GB/s liefern. Die SIMD-Verarbeitungseinheit unterstützt im Unterschied zur Bonell-Architektur nun Vektoroperationen auf die doppelte Wortbreite von 256 bit. Die zugehörige Befehlssatzerweiterung wird Advanced Vector Extensions (AVX) genannt. Sie ist rückwärtskompatibel zu den bekannten Vektorbefehlssätzen MMX (64 bit) und SSE (128 bit). Tabelle 7.3 zeigt in Kurzübersicht das System im Vergleich zur bekannten Atom-Plattform.

7.2.2 Messungen und Laufzeitvergleich

Die Ermittlung der Performance-Werte folgt der gleichen Profiling-Methodik wie in Kapitel 5 beschrieben. Es soll erneut der Fall der DAB-Signalverarbeitungskette beim Empfang einer Radiostation mit hoher Bitrate ($r_b=192$ kbit/s) sowie Decodierung des obligatorischen FIC-Datenkanals ($r_b=32$ kbit/s) betrachtet werden. Es wird die identische Appli-

²So ist die TDP der Intel Core i3-i2310M CPU mit 35 W spezifiziert, die TDP des Intel Atom N270 ist mit N270 2.5 W angegeben.

DAB-Demodulation(FIC $r=32$ kbit/s, $R_c=\frac{1}{3}$ und MSC $r=192$ kbit/s, $R_c=\frac{1}{2}$)

















Code-Bereich	Anteilige Prozessorlast	
Intel Bonell: $f_{Clk}=1.6$ GHz		
Δf_c -Schätzung	0.95 %	
Δf_c -Korrektur	0.35 %	
Fourier-Transformation	1.35 %	
DQPSK-Demapper	1.27 %	
Zeit-Deinterleaver	0.69 %	
Viterbi-Fehlerschutzdecoder	2.71 %	
Treibermodule HF-Frontend	2.72 %	
Betriebssystem, Sonstiges	2.03 %	
	in Summe: 12.1 %	
Intel Sandy Bridge: $f_{Clk}=2.1$ GHz		
Δf_c -Schätzung	0.21 %	
Δf_c -Korrektur	0.06 %	
Fourier-Transformation	0.28 %	
DQPSK-Demapper	0.67 %	
Zeit-Deinterleaver	0.09 %	
Viterbi-Fehlerschutzdecoder	0.85 %	
Treibermodule HF-Frontend	0.80 %	
Betriebssystem, Sonstiges	0.36 %	
	in Summe: 3.4 %	

Tabelle 7.4: Vergleich der benötigten Prozessorressourcen der SSE-optimierten DAB-Signalverarbeitung der low-cost Plattform zu einem Workstation-System.

kationssoftware mit SSE-Optimierung ausgeführt, wie sie für die Intel Atom-Plattform entwickelt wurde. Dies bedeutet, dass von der Applikation nur das SSE-, aber nicht das AVX-Feature der Sandy Bridge-Befehlssatzarchitektur genutzt wird und somit das Potential der Mikroarchitektur des Intel Core i3-2310M noch nicht vollständig ausgeschöpft ist. Das Ergebnis der Performance-Messung ist in Tabelle 7.4 dargestellt. Für einen CPU-Kern der Intel Atom-Familie mit 1.6 GHz stellte sich für dieses Szenario bekanntermaßen nach Kapitel 5 eine Gesamtlast von 12% ein. Die komplexere Sandy Bridge-Mikroarchitektur stellt sich erwartungsgemäß weit leistungsfähiger dar. Die Auslastung eines CPU-Kerns des vollwertigen Notebook/Desktop-x86-Systems beträgt weniger als 3.5%. Bezogen auf den Dualkern-Prozessor wird die Gesamtplattform folglich weniger als 2% durch die Signalverarbeitung belastet. Die Untersuchung zeigt, dass auf heutigen high-end Notebook/Desktop-Prozessoren softwarebasierte Signalverarbeitung für Digitalradio-Übertragungsverfahren

7.3 Vergleich des prozessorbasierten Lösungsansatzes zu einem FPGA-basierten System

moderater Datenrate mit geringer Prozessorlast möglich ist.

7.3 Vergleich des prozessorbasierten Lösungsansatzes zu einem FPGA-basierten System

Wie im Einführungskapitel erwähnt, existiert die Technologie der FPGAs als zweites Mittel zur Realisierung rekonfigurierbarer Signalverarbeitungssysteme. Trotz gleicher verwendeter Hauptalgorithmen sind die entsprechenden Signalverarbeitungsketten in der Regel konzeptbedingt sehr unterschiedlich aufgebaut. Die Hauptunterschiede der Entwurfsparadigmen einer Signalverarbeitungskette im Prozessor beziehungsweise FPGA werden dabei meist wie folgt sein:

- Reduktion der Rechenoperationszahl unter Inkaufnahme erhöhter Speicheranforderung und irregulären Datenflusses im Prozessor (beispielsweise Anwendung lückenden Time Slicing-Betriebs).
- Stetiger, gleichförmiger Datenfluss zur Minimierung des Zwischenspeicherbedarfs für die Pufferung von Teilergebnissen unter Inkaufnahme erhöhter Rechenoperationszahl im FPGA.

Die vorgestellten prozessorbasierten Systeme für das DAB-Radiosystem sollen nun in den Kontext eines adäquaten FPGA-Entwurfs gestellt werden. Um Vergleichswerte anzugeben wird im Folgenden auf die Ressourcenverbrauchsangaben aus Feilen et al. [103] referenziert. Dort wurde eine ähnliche Signalverarbeitung eines DAB-Empfängers für die low-cost FPGA Familie Xilinx Spartan-6 [104] entwickelt.

7.3.1 Einordnung eines FPGA-Entwurfs

Zum Ressourcenvergleich mit [103] wird der dortige Block einer digitalen Pegelregelung am Eingang des Systems außen vor gelassen, denn in den bisher durchgeführten Untersuchungen wird eine Signalpegeleinregelung (Automatic Gain Control, AGC) als elementarer Bestandteil des HF-Frontends, dort noch vor der Analog-Digital-Wandlung, durchgeführt. Eine weitere dedizierte Pegelregelung in der Domäne der digitalen Signalverarbeitung wurde deshalb als für den vorliegenden Anwendungsfall nicht notwendig erachtet.

Um die Vergleichbarkeit der Signalverarbeitungsketten weiterhin zu verbessern, werden die folgenden Funktionsblöcke aus [103] gruppiert: Das Ausschneiden des Nutzanteils und Verwerfen des Schutzintervall-Anteils im Eingangsdatenstrom geschieht im System aus Kapitel 4 im Zuge der Trägerfrequenzkorrektur, entsprechend wird hinsichtlich [103] der Block *Guard Removal* mit der Trägerfrequenzsignalkorrektur in der Betrachtung zusammengefasst. Gleiches gilt für die Fouriertransformation und das Frequenzinterleaving sowie den DQPSK-Demapper und den dortigen Block *Bitcut*. Der Zeitinterleaver in [103] wurde aufgrund des nötigen Speicherbedarfs mittels externem RAM implementiert und liegt damit außerhalb des in [103] angegebenen Datensets zur Ressourcenverbrauch-Analyse.

DAB Demodulationskette FPGA
Xilinx Spartan-6 Architektur

	CLB-Slices	Block-RAMs	DSP-Slices
<i>Funktionsblock:</i>			
Schätzung Δt -Fehler	64	0	0
Schätzung Δf_c -Fehler	291	2	5
Korrektur Δf_c -Fehler	134	0	4
Fourier-Transformation	407	10	8
DQPSK-Demapper	75	2	6
Zeit-Deinterleaver	extern, nicht angegeben		
Viterbi-Fehlerschutzdecoder	713	3	0
in Summe:	1684	17	23
<i>FPGA-Gesamtauslastung:</i>			
Xilinx XC6SLX16	73.9 %	53.1 %	71.9 %
Xilinx XC6SLX25	44.8 %	32.7 %	60.5 %
Xilinx XC6SLX45	24.7 %	14.7 %	39.7 %
Xilinx XC6SLX150	7.3 %	6.3 %	12.8 %

Tabelle 7.5: Ressourcenbelegung des DAB-Empfangssystems aus [103], abgebildet für die Synthese auf verschiedenen FPGAs der Xilinx Spartan-6 Familie [104].

Vergleich anhand von Chipauslastung

Während der Ressourcenverbrauch beziehungsweise die Auslastung im Prozessor als anteilig konsumierte Zeit der verfügbaren Verarbeitungstaktzyklen angegeben ist, wird der Ressourcenbedarf im FPGA anhand des Anteils der belegten Logikzellen beschrieben. Neben Standardlogikzellen, Configurable Logic Block (CLB)-Slices bei Xilinx genannt, existiert typischerweise im modernen FPGA und so auch im Xilinx Spartan-6 eine beschränkte Menge spezialisierter Elemente, welche Speicherbänke (Block-RAMs) oder arithmetische Rechenwerke (DSP-Slices, insbesondere für Multiplikationsoperationen) zur Verfügung stellen [104].

Ein sinnvoller direkter Vergleich zwischen Prozessoren und FPGA anhand von Metriken fällt schwer. Es soll deshalb versucht werden, den Vergleich praxisnah anhand der Suche eines auslastungsäquivalenten FPGAs gemäß der Prozessorlast der Systeme aus Kapitel 5 und 6 zu leisten:

- Intel Atom (7.3 % @ 1.6 GHz):
circa Xilinx XC6SLX150 (7.3 % CLB, 6.3 % BRAM, 12.8 % DSP)
- ARM Cortex A8, (20.2 % @ 1000 MHz, 29.3 % @ 600 MHz):
circa Xilinx XC6SLX45 (24.7 % CLB, 14.7 % BRAM, 39.7 % DSP)

7.4 Abschätzung eines Softwaredemodulators für DVB-T

- TI C64x (43.1 % @ 600 MHz):
circa Xilinx XC6SLX25 (44.8 % CLB, 53.1 % BRAM, 71.9 % DSP)

Diese Vergleichbarkeitsangabe kann nur als grober Richtwert gesehen werden und gilt nur für den vorliegend untersuchten Anwendungsfall der Implementierung einer DAB-Demodulationsapplikation.

7.3.2 Resümee zu einer möglichen Präferenz zwischen FPGA und Prozessor

Eine belastbare, generell gültige Präferenzempfehlung zwischen FPGA- und Prozessor-Signalverarbeitungskette abzuleiten fällt schwer. Ein vom umgebenden System losgelöster Vergleich zwischen FPGA- und Prozessor-Signalverarbeitung sowie Ableitung einer Aussage hiervon erscheint nicht praxisnah und nicht fair: Es ist zum einen zu bedenken, dass auch hinsichtlich der Gesamtsystemintegration Unterschiede zwischen FPGA- und Prozessor-Realisierung existieren werden. Hier ergibt sich der folgende Zwiespalt:

- Jegliche praktische Umsetzung eines rekonfigurierbaren Demodulatorkerns erfordert die Anbindung an ein HF-Frontend durch ein Hardware-Interface (oft als engl. Glue-Logic bezeichnet). Das logikbasierte SDR-System kann diesen Schaltungsteil im eigenen FPGA direkt integrieren.
- Im Gesamtsystem muss stets ein Steuer-/Applikationsprozessor die primär für Mikrocontrollersoftware ausgelegten Aufgaben der höheren Protokollebenen des Übertragungsschemas und das Hauptprogramm der Nutzapplikation sowie oft eine Mensch-Maschine- oder Maschine-Maschine-Schnittstelle bedienen. Das prozessorbasierte Signalverarbeitungssystem kann, wie in der Arbeit gezeigt, mit dem Aufgabenbereich des Applikationsprozessors fusioniert werden.

Des Weiteren existieren in der Realität stets Bauteilesynergien zu dritten in der Applikationsschaltung realisierten Funktionen, welche je nach hierfür benötigtem Chiptyp die Wirtschaftlichkeit eines jeden der beiden Ansätze in der Praxis beeinflussen werden. Eine Vorteilhaftigkeitsbewertung wäre folglich vorab in genereller Weise fragwürdig und erscheint nur bei vollständig vorliegendem, konkretem Anwendungssystem sinnvoll. Die vorgestellte Arbeit soll und kann dazu nur als Basis die Daten über den Ressourcenverbrauch auf Seite der Prozessoren stellen, für die FPGA-Variante sind diese beispielsweise im zitierten System nach [103] zu finden. Es sind dann stets im Gesamt-Usecase der zu betrachtenden Elektronikplattform die Zusatzkosten der rekonfigurierbaren Logikbaugruppe gegen die Kosten des zusätzlichen Leistungsvorhaltes auf dem Prozessor aufzuwiegen.

7.4 Abschätzung eines Softwaredemodulators für DVB-T

Die Untersuchungen der vorliegenden Arbeit erfolgten als Fallstudie anhand einer Implementierung für den digitalen Hörfunkstandard DAB. Als komplementäres Verfahren für die digitale Fernsehübertragung existiert der Standard DVB-T [3]. Der folgende Abschnitt soll die Systeme gegenüberstellen und versuchen, die gewonnenen Ergebnisse in Näherung auf den Anwendungsfall eines softwarebasierten DVB-T-Demodulators zu extrapolieren.

7.4.1 Eigenschaften von DVB-T

Bei DVB-T handelt es sich um ein technisch mit dem DAB-System eng verwandtes System. Im Folgenden sollen für die durchgeführte Abschätzung relevante Unterschiede herausgestellt werden.

Multiplexbildung

Während DAB als Multiplex-Grundmechanismus einzelne isochrone Datenübertragungskanäle anbietet, wird bei DVB-T primär nur ein einzelner synchroner Datenkanal, ein MPEG-2 Transportstrom (MPEG-2 TS), übertragen³. Die zur Verfügung stehende Gesamtdatenrate ist im MPEG-2 Transportstrom gebündelt und wird asynchron beziehungsweise on-demand unter sämtlichen Diensten aufgeteilt. Es besteht also die technische Möglichkeit, variable Bitraten für die Subkanäle zu implementieren und Übertragungskapazität zwischen diesen ad-hoc auszutauschen⁴.

Als Folge ist jedoch kein Zeitschlitzbetrieb möglich. Die zugehörigen Pakete der gemultiplexten Teilservices können im MPEG-2 Transportstrom spontan auftreten. Sie haben somit keine feste Position im Übertragungsrahmen und damit im Zeitablaufschema. Das Konzept des MPEG-2 Transportstroms beinhaltet keine Möglichkeit zur Positionssignalisierung einzelner Datenpakete im Vorfeld. Der Empfänger muss deshalb stets den vollständigen Datenstrom des kompletten Multiplexangebotes demodulieren und mitlesen, erst dann kann die Extraktion des oder der tatsächlich benötigten Teilservices erfolgen.

Modulation

Das Modulationsverfahren erlaubt keine unkohärente Demodulation. Für das kohärente Demapping der OFDM-Unterträger ist vorab eine vollständige Entzerrung des Signals zwingend. Hierzu muss der Zustand der zeitvarianten Kanalübertragungsfunktion im Frequenzbereich $H_i[K]$ für jedes Symbol i und dort sämtliche Unterträger K geschätzt und zeitlich fortgeschrieben werden. Ein entsprechender Funktionsblock ist in der vorliegenden, auf dem DAB-Modell basierenden Abschätzung nicht enthalten. Es muss bedacht werden, dass die mathematisch anspruchsvollen Entzerrungsverfahren in einer Implementierung zusätzlich nennenswerte Rechenleistung erfordern werden.

