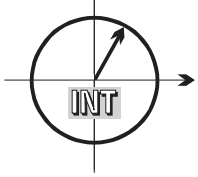
Forschungsberichte aus dem
Institut für Nachrichtentechnik
der Universität Karlsruhe (TH)



Arnd-Ragnar Rhiemeier

Modulares Software Defined Radio

Band 9

Druck: Druckerei Ernst Grässer, Humboldtstr. 1
76131 Karlsruhe, Tel. 0721/615050

ISSN: 1433-3821

Forschungsberichte aus dem Institut für Nachrichtentechnik der Universität Karlsruhe (TH)

Herausgeber: Prof. Dr. rer. nat. Friedrich Jondral

Band 1 Marcel Kohl
Simulationsmodelle für die Bewertung von Satellitenübertragungsstrecken im 20/30 GHz Bereich

Band 2 Christoph Delfs
Zeit-Frequenz-Signalanalyse: Lineare und quadratische Verfahren sowie vergleichende Untersuchungen zur Klassifikation von Klaviertönen

Band 3 Gunnar Wetzker
Maximum-Likelihood Akquisition von Direct Sequence Spread-Spectrum Signalen

Band 4 Anne Wiesler
Parametergesteuertes Software Radio für Mobilfunksysteme

Band 5 Karl Lütjen
Systeme und Verfahren für strukturelle Musteranalysen mit Produktionsnetzen

Band 6 Ralf Machauer
Multicode-Detektion im UMTS

Band 7 Gunther M. A. Sessler
Schnell konvergierender Polynomial Expansion Multiuser Detektor mit niedriger Komplexität

Band 8 Henrik Schober
Breitbandige OFDM Funkübertragung bei hohen Teilnehmergeschwindigkeiten

Forschungsberichte aus dem Institut für Nachrichtentechnik der Universität Karlsruhe (TH)

Herausgeber: Prof. Dr. rer. nat. Friedrich Jondral

Band 9 Arnd-Ragnar Rhiemeier
Modulares Software Defined Radio

Vorwort des Herausgebers

Seit Joseph Mitola den Begriff Software Radio (SR) in die Diskussion gebracht hat, sind fast 15 Jahre vergangen. Es wurde schnell deutlich, dass die Realisierung idealer SRs (breitbandige Abtastung des Antennenausgangs) problematisch ist und für die absehbare Zukunft auch bleibt. Als praktische Alternative bietet sich das Software Defined Radio (SDR) an, das die Digitalisierung des Empfangssignals erst nach einer Zwischenfrequenzfilterung oder direkt im komplexen Basisband vornimmt.

Zur Konstruktion von SDRs wurden bisher verschiedene Wege eingeschlagen. Einer dieser Wege ist das Parameter Controlled (PaC-) SDR, in dem die gesamte Verarbeitung des Transceivers über einen Parametersatz, der im Wesentlichen einen Standard darstellt, gesteuert wird[1]. Voraussetzung für diesen Ansatz ist die Kenntnis der zu unterstützenden Standards. Der Austausch von Signalverarbeitungseinheiten (z.B. von En- und Decodern) ist daher in einem PaC-SDR schwierig.

Um die Diskussion über SDRs auf einer höheren Ebene führen zu können, muss insbesondere von den Signalverarbeitungsalgorithmen der physikalischen Schicht abstrahiert werden. Dazu wird die gesamte Signalverarbeitung eines Senders bzw. eines Empfängers als gerichteter Graph dargestellt, in dessen Knoten Software Module ablaufen. Diese Module sind in einem SDR nicht fest vorgegeben sondern in einem gewissen Rahmen austauschbar. Aus dieser Vorstellung stammt der Begriff Modulares Software Defined Radio (Mod-SDR).

Charakteristische Größen für die Module sind ihre Laufzeit und ihr Speicherbedarf. Im Weiteren wird nun davon ausgegangen, dass die Realisierung von Mod-SDRs eine Mehrprozessorhardware voraussetzt. So ergibt sich natürlich die Frage nach der Bewertung verschiedener modularer Realisierungen desselben Übertragungsverfahrens auf einer Mehrprozessorplattform. Die vorliegende Arbeit misst die Qualität der beschriebenen Entwicklungen nicht wie üblich an der Darstellung der mittleren Bitfehlerrate über dem Signal-Rausch-Verhältnis sondern schlägt andere Maße wie den (relativen) Speedup bei Mehrprozessorsystemen, die durch die Signalverarbeitung verursachte Laufzeitverzögerung oder den Speicherbedarf als Gütemaße vor.

[1] Anne Wiesler: Parametergesteuertes Software Radio für Mobilfunksysteme. Dissertation, Forschungsberichte aus dem Institut für Nachrichtentechnik der Universität Karlsruhe (TH), Band 4, 2001, ISSN 1433-3821

Mit seiner Dissertation Modulares Software Defined Radio betritt Herr Rhiemeier Neuland bei der Beschreibung und Entwicklung von Funkgeräten. Dabei bestätigt er einen Trend, der unterschwellig seit der Verabschiedung der Software Communications Architecture (SCA) durch das SDR Forum im Jahr 1999 zu spüren ist: Der Entwickler für die digitale Signalverarbeitung in Funkgeräten muss in Zukunft neben der Algorithmik sender- und empfängerspezifischer Aufgaben (Kanalschätzung, Entzerrung, Modulation/Demodulation, Encodierung/Decodierung etc.) die objektorientierte Programmierung im Auge haben, um höchste Flexibilität in seine Geräte implementieren zu können.

Karlsruhe im Dezember 2004
Friedrich Jondral

Modulares Software Defined Radio

Zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS

von der Fakultät für
Elektrotechnik und Informationstechnik
der Universität Fridericiana Karlsruhe

genehmigte

DISSERTATION

von

Dipl.-Ing. Arnd-Ragnar Rhiemeier

aus

Bochum

Tag der mündlichen Prüfung:	04.11.2004
Hauptreferent:	Prof. Dr. rer. nat. Friedrich Jondral
Korreferent:	Prof. Dr. rer. nat. Hartmut Schmeck

Danksagung

Die vorliegende Dissertation entstand im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Nachrichtentechnik der Universität Karlsruhe (TH). Mein besonderer Dank gilt Herrn Prof. Dr. rer. nat. Friedrich Jondral, dem Leiter dieses Instituts, für sein Vertrauen auf meine Fähigkeiten in Forschung und Lehre ebenso wie für die sehr gute Zusammenarbeit und seine Unterstützung bei der Durchführung meiner Arbeit. Dazu gehört jede einzelne Diskussion, die wir im Laufe der Zeit geführt haben. Für seine Bereitschaft, das Hauptreferat zu übernehmen, danke ich ebenfalls.

Herrn Prof. Dr. rer. nat. Hartmut Schmeck, Leiter des Instituts für Angewandte Informatik und Formale Beschreibungsverfahren, Fakultät für Wirtschaftswissenschaften der Universität Karlsruhe (TH), danke ich für sein Interesse an meiner Arbeit und für die Übernahme des Korreferats.

Allen Kollegen, mit denen ich im Laufe meiner Tätigkeit am Institut für Nachrichtentechnik zusammengearbeitet habe, möchte ich meinen Dank aussprechen. Besonders angenehm war die Zusammenarbeit mit meinem Kollegen Timo Weiß, dessen unkomplizierte und zuverlässige Art ich sehr schätze. Des Weiteren behalte ich Herrn Dr.-Ing. Ralf Machauer stets in guter Erinnerung. Er hat in wichtigen Momenten das Wenige und Richtige gesagt, das ihn als einen hervorragenden Kollegen auszeichnet.

Mein Dank gilt auch Herrn Dipl.-Ing. (FH) Reiner Linnenkohl für sein tägliches Engagement beim Betrieb aller Rechner- und Netzsysteme. Weiterhin möchte ich unsere Sekretärinnen, besonders Frau Gabriele Kuntermann, und unsere Grafikerin, Frau Angelika Olbrich, in meinen Dank einbeziehen. Sie alle tragen auf ihre Weise zu der sehr guten Arbeitsatmosphäre am Institut für Nachrichtentechnik bei.

Schließlich danke ich ganz besonders meinen Eltern, deren Vertrauen und Rat mir stets eine wertvolle Unterstützung waren, und meiner zukünftigen Frau Verena für ihr Verständnis.

Zusammenfassung

Seit dem Bekanntwerden des Begriffs Software Radio vor rund 10 Jahren sind zahlreiche Beiträge auf diesem Gebiet veröffentlicht worden, sowohl zum Konzept des idealen Software Radio als auch zu realisierbaren Kompromisslösungen, die unter dem Begriff Software Defined Radio (SDR) zusammengefasst werden. Bei nahezu allen Arbeiten steht lediglich ein Teilaspekt der gesamten Signalverarbeitung im Vordergrund, und eine bestimmte Entwurfsaufgabe wird isoliert von anderen Vorgängen in einem SDR-Endgerät betrachtet. Insbesondere die Beiträge zur Signalverarbeitung im komplexen Basisband eines SDR konzentrieren sich stark auf die Algorithmen der Nachrichtentechnik. Das Zusammenspiel vieler verschiedener Algorithmen in einem größeren Verbund logischer Abhängigkeiten und die Abbildung derartiger Strukturen auf ein flexibles Hardware-System sind in der SDR-Literatur bisher unbeachtet geblieben.

Unter dem Begriff des Modularen Software Defined Radio (Mod-SDR) werden in dieser Arbeit alle Arten von SDRs verstanden, bei denen sowohl die Hardware als auch die Software modular aufgebaut ist. Eine wesentliche Grundannahme besteht darin, dass das SDR-Konzept am ehesten auf einer einfach erweiterbaren Multiprozessor-Hardware in die Realität umzusetzen sein wird. Unter Software werden alle Formen von ausführbarem Maschinencode verstanden, der Aufgaben der digitalen Signalverarbeitung auf der physikalischen Schicht eines mobilen Funkgerätes bewältigt. Allgemein gültige Richtlinien für den Entwurf und Betrieb solcher Mod-SDRs herzuleiten ist das Ziel der vorliegenden Arbeit.