Fehlerschutzverfahren

Der Faltungscodiercode des DVB-T Systems ähnelt demjenigen des DAB-Systems stark. Er besitzt ebenfalls 64 Zustände, jedoch hat der weniger robuste Muttercode eine Rate $R_c = 1/2$ anstelle $R_c = 1/4$. Er verarbeitet pro Nutzdatenbit somit nur ein Tupel aus zwei statt vier Eingangsbits. Des Weiteren erfolgt die Ratenadaptation mittels Punktierung nicht wie bei DAB bit-flexibel, sondern nach einem einfacheren, fixen Schema.

³Als Sonderfall muss das DVB-T-Schema mit hierarchischer Modulation erwähnt werden. Hier existiert ein zweiter, vom ersten unabhängiger und durch ein Codierungsschema in unterschiedlicher Weise geschützter Transportstrom im gleichen Kanal. Obwohl dieses Verfahren zwar standardisiert ist, wurde es bis zum Zeitpunkt der Arbeit noch in keinem Land kommerziell eingesetzt.

⁴Im Falle von schnellen Bewegungen in Videosequenzen kann so kurzzeitig eine erhöhte Datenrate encodiert werden.

Im DVB-T System ist nach dem Viterbi-Decoder ein weiterer Fehlerschutzmechanismus implementiert. Ein Reed Solomon-Decoder wird eingesetzt, um Restfehler zu eliminieren. Für die hier vorgenommene qualitative Abschätzung, basierend auf den Laufzeitdaten des DAB-Systems, soll pauschal angenommen werden, dass sich die potentiellen Vereinfachungen bei der Implementierung des Viterbi-Decoders für DVB-T mit dem Zusatzaufwand für den nachgeschalteten Reed Solomon-Decoder kompensieren.

7.4.2 Komplexität eines gewählten Übertragungsszenarios

Das DVB-T Schema 16-QAM, $R_c=2/3$, $T_G=1/32T_S$ mit $r_b=13.27$ Mbit/s stellt in Deutschland die kommerziell genutzte Übertragungsparametrisierung mit der niedrigsten Datenrate dar. Es kommt als vergleichsweise relativ robustes Modulationsschema für ausgedehnte Landessendernetze häufig zur Anwendung. Dieses Basisschema soll für die folgende Abschätzung verwendet werden⁵.

Das DVB-T System basiert auf der Taktbasis $f_s=(64/7)$ MSa/s=9.143 MSa/s, folglich skaliert der vom Empfangsfrontend angelieferte Strom der Basisbanddaten im Volumen mit dem Faktor 4.5 verglichen zum DAB-Referenzszenario $f_s=2.048$ MSa/s. Entsprechend wird das Verhalten der Komplexität des Datentransports angenommen. Für die Betrachtung des OFDM-Demodulators, bestehend aus Frequenzkorrektur, Fourier-Transformation und Demapper, muss zusätzlich ein weiterer Faktor berücksichtigt werden, da dieser nicht mehr im Time Slicing-Betrieb arbeiten kann. Beim DAB-Referenzszenario für die Demodulation mit $r_b=(192+32)$ kbit/s betrug der Tastgrad des Time Slicings $DC=26.3\%$. Entsprechend ergäbe sich bei fix $DC=100\%$ Tastgrad eine stets mindestens 3.8fach erhöhte Komplexität. Im Produkt folgt für den OFDM-Demodulatorblock als Schätzung eine um den Faktor $4.5 \cdot 3.8 \approx 17$ gesteigerte Komplexität.

Für das betrachtete DVB-T Schema moderater Nutzdatenrate gilt $r_b=13.27$ Mbit/s. Ausgehend von dem DAB-Szenario hoher Datenrate $r_b=(192+32)$ kbit/s ergibt sich ein Skalierungsfaktor für die Last des Fehlerschutzdecoders von circa 59.

Diese Faktoren, zusammengefasst in Tabelle 7.6, sollen nun in Kombination zu den gemessenen Ressourcenverbrauchswerten für das DAB-System aus Kapitel 5 und 6 gesetzt werden.

7.4.3 Ergebnisse der Abschätzung

Für das DVB-T Szenario mit niedriger Datenrate $r_b=13.27$ Mbit/s zeigt Tabelle 7.7, ausgehend von den bislang erarbeiteten DAB-Performancekennzahlen und extrapoliert nach den Hypothesen des letzten Abschnittes, die Abschätzungsergebnisse. Auf der x86 CPU-Architektur Intel Sandy Bridge wäre bei 2.1 GHz die DVB-T-Demodulation mit einer Last von circa 76.2% auf einem Kern möglich. Dies entspricht in Anbetracht von Echtzeitbetrieb praktisch effektiver Volllast des CPU-Kerns. Die x86 CPU-Architektur Intel Bonell bei 1.6 GHz Takt wäre rechnerisch mit dem fiktiven Wert 227% ausgelastet. Zur Demodulation

⁵Es muss jedoch darauf hingewiesen werden, dass für einen standardkonformen DVB-T-Empfänger der Kanaldecoder selbstverständlich in der Lage sein muss, sämtliche spezifizierten Modulationsvarianten zu demodulieren. Diese weiteren Übertragungsschemen sind mit Raten bis zu $r_b=31.67$ Mbit/s spezifiziert (64-QAM, $R_c=7/8$, $T_G=1/32T_S$).

Extrapolation der Ressourcenanforderungen DAB zu DVB-T

Annahme: typisches DVB-T Szenario moderater Datenrate

Systemkomponente	DAB	DVB-T	Komplexitäts-Faktor
<i>OFDM-Demodulator:</i>			
Samplingtakt	2048 kSa/s	9143 kSa/s	×4.5
Timeslicing-Tastgrad	26.3 %	100 %	×3.8
			Total: ×17
<i>Kanaldecoder:</i>			
Datenrate	224 kbit/s	13.27 Mbit/s	×59

Tabelle 7.6: Vergleich der die Ressourcenanforderungen maßgeblich bestimmenden Parameter der Funkübertragungsszenarios von DAB und DVB-T.

wäre folglich das Signalverarbeitungssystem auf mehrere CPU-Kerne zu partitionieren, es wäre rechnerisch also mindestens ein Intel Atom-Dreikernsystem nötig.

Als nächstes soll die GPU-Mikroarchitektur betrachtet werden⁶. In Kapitel 5 zeigte sich für das DAB-System, dass zwar beispielsweise für die FFT-Algorithmen die GPU Vorteile zeigt, der Algorithmus des Viterbi-Kanaldecoders auf der GPU aber in Relation betrachtet weniger effizient zu lösen ist als auf der SIMD-fähigen CPU. Da gerade dieser logikorientierte Funktionsblock den umfangreichsten Anteil darstellt, und dies bei gesteigerter Datenrate r_b noch intensiver beim Schritt von DAB zu DVB-T, bleibt der Nutzen des Einsatzes der GPU auch für DVB-T fraglich. Sicherlich bietet der für die GPU des Typs NVidia Ion geschätzte Wert durch spezifische Optimierung noch Potential für Verbesserungen. Doch zeigt sich klar anhand einer hypothetischen Überlastung des Chips um den Faktor 15 (1470 % rechnerische GPU-Gesamtauslastung), dass eine GPU aus der Klasse der eingebetteten IGP's in der Leistung bei Weitem nicht ausreichen wird: Es wären 480 anstelle von 16 GPU-internen Rechenwerken (60 anstelle von zwei SM-Teilprozessoren der G9x-Architektur) und somit eine highest-end GPU erforderlich⁷.

Ein ARM Cortex A8 Kern bei 600 MHz müsste rein rechnerisch 783 % Last für die gestellte Aufgabe bewältigen. Bei einer auf 1.2 GHz verdoppelten Taktrate könnte somit ein ARM-SoC mit vier Prozessorkernen unter Volllast möglicherweise ausreichen⁸, um ohne jegliche Hardwarebeschleuniger in reiner Software das DVB-T Signal des evaluierten 13.27 Mbit/s-Basiszenarios zu demodulieren.

⁶In der Abschätzung für die GPU-Architektur findet sich verständlicherweise kein Schätzwert für eine Ressourcenanforderung bezüglich des Transports der Basisbanddaten, da diese Aufgabe konzeptbedingt nicht von der GPU, sondern stets von deren Host-CPU zu bewältigen ist.

⁷Dies entspricht beispielsweise dem Grafikadapter NVidia Geforce GTX 295, bestehend aus zwei GPU-Chips GT200b mit insgesamt exakt 480 Rechenwerken. Die elektrische Leistungsaufnahme liegt im Betrieb laut Hersteller bei bis zu 289 W.

⁸Als Systeme mit vier ARM CPU-Kernen und Taktfrequenzen ≥ 1.2 GHz existieren zum aktuellen Zeitpunkt als Spitzensegment der ARM-Klasse die SoCs NVidia Tegra 3 und 4, Samsung Exynos 4412 und 54x0 sowie Freescale i.MX6 Quad.

Abschätzung DVB-T Demodulation

(Szenario $r_b=13.27$ Mbit/s)
























Code-Bereich	Geschätzte Prozessorauslastung	
CPU Intel SandyBridge: $f_{Clk}=2.1$ GHz		
Korrektur Δf_c -Fehler ($\times 17$)	1.0 %	
Fourier-Transformation ($\times 17$)	4.8 %	
Demapper ($\times 17$)	11.4 %	
Zeit-Deinterleaver ($\times 59$)	5.3 %	
Fehlerschutz-Decoder ($\times 59$)	50.2 %	
Treibermodule HF-Frontend ($\times 4.5$)	3.6 %	
in Summe: 76.2 %		
CPU Intel Bonell: $f_{Clk}=1.6$ GHz		
Korrektur Δf_c -Fehler ($\times 17$)	6.0 %	
Fourier Transformation ($\times 17$)	22.9 %	
Demapper ($\times 17$)	21.6 %	
Zeit-Deinterleaver ($\times 59$)	4.1 %	
Fehlerschutz-Decoder ($\times 59$)	159.9 %	
Treibermodule HF-Frontend ($\times 4.5$)	12.2 %	
in Summe: 226.7 %		
GPU NVidia MCP79: $f_{Clk}=1.1$ GHz, 2 SM-Prozessoren nach G9x-Architektur		
Korrektur Δf_c -Fehler ($\times 17$)	6.8 %	
Fourier Transformation ($\times 17$)	10.2 %	
Demapper ($\times 17$)	25.5 %	
Zeit-Deinterleaver ($\times 59$)	70.8 %	
Fehlerschutz-Decoder ($\times 59$)	1357 %	
Treibermodule HF-Frontend ($\times 4.5$)	(nicht anwendbar)	
in Summe: 1470 %		
CPU ARM Cortex A8: $f_{Clk}=600$ MHz		
Korrektur Δf_c -Fehler ($\times 17$)	47.6 %	
Fourier-Transformation ($\times 17$)	35.7 %	
Demapper ($\times 17$)	44.2 %	
Zeit-Deinterleaver ($\times 59$)	82.6 %	
Fehlerschutz-Decoder ($\times 59$)	542.8 %	
Treibermodule HF-Frontend ($\times 4.5$)	30.2 %	
in Summe: 783.1 %		

Tabelle 7.7: Abschätzung des Ressourcenverbrauchs einer vollständig softwarebasierten Realisierung eines DVB-T-Decoders.

7.5 Prototypische Integration des Empfängers in ein Versuchsfahrzeug

Im Rahmen der Arbeit wurden theoretische Überlegungen in Prototypaufbauten umgesetzt und diese Labormuster hinsichtlich ihrer Leistungsfähigkeit evaluiert. Das vorliegende Kapitel widmet sich der Beschreibung des Aufbaus eines vollständigen Konzeptfahrzeuges, in welchem die untersuchte Herangehensweise für die Implementierung eines Radioempfängersystems gesamtheitlich dargestellt wurde. Neben der erfolgreichen praxisnahen Evaluierung des Softwaredemodulators konnte im Versuchsträger erstmalig eine Anbindung eines DAB-Receiver an ein Ethernet-basiertes Fahrzeugboardnetz demonstriert werden.

Das Demonstratorfahrzeug, dessen Aufbau in Abbildung 7.1 in schematischer Weise dargestellt ist, unterscheidet sich von heutigen Radiosystemarchitekturen für Fahrzeuge in zweierlei Hinsicht:

- Der radiostandardspezifische Demodulator wird, gemäß den Untersuchungen dieser Arbeit, als reine Softwareapplikation ohne dedizierte Hardwarebeschleunigung auf der general-purpose CPU einer eingebetteten low-cost x86 Plattform ausgeführt.
- Anstelle der Nutzung etablierter, proprietärer Feldbussysteme aus der Automobilbranche werden der digitale Echtzeit-Audiostream sowie Steuerinformation via Internet Protocol (IP) über ein Ethernet-Medium im Fahrzeug verteilt.

Hierdurch wird erhöhte Flexibilität sowohl auf der Seite der Algorithmenberechnung als auch auf Seite der Vernetzungsarchitektur ermöglicht. Es wird ein vollständiges, verteiltes Fahrzeugaudiosystem vorgestellt. Der Kontrolldatenfluss wird über ein IP-basiertes Remote Procedure Call (RPC) Framework abgewickelt. Für den Echtzeit-Transport der digitalen Audiodaten im Fahrzeugnetzwerk mittels paketorientierter IP-Technologie kommt ein zeitstempelbasiertes Streamingverfahren zum Einsatz.

7.5.1 Automotive Ethernet/IP

Über die vergangenen zwei Jahrzehnte hinweg lässt sich im Bereich des Fahrzeugboardnetzes eine starke Komplexitätszunahme erkennen: Aufgrund der generellen Etablierung von Elektronikkomponenten im Fahrzeug stieg zuerst die Anzahl zu vernetzender Steuergeräte im Gesamtfahrzeug stark an, resultierend in einer wachsenden mechanischen Komplexität des Kabelbaums. Waren zu Beginn die auf Feldbussen transportierten Datenmengen noch gering bis moderat, so zeichnet sich in jüngster Vergangenheit nun ein vermehrter Bedarf an Datenrate auf den einzelnen Signalleitungen ab. Gründe hierfür liegen zum einen in der zunehmenden Integration von Unterhaltungsmedien ins Auto, zum anderen aber auch in der Einführung neuartiger Fahrerassistenzsysteme (engl. Advanced Driver Assistance Systems, ADAS) wie radar- und kamerabasierte Umfeldüberwachung oder Headup- und Entertainment-Displays [105].