Das SDR-Konzept verspricht eine große Flexibilität bei der drahtlosen Anbindung mobiler Endgeräte an das Festnetz. Die vielfältigen neuen Anwendungsmöglichkeiten, die mit dieser Flexibilität einhergehen, werden in der Literatur nur all zu oft als die großen Vorteile von SDR angeführt. Die Nachteile von SDR sind zwar durchaus bekannt, aber es wird üblicherweise angenommen, dass sich ihr Gewicht im Zuge des Fortschritts in der Mikro- und Nanotechnologie stetig verringern wird. Insgesamt existiert eine Vielzahl unterschiedlichster Sichtweisen zum Thema SDR, und daher gilt es, zunächst alle wesentlichen technischen und wirtschaftlichen Aspekte für den Einsatz flexibler, modularer SDR-Systeme zu verstehen, um dann entsprechende Schlussfolgerungen für die Signalverarbeitung auf der physikalischen Schicht zu ziehen: Mod-SDR wird im Rahmen dieser Arbeit als der Prototyp eines eingebetteten Echtzeitsystems in der Nachrichtentechnik interpretiert.

Diese Sichtweise erfordert zuerst eine neuartige und abstrakte Modellbildung, vor allem für die Signalverarbeitungssoftware. Das in dieser Arbeit eingeführte Modell bedient sich gerichteter azyklischer Graphen und weist den entscheidenden Vorteil auf, sowohl vom Stand der Technik als auch von konkreten Realisierungen heutiger Mobilfunkstandards unabhängig zu sein. In einem zweiten Schritt wird die Entwurfsaufgabe auf der physikalischen Schicht eines Mod-SDR mathematisch präzise erfasst und dabei formal in das Gebiet der linearen ganzzahligen Optimierung eingeordnet. Es stellt sich schnell heraus, dass das Auffinden der optimalen Lösung NP-schwer und damit für den praktischen Einsatz in einem Mod-SDR nicht geeignet ist.

Um in der Praxis schnell eine hinreichend gute Lösung für die Mod-SDR-Entwurfsaufgabe hervorzubringen, bedarf es verschiedener Methoden der Partitionierung und der Ablaufplanung. In dieser Arbeit werden dazu folgende Ansätze verwendet: Erstens der Algorithmus von Hu, der unter speziellen Randbedingungen die parallelen Strukturen eines Graphen optimal ausnutzt. Zweitens ein heuristisches Verfahren nach Kernighan und Lin, das eine lokale Suchstrategie im Raum der Lösungen verfolgt. Drittens die spektrale Partitionierung, die einen Graphen als Ganzes bewertet und eine Anwendung der algebraischen Graphentheorie darstellt.

Eine radikal andere Betriebsart wird von den Verfahren Graph Duplication Pipelining (GDP) und Half-Frame Pipelining (HFP) angestrebt. Die entsprechende Vorgehensweise in der Partitionierung und Ablaufplanung ist überaus einfach und dennoch sehr erfolgreich. Aus den Untersuchungen zur Leistungsfähigkeit der verschiedenen Ansätze geht das Pipelining-Prinzip als die zentrale Richtlinie für den Entwurf und Betrieb von Mod-SDR-Systemen hervor.

Inhaltsverzeichnis

1 Einleitung

Der Begriff Software Radio ist 1995 zum ersten Mal in das Bewusstsein einer breiten Öffentlichkeit gerückt [61]. Der Begriff wurde wesentlich von Mitola geprägt, der sich bereits zuvor [59, 60] mit der Flexibilisierung und der Interoperabilität von Mobilfunkgeräten beschäftigt hatte. Die Grundidee besteht darin, die gesamte verfügbare Information innerhalb eines breiten Frequenzbandes zunächst vollständig zu digitalisieren und dann abhängig von den Bedürfnissen des Nutzers und seiner aktuellen Kommunikationssituation beliebige Unterbänder zu demodulieren und deren Information weiter zu verarbeiten. Dabei können in den Unterbändern verschiedenste Mobilfunkstandards angetroffen werden.

Um verschiedenste Standards bedienen zu können, muss die Signalverarbeitungskette sehr flexibel gehalten werden. Insbesondere ist es unmöglich, zur Zeit des Systementwurfs alle Nutzersituationen vorherzusehen und Spezialhardware für alle möglichen Mobilfunkstandards bereitzuhalten. Auch ist es beim Entwurf von Spezialhardware sehr oft nicht möglich, denjenigen Änderungen zu folgen, die sich aus einem laufenden Standardisierungsprozess ergeben. Zuletzt ist es unmöglich, die Entwicklung völlig neuartiger Mobilfunkstandards vorherzusehen und Hardware dafür zu entwickeln, weil die Signalverarbeitungskette zum Zeitpunkt der Hardware-Entwicklung schlichtweg unbekannt ist.

Die Lösung für dieses Problem besteht darin, die Signalverarbeitung nicht mehr durch Spezialhardware zu bewältigen, sondern mit Hilfe von Software. Diese Software soll auf einer allgemein gültigen Hardware ablaufen. Weil die Funktionalität des Gerätes wesentlich durch Software definiert wird, spricht man von einem idealen Software Radio.

Die Vorteile liegen in der großen Flexibilität von Software und in der Wiederverwendung von Hardware-Komponenten für verschiedenste Mobilfunkstandards. Software kann eingesetzt werden, um alle möglichen Signalverarbeitungsaufgaben auf allen Schichten des OSI-Modells [41] zu lösen. Von besonderem Interesse für die Nachrichtentechnik sind jedoch die Aufgaben der beiden untersten Schichten, also der physikalischen Schicht und der Datensicherungsschicht. Gerade die Realisierung dieser Schichten in Software erlaubt erst die äußerst flexible drahtlose Anbindung eines Nutzers an das Festnetz.

Die Betreiber von Mobilfunknetzen könnten durchaus ein Interesse an der Verbreitung von Software Radios haben [33], weil die Nutzer solcher Geräte entsprechend ihrer momentanen Mobilität verschiedene Arten des drahtlosen Zugangs zum Netz ein und desselben Betreibers wählen können und erwartet wird, dass sich auf diese Weise die Verbindungszeiten im Durchschnitt erhöhen. Zusätzlich nennen Netzbetreiber die Möglichkeiten für ein differenziertes Diensteangebot sowie einfache Fehlerbehebung, Wartung und Aufrüstung von Geräten (vorrangig in Basisstationen) als Vorteile von Software Radio [32].

Für mobile Endgeräte liegen die Nachteile eines idealen Software Radios eindeutig in der erhöhten Leistungsaufnahme und in den hohen Stückkosten flexibler Hardware-Komponenten (DSP, FPGA) im Vergleich zu Spezialhardware (ASIC, ASSP). Eine umfassende Sammlung von Beiträgen zu Hardware-Komponenten und zur Abwägung von Vor- und Nachteilen von idealen Software Radios findet sich in [45]. Im Folgenden sollen lediglich zwei Argumentationslinien herausgegriffen werden, die letztlich auf die Modelle und Lösungsansätze der vorliegenden Arbeit geführt haben.

Aus technischer Sicht ist es nicht sinnvoll, ein sehr breites Frequenzband zu digitalisieren und dann die Auswahl eines relativ schmalen Nutzbandes mit Hilfe von Digitalfiltern vorzunehmen. Dazu wäre nicht nur ein A/D-Umsetzer mit extrem hoher Abtastrate notwendig, sondern gleichzeitig eine große Wortbreite der gewandelten Werte, um die hohe Dynamik der Empfangssignale im gesamten breiten Frequenzband zu beherrschen. Es ist bekannt [108], dass zwischen Abtastrate und Wortbreite eines A/D-Umsetzers eine Beziehung herrscht, die dem Zeit-Bandbreite-Produkt in der Nachrichtentechnik ähnelt: Man kann A/D-Umsetzer herstellen, die entweder eine hohe Abtastrate bei kleiner Wortbreite erzielen oder umgekehrt, aber nicht gleichzeitig beides in unbeschränktem Maße. Bei konstanter Abtastrate verschiebt sich die Technologiegrenze alle 8 Jahre etwa um $1{,}5$ bit [108]. Demnach ist nicht zu erwarten, dass sich das Wandlerproblem durch schlichtes Abwarten in den kommenden Jahren lösen wird. Zudem ist die Herstellung solcher Wandler zunächst sehr teuer und die elektrische Leistungsaufnahme enorm. Der technologische Aufwand ist ebenso wenig zu rechtfertigen wie die erhöhte Leistungsaufnahme, wenn man bedenkt, dass unmittelbar nach der Wandlung durch die Reduktion der Bandbreite auf ein relativ schmales Nutzband der Großteil aller digitalisierten Informationen bereits wieder verworfen wird. Das ideale Software Radio ist damit per se teuer und bezüglich der elektrischen Leistungsaufnahme ineffizient.

Aus betriebswirtschaftlicher Sicht ist es für die Hersteller von Endgeräten äußerst

zweifelhaft, ob sich mit einem idealen Software Radio ein tragfähiges Geschäftsmodell aufbauen lässt. Angenommen, ein solches Gerät kann hergestellt und vom Endkunden zu einem akzeptablen Preis gekauft werden. Dann könnte es auch möglich werden, ein und dasselbe Gerät allein durch den Austausch von Software zu künftigen Mobilfunkstandards kompatibel zu machen. Signalverarbeitungssoftware für Software Radio würde zu einem neuen Markt werden, der auch für Dritte offen ist, so z.B. für unabhängige Software-Anbieter. Sicherlich besteht die Möglichkeit, dass der Hersteller eines Software Radios nicht nur als Hardware- sondern auch als Software-Anbieter am Markt auftritt. Jedoch würde sich der ursprüngliche Hersteller von Endgeräten dadurch selbst unter Druck setzen: Er müsste seine eigenen Kompetenzen erweitern und sich gleichzeitig der Konkurrenz durch Dritte stellen. Beides birgt unternehmerische Risiken.

Die geschilderte Art der Aufrüstbarkeit durch Software (*engl.* future upgradability) wurde lange Zeit als einer der Vorzüge von Software Radios angeführt. Nach der vorangegangenen Überlegung ist in der Tat jedoch das Gegenteil der Fall. Dennoch zeigen namhafte Hersteller von mobilen Endgeräten Interesse an dem Thema Software Radio [91, 116, 117]. Dabei wird vor allem auf die Flexibilität hingewiesen, die ein Software Radio mit sich bringt (*engl.* "Flexibility sells, future upgradability does not." [16]). Dass es sich hierbei im Wesentlichen um die Flexibilität der Unternehmen handelt und nicht um die des Endkunden, wird im folgenden Abschnitt erläutert.