Vor diesem Hintergrund wird aktuell nach Konzepten und Übertragungsmedien für moderne Boardnetzgenerationen gesucht. Zur Verteilung von Multimediainhalten wurde im Automobilbereich bislang der proprietäre Media Oriented Systems Transport (MOST)-Feldbus eingesetzt. Favorisiert durch das Gremium Open Alliance [106] zeichnet sich jüngst eine an das Fahrzeugumfeld angepasste Modifikation des Ethernet-Mediums als gemeinhin

7.5 Prototypische Integration des Empfängers in ein Versuchsfahrzeug

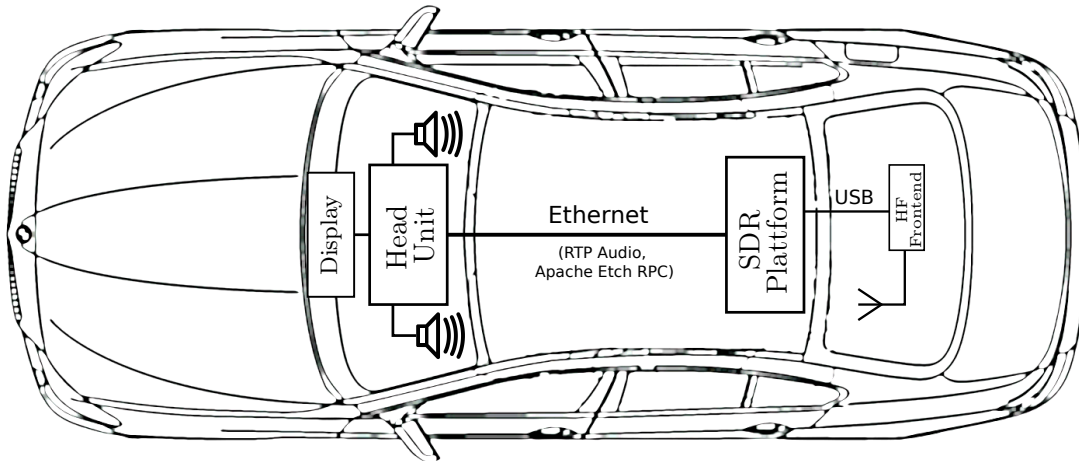


Abbildung 7.1: Konzept des Fahrzeugdemonstrators mit verteilter Audiosignalquelle (DAB-Radioempfangsmodul) und -senke (Head Unit).

akzeptierter zukünftiger Übertragungsstandard ab. Dabei handelt es sich um eine Ethernet-Variante, welche mittels zweiadrigem, ungeschirmtem Twisted Pair-Kabel einen physikalischen Kanal mit 100 Mbit/s full-duplex Datenrate bereitstellt. Der Paradigmenwechsel vom proprietären Bus der Automobilbranche hin zu einer bereits in anderen Märkten erprobten, weitgehenden Standardlösung soll zugleich hohe Bandbreite wie auch niedrige Kosten versprechen. Für die logische Kommunikation oberhalb des physikalischen Mediums Ethernet ist ebenfalls vornehmlich die Anlehnung an bereits etablierte Standardprotokolle vorgesehen [107]. Dies sind im Speziellen diejenigen der Protokollfamilie TCP/IP, welche im IT-Bereich bereits seit Jahren bekannt und im Einsatz bewährt sind.

7.5.2 SDR-Demodulator und Audioquelle

Als eingebettetes System für die zentrale SDR-Plattform wird ein Modul auf Basis der industrial-grade CPU Intel Atom Z530 gewählt (Tabelle 7.8). Diese basiert intern auf der identischen Mikroarchitektur wie die in Kapitel 5 untersuchte CPU Intel Atom N270. Die Plattform ist mit 512 MB RAM als Arbeitsspeicher bestückt, eine programmierbare NVi-

Demonstrator-Plattform

CPU Intel Atom Z530	
Mikroarchitektur	Intel Bonell, x86/IA-32
Prozessorkerne	1
Kerntakt	1600 MHz
Chipsatz	Intel US15W
DRAM Arbeitsspeicher	DDR2-667 (Single Channel)

Tabelle 7.8: Systemplattform des Fahrzeugversuchsträgers.

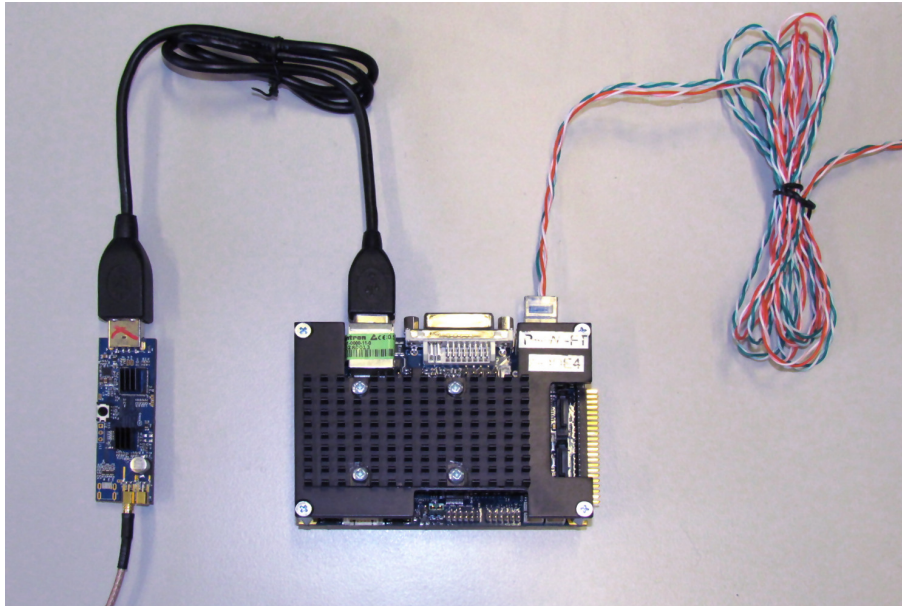


Abbildung 7.2: Die Plattform zur Softwaredemodulation ist via USB-Bus mit dem Modul des HF-Frontends und via Ethernet UTP (Unshielded Twisted Pair) mit dem Head Unit-Emulator verbunden.

die GPU ist jedoch anders als im System aus Kapitel 5 nicht vorhanden⁹. Zum Einsatz kommt für die Digitalradio-Demodulation die vorgestellte x86-Variante der Signalverarbeitungskette mit SSE-SIMD-Optimierung. Ein externes Frontend-Modul mischt in gewohnter Weise das hochfrequente Antennensignal in Basisbandlage und führt anschließend die Analog-Digital-Wandlung durch. Wie bereits in den prototypischen Laboraufbauten wird wieder das USB-Interface gewählt, um die digitalen Basisbandsignale der Rechenplattform zur Verfügung zu stellen (siehe Abbildung 7.2). Der Bus erlaubt vorteilhaft die entfernte Platzierung des Frontends nahe der Fahrzeugantenne beziehungsweise die direkte Integration in das dortige Antennenverstärkermodul. So kann gegebenenfalls in Serienfahrzeugen ein Kostenvorteil durch Elimination vergleichsweise kostenaufwendiger Koaxialleitungen erreicht werden.

Da der digitale Empfangsdatenstrom und so auch die decodierten digitalen Audiodaten phasenstarr aus dem vorgegebenen Takt der Radiosendeanstalt resultieren, müssen im Fahrzeugaudiosystem der Radiodemodulator und seine Streamingquelle als Taktmaster agieren. Die SDR-Plattform ist über ungeschirmte Ethernet-Zweidrahtleitung an einen Ethernet-Switch des prototypischen Fahrzeugboardnetzes verbunden.

7.5.3 Audiosenke und Bedieneinheit

Über diesen zentralen Switch ist auch die sogenannte Head Unit des Fahrzeugs angeschlossen. Hierfür dient ein identisches eingebettetes System auf Intel Atom-Basis, das über

⁹Im Chipsatz Intel US15W ist die in Kapitel 4 eingeführte PowerVR SGX-GPU integriert, für welche die Nutzbarkeit für GPGPU-Anwendungen vermutet wurde und jüngst offiziell bestätigt ist, die jedoch im Rahmen der Arbeit wie dargelegt nicht genauer betrachtet werden konnte.

7.5 Prototypische Integration des Empfängers in ein Versuchsfahrzeug

Anschluss an den Leistungsverstärkerapparat der Kabinenlautsprecher die Funktion eines digitalen Audioverstärkers darstellt. Um den MPEG-Audiostrom zu empfangen und zu decodieren bedient sich das vorgestellte Setup der Routinen des Videolan VLC-Frameworks. Diese Senke adaptiert durch Polyphasen-Resampling der Audiodaten ihren Takt an den der Streaming-Quelle. Weiterhin fungiert dieses System auch als experimentelle zentrale Bedieneinheit (HMI), welche die fahrzeugintegrierte Displayansteuerung und Auswertung der Bedienelemente übernimmt. Der zentrale Haptikcontroller *iDrive* des Fahrzeugs ist dabei angebunden über einen CAN-Feldbusadapter. Mittels eines LVDS-Signalkonverters wird die interne Displaykonsole angesteuert.

7.5.4 IP-basiertes Audio Streaming

Alle 24 ms wird ein vollständiger Rahmen codierter MPEG-Audiodaten vom DAB-Empfängerkern bereitgestellt und entsprechend für den Versand via Ethernet-Bus vorbereitet. Die resultierende Größe der Nutzdaten p in der Datenbyte-Einheit B ist in trivialer Weise dann bestimmt nach folgender Formel:

$$p = \frac{1\text{B}}{8\text{bit}} \cdot 24\text{ms} \cdot r_b \quad (7.1)$$

Hierbei ist r_b die Bitrate des empfangenen DAB Audio Service Streams. Mit der Annahme einer beispielhaften Bitrate von 192 kbit/s ergibt sich eine MPEG-Rahmenlänge von 576 B.

Für die Funktion des Echtzeit-Audiostreamings kommt das Real-Time Transport Protocol (RTP) gemäß RFC3550 [113] zum Einsatz. Die Hauptcharakteristik des RTP-Formats liegt in der Bereitstellung von Zeitstempeln zu den jeweiligen Fragmenten des vom Prinzip

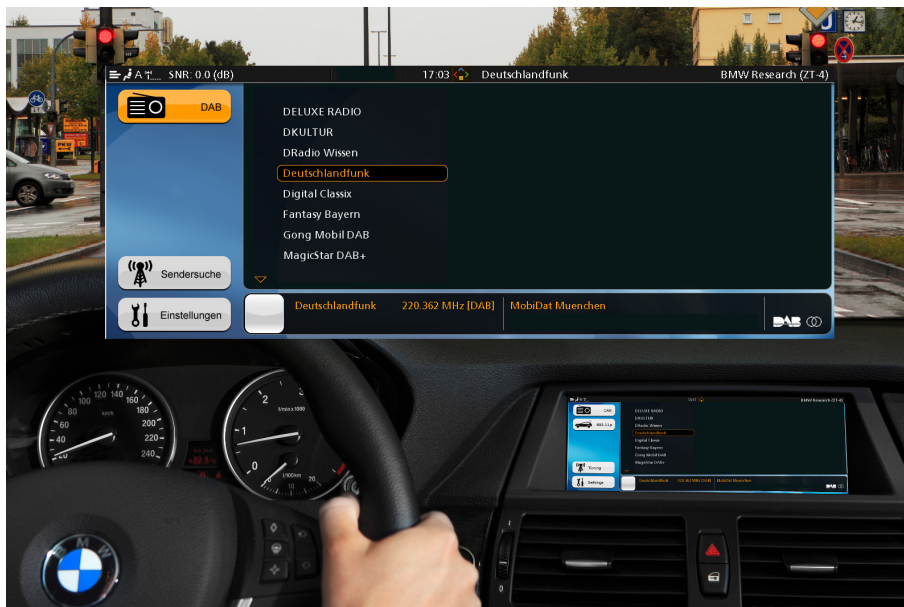


Abbildung 7.3: Der Head Unit-Emulator nutzt einen Embedded PC, welcher der Softwaredemonstrationsplattform gleicht. Für das Benutzerinterface sind die Anzeige-, Audio- und Bedienelemente des Fahrzeugs angebunden.

7 Weitere Untersuchungen im Umfeld der entwickelten Signalverarbeitungskette

kontinuierlichen Nutzdatenstroms. Dies ermöglicht der einen oder auch mehreren RTP-Senken eine zeitgenaue, synchrone Präsentation der Nutzdaten. Daneben ist ein Erkennen von Paketverlusten und, bei fehlerhafter Empfangsreihenfolge, ein Sortieren der Datagramme an der RTP-Senke möglich.

RTP bedient sich im Protokollstapel der untergeordneten Protokollschichten UDP nach RFC768 [114] und IP nach RFC791 [115]. Bedingt durch diese Hierarchie ergibt sich für die Nutzdaten ein Overhead, welcher die Paketheader der Layer nach Tabelle 7.9 umfasst. In Summe werden so 58 B den Nutzdaten eines jeden Paketes hinzugefügt. Dies erhöht die effektive Streamingrate r_{total} auf der Schicht des Ethernet-Mediums um 19.33 kbit/s.

$$r_{\text{total}} = r_b + \frac{58 \cdot 8 \text{ bit}}{24 \text{ ms}} = r_b + 19.33 \text{ kbit/s} \quad (7.2)$$

Betrachtet für die exemplarische Audiorate von 192 kbit/s wird somit beim IP-basierten Audiotransport durch Protokolloverhead circa 10 % Datenratenbedarf zur Signalisierung zusätzlich hinzugefügt (Tabelle 7.10). Dieser Overhead könnte potenziell reduziert werden, indem MPEG-Rahmen von 24 ms Länge in Pakete gruppiert würden, was jedoch die Latenz des Systems erhöhen würde und auch unter dem Aspekt der Paketfragmentierung kritisch erscheint (siehe nächster Absatz). Für eine DAB-Audioservicerate $r_b = 384 \text{ kbit/s}$, welche als absolut mögliches Maximum gemäß DAB-Standard angesehen werden kann, stellt sich die Gesamtdatenrate r_{total} ein auf den Wert 403 kbit/s. Dieser Wert ist, auch bei Querdatenaufkommen im Netzwerk, auf der anvisierten 100 Mbit/s Feldbusverbindung zu realisieren. Rahmen auf der Ethernet-Schicht sind in ihrer Größe limitiert durch den Wert der sogenannten Maximum Transmission Unit (MTU, siehe [115]): Die Nutzdatenlänge ohne Ethernet-Header darf den Wert von 1500 B nicht überschreiten, ansonsten muss das Paket auf IP-Protokollschicht fragmentiert werden. Für $r = 384 \text{ kbit/s}$ wird ein vollständiges 24 ms Audio-Paket eine Größe von $p = 1152 \text{ B}$ plus 40 B Headerinformationen der Protokolle IP, UDP, RTP aufweisen. Dies liegt im Rahmen der Begrenzungen des MTU-Wertes, somit ist im gesamten Usecase-Szenario keine IP-Layer-Fragmentierung notwendig.

RTP/IP-Protokolloverhead, absolut

Protokoll	Header-Länge
Ethernet*)	18 B
IP	20 B
UDP	8 B
RTP	12 B
<i>in Summe, pro Paket:</i>	58 B

*) ohne VLAN-Tag.

Tabelle 7.9: Zusammenstellung des bei RTP/IP/Ethernet-basiertem Audiostreaming entstehenden absoluten Overheads pro Datenpaket.

RTP/IP-Protokolloverhead, relativ

Audio Bitrate r_b	Overhead (+19.33 kbit/s)
64 kbit/s	30.2 %
96 kbit/s	20.1 %
128 kbit/s	15.1 %
192 kbit/s	10.1 %
256 kbit/s	7.6 %
384 kbit/s	5.0 %

Tabelle 7.10: Abbildung des durch RTP/IP/Ethernet-basierten Audiostreaming entstehenden relativen Overheads bei typischen DAB Audiodatenraten r_b .