1.1 Randbedingungen

Traditionell wurden Funkgeräte entworfen und gebaut, um die Anforderungen einer bestimmten Nutzergruppe (z.B. Militär, BOS, Luftfahrt, Privatpersonen, Nachrichtenagenturen) zu befriedigen, die unter festgelegten Randbedingungen und über eine bestimmte Klasse von Kanälen kommunizieren. Sprachkommunikation stand dabei zunächst im Vordergrund. Folglich waren viele Aspekte in Bezug auf die analogen Hochfrequenzbaugruppen und die digitale Signalverarbeitung im Basisband wohldefiniert. Produkte wurden selbstverständlich mit Spezialhardware realisiert, insbesondere für einen Massenmarkt wie den zivilen Mobilfunk der zweiten Generation. Mittlerweile sind die Nutzeranforderungen jedoch alles andere als wohldefiniert, und dafür gibt es mehrere Gründe: Der wachsende Anteil von Daten- und Multimediadiensten (die den Anwendern aus leitungsgebundenen Netzen bekannt

sind), die Vielfalt der Mobilfunkstandards und nicht zuletzt die Verunsicherung der Endkunden über das Preis-/Leistungsverhältnis der dritten Mobilfunkgeneration.

Der Anteil von Daten- und Multimediainhalten am gesamten Kommunikationsaufkommen ist in den letzten Jahren deutlich gestiegen. Insbesondere die Anbindung privater Haushalte an das Internet über ISDN und ADSL hat dazu beigetragen, den Anteil paketorientierter Dienste, die auf dem Internetprotokoll aufbauen, zu vergrößern. Eine große Anzahl von Anwendern kennt die Vorteile von HTTP-, FTP- und SMTP-basierten Diensten und das (häufig kostenlose) Herunterladen von Anwendungssoftware. Es ist anzunehmen, dass diese Dienste in den kommenden Jahren mehr und mehr auf mobile Geräte übertragen werden und dass diese von den Nutzern auch nachgefragt werden. Dazu ist es unabdingbar, die passenden Übertragungstechniken zu beherrschen, und zwar gerade auch im Endgerät, abhängig von der aktuellen, vorher nicht festgelegten Kommunikationssituation des Nutzers. Entwurfstechniken aus dem Bereich Software Radio können dabei von Vorteil sein.

In leitungsgebundenen Netzen ist derzeit ein Trend zu beobachten, der langfristig darauf abzielt, die Leitungsvermittlung abzuschaffen und jegliche Kommunikation paketorientiert abzuwickeln. VoIP ist ein extremes Beispiel dafür. Mit diesem Trend geht die Konvergenz verschiedener Netze zu einem einheitlichen Netz (*engl.* all-IP network), das mit dem IPv6 arbeitet, einher.

Ganz im Gegenteil dazu beobachtet man im Bereich der drahtlosen Zugangsnetze eine wachsende Vielfalt von Mobilfunkstandards: Weltweit sind verschiedenste Standards der zweiten Generation im Einsatz (GSM, TETRA, TETRAPOL, PHS, IS-54, IS-136, IS-95A/B). Diejenigen Varianten, die zur Generation $2{,}5$ gezählt werden, umfassen technische Änderungen (z.B. 8-PSK in EDGE anstelle von GMSK bei GSM Phase 1) oder die Bündelung von physikalischen Kanälen (HSCSD) zur Erhöhung der Nutzdatenrate oder die Umstellung der Luftschnittstelle auf paketorientierte Übertragung (GPRS). Die Einführung eines umfassenden Standards zur mobilen Telekommunikation (*engl.* Universal Mobile Telecommunications System, UMTS) als einziger Standard der dritten Generation war ursprünglich gedacht, um dieser Vielfalt ein Ende zu setzen. Allerdings kann der Versuch als gescheitert angesehen werden, weil die ITU unter dem Kürzel IMT-2000 nicht ein einziges, sondern fünf verschiedene Luftschnittstellenkonzepte zusammenfasst [42], darunter auch die zu IS-136 rückwärtskompatible Variante TDMA Single Carrier sowie DECT, einen Standard der zweiten Generation für Schurlostelefonie. In Europa versteht man unter dem Kürzel UMTS hauptsächlich UTRA-FDD und UTRA-TDD mit DS-CDMA als Vielfachzugriffsverfahren, während die Firma

Qualcomm die Variante MC-CDMA (cdma2000 1X/3X) favorisiert, die ebenfalls rückwärtskompatibel ist, und zwar zum nordamerikanischen IS-95B (cdmaOne). Nicht nur technische Überlegungen, sondern vor allem wirtschaftliche Interessen werden für die Entscheidung der ITU zu Gunsten einer solchen Kompromisslösung verantwortlich gemacht. Damit wird letztlich der Erfolg am Telekommunikationsmarkt über die genaue technische Ausprägung der dritten Generation entscheiden, nicht ein Standardisierungsgremium.

Neben dem flächendeckenden zellularen Mobilfunk der dritten Generation rücken auch WLANs ins Interesse von Nutzern und Geräteherstellern. Hier sind insbesondere die Standards HIPERLAN/2 und IEEE802.11a/b zu nennen. Mittlerweile sehen die Hersteller in UMTS und WLAN eine Kombination von komplementären Lösungen für unterschiedliche Nutzersituationen: Auf der einen Seite UMTS für hohe Mobilität und niedrige Datenraten in flächendeckenden Netzen, auf der anderen Seite WLAN für nomadische Anwendungen und höhere Datenraten in der unmittelbaren Umgebung von isolierten Zugangspunkten (*engl.* hot spots). Bereits bei der technischen Beherrschung dieser Kombination kommen Techniken aus dem Bereich Software Radio zum Einsatz [11, 66, 117].

Die oben genannten Vertreter der zweiten und dritten Generation sind nur ein unvollständiger Ausschnitt aus der zunehmenden Vielfalt von Mobilfunkstandards. Der Grund dafür liegt im weitesten Sinne in der Vielfalt der Situationen und Anforderungen mobiler Nutzer. Je besser man einen Standard an die augenblickliche Lage des Nutzers anpasst, desto besser die Ergebnisse bezüglich der Übertragungsqualität. Bei dieser Anpassung sind die physikalische Schicht und die Datensicherungsschicht von besonderer Bedeutung, da höhere Schichten letztlich nur auf die Fähigkeiten der untersten Schichten aufbauen können. Daher beschäftigt sich die vorliegende Arbeit mit Software, welche in einem Software Radio die Signalverarbeitungsaufgaben auf diesen untersten Schichten des OSI-Modells erfüllt.

Ein weiterer Grund dafür, den traditionellen Entwurf von Funkgeräten grundsätzlich zu überdenken, besteht in betriebswirtschaftlichen Überlegungen. Beispielsweise zeigt die Einführung von UMTS in Europa ganz deutlich: Wenn die Nachfrage nach einem Mobilfunkstandard hinter den Erwartungen eines bestimmten Geschäftsmodells zurückbleibt, oder wenn zufolge einer Trendwende plötzlich andere Dienste (und damit auch andere zugrunde liegende Übertragungstechniken) nachgefragt werden, dann wäre es für Hersteller ebenso wie für die Netzbetreiber sehr vorteilhaft, auf diese Situation schnell und angemessen reagieren zu können. Das schließt die Wiederverwendung (oder modulare Erweiterung) eines bereits fertig

entwickelten Hardware-Konzeptes ein, so dass nur Software ausgetauscht werden müsste. Software Radio zeigt genau für diese Situation das benötigte Lösungspotential.

Man kann aus den geschilderten Randbedingungen den Schluss ziehen, dass die Flexibilität von Software Radio also nicht in vollem Umfang an den Endkunden weitergegeben, sondern weitgehend von Geräteherstellern kontrolliert werden wird. Der genaue Hardware-Aufbau und alle Aspekte, welche die interne Organisation der Software und das Betriebssystem betreffen, werden nach wie vor firmenspezifisches Know-how bleiben. Diese Prognose spricht nicht gegen die Einführung offener Schnittstellen [89], sondern lediglich für spezifische Implementierungen von Software Radios. So können sich Hersteller weiterhin voneinander differenzieren und trotz kurzer Produktzyklen Entwicklungskosten über mehrere Geräteversionen und verschiedene Produktlinien amortisieren. Software Radio ist also als **der** Prototyp des eingebetteten Systems (*engl.* embedded system) in der Telekommunikation aufzufassen.

1.2 Modulares Software Defined Radio

In der Begriffswelt wird zuweilen zwischen Software Radio und Software Defined Radio unterschieden [114], wobei der erste Begriff das ideale Software Radio meint, das in der Einleitung bereits diskutiert worden ist. Um klarzustellen, dass die vorliegende Arbeit von realistischen Randbedingungen ausgeht, wird im Folgenden nur noch der Begriff des Software Defined Radio (SDR) gebraucht. Unter diesem Sammelbegriff werden alle Techniken verstanden, die das Ideal an einer oder an mehreren Stellen kompromittieren, jedoch dafür sorgen, dass die Grundideen und Vorteile von Software Radio auf verfügbarer Hardware technisch anwendbar und wirtschaftlich sinnvoll nutzbar werden.

Die Flexibilität eines Software Defined Radios kann um so besser für verschiedene Mobilfunkgeräte genutzt werden, je allgemeiner und skalierbarer der Entwurfsansatz ist. Gerade bei der Einführung einer neuen Gerätegeneration ist schnell einsichtig, dass eine bestimmte Hardware, egal wie leistungsfähig sie im Moment erscheinen mag, beim Übergang auf wesentlich kompliziertere Verfahren zur drahtlosen Datenübertragung irgendwann an ihre Grenzen stößt. Es wäre in solchen Fällen günstig, wenn diese Grenzüberschreitung zwar die aktuelle Hardware-Realisierung, nicht aber das gesamte Hardware-Konzept beträfe. Die Entwicklungsteams

eines Unternehmens würden dann mit dem gleichen vertrauten Konzept weiterarbeiten können und müssten lediglich für eine Steigerung der Rechenleistung sorgen, so dass der neue Mobilfunkstandard beherrschbar wird. Das Hardware-Konzept für ein Software Defined Radio sollte also modular sein in dem Sinne, dass Leistungssteigerungen allein durch Hinzufügen von einzelnen Zusatzkomponenten (z.B. weitere DSPs oder eines speziellen Coprozessors für die Beschleunigung bestimmter Signalverarbeitungsaufgaben) möglich sind. Damit ist offensichtlich, dass es sich bei der Hardware eines sinnvoll aufgebauten Software Defined Radios im Allgemeinen um ein Multiprozessorsystem handeln wird.