7.5.5 IP-basierte Middleware

Neben der Übertragung des Audiostroms ist in einem verteilten Multimediasystem auch der Austausch von Kontroll- und Steuerdaten nötig. Die Implementierung der Radiokomponente muss Senderlisten im Systemverbund bereitstellen und in der Lage sein, Tuning-Kommandos von einer entfernten Bedieneinheit entgegenzunehmen.

Im Sinne einer Koordinierung der auf verschiedenen Plattformen entfernt ausgeführten Teilprozesse sind Methoden zur Interprozesskommunikation notwendig. Für die konzeptionelle Art des Informationsaustausches bietet sich das Modell der Remote Procedure Calls (RPC) an. Beim RPC-Interaktionsmechanismus handelt es sich im Grunde um eine Transaktion nach dem klassischen Client-Server-Modell. Die vom Server angebotenen Dienste stellen eine Möglichkeit dar, auf seiner Plattform bestimmte Prozeduren seines Anwendungsprozesses aufgrund externer Anforderung zur Ausführung zu bringen. Die sogenannte Middleware abstrahiert von den technischen Verfahren, welche zur Realisierung eines solchen RPCs nötig sind. Durch sie erscheint für den Prozess des Clients die Ausführung einer Methode auf dem entfernten Server im Idealfall vollkommen transparent, das heißt identisch dem Aufruf einer Funktion auf der lokalen Plattform.

Entsprechend der Definition von Issarny et al. [116] spezifiziert eine Middleware durch eine verbindliche Interface Description Language (IDL)

- ein Transportprotokoll für den Nachrichtenaustausch inklusive der zugehörigen Syntax und Semantik für einen entfernten Befehlsaufruf,
- ein Koordinierungsmodell für die Abarbeitung der Interaktionsnachrichten,
- ein Adressierungsschema für die Absender- und Zielgeräte sowie gegebenenfalls ein Service Discovery Protocol (SDP), um Ressourcen im verteilten Systemverbund anzukündigen und zu entdecken.

Als Middleware für den möglicherweise zukünftigen Einsatz im Automobilbereich konkurrierten zum Zeitpunkt der Arbeit die RPC-Frameworks SOME/IP und Apache Etch [108].

Für den vorgestellten Prototyp wurde eine Implementierung gewählt, welche sich an den Spezifikationen von Apache Etch orientiert. Da die existierenden Apache Etch-Bindings primär auf Workstations aus dem IT-Bereich statt auf eingebettete Systeme zielen, wurde im Rahmen der Arbeit eine eigene, schlanke Implementierung erarbeitet. Diese Implementierung richtet sich speziell an die Anforderungen von Embedded Computing-Anwendungen im Sinne eines auch für ressourcenbeschränkte Mikrocontrollerarchitekturen tauglichen Programmierstils. Sie wurde in [117] weiterentwickelt und im Detail evaluiert.

7.5.6 Resümee zum Demonstrator-Gesamtfahrzeug

Mit dem Demonstrator konnte das vorgestellte Konzept aus softwarebasierter Demodulation von Digitalradio und IP-basierter fahrzeuginterner Kommunikation dargestellt werden. Stabiler Empfang und synchrone, lückenlose Audiowiedergabe des Systems wurden erfolgreich demonstriert. Die softwarebasierte Demodulation beansprucht auf dem eingesetzten Applikationsprozessor nur moderate Ressourcen, ebenfalls ist der Transport des codierten RTP-Audiostroms über ein IP/Ethernet-basiertes Fahrzeugboardnetz problemlos möglich. Tabelle 7.11 fasst die hinsichtlich des Usecases wesentlichen Daten des prototypischen Setups abschließend zusammen.

DAB-Softwaredemodulation und Ethernet-Audiostreaming	
Systemkomponente	Benötigte Ressourcen
Prozessor:	
DAB-Softwaredemodulation (Tuner)	äquivalent 325 MHz* (20.3% Last @ 1.6 GHz Intel Bonell)
MPEG-Audiodecodierung (Head Unit)	äquivalent 100 MHz* (6.1% Last @ 1.6 GHz Intel Bonell)
Buskapazität:	
Audiostreaming (Ethernet)	403 kbit/s* (0.4% Last von 100 Mbit/s)
HF Frontend-Bus (USB)	64 Mbit/s (13.3% Last von 480 Mbit/s)

*) für Extremfall standardkonformer DAB Servicebitraten bis zu $r_b=384$ kbit/s

Tabelle 7.11: Zusammenstellung der benötigten Systemressourcen des vorgestellten Demonstrator-Infotainmentsystems.

8 Zusammenfassung und Ausblick

8.1	Diskussion der CPU-Mikroarchitekturen mit integriertem SIMD-Coprozessor	183
8.2	Diskussion der GPU-Mikroarchitekturen	184
8.3	Folgerungen für die Plattformauslegung und Produktentwicklung einer Automotive Entertainment-Plattform	189
8.4	Ausblick	191

Zur Untersuchung der flexiblen Umsetzung der Basisbandsignalverarbeitung von Sende- und Empfangsbaugruppen in reprogrammierbaren Strukturen wurde deren Realisierung mit Fokus auf Prozessorplattformen im Detail betrachtet. Als Fallstudie wurde eine mathematisch effiziente Signalverarbeitungskette für die standardkonforme Demodulation von Digital Audio Broadcasting (DAB) erarbeitet und plattformspezifisch optimierte Implementierungsvarianten ausgeleitet. Ein Hauptuntersuchungsgegenstand war die Evaluierung von general-purpose Mikroprozessorarchitekturen für Radiosignalverarbeitungsaufgaben. Als besonderer Aspekt erfolgte außerdem die Evaluierung von im System verbauten, freiprogrammierbaren Grafikbeschleunigerstrukturen zur Nutzung für Radiosignalverarbeitungsaufgaben. Die Untersuchung wurde auf zwei hybriden low-cost x86-CPU/GPU-Plattformen sowie einer hybriden ARM-CPU/DSP-Plattform durchgeführt. Mit dem Anspruch an einen fairen Architekturvergleich insbesondere zwischen CPU und GPU wurde sowohl der Programmcode der GPU als auch derjenige der CPU vor der Laufzeitanalyse einer intensiven Optimierung unterzogen. Auf allen Architekturen wurden die Basisalgorithmen auf eine bestmögliche Passung zu den jeweiligen Hardwarespezifika adaptiert. Dies umfasst in sämtlichen Fällen die Nutzung vorhandener SIMD-Rechenwerke sowie die Beachtung der prozessorspezifischen Speicherzugriffsmuster.

8.1 Diskussion der CPU-Mikroarchitekturen mit integriertem SIMD-Coprozessor

Der Einsatz der SIMD-Coprozessorsysteme SSE und NEON mit 128 bit-breiten Vektor-Registern erlaubt auf den Applikationsprozessoren x86 und ARM eine Parallelisierung der Datenverarbeitung. Für die im vorliegenden Softwareradioanwendungsfall hauptsächlich

auf 16 bit-Arithmetik basierenden Signalverarbeitungsalgorithmen folgt ein möglicher Parallelitätsgrad von acht. Die dazu nötige Umsetzung mittels direkter Maschinenbefehle ist aufwändig, jedoch lohnenswert. In den untersuchten Realisierungen der Algorithmen konnten stets nennenswerte Beschleunigungsfaktoren im Bereich drei bis acht durch Nutzung der SIMD-Fähigkeiten der CPUs erreicht werden.

Die Untersuchungen zeigen für den gewählten Anwendungsfall, dass aktuelle, initial für general-purpose Aufgaben vorgesehene x86- und ARM-Applikationsprozessoren durch die integrierten SIMD-Befehlssatzerweiterungen die Performance dedizierter DSP-Signalprozessoren erreichen können.

x86/SSE-Architektur

Der Befehlssatz der x86/SSE-Architektur erweist sich trotz des Vektorcharakters als gut geeignet für die Verarbeitung verschachtelter Datensätze mit irregulärer Elementezuordnung. Durch Interleavingmechanismen können im SIMD-Vektor die einzelnen Komponenten extrahiert sowie neu angeordnet und damit effizient für die folgende datenparallele Berechnung vorbereitet werden. Bei den arithmetischen Operationen kann die Möglichkeit zur Präzisionspropagation der Werte im Vektor vorteilhaft genutzt und arithmetische Genauigkeit gegen Parallelitätsgrad getauscht werden. Die SSE-Registerbank umfasst nur acht Plätze und verhindert dadurch für die gestellten Algorithmen teilweise höhere Effizienz. So muss vor allem die Viterbi-Implementierung fortgesetzt Zwischenergebnisse prozessorextern ablegen. Auch für die dennoch effiziente Fourier-Transformation gibt die Registeranzahl eine Schranke, da komplexere algorithmische Mechanismen zur weiteren Erhöhung der Datenlokalität aufgrund der beschränkten internen Register nicht umsetzbar sind.

ARM/NEON-Architektur

Für den Advanced-SIMD-Befehlssatz der konzeptionell ähnlichen NEON-Architektur des ARM Cortex A gelten die bereits erwähnten Charakteristika ebenfalls. Die Flexibilität der Interleavingmechanismen des NEON-Systems geht sogar über diejenige der x86/SSE-Vektorbefehlssatzarchitektur hinaus. Bereits direkt beim Lade- und Speichervorgang können Datenmuster entflochten werden. Dies erweist sich beispielsweise im untersuchten Themenfeld der Radiosignalverarbeitung als besonders hilfreich für die Bearbeitung von komplexwertigen Datensätzen mit Real- und Imaginärteilen. Die doppelte Kapazität der NEON-Registerbank im Vergleich zu derjenigen der SSE-Architektur lässt sich stets vorteilhaft nutzen und hilft den untersuchten Algorithmen stark, Zugriffe auf temporäre Werte im langsamen prozessorkernexternen Speicher zu vermeiden.

8.2 Diskussion der GPU-Mikroarchitekturen

Die Mikroarchitekturen der programmierbaren GPUs sind naturgemäß für die Algorithmen des initialen Anwendungsfalls 3D-Grafikbeschleunigung optimiert. Hinsichtlich der Nutzung für fachfremde Algorithmen wird in der Literatur oft von überdurchschnittlicher Rechenperformance der GPU-Mikroarchitektur und über aufgrund ihrer hohen Parallelität realisierte Geschwindigkeitsgewinne berichtet, auch für den Fall der Radiosignalverarbeitung.

Neuheitswert der GPU-Architektur

Die GPU wird vielfach in der Literatur im Vergleich zu Standardprozessoren als neuartige, im Unterschied zu diesen massiv-parallele Architektur geführt. Dies steht nicht in Übereinstimmung zu der durchgeführten detaillierten Analyse der GPU-Hardwarearchitektur mit Vergleich zu konventionellen Mikroarchitekturen.

Es zeigte sich, dass sich die Anzahl an vorhandenen Rechenwerken in beiden Konzepten in vergleichbaren Größenordnungen bewegt. Die von den GPU-Herstellern gelieferten Angaben zur Anzahl vorhandener Recheneinheiten, insbesondere mit der Bezeichnung "Kerne", sind kritisch auf die Definition der jeweilig proklamierten Struktur der Verarbeitungseinheiten zu hinterfragen, bevor ein Vergleich von GPUs zu gewöhnlichen CPUs angestellt wird. Denn heutige CPU-Prozessoren stellen mitnichten "Single Instruction Single Data"-Prozessoren dar, sie besitzen ebenfalls eine Vielzahl nebenläufiger Verarbeitungseinheiten. So hat als Beispiel die NVidia Ion-GPU zweimal acht Rechenwerke zur Verfügung, die über ein SIMD-Konzept gekoppelt sind, jedoch nach offizieller Terminologie des Herstellers als eigenständige "Kerne" benannt werden. Dem gegenübergestellt besitzt eine Intel Atom Singlecore CPU in ihrer dual-superskalaren SSE-Verarbeitungseinheit das Potential, mit diesem einzelnen CPU-"Kern" bei 16 bit Genauigkeit ebenfalls zweimal acht arithmetische Operationen parallel zu berechnen.

Die untersuchten GPU-Klassen präsentieren sich zusammenfassend als ein, bezogen auf den Stand der Technik, nicht im Rahmen außergewöhnlicher Strukturen liegendes Prozessordesign. Die GPU-Architektur basiert im Wesentlichen auf dem bekannten Konzept eines symmetrischen Multiprozessor-DSPs mit SIMD-Verarbeitungseinheiten für Fließkomma-Arithmetik, speziell entwickelt und optimiert für ihren Hauptanwendungsfall der 3D-Grafik im Consumer-PC-Segment. Die Programmierweise der GPU-SDKs lehnt sich an die wissenschaftlichen Erkenntnisse und praktischen Herangehensweisen datenparalleler Verarbeitung aus dem Umfeld historischer Many-Core- und Vektor-Supercomputer an. Die Softwarekonzepte der die GPU in das CPU-Gesamtsystem einbindenden GPGPU-Computing Frameworks ähneln den in eingebetteten Systemen etablierten Methoden heterogener Applikationsprozessor-/DSP-Systemkombinationen.

Strenge Datenparallelität als limitierende Einschränkung

Anders als typische Algorithmen zur Grafikverarbeitung sind diejenigen der Nachrichtentechnik oft nicht hinsichtlich Speicherzugriff von Regularität und strenger Datenparallelität geprägt. Die Ursache hierfür kann unter anderem im nachrichtentechnischen Grundkonzept der Nutzung von Transinformation gesehen werden: Für die Theorie der modernen Nachrichtensignalverarbeitung stellt das Sammeln, Verwürfeln und Verteilen von Transinformation zwischen im Idealfall sämtlichen Datenpunkten des Datensatzes ein grundlegendes Paradigma dar. Eine Architektur, welche für strikte Datenparallelität entworfen ist, also Datensätze ohne jegliche Abhängigkeiten zueinander optimal verarbeitet und verschränkte Datenmuster meidet, widerspricht diesem Grundsatz vollständig. Entsprechend zeigten sich bei der vorliegenden Untersuchung auf der GPU diejenigen Operationen als stark die Laufzeitperformance limitierende Codefragmente, an denen Datenpunkte gemäß unregelmäßiger, (pseudo-)zufälliger Muster verknüpft werden müssen.

Weiterhin kann auf der GPU im Programmfluss die Parallelität nicht von Instrukti-

on zu Instruktion ad-hoc festgelegt werden. Dementsprechend können Algorithmen mit inhomogener Parallelitätsdimension nicht optimal implementiert werden. Dies wurde im vorliegenden Applikationsbeispiel am rechenintensivsten Algorithmus, dem Viterbi-Fehler-schutzdecoder, welcher stets zwischen vier bis 64-facher Parallelität wechseln muss, deutlich sichtbar.

Datentypen und numerische Präzision

Die GPU bietet effiziente Berechnungen für den nativen Datentyp von 32 bit Fließkommawerten. Dieser Datentyp war in der untersuchten Anwendung allerdings auch dann zu wählen, obwohl die Signalverarbeitungskette die Genauigkeitsanforderungen mit 16 bit Wortbreite hätte erfüllen können.

Die Verwendung von Fließkommazahlen im Signalverarbeitungssystem erleichtert zwar dem Programmierer die Umsetzung der Algorithmen, da hinsichtlich des verfügbaren hohen Dynamikbereiches bei Fließkommaarithmetik eine genaue Betrachtung und Kontrolle des Signalpegels beim Entwurf eines jeden Berechnungsschritts vernachlässigt werden kann. Für die Funktionalität der Baugruppe selbst ist in einem Radiosignalverarbeitungssystem jedoch kein Vorteil durch eine Fließkommaarithmetik unterstützende Mikroarchitektur gegeben. Im Gegenteil kann durch die Wahl von Fixpunktarithmetik mit reduzierter Wortbreite die Datenmenge und somit Speichertransferbandbreite oft gegenüber der Fließkommavariante deutlich reduziert werden.

Auch erscheint die Möglichkeit zur flexiblen Rekonfiguration der SIMD-Datenpfade auf einer parallelen Rechenwerksarchitektur als wichtiges Hilfsmittel. Kann die Wortbreite der einzelnen Vektorelemente verringert werden, dann kann die SIMD-Verarbeitungseinheit mit ihrer konstanten Bitbreite mehr Elemente parallel verarbeiten. So kann stets gemäß der Anforderungen des jeweiligen Rechenschrittes die numerische Präzision gegenüber erzielbarem Parallelitätsgrad abgewogen und die vorhandene Rechenwerkslogik optimal genutzt werden. Ist in einer Vektorprozessor-Architektur wie den evaluierten GPUs keine Wortbreitenrekonfiguration der Verarbeitungseinheit vorgesehen, so ist der SIMD-Parallelitätsgrad konstant und es müssen Rechenoperationen oft mit zu hoher numerischer Präzision bearbeitet werden. Entsprechend ist die numerische Auslastung der Verarbeitungseinheit niedrig und zugleich die belegte Speichertransferbandbreite unökonomisch hoch.

Leistungsfähigkeit für Signalverarbeitungsalgorithmen

In der Literatur werden für GPU-Berechnungen hohe Geschwindigkeitsgewinne gegenüber CPU-Implementierungen berichtet. Im Laufe der Untersuchung zeigte sich jedoch, dass dies mit Blick auf das hier geprüfte praktische Anwendungsszenario nicht generell bestätigt werden kann. Die GPU, angesehen als Prozessorklasse, kann sich in Summe von der Performance heutiger CPUs, welche allesamt SIMD-Befehlssatzerweiterungen implementieren, nicht positiv absetzen.

Zwar erweisen sich die Architekturen der GPUs im Vergleich als sehr leistungsfähig bei der Berechnung der Fourier-Transformation. Die regelmäßigen Muster der Datenverarbeitung scheinen sich gut abbilden zu lassen, auch ermöglicht die große interne Registerbank das Halten vieler Teilergebnisterme im Prozessorkern und somit eine schnelle Verarbeitung. Jedoch treten insbesondere bei der Umsetzung des für die vorliegende nachrichten-

technische Anwendung wichtigsten und somit maßgeblichen Algorithmus, dem stark an Logikoperationen orientierten Viterbi-Decoder, Ineffizienzen zu Tage. Auf den vorliegenden Chips konnten deshalb sowohl für die NVidia Ion-GPU als auch die AMD Radeon HD6310-GPU im Totalen über sämtliche Algorithmen der Verarbeitungskette keine Lastwerte dieser Prozessoren besser als 26.9 % bzw. 31.1 % erreicht werden. Hier können die vektorfähigen low-cost general-purpose Prozessoren weit effizienter arbeiten (6.1 % beziehungsweise 7.3 % anteilige Prozessorlast der Signalverarbeitung auf dem Intel Atom N270 beziehungsweise auf einem x86-Kern der AMD E-350 CPU).

Maximaler und realer Durchsatz von Operationen pro Zeit

Als Argument für die Leistungsfähigkeit von Prozessoren und im Besonderen GPUs wird oft deren maximale Anzahl arithmetischer Operationen pro Sekunde angeführt.

Der Wert der maximal erreichbaren Anzahl an arithmetischen Rechenoperationen pro Zeit ist keine Bewertungsmetrik für die Eignung einer Prozessorarchitektur für ein Anwendungsszenario. In einem realen Einsatzszenario wird auf einem Allzweckrechner die tatsächliche Auslastung der vorhandenen Verarbeitungseinheiten in der Regel nicht das Optimum erreichen können. Der Bezug zwischen Parallelitätsstruktur der Prozessorhardware und des ausgeführten Algorithmus als ein Grund dafür wurde bereits erwähnt. Weiterhin ist als Argument hierfür zu betonen, dass arithmetische Operationen selbst nur ein Teil eines Computerprogramms sind. Es darf nicht vernachlässigt werden, dass die Leistungsfähigkeit einer Rechenarchitektur in ebenbürtiger Weise dadurch bestimmt wird, wie performant und wie flexibel Daten zu den Verarbeitungseinheiten hin und von diesen fort transportiert werden können. Diese obligatorischen Hilfsoperationen, welche stets neben den eigentlichen Arithmetikbefehlen ausgeführt werden müssen, bedingen die Möglichkeit, dass die Implementierung eines Algorithmus – verglichen zu einer Mikroarchitektur mit hohem arithmetischem Maximaldurchsatz – auf einer Mikroarchitektur mit geringerem arithmetischem Maximaldurchsatz, aber flexiblerem Datenzugriff, trotzdem deutlich performanter ausfallen kann.

Massive Parallelität und deren Nutzung

Im Zuge der Untersuchung wurde zudem offensichtlich, dass für die vorgeschriebenen Algorithmen deutliche Schranken hinsichtlich des erschließbaren Grades an Parallelität existieren. Ist der Parallelitätsgrad eines Algorithmus erschöpft, so lässt sich dieser nur durch den Datenfluss aufstauende Signalpufferung und parallele Stapelverarbeitung mehrerer über die Zeit gesammelter unabhängiger Datensätze weiter erhöhen. Somit wird allerdings auch stets zusätzliche Latenz in das Gesamtsystem eingeführt, für die in praktischen Anwendungen jedoch Grenzen bestehen können. Folglich ist ein hoher Parallelitätsgrad der Hardware kein prinzipieller Garant für gesamtheitlich zufriedenstellende Ausführung einer gegebenen Applikation.

Die praktischen Untersuchungen bestätigen dies. Für den gegebenen Anwendungsfall DAB ist ein massiv hoher Parallelitätsgrad der Verarbeitungshardware für die tatsächliche Umsetzung nicht vorteilhaft nutzbar und darf somit nicht als Maß für die potentielle Leistungsfähigkeit der zu nutzenden Plattform betrachtet werden. Als Beispiel dient erneut der laufzeitmaßgebliche Funktionsblock des Viterbi-Fehlerschutzdecoders mit maximal 64-

facher Parallelität. Vielmehr zeigte sich im Rahmen der Studie wie bereits angedeutet, dass für Radiosignalverarbeitung neben Parallelität an sich die Möglichkeiten einer Mikroarchitektur zur schnellen Vektorpermutation und zur Rekonfiguration der Parallelität, das heißt zur ad-hoc Festlegung der Dimension des Workload-Arrays pro Rechenschritt, als wichtig für eine effiziente Bewältigung der Rechenlast anzusehen ist.

Verbleibende Last der Host-CPU und Profilingmethodik der GPU

Die ursprüngliche Intention basierte auf der Idee, die GPU im System beim vorliegenden Anwendungsfall für eine Entlastung des Hauptprozessors zu nutzen. Auf den Testplattformen wurde interessanterweise beobachtet, dass die CPU-Entlastung, welche durch Auslagerung von Algorithmen auf die externe GPU erreicht wurde, durch den Systemoverhead, der im GPU-Framework zur Kontrolle des heterogenen Multicore-Systems CPU-GPU nötig ist, konsumiert wird. Für die Intel Atom/NVidia Ion-Plattform wurde die Host-CPU durch den asymmetrischen Multiprocessing-Betrieb der GPU sogar signifikant stärker belastet, verglichen mit der Vornahme der Berechnung durch die CPU alleine. Die GPU konnte also überraschenderweise nicht zu einer CPU-Entlastung beitragen.

Dieser Overhead kann nur mit einer systemweiten Profilinganalyse der Host-CPU gleichzeitig zur GPU-Ausführung sichtbar gemacht werden, da er maßgeblich in der Systemsoftware begründet ist. Eine Profilinganalyse nur der eigenen Codebasis der Applikation ist folglich nicht ausreichend und führt unter Umständen zu nicht die Realität korrekt abbildenden Ergebnissen. Deshalb ist darauf hinzuweisen, dass zu einer Laufzeituntersuchung eines GPU-Programms stets auch eine begleitende Analyse des CPU-Hosts einschließlich des Laufzeitverhaltens sämtlicher Betriebssystem- und Treiberanteile durchgeführt werden muss.

GPU-Nutzung und Echtzeitbetrieb

Die bis heute verfügbaren Laufzeit-Frameworks für nichtgrafische Nutzung der GPU bieten noch keinerlei Unterstützung für den Systembetrieb mit harter Echtzeitanforderung. Während dies beispielsweise für die Nutzung zu simulativen Berechnungen keine Limitierung darstellt, steht dies jedoch im Konflikt zu den elementaren Grundanforderungen des Usecases zuverlässiger und produkttauglicher Signalverarbeitungssysteme. Hier müsste auf Bereitstellung entsprechender echtzeitfähiger Frameworks seitens der GPU-Chiphersteller hingewirkt werden, so dass Berechnungen nicht nur auf Basis best-effort durchgeführt werden.

Einsatzszenarien für die GPU

Die Datenverarbeitung auf der GPU sollte sich auf streng datenparallele Algorithmen beschränken, überkreuzte Datenzugriffsmuster führen in der Regel zu Effizienzverlusten. Aufgrund bislang fehlender Unterstützung durch die Programmierframeworks darf das anvisierte Applikationsszenario keine Echtzeitanforderungen besitzen. Durch die Verwendung von Hochsprache-Dialekten kann die Programmierung gegenüber der Assemblerorientierten Programmierung der x86- und ARM-SIMD-Systeme vergleichsweise schnell

umgesetzt werden. Entsprechend zeichnet sich nach Einschätzung des Autors als Anwendungsszenario der GPU-Nutzung beispielsweise ein Einsatz im Simulations- und Rapid-Prototyping-Umfeld ab.

8.3 Folgerungen für die Plattformauslegung und Produktentwicklung einer Automotive Entertainment-Plattform

Ausgangspunkt der Untersuchung war die Klärung der Realisierbarkeit eines Radiosignalverarbeitungssystems anstelle mithilfe von ASICs alleine unter Verwendung des low-cost Applikationsprozessors, hierbei mit Fokus auf die beiden möglicherweise einzusetzenden Architekturen x86 und ARM. Weiterhin bestand der Wunsch, in einer konkreten Produktentwicklung gegebenenfalls eine aktuelle programmierbare GPU für die Radiosignalverarbeitung einzusetzen. Die praktische Realisierbarkeit konnte für digitale Hörfunkanwendungen bestätigt werden.

Vergleich zwischen x86- und ARM-Architektur

Im absoluten Vergleich zeigen sich für den gewählten Anwendungsfall die vorliegenden Implementierungen der x86-Mikroarchitekturen erwartungsgemäß performanter als die der untersuchten ARM Cortex A8-Architektur. Diese absoluten Werte skalieren jedoch sinnvoll, wenn für die Plattform Taktfrequenz und Energieaufnahme (sowie auch Preis) pro Prozessorkern mitberücksichtigt werden. Aufgrund der Vielzahl von verfügbaren SoC-Varianten speziell für Embedded-Anwendungen, der neuerdings verfügbaren und gut skalierbaren ARM Mehrkern-CPU's sowie der größeren Auswahl an Herstellern und somit Lieferanten für Prozessoren nach ARM-Architektur wird für eine automotiv Multimediaplatform die Entscheidung für die ARM-Architektur vom Verfasser nahegelegt. Während bei aktuellen x86-Systemen der SIMD-Coprozessor obligatorisch vorhanden ist, muss für ARM-Systeme auf das Vorhandensein eines solchen im gewählten Chipexemplar geachtet werden, wenn anforderungsstarke Signalverarbeitungsaufgaben zu bearbeiten sind.

Nutzung der GPU für Radiosignalverarbeitung

Als Fazit der durchgeführten Untersuchungen kann mit Bezug auf die evaluierten Plattformen die Empfehlung festgehalten werden, für Echtzeit-Radiosignalverarbeitung auf einem x86-System die GPU nicht in Betracht zu ziehen. Im Hardwaredesignprozess sollten bei der Auswahl eines Grafikbeschleunigers aus der Motivation des Einsatzes für eine Radiosignalverarbeitung heraus keine extra GPU-Leistungsressourcen begründet werden. Die Empfehlung, die GPU nicht zu nutzen, schließt aber auch den Fall einer ohnehin im x86-System bereits vorhandenen GPU ein. Es ergibt sich durch den Einsatz der GPU keine Entlastung der Haupt-CPU, noch ergibt sich eine effizientere Nutzung des Gesamtsystems, vielmehr werden nun die Ressourcen zweier Prozessoren allokiert. Außerdem steigt die elektrische Gesamtleistungsaufnahme. Der vorgestellte GPU-Forschungsdemonstrator zur Evaluierung des Systems operierte unter exklusiver Systemnutzung und konnte deshalb ohne Echtzeitunterstützung des GPU-Frameworks die Zeitablaufanforderungen erfüllen und stabilen

Empfang auch unter Nutzung der GPU demonstrieren. Jedoch wäre wie bereits dargestellt in der Multitaskingumgebung eines realen Infotainmentsteuergerätes ohne Sicherung von Echtzeitmechanismen auf der GPU mit gravierenden Problemen zu rechnen¹.

Nutzung eines DSPs zusätzlich zu einem ARM-System

Hinsichtlich eines Applikationsprozessorkerns mit ARM NEON-Erweiterung bewegen sich die benötigten Laufzeitressourcen einer Digitalradio-Implementierung näherungsweise in einer ähnlichen Größenordnung des zum Vergleich exemplarisch herangezogenen bekannten DSPs TMS320C64x von TI. Die Signalverarbeitungsroutinen können vom Betrachtungspunkt der Prozessorlast her also direkt auf einer heutigen ARM-Applikationsprozessor-Mikroarchitektur realisiert werden.

Die Erweiterung der ARM-Plattform um ein dediziertes DSP-Subsystem mit einem zusätzlichen architekturfremden Prozessorkern ist also nicht zwingend notwendig. Einer einheitlichen Systemstruktur nur auf ARM-Basis, gegebenenfalls als homogenes Multicore-System ausgeführt, ist nach Auffassung des Autors zusätzlich hinsichtlich praktischer Gründe der Vorzug zu geben. Die homogene Multicore-Struktur bietet auf einfache Weise die Möglichkeit, über die Anzahl der Kerne die Leistungsfähigkeit des Gesamtsystems vergleichsweise einfach zu skalieren oder, falls die Signalverarbeitung deaktiviert ist, dessen Laufzeitressourcen flexibel zusätzlich für dritte Applikationsprozesse zu nutzen. Sie reduziert ferner sowohl im initialen als auch evolutionären Softwareentstehungsprozess den Entwicklungsaufwand gegenüber einer heterogenen Plattformstruktur. Des Weiteren kann die vom eigentlichen Halbleiterhersteller unabhängige Befehlssatzarchitektur des ARM NEON-Konzeptes Vorteile hinsichtlich Softwarekompatibilität und Software-Reuse zwischen Prozessorprodukten verschiedener Halbleiterhersteller bieten (und somit für CPU-Hardware annähernd Second-Source Lieferoptionen eröffnen), wohingegen für dedizierte DSP-Architekturen jeder Halbleiterhersteller eigene Befehlssatzarchitekturen einführt und somit eine auf sein Produkt festgelegte Softwareentwicklung fordert.