Unter diesen Bedingungen muss auch die Software zur Signalverarbeitung zunächst auf das Gesamtsystem verteilt (*engl.* partitioning) und ein Ablaufplan für alle Module erstellt (*engl.* scheduling) werden. Folglich braucht ein SDR ein Betriebssystem, das sich um diese Aufgaben kümmert. Dieses Betriebssystem wird sicherlich herstellerspezifisch sein und zu derjenigen Software gehören, die zusammen mit der Hardware an den Endkunden ausgeliefert wird (*engl.* firmware), aber weder vom Nutzer noch vom Netzbetreiber geändert werden kann. Um nicht für jede modulare Hardware-Konfiguration ein neues Betriebssystem entwickeln zu müssen, gilt es, allgemeine Richtlinien für den Entwurf und den Betrieb von Software Defined Radios zu finden, die mit allen möglichen Hardware-Konfigurationen zurechtkommen. Das Betriebssystem ist also das klassische Bindeglied zwischen Hardware und Software eines SDRs.

Auch Software in einem Software Defined Radio ist modular. Abhängig davon, welcher Mobilfunkstandard bedient werden soll, sind verschiedenste nachrichtentechnische Funktionen notwendig, so z.B. Quellencodierung, Kanalcodierung, Symbolerzeugung, Rahmenaufbau, Entzerrung und lineare Signaltransformation durch Filter oder FFT. Diese klassischen Funktionen können entweder vom Hersteller selbst entwickelt werden und aus kleineren Funktionsblöcken zusammengesetzt sein, oder sie werden von spezialisierten Anbietern in Form von monolithischen Funktionskernen (*engl.* IP cores) eingekauft und in die Softwaresammlung integriert. Die kleinsten zusammenhängenden Einheiten von ausführbaren Maschinenbefehlen werden im Folgenden Module genannt. Klassische nachrichtentechnische Funktionen sind in einem SDR hierarchisch aufgebaut und können wiederum aus Funktionen, Softwaremodulen oder Kombinationen aus beidem bestehen, je nachdem welche Hierarchieebene man betrachtet. Aus Sicht eines Prozessors gibt es sicherlich nur die unterste Ebene, in der ausschließlich Module vorhanden sind. Im Hinblick auf ein möglichst einfaches Betriebssystem wird angenommen, dass

die Ausführung von Modulen nicht unterbrochen werden kann: Ein Modul kann gestartet werden, wenn alle Eingangsdaten verfügbar sind. Das Modul belegt Datenspeicher, während es Eingangsdaten verarbeitet und Ausgangsdaten produziert. Die Ausführung des Moduls wird erst mit dem letzten Maschinenbefehl beendet. Zwischen Start und Ende der Ausführung eines Moduls m vergeht die Laufzeit p_m (*engl.* processing runtime).

Es ist im Rahmen der vorliegenden Arbeit von besonderer Wichtigkeit, dass die Laufzeit als die wesentliche charakteristische Größe zur Beschreibung eines Moduls erkannt wird, da sie unabhängig ist von der konkreten Hardware, auf der das Modul abläuft. Diese laufzeitorientierte Sichtweise wird den Kern der Software-Modelle für Modulares Software Defined Radio bilden.

Mit dem Umfang, in dem die komplexen Funktionen eines beliebigen Mobilfunkstandards aus kleineren Funktionsblöcken und Modulen zusammengesetzt werden, erhöht sich die Wahrscheinlichkeit, dass der Hersteller bei der Umstellung seines SDRs auf einen anderen Standard Funktionsfragmente und Module wiederverwenden kann (*engl.* code reuse). Das birgt den Vorteil geringerer Entwicklungszeiten für ein neues Produkt und ebenso den Vorteil der weitgehenden Fehlerfreiheit bereits getesteter Funktionen und Softwaremodule. Im Weiteren sollen unter dem Begriff Modulares Software Defined Radio (Mod-SDR) alle Arten von Software Defined Radios verstanden werden, bei denen die Hardware aus einem einfach zu erweiternden Multiprozessorsystem besteht und Software für die Signalverarbeitungsaufgaben der physikalischen Schicht aus Modulen zusammengesetzt ist. Allgemein gültige Richtlinien für den Entwurf und Betrieb solcher Modularer Software Defined Radios sind Gegenstand der Untersuchungen in der vorliegenden Arbeit.

1.3 Bisherige Arbeiten im Bereich Software Radio

In der Literatur finden sich zahlreiche Beiträge zum grundlegenden Konzept des idealen Software Radios [13, 61, 63, 64] und zu Kompromisslösungen, also Software Defined Radios [102]. Die Ursprünge, Entstehungsgeschichte und verschiedene Sichtweisen zu SDR sind in [103] umfassend beschrieben. Mittlerweile werden auch alle Aufgaben der Netzwerk- und Nutzerverwaltung in die Überlegungen zur Realisierung flexibler Mobilfunkanwendungen einbezogen [19]. Dies schließt adaptive Protokollschichten, Organisation des Diensteangebots, Sicherheitsaspekte, ”intelligente” Endgeräte (*engl.* ”Cognitive Radios” [64, 65]), flexible Frequenzver-

gabe (”Spectrum Pooling” [52, 68, 111]) und Modulationsartenerkennung [67] ein. Die zuletzt genannten, sehr umfangreichen Aufgaben gehen weit über den Rahmen der vorliegenden Arbeit hinaus und werden im Folgenden nicht weiter behandelt.

Frühe Beiträge zu flexiblen Endgeräten erläutern meist Bauteile, Komponenten und technologische Randbedingungen [4, 17, 48, 88], ziehen jedoch kaum konkrete Schlussfolgerungen bezüglich des Entwurfs und Betriebs von SDRs. Die fortgeschrittenen Artikel befassen sich fast ausschließlich mit einem bestimmten Teil der Signalverarbeitungskette, so z.B. mit MEMS [84, 85] für den Antennen- und Filterentwurf [7, 73], der Wahl der Zwischenfrequenzen im analogen Hochfrequenzteil des Empfängers [7, 21, 58, 90, 114], A/D- und D/A-Umsetzung [46, 96], der Abtastratenumsetzung [2, 3, 37] oder der Kanalcodierung [4, 100, 105]. Manche Ergebnisse beziehen sich auch gar nicht zwingend auf das Konzept des SDRs, sondern auf die klassische Nachrichtentechnik [118]. In diesen Artikeln gibt es oft einen Bruch zwischen einer Einführung, die viel von SDR verspricht, und einem Teil mit konkreten Ergebnissen in der Form BER = f (SNR), die mit Hilfe von Spezialhardware ebenso hätten erzielt werden können wie mit einer Softwarerealisierung [35, 44]. Systematische Aspekte von SDR werden häufig vermischt mit den Aufgaben der Signalverarbeitung aus anderen Bereichen der Nachrichtentechnik: Intelligente Antennensysteme (*engl.* smart antenna systems) [39, 71, 72] und MUD [94]. Diese Analyse soll die erzielten Ergebnisse in keiner Weise schmälern, jedoch muss man sich klarmachen, dass sie nicht unmittelbar eine Methodik zum Entwurf und Betrieb eines Software Defined Radios betreffen, sondern lediglich spezielle Funktionen der Signalverarbeitung, die hinsichtlich der Integration in ein Mod-SDR methodisch genau so zu behandeln sind wie alle klassischen nachrichtentechnischen Funktionen.

Eine Arbeit [62], die sich dem Thema Software Defined Radio von theoretischer Seite nähert, zeigt deutliche Schwächen im Hinblick auf die Begriffsbildung (*engl.* ”topology”) und auf die statistischen Eigenschaften der in einem SDR benötigten Rechenleistung (*engl.* ”statistical demand for processing power”). Die Ergebnisse der genannten Arbeit gehen letztlich nicht über die bekannten Aussagen der theoretischen Informatik hinaus (Church-Turing-These [113]).

Ein wichtiger Schritt auf dem Weg zu konkreten Entwurfsrichtlinien für Software Defined Radio besteht in dem Ansatz PaC-SDR [43, 114], der die Signalverarbeitung im digitalen Basisband betrifft. Dabei wird eine zum Entwurfszeitpunkt festgelegte Menge an Mobilfunkstandards auf ihre Gemeinsamkeiten und Unterschiede hin untersucht. Eine grundlegende Hypothese von PaC-SDR besteht darin,

dass die Gemeinsamkeiten bisher weit überwogen haben und sich das auch in Zukunft nicht ändern wird, ungeachtet der Möglichkeit, ganz andersartige Standards zu entwerfen. Der Grund liegt im Wesentlichen darin, dass die klassische Nachrichtentechnik Antworten auf die Eigenschaften des Mobilfunkkanals entwickelt hat und sich letztere auch in Zukunft nicht ändern werden. Das Entwurfsziel von PaC-SDR besteht nun darin, die Algorithmen der Signalverarbeitung so allgemein zu halten, dass sie sich lediglich durch die Angabe eines Parametersatzes von einem auf den anderen Standard umschalten lassen. Die Idee kann der Realisierung von schnellem vertikalen Handover dienen, aber auch der Interoperabilität zwischen Nutzergruppen (BOS und Militär, multinationale Einsätze). Das Konzept des Betriebs allgemein gültiger, parametrisierter Softwaremodule zieht sich durch alle hierarchischen Funktionen der untersten OSI-Schichten. Zum ersten Mal geht es bei PaC-SDR also nicht um eine Einzellösung, sondern um ein Konzept, das einzig und allein SDR zu Eigen sein kann.

Der Hauptvorteil von PaC-SDR liegt im geringen Datenvolumen der Parametersätze im Vergleich zur gesamten Software. Gerade dadurch, dass man eben nicht die Gesamtheit der nachrichtentechnischen Funktionen beim Wechsel auf einen neuen Mobilfunkstandard laden muss, sondern lediglich einen neuen Parametersatz, kann man schnelles vertikales Handover realisieren. Weiterhin ist die Wiederverwendbarkeit von Modulen und Funktionen selbst in komplizierten Anwendungen (z.B. zyklische Faltungscodes [74] oder Bandspreizen [76]) gezeigt worden.

Allerdings stößt man bei den Aufgaben der Kanalcodierung auch schnell an die Grenzen von PaC-SDR [74]: Zum