Im konkreten Fall werden verfügbare homogene ARM Cortex A8/A9-Mehrkern-Kombinationen mit NEON-SIMD-Befehlssatzerweiterung als zukunftssträchtigere Alternative gegenüber verfügbaren heterogenen Kombinationen aus ARM CPUs und zusätzlichem Signalverarbeitungs-DSP TMS320C64x angesehen.

Möglichkeiten und Grenzen des Einsatzszenarios der Softwaredemodulation

Im Rahmen der Arbeit konnte anhand des Digitalradio-Standards DAB gezeigt werden, dass moderne digitale Hörfunkstandards auch mit low-cost Systemen flexibel als reine Softwarelösung realisierbar sind. Auf general-purpose Prozessoren ist mit vergleichsweise moderaten Ressourcenanforderungen diese Demodulation umzusetzen und kann so als gängige Implementierungsalternative zu Demodulator-ASICs in Betracht kommen.

Die Untersuchung zeigte jedoch im Rahmen einer extrapolierenden Abschätzung ebenfalls, dass die Anforderungen hochratiger Übertragungsverfahren für TV-Übertragung, wenn überhaupt, nur mit heutigen high-end Rechensystemen als ein SDR-Konzept sinnvoll

¹Für den Anwendungsfall des DAB-Demodulators resultiert der Verlust der Empfängersynchronisation in einer zeitintensiven Neusynchronisation. Da das Neubefüllen des Interleaverspeichers nötig ist, entsteht eine hörbare Lücke im Audiodatenstrom von circa einer halben Sekunde.

erfüllbar sind und der ökonomische Einsatz in einem stark preisgetriebenen embedded-computing Marktsegment fraglich erscheint. Die durchgeführte Abschätzung lässt vermuten, dass dort die Umsetzung eines Softwaredemodulators für DVB-T in naher Zukunft nicht auf einfache Weise wirtschaftlich zu realisieren sein wird. Die für DVB-T als reine Softwarelösung nötige Rechenleistung entspricht zum heutigen Zeitpunkt in etwa dem gesamten Potential eines einzelnen Prozessorkerns eines vollwertigen, nicht-embedded PC-Systems. Es erscheint auf nahe Sicht schwer, einen Business-Case für ein stark preisgetriebenes Produkt zu finden, in dem diese Ausmaße des nötigen Leistungsvorhalts auf dem general-purpose Prozessor für die vollflexible Softwaredemodulation gegen die Kosten einer dedizierten ASIC-Demodulatorlösung bestehen. Die Untersuchung lässt ferner eindeutig den Schluss zu, dass auch durch den Einsatz der GPU hierzu keine Hilfe zu erwarten ist.

Folglich kann die softwarebasierte Demodulation für moderne digitale Hörfunksysteme auf dem general-purpose Applikationsprozessor als Designvariante durchaus empfohlen werden, für TV- und andere höchstratige Systeme erscheint jedoch die Nutzung spezieller ASIC-Lösungen mit integrierter Hardwarebeschleunigung heute als Variante der Wahl.

8.4 Ausblick

Im abschließenden Kapitel wurde der prototypische Aufbau erläutert, welcher die prinzipielle Machbarkeit eines Automotive Entertainment-Systems mit vollständig softwarebasierter und somit flexibel austauschbarer Radiosignalverarbeitung für Audioempfang anhand eines Fahrzeugdemonstrators mit low-cost general-purpose CPU-Plattform erfolgreich darstellt. Für eine Umsetzung als produkttaugliche Systemintegration sind die Module in entsprechende automotive-qualifizierte Softwarearchitekturen einzuarbeiten. Dies beinhaltet speziell diejenigen Schritte, welche nötig sind, um die durch die vorgestellten Algorithmen realisierte Kundenfunktion Audioplayback mit Echtzeitbezug auch gegen extern im System verursachte Systemlast kritischen Ausmaßes zu schützen und somit im Sinne harter Echtzeitanforderungen abzusichern.

Die in dieser Studie vorgestellten Performancemesswerte auf Basis des Digitalradioverfahrens DAB können als erste Grundlage zur Abschätzung der Rechenanforderungen verwandter Übertragungstechnologien aus dem Umfeld der Radiokommunikation dienen. Hier ist, wie in Kapitel 7 für das Digitalfernsehverfahren DVB-T durchgeführt, eine Skalierung des Ressourcenbedarfs der betroffenen Funktionsblöcke anhand der jeweiligen Übertragungsbitraten möglich.

Die an vielen Stellen in der wissenschaftlichen als auch der populärwissenschaftlichen Literatur vertretene These einer potentiell generell überlegenen Leistungsfähigkeit von GPU-Prozessoren gegenüber konventionellen CPUs konnte anhand der Ergebnisse im Themenfeld dieser Arbeit nicht gestützt werden. Die in der Fallstudie erarbeiteten Laufzeitergebnisse lassen im Gesamten sogar Zweifel daran aufkommen. Um die Entstehung beziehungsweise die Aussagen der präsentierten Werte einordnen zu können sowie um sie zu validieren, erfolgte im Vorfeld der Versuchsdurchführung eine intensive Analyse des Aufbaus der internen Prozessormikroarchitekturen. So erscheinen auch die später gemessenen Laufzeitwerte im Mikroarchitekturkontext plausibel. Hier wurde deutlich, dass sich GPU-Prozessoren ausschließlich an bekannten Mikroarchitekturprinzipien bedienen, entsprechend ist aus dieser Sicht ein gewichtiger genereller Leistungsunterschied der integrierten Schaltungen GPU

8 Zusammenfassung und Ausblick

und CPU auch logisch nicht zu erwarten.

Die Arbeit soll im Tenor die potentielle Leistungsfähigkeit existenter GPU-Chips nicht ableugnen, aber eine allgemein realistischere Sichtweise auf diese noch junge Prozessorklasse anregen. In diesem Sinne soll die Arbeit einen Beitrag für zukünftige Untersuchungen liefern, in der Diskussion um den Architekturvergleich CPU/GPU verstärkt auf eine faire Vergleichsmethodik zu achten: Nicht nur auf der GPU sind architekturenspezifische Optimierungen des Codes möglich und nötig. Dies gilt auch auf der klassischen CPU, speziell muss dort ein Code mit dem Anspruch einer Leistungsfähigkeitsbewertung des Prozessors zwingend an die SIMD-Vektorprozessorfähigkeiten der heutigen CPU-Prozessoren angepasst werden.

Abkürzungsverzeichnis

AAC	Advanced Audio Coding
ACS	Accumulate Compare Select
ADC	Analog Digital Converter
AGC	Automatic Gain Control
AGU	Address Generation Unit
ALU	Arithmetic-Logic Unit
AM	Amplitude Modulation
AMD	Advanced Micro Devices Inc.
ANSI	American National Standards Institute
API	Application Programming Interface
APU	Accelerated Processing Unit
ARM	Advanced RISC Machines Ltd.
ARQ	Automatic Repeat Request
ASIC	Application Specific Integrated Circuit
ASIP	Application Specific Instruction-Set Processor
AVX	Advanced Vector Extensions
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
CAN	Controller Area Network
CAZAC	Constant Amplitude Zero Autocorrelation
CFO	Carrier Frequency Offset
CIF	Common Interleaved Frame
CISC	Complex Instruction Set Computer
CP	Cyclic Prefix

Abkürzungsverzeichnis

CPU	Central Processing Unit
CU	Capacity Unit
CUDA	Common Unified Device Architecture
DAB	Digital Audio Broadcasting
DAC	Digital Analog Converter
DIF	Decimation In Frequency
DMA	Direct Memory Access
DMB	Digital Media Broadcasting
DQPSK	Differential Quadrature Phase-Shift Keying
DRAM	Dynamic Random-Access Memory
DSL	Digital Subscriber Line
DSP	Digital Signal Processor
DVB-T	Digital Video Broadcasting Terrestrial
ETSI	European Telecommunications Standards Institute
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FIC	Fast Information Channel
FID	Fast Information Data
FIG	Fast Information Group
FM	Frequency Modulation
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
FSB	Front Side Bus
GI	Guard Interval
GPGPU	General-Purpose Computing on GPU
GPP	General-Purpose Processor
GPU	Graphics Processing Unit
GSM	Global System for Mobile Communications

HAL	Hardware Abstraction Layer
HF	Hochfrequenz
HMI	Human Machine Interface
ICI	Inter-Carrier Interference
IGP	Integrated Graphics Processor
ILP	Instruction Level Parallelism
IP	Internet Protocol
IQ	Inphase/Quadrature
ISI	Inter-Symbol Interference
LDPC	Low-Density Parity Check
LUT	Look-up Table
LVDS	Low-Voltage Differential Signaling
MCI	Multiplex Configuration Information
MIMD	Multiple Instruction Multiple Data
MMU	Memory Management Unit
MMX	Multimedia Extensions
MPEG	Moving Picture Expert Group
MSC	Main Service Channel
NEON	<i>Markenname für:</i> ARM Advanced SIMD Extension
OFDM	Orthogonal Frequency Division Multiplex
OPS	Operations-Per-Second
PAR	Peak-to-Average Ratio
PCIe	Peripheral Component Interconnect Express
PRBS	Pseudo-Random Binary Sequence
PRS	Phase Reference Symbol
PSK	Phase-Shift Keying
PTX	Parallel Thread Execution
QAM	Quadrature Amplitude Modulation

Abkürzungsverzeichnis

QoS	Quality-of-Service
QPSK	Quadrature Phase-Shift Keying
RAM	Random-Access Memory
RF	Radio Frequency
RISC	Reduced Instruction Set Computer
RPC	Remote Procedure Call
RS	Reed Solomon
RTP	Real-Time Transport Protocol
SC	Synchronisation Channel
SDK	Software Development Kit
SDR	Software Defined Radio
SFDR	Spurious Free Dynamic Range
SFN	Single Frequency Network
SFU	Special Function Unit
SIMD	Single Instruction Multiple Data
SIMT	Single Instruction Multiple Threads
SM	Streaming Multiprocessor
SNR	Signal-to-Noise Ratio
SoC	System on Chip
SRAM	Static Random-Access Memory
SSE	Streaming SIMD Extensions
TDP	Thermal Design Power
TS	Transport Stream
UDP	User Datagram Protocol
UKW	Ultrakurzwelle
USB	Universal Serial Bus
USSE	Universal Scalable Shader Engine
VHF	Very High Frequency
VLIW	Very Long Instruction Word

Abbildungsverzeichnis

1.1	Visualisierung einer beispielhaften Auswahl der durch ein Multistandardkommunikationsgerät abzudeckenden Funkübertragungsverfahren.	3
2.1	Übersicht der einzelnen Komponenten beziehungsweise disjunkten Themengebiete, welche für das Produkt eines vollständigen multistandardfähigen Funksystems allesamt zu bearbeiten sind.	22
2.2	Grundschialtung der IQ-Mischverfahren.	30
2.3	Prinzip der Signalabtastung mit zeitdiskreter Repräsentation eines beispielhaften Signals.	31
2.4	Beispiele für die Signalkonstellationen in der komplexwertigen Basisbandebene $\underline{s}[k]$ verschiedener grundlegender QAM-/PSK-Modulationsverfahren bei kohärenter Abtastung im Symboltakt.	34
3.1	Übersicht der Plattformarchitekturen und deren Varianten, die zur Implementierung des eingebetteten Infotainment-Zielsystems konkret zur Auswahl stehen.	40
3.2	Aufbau der Intel Bonell-Mikroarchitektur am Beispiel einer Singlecore-CPU Intel Atom N270.	43
3.3	Mikroarchitektur des DSP-Kerns Texas Instruments C64x.	47
3.4	Mikroarchitektur der G9x-GPUs am Beispiel der NVidia MCP79 Ion.	50
3.5	Mikroarchitektur R800 Evergreen der AMD Radeon HD6310 GPU.	52
4.1	Multiplexhierarchie des DAB-Datenstroms mit beispielhaft drei Nutzdaten-services unterschiedlichen Formats.	71
4.2	Zeitliches Schema der Rahmenstruktur in DAB.	71
4.3	Blockschaltbild des DAB-Faltungscoders mit der Coderate $R_m=1/4$	72
4.4	Der Mechanismus des Energy Dispersals erfolgt durch Verknüpfen der Nutzdaten mit einer von einem linear rückgekoppelten Schieberegister erzeugten PRBS-Folge.	73
4.5	Gemessenes Empfangssignal $\hat{\underline{s}}[k]$ im detektierten Bereich des DAB-Nullsymbols. Die reduzierte Signalleistung ist im Zeitbereichssignal deutlich zu erkennen. Zur linken Seite sind Teile des letzten Symbols des vorangegangenen DAB-Rahmens, zur rechten Seite der Beginn des PRS Symbols des nächsten Rahmens erkennbar.	81
4.6	Schematische Darstellung der empfangenen OFDM-Symbole und Wahl der Länge des Nullsymbol-Fensters.	81
4.7	Autokorrelation $R_{s_0s_0}[k]$ mit näherungsweise Einheitspulsverlauf, bedingt durch die CAZAC-Eigenschaft der Folge des PRS-Symbols.	83

Abbildungsverzeichnis

4.8	Beispielhafter Verlauf der Kreuzkorrelation $R_{\hat{s}_0 s_0}[k]$ in Abhängigkeit der Schätzung des Frequenzoffsets ΔF_c . Die Berechnung über das Verfahren der schnellen Faltung, resultierend in einem zyklisch auf $k = 0 \dots L_S$ beschränkten Indexintervall.	84
4.9	Schematische Darstellung der Aufwandsreduktion der für das Tracking des Frequenzfehlers nötigen Autokorrelationsberechnung über a priori-Wissen bezüglich der Position des zyklischen Symbolpräfix.	88
4.10	Betragsspektrum nach der Fourier-Transformation im OFDM-Demodulator. Im Falle von Signalreflexionen ergeben sich stark variierende Pegel für die einzelnen Subträger des OFDM-Symbols.	93
4.11	Signalkonstellation in der komplexen IQ-Ebene des inkohärenten $\pi/4$ -DQPSK-Demodulators für ein Funkszenario ohne nennenswerte beziehungsweise mit starken Mehrwegereflektionen.	95
4.12	Schematische Darstellung des Datenflusses des Fehlerschutzdecoders, ohne Backtrace als zweitem Teil des Viterbi-Algorithmus. Zur Übersichtlichkeit sind nicht sämtliche 64 Zustände des Viterbi-Trellis dargestellt.	98
4.13	Schematische Übersicht beziehungsweise Visualisierung des Datenflusses im entwickelten Empfängerstack.	102
4.14	Vergleich der vorgestellten DAB-Signalverarbeitungskette mit Testequipment nach ETSI TR 101 496-3 [89] und der theoretischen Grenze für DQPSK-Modulation mit Faltungscode für eine Coderate $R_c = 1/2$	105
5.1	Systemarchitektur einer x86-Plattform mit integrierter GPU. Spezifisch für das vorliegende Anwendungsszenario ist ein Datenpfad von einem Antennensignal-Subsystem zur Audioausgabe hinzugefügt.	113
5.2	Überblick über die Hierarchie der für die Laufzeitprofilanalyse relevanten Codemodule von Betriebssystem und Gerätetreibern für den untersuchten Anwendungsfall CUDA-basierten GPGPU-Computings.	115
5.3	Insgesamt sind 11 Stufen zu 1024 strukturidentischen Radix 2-Berechnungselementen für die geforderte 2048 Punkt-FFT notwendig. Durch adäquate Gruppierung und Zusammenfassung bei der Berechnung kann die Datenlokalität benachbarter Radix 2-Elemente genutzt und Zugriffe auf den Hauptspeicher zum Speichern und Lesen von Zwischenergebnissen reduziert werden. Das Bild zeigt als Ausschnitt den Datenfluss zweier FFT-Stufen mit je zwei Radix 2-Elementen.	121
5.4	Durch Entflechten der Indexpositionen der Zwischenergebnisspeicherplätze lässt sich der Datenfluss der einzelnen FFT-Stufen vereinheitlichen.	122
5.5	Funktionsschema der Variante der DAB-Empfangskette mit Auslagerung der Signalverarbeitungskernel von der Host-CPU auf die GPU.	128
5.6	Vergleich der Prozessorauslastung der DAB-Softwaredemodulationsroutinen für die drei Implementierungsvarianten auf dem jeweils genutzten Compute-Device.	142
5.7	Vergleich der Prozessorauslastung der x86-Host-CPU während live/on-air Empfang ohne und mit zusätzlicher Nutzung des GPU-Prozessors.	143

5.8	Messung der okkupierten CPU-Ressourcen auf den beiden x86-Plattformen bei live-/on-air-Empfang mit verschiedenen Service Bitraten r_b	145
6.1	Last des ARM Cortex A8 CPU-Kerns bei Betrieb der DAB-Empfängerapplikation ohne Nutzung des NEON SIMD-Coprozessors in Abhängigkeit der DAB-Service-Bitrate r_b	149
6.2	Last des ARM Cortex A8 CPU-Kerns bei Betrieb der DAB-Empfängerapplikation unter Nutzung des NEON SIMD-Coprozessors in Abhängigkeit der DAB-Service-Bitrate r_b	151
6.3	Anpassung der Datenflussarchitektur der Signalverarbeitungskette an das volumenbegrenzte Cachesystem des DSPs.	154
6.4	Last des TI C64x DSP-Kerns bei Betrieb der DAB-Signalverarbeitung, in Abhängigkeit der DAB-Service-Bitrate r_b	156
6.5	Gegenüberstellung der gemessenen Last des ARM Cortex A8 CPU-Kerns bei Betrieb der DAB-Empfängerapplikation ohne und mit Nutzung der NEON SIMD-Instruktionen.	157
6.6	Last des ARM Cortex A8 CPU-Kerns bei Betrieb der DAB-Empfängerapplikation unter Nutzung des NEON SIMD-Coprozessors in Abhängigkeit der Prozessortaktfrequenz f_{Clk}	158
7.1	Konzept des Fahrzeugdemonstrators mit verteilter Audiosignalquelle (DAB-Radioempfangsmodul) und -senke (Head Unit).	177
7.2	Die Plattform zur Softwaredemodulation ist via USB-Bus mit dem Modul des HF-Frontends und via Ethernet UTP (Unshielded Twisted Pair) mit dem Head Unit-Emulator verbunden.	178
7.3	Der Head Unit-Emulator nutzt einen Embedded PC, welcher der Softwaredemodulationsplattform gleicht. Für das Benutzerinterface sind die Anzeige-, Audio- und Bedienelemente des Fahrzeugs angebunden.	179

Tabellenverzeichnis

3.1	Vergleich der Mikroarchitekturen bezüglich der theoretischen arithmetischen Leistungsfähigkeit je isoliertem Prozessorkern (darunter: Werte für den Gesamtchip der hier betrachteten Implementierungsvarianten, falls symmetrisches Multicore-Design).	66
4.1	Charakteristika des DAB-Systems im sogenannten Mode I.	70
4.2	Individuelle Dauer der Nutzperiode eines OFDM-Symbols sowie des Rahmens gemäß [2] in den vier DAB-Übertragungsmodi, angewendet auf die systemtypische Abtastfrequenz $f_s=2048$ kSa/s.	77
4.3	Konfiguration einiger realer, beispielhaft ausgewählter DAB-Services bei Angabe des zugehörigen Anteils der im OFDM-Signal tatsächlich betroffenen Anteile.	90
4.4	Auflistung der wesentlichen benötigten Speicherressourcen des DAB-Decoders.	104
5.1	Die im Dokument zur Evaluierung genutzte low-cost x86/GPU Plattform auf Intel Atom/NVidia Ion-Basis.	111
5.2	Last des Intel Atom CPU-Kerns bei Betrieb der DAB-Empfängerapplikation ohne plattformspezifische Optimierung.	117
5.3	Last des Intel Atom CPU-Kerns bei Betrieb der DAB-Empfängerapplikation mit Nutzung des SSE-Coprozessors.	125
5.4	Profiling der DAB-spezifischen Signalverarbeitungskernel für die Demodulation eines 96 ms DAB-Rahmens auf der x86/GPU-Plattform Intel Atom N270/NVidia MCP79 Ion.	131
5.5	Gemessene GPU-Rechenaktivität der Compute-Kernel im chronologischen Ablauf für die Demodulation eines 96 ms DAB-Rahmens auf der NVidia Ion-Plattform.	132
5.6	Vergleich der Lastanteile der DAB-Softwaredemodulation für die Variante unter Nutzung der GPU der NVidia Ion-Plattform.	133
5.7	Die zweite evaluierte Systemplattform auf Basis des kombinierten x86/GPU AMD Fusion-Konzeptes.	135
5.8	Gemessene Prozessorauslastung eines x86-Kerns des AMD E-350 SoC bei Ausführung des reinen C-Codes des Referenzmodells, das heißt ohne die Nutzung der SIMD-Fähigkeiten der SSE-Einheit als auch des speziell SSE-optimierten Codes.	137
5.9	Profiling der DAB-spezifischen Signalverarbeitungskernel für die Demodulation eines 96 ms DAB-Rahmens auf der x86/GPU-Plattform AMD E-350 Fusion.	139

Tabellenverzeichnis

5.10	Prozessorauslastung der GPU des AMD E-350 SoC sowie die verbleibende Last des AMD E-350 Bobcat-Kerns als die GPU überwachende Host-CPU während der Demodulation eines DAB-Signals.	140
6.1	Daten der zur Untersuchung genutzten kombinierten ARM-/DSP-Plattform.	148
6.2	Last des CPU-Kerns ARM Cortex A8 bei Betrieb der DAB-Empfängerapplikation ohne und mit Nutzung des NEON SIMD-Coprozessors.	152
6.3	Vergleich der Prozessorlast des DAB-Demodulatorkerns der Implementierungsvariante für den TI C64x-DSPs mit derjenigen der ARM NEON-Architektur bei gleicher Taktung des Prozessorkerns.	159
7.1	Leistungsaufnahme der Intel Atom/NVidia Ion-Testplattform bei Ausführung des DAB-Digitalradioszenarios.	165
7.2	Leistungsaufnahme der ARM Cortex A8-Testplattform bei Ausführung des DAB-Digitalradioszenarios.	165
7.3	Vergleich der in Kapitel 5 evaluierten low-cost Plattform zu einem nun betrachteten typischen Workstation-System.	167
7.4	Vergleich der benötigten Prozessorressourcen der SSE-optimierten DAB-Signalverarbeitung der low-cost Plattform zu einem Workstation-System.	168
7.5	Ressourcenbelegung des DAB-Empfangssystems aus [103], abgebildet für die Synthese auf verschiedenen FPGAs der Xilinx Spartan-6 Familie [104].	170
7.6	Vergleich der die Ressourcenanforderungen maßgeblich bestimmenden Parameter der Funkübertragungsszenarios von DAB und DVB-T.	174
7.7	Abschätzung des Ressourcenverbrauchs einer vollständig softwarebasierten Realisierung eines DVB-T-Decoders.	175
7.8	Systemplattform des Fahrzeugversuchsträgers.	177
7.9	Zusammenstellung des bei RTP/IP/Ethernet-basiertem Audiostreaming entstehenden absoluten Overheads pro Datenpaket.	180
7.10	Abbildung des durch RTP/IP/Ethernet-basierten Audiostreaming entstehenden relativen Overheads bei typischen DAB Audiodatenraten r_b	181
7.11	Zusammenstellung der benötigten Systemressourcen des vorgestellten Demonstrator-Infotainmentsystems.	182

Literaturverzeichnis

- [1] W. Tuttlebee, *Software-defined radio: facets of a developing technology*, IEEE Personal Communications Magazine, Band 6, Seiten 38–44, 1999.
- [2] European Telecommunications Standards Institute (ETSI), *Radio Broadcasting Systems: Digital Audio Broadcasting to mobile portable and fixed receivers*, European Standard (Telecommunications Series), Version 1.4.1, Januar 2006.
- [3] European Telecommunications Standards Institute (ETSI), *Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television*, European Standard, EN 300 744, Version 1.5.1, Juni 2004.
- [4] Texas Instruments Incorporated, *TI Introduces Industry's First Open Software and DSP-Based Solution for Eureka DAB Digital Radio Receivers*, online: <http://www.ti.com/sc/docs/news/2001/01002.htm>, Januar 2001.
- [5] Texas Instruments Incorporated, *TMS320DRE310 Eureka DAB Digital Radio Solution*, Product Bulletin SPRT273, Juni 2003.
- [6] Fraunhofer Institut für Integrierte Schaltungen, *Software for Efficient Implementation of DAB/DAB+/T-DMB Radio Receivers*, Information Sheet, 2011, online: http://www.iis.fraunhofer.de/content/dam/iis/de/dokumente/db/DAB_ReceiverKit.pdf.
- [7] C. Crawford, T. King-Smith, *Digital Radio - A Receiver Manufacturer's Viewpoint*, EBU Technical Review 2008/2, Juni 2008.
- [8] Frontier Silicon Ltd., *Digital radio solutions. Chips, modules and software for consumer and automotive devices*, online: <http://www.frontier-silicon.com/digital-radio-solutions>.
- [9] R. Schiffelers, *Digital Radio in Car Infotainment systems*, Digital Radio Conference 2012, EBU New Radio Group, Brüssel, Oktober 2012.
- [10] Realtek Semiconductor Corporation, *emphRTL2832U: DVB-T COFDM Demodulator and USB 2.0*, online: <http://www.realtek.com.tw>.
- [11] A. Müller, *DAB Receiver Implementation*, Eidgenössische Technische Hochschule Zürich, Semester Thesis Project, Spring Term 2008, Juni 2008.
- [12] *Performance on ARM Cortex-A8*, online: <http://lists.gnu.org/archive/html/discuss-gnuradio/2011-07/msg00201.html>.
- [13] J. van Katwijk, *SDR-J*, online: <http://www.sdr-j.tk>.

- [14] M. Frigo, S.G. Johnson, *The Design and Implementation of FFTW3*, Proceedings of the IEEE Special Issue on Program Generation, Optimization, and Platform Adaptation, Band 93, Nummer 2, Seiten 216–231, Januar 2005.
- [15] F. de Mesmay, S. Chellappa, F. Franchetti, M. Püschel, *Computer Generation of Efficient Software Viterbi Decoders*, Proceedings of the 5th International Conference on High Performance Embedded Architectures and Compilers Lecture Notes in Computer Science (HiPEAC 2010), Seiten 353–368, Pisa, Italy, Januar 2010.
- [16] S.-M. Tseng, Y.-T. Hsu, M.-C. Chang, H.-L. Chan, *A Notebook PC Based Real-Time Software Radio DAB Receiver*, IEICE Transactions on Communications, Band E89-B, Nummer 12, Seiten 2270–2271, Oktober 2006.
- [17] Y. Xiong, Y. Huang, M. Evans, T. Cronk, *A SDR for Realtime DAB COFDM Demodulation with ARM Processor*, Proceeding of IEE Colloquium on DSP enabled Radio, Seiten 1–7, September 2003.
- [18] K. Borre et al., *A Software-Defined GPS and Galileo Receiver: A Single-Frequency Approach*, Birkhäuser Boston, August 2007.
- [19] Fraunhofer Institut für Integrierte Schaltungen, *Fraunhofer Software Radio – Professional Monitoring Receiver for DRM Signals*, Information Sheet, 2009, online: http://www.iis.fraunhofer.de/content/dam/iis/en/dokumente/Digital-Broadcasting/100825_FraunhoferSoftwareRadio2010low.pdf.
- [20] V. Fischer et al., *Dream DRM Receiver – A software radio for AM and Digital Radio Mondiale (DRM)*, SourceForge Source Code Repository, on-line: <http://sourceforge.net/projects/drm/>.
- [21] V. Pellegrini, M. Di Dio, L. Rose. M. Luise, *A Real-Time, Fully-Software Receiver for DVB-T Signals based on the USRP*, 6th Karlsruhe Workshop on Software Radios (WSR 2010), Karlsruhe, Germany, März 2010.
- [22] Mirics Limited, *Mirics FlexiTV*, Product Information Leaflet, online: <http://www.mirics.com/node/6>.
- [23] J. Kim, S. Hyeon, S. Choi, *Implementation of an SDR System Using Graphics Processing Unit*, IEEE Communications Magazine, Band 48, Nummer 3, Seiten 156–162, März 2010.
- [24] H. Yang, S. Choi, *Implementation of a WiBro System Using GPU*, Proceedings of IEEE International Conference on Network Infrastructure and Digital Content, Band 2, Seiten 713–716, Sept. 2010.
- [25] NVidia Corporation, *GeForce 9800 GTX Specifications*, <http://www.geforce.com/hardware/desktop-gpus/geforce-9800-gtx/specifications>, 2013.
- [26] Texas Instruments Incorporated, *TMS320C6414/5/6 Power Consumption Summary*, Application Report SPRA811C, Juli 2003.

- [27] V. Volkov, B. Kazian, *Fitting FFT onto the G80 Architecture*, University of California, Berkeley, Mai 2008.
- [28] D. Zhang et al., *An Implementation of Viterbi Algorithm on GPU*, Proceedings of International Conference on Information Science and Engineering, Seiten 121–124, April 2010.
- [29] C.S. Lin et al., *A Tiling-Scheme Viterbi Decoder in Software Defined Radio for GPUs*, Proceedings of International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM), Band 7, Seiten 1–4, September 2011.
- [30] C.-C. Chang, Y.-L. Chang, M.-Y. Huang, B. Huang, *Accelerating Regular LDPC Code Decoders on GPUs*, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, Band 4, Nummer 3, September 2011.
- [31] H. Ji, J. Cho, W. Sung, *Massively Parallel Implementation of Cyclic LDPC Codes on a General Purpose Graphics Processing Unit*, IEEE Workshop on Signal Processing Systems (SiPS), Seiten 285–290, Oktober 2009.
- [32] R. Vuduc, A. Chandramowlishwaran, J. Choi, M. Guney, A. Shringarpure, *On the Limits of GPU Acceleration*, 2nd USENIX Conference on Hot Topics in Parallelism, Berkeley, Juni, 2010.
- [33] International Organization for Standardization, *emphISO/IEC 9899:1999 Information Technology — Programming Languages — C*, Geneva, Switzerland, Dezember 1999.
- [34] Nvidia Corporation, *Nvidia CUDA Parallel Programming Toolkit 4.1*, Jan. 2012, online: <http://www.nvidia.com/cuda>.
- [35] Advanced Micro Devices Incorp., *AMD Accelerated Parallel Processing (APP) SDK*, Version 2.5, August 2011.
- [36] Advanced Micro Devices Incorp., *AMD Accelerated Parallel Processing Math Libraries (APPML)*, Version 1.6, Januar 2012.
- [37] Khronos OpenCL Working Group, *OpenCL. The open standard for parallel programming of heterogeneous systems*, November 2011, online: <http://www.khronos.org/opencl>.
- [38] R. Gerber, A.J.C. Bik, K.B. Smith, X. Tian, *The Software Optimization Cookbook: High-Performance Recipes for IA-32 Platforms*, Intel Press, Dezember 2005.
- [39] Intel Corporation, *Intel 64 and IA-32 Architectures Optimization Reference Manual*, online: <http://www.intel.com/content/dam/doc/manual/64-ia-32-architectures-optimization-manual.pdf>, Juni 2011.
- [40] A. Fog, *Optimizing Software in C++*, online: http://www.agner.org/optimize/optimizing_cpp.pdf.
- [41] A. Fog, *Optimizing Subroutines in Assembly Language – An Optimization Guide for x86 Platforms*, online: http://agner.org/optimize/optimizing_assembly.pdf.

- [42] R. Dasgupta, *Embedded DSP Software Optimization: Strategies and Techniques*, IEEE Conference on Emerging Applications of Information Technology (EAIT), Seiten 250–255, November 2012.
- [43] International Telecommunication Union, *Reference Model of Open Systems Interconnection for CCITT Applications*, ITU-T Recommendation X.200, November 1988.
- [44] J. Proakis, D.G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 4. Auflage, Prentice Hall, April 2006.
- [45] J.H. Reed, *Software Radio: A Modern Approach to Radio Engineering*, Prentice Hall Press, New York, USA, Mai 2002.
- [46] Software Freedom Law Center Inc., *FCC Rules on FOSS and Software-Defined Radio*, White Paper, New York, USA, July 2007.
- [47] L. Sha et al., *Real Time Scheduling Theory: A Historical Perspective*, Kluwer Academic Publishers, Real-Time Systems, Band 28, Nummer 2, Seiten 101–155, November 2004.
- [48] J. Proakis, M. Salehi, *Digital Communications*, 4. Auflage, McGraw-Hill, August 2000.
- [49] NVidia Corporation, C. Woolley, *GPU Optimization Fundamentals*, Accelerating Computational Science Symposium 2012, Oak Ridge Leadership Computing Facility, Washington, USA, März 2012.
- [50] G. Gerosa et al., *A sub 2W low power IA Processor for Mobile Internet Devices in 45nm Hi-K metal gate CMOS*, Proceedings of IEEE Solid-State Circuits Conference 2008, Seiten 17–20, San Francisco, USA, Februar 2008.
- [51] A. Fog, *The microarchitecture of Intel, AMD and VIA CPUs: An optimization guide for assembly programmers and compiler makers*, Copenhagen University College of Engineering, June 2011, online: <http://www.agner.org/optimize/microarchitecture.pdf>.
- [52] Intel Corporation, *Mobile Intel Atom Processor N270 Single Core: Datasheet*, 2008.
- [53] B. Burgess et al., *Bobcat: AMD's Low-Power x86 Processor*, IEEE Computer Society, IEEE Micro, Band 31, Heft 2, Seiten 16–25, März 2011.
- [54] D. Foley, *A Low-Power Integrated x86-64 and Graphics Processor for Mobile Computing Devices*, IEEE Journal of Solid-State Circuits, Band 47, Heft 1, Seiten 220–231, Januar 2012.
- [55] Advanced Micro Devices Incorp., *Family 16h Models 00h-0Fh AMD E-Series Mobile Processor Product Data Sheet*, Revision 3.00, 2013.
- [56] ARM Limited, *ARM Architecture Reference Manual ARMv7-A and ARMv7-R Edition*, ARM DDI0406B, 2008.
- [57] ARM Limited, *Cortex-A8 Technical Reference Manual*, ARM DDI0344J, 2009.
- [58] ARM Limited, *Cortex-A9 Technical Reference Manual*, ARM DDI0388F, 2010.

- [59] Texas Instruments Incorp., *Low Power Consumption and a Competitive Price Tag Make The Six-Core TMS320C6472 Ideal for High-Performance Applications*, Technical White Paper, SPRY130, Oktober 2009.
- [60] S. Agarwala et al., *A 600-MHz VLIW DSP*, IEEE Journal of Solid-State Circuits, Band 37, Heft 11, Seiten 1532–1544, November 2002.
- [61] S. Agarwala et al., *A 800-MHz System-on-Chip for Wireless Infrastructure Applications*, Proceedings of 17th International Conference on VLSI Design, Seiten 381–389, Januar 2004.
- [62] Texas Instruments Incorp., *DM3730, DM3725 Digital Media Processors*, Product Information, SPRS685, August 2010.
- [63] Texas Instruments Incorp., *Cache Usage in High-Performance DSP Applications With the TMS320C64x*, Application Report, SPRA756, Dezember 2001.
- [64] H. Wong, M.-M. Papadopoulou, M. Sadooghi-Alvandi, A. Moshovos, *Demystifying GPU microarchitecture through microbenchmarking*, IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Seiten 235–246, März 2010.
- [65] Nvidia Corporation, *NVidia Compute PTX: Parallel Thread Execution ISA*, Version 2.3, März 2011.
- [66] NVidia Corporation, *NVidia CUDA Programming Guide*, Version 4.0, Mai 2011.
- [67] Nvidia Corporation, *Nvidia CUDA C Best Practices Guide*, Version 4.1, Januar 2012.
- [68] Advanced Micro Devices Incorporated, *Evergreen Family Instruction Set Architecture: Instructions and Microcode*, Reference Guide, November 2011.
- [69] Imagination Technologies, *PowerVR SGX Series5XT IP Core Family*, Product Bulletin, 2011.
- [70] Imagination Technologies Ltd., *Imagination Technologies Reveals Extended PowerVR SGX Graphics & Video Core Family – Shader-based Programmable GPU IP for Consumer, Automotive and PC Devices*, Press Release, Januar 2007.
- [71] Texas Instruments Incorporated, *AM335x ARM Cortex-A8 Microprocessors (MPUs)*, Technical Reference Manual, SPRUH73, Oktober 2011.
- [72] P. Clarke, *Imagination offers Super-Threading Meta Processor IP Core*, Product News, EE-Times, online: http://www.eetimes.com/document.asp?doc_id=1295285, Februar 2005.
- [73] P.J. Hatcher et al., *Data-parallel programming on MIMD computers*, IEEE Transactions on Parallel and Distributed Systems, Band 2, Heft 3, Seiten 377–383, Juli 1991.
- [74] K.L. Kloker, Motorola Inc., *The Motorola DSP56000 Digital Signal Processor*, IEEE Micro, Band 6, Heft 6, Seiten 29–48, Dezember 1986.

- [75] Thinking Machines Corporation, *C* Programming Guide*, 1990.
- [76] Cray Research Incorporated, *An Introduction to the Cray-1 Computer*, 1975.
- [77] W. Abu-Sufah, A.D. Malony, *Vector Processing on the Alliant FX/8 Multiprocessor*, International Conference on Parallel Processing, Seiten 559–566, August 1986.
- [78] B. Duzett, R. Buck, *An Overview of the nCUBE 3 Supercomputer*, Proceedings of the Fourth Symposium on the Frontiers of Massively Parallel Computation, Oktober 1992.
- [79] C. Schmidt-Knorreck, M. Ihmig, R. Knopp, A. Herkersdorf, *Multi-Standard Processing using DAB and 802.11p on Software Defined Radio Platforms*, 7th Karlsruhe Workshop on Software Radios (WSR 2012), Karlsruhe, Germany, March 7-8, 2012.
- [80] European Telecommunications Standards Institute (ETSI), *Digital Audio Broadcasting (DAB); Transport of Advanced Audio Coding (AAC) Audio*, Technical Specification, TS 102 563, Version 1.2.1, Mai 2010.
- [81] European Telecommunications Standards Institute (ETSI), *Digital Audio Broadcasting (DAB); Data Broadcasting – MPEG-2 TS streaming*, Technical Specification, TS 102 427, Version 1.1.1, Juli 2005.
- [82] European Telecommunications Standards Institute (ETSI), *Digital Audio Broadcasting (DAB); DMB video service; User application specification*, Technical Specification, TS 102 428, Version 1.2.1, April 2009.
- [83] A. J. Viterbi, *Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm*, IEEE Transactions on Information Theory, Band 3, Heft 2, April 1967.
- [84] U. H. Rohrs, L. P. Linde, *Cazac Code Tracking Loop*, Proceedings of IEEE 1990 South African Symposium on Communications and Signal Processing (COMSIG), Seiten 142–146, Juni 1990.
- [85] C.-R. Sheu, Y.-L. Huang, and C.-C. Huang, *Joint Symbol Frame, and Carrier Synchronization for Eureka 147 DAB System*, Proceedings of the International Conference on Universal Personal Communications (ICUPC'97), Seiten 693-697, September 1997.
- [86] T. Hewavithana and M. Brookes, *Soft Decisions for DQPSK Demodulation for the Viterbi Decoding of the Convolutional Codes*, Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Band 4, Seiten 17–20, April 2003.
- [87] G. Forney, *Burst-Correcting Codes for the Classic Bursty Channel*, IEEE Transactions on Communication Technology, Band 19, Heft 5, Teil 1, Seiten 772–781, Oktober 1971.
- [88] S. Graves, *Memory Allocation Strategy in Safety-Critical Mil Systems*, COTS Journal of Military Electronics and Computing, Februar 2010.

- [89] European Telecommunications Standards Institute (ETSI), *Digital Audio Broadcasting (DAB); Guidelines and rules for implementation and operation; Part 3: Broadcast network*, Technical Report, TR 101 496-3, Version 1.1.2, Mai 2001.
- [90] I.N. Bronstein, K.A. Semendjajew, *Taschenbuch der Mathematik*, 25. Auflage, BG Teubner Verlagsgesellschaft, Stuttgart Leipzig, und Verlag Nauka, Moskau, 1991.
- [91] Microsoft Corporation, *MS Windows NT Workstation 4.0 Resource Guide*, MS Windows NT Workstation 4.0 Resource Kit, 1995.
- [92] D. Stewart, M. Barr, *Introduction to Rate Monotonic Scheduling*, Beginner's Corner, Embedded Systems Programming, Februar 2002. online: www.embedded.com/showArticle.jhtml?articleID=9900522.
- [93] E. Chu, A. George, *Inside the FFT Black Box: Serial and Parallel Fast Fourier Transform Algorithms*, CRC Press, Juni 2000.
- [94] C.S. Burrus, T.W. Parks, *DFT/FFT and Convolution Algorithms: Theory and Implementation (Topics in Digital Signal Processing)*, John Wiley & Sons, Januar 1985.
- [95] M.C. Pease, *An Adaption of the Fast Fourier Transform for Parallel Processing*, Journal of the Association for Computing Machinery (ACM), Band 15, Heft 2, Seiten 252–264, April 1968.
- [96] Nvidia Corporation, *CUDA Toolkit 4.1 CUFFT Library*, Programming Guide, Oktober 2011.
- [97] O. Haag, *Optimierung und Weiterentwicklung eines Software Defined DAB Radio Receivers auf einer DSP-Plattform*, Diplomarbeit, Fachhochschule Kempten, Fachbereich Elektrotechnik, Labor für Nachrichtentechnik, Mai 2011.
- [98] Texas Instruments Corporation, *TMS320 DSP Algorithm Standard – Rules and Guidelines*, User's Guide SPRU352, Juni 2005.
- [99] Texas Instruments Corporation, *DSP/BIOS Timers and Benchmarking Tips*, Application Report SPRA829, Juli 2002.
- [100] Texas Instruments Corporation, *Viterbi Decoding Techniques for the TMS320C55x DSP Generation*, Application Report SPRA776, April 2009.
- [101] Texas Instruments Corporation, *TMS320C55x DSP CPU Reference Guide*, Technical Datasheet SPRU371, Februar 2004.
- [102] Texas Instruments Corporation, *TMS320C64x DSP Viterbi-Decoder Coprocessor (VCP)*, Reference Guide SPRU533, September 2004.
- [103] M. Feilen, A. Iliopoulos, M. Ihmig, W. Stechele, *Partitioning and Context Switching for a Reconfigurable FPGA-based DAB Receiver*, Conference on Design and Architectures for Signal and Image Processing (DASIP), Karlsruhe, Germany, Oktober 2012.

- [104] Xilinx Incorporated, *Spartan-6 Family Overview*, Product Specification, Oktober 2011.
- [105] H.-T. Lim, B. Krebs, L. Völker, and P. Zahrer, *Performance Evaluation of the Inter-Domain Communication in a Switched Ethernet Based In-Car Network*, 36th Annual IEEE Conference on Local Computer Networks (LCN), Seiten 101–108, Bonn, Oktober 2011.
- [106] OPEN Alliance, *OPEN Alliance SIG*, online: <http://www.opensig.org/about.php>.
- [107] L.L. Bello, *The Case for Ethernet in Automotive Communications*, Special Issue on the 10th International Workshop on Real-Time Networks, Band 8, Nummer 4, Dezember 2011.
- [108] Apache Etch community, *Apache Etch*, online: <http://incubator.apache.org/etch>.
- [109] K. Weckemann, H.-T Lim, and D. Herrscher, *Practical experiences on a communication middleware for IP-based in-car networks*, Proceedings of the 5th International Conference on Communication System Software and Middleware (COMSWARE), Verona, Italy, Juli 2011.
- [110] F. van der Laar, N. Philips, and J. Huisken, *Towards the next generation of DAB receivers*, EBU Technical Review Summer 1997, Seiten 46–59, 1997.
- [111] K. Taura, M. Tsujishita, M. Takeda, H. Kato, M. Ishida, and Y. Ishida, *A digital audio broadcasting (DAB) receiver*, IEEE Transactions on Consumer Electronics, Band 42, Nummer 3, Seiten 322–327, August 2002.
- [112] L. Stolz, M. Feilen, and W. Stechele, *An Optimized Software Defined DAB Receiver for x86 Platforms*, Proceedings of the Workshop on Software Radio, Band 8, Karlsruhe, März 2012.
- [113] H. Schulzrinne et al., *RTP: A Transport Protocol for Real-Time Applications*, Request For Comments 3550, Internet Engineering Task Force, July 2003.
- [114] J. Postel et al., *User Datagram Protocol*, Request For Comments 768, Internet Engineering Task Force, August 1980.
- [115] J. Postel et al., *Internet Protocol – DARPA Internet Programm, Protocol Specification*, Request For Comments 791, Internet Engineering Task Force, September 1981.
- [116] V. Issarny, M. Caporuscio, and N. Georgantas, *A Perspective on the Future of Middleware-based Software Engineering*, Proceedings of the Conference on The Future of Software Engineering, Mai 2007.
- [117] K. Weckemann, F. Satzger, L. Stolz, D. Herrscher, and C. Linnhoff-Popien, *Lessons from a Minimal Middleware for IP-Based In-Car Communication*, Proceedings of the IEEE Intelligent Vehicles Symposium, Juni 2012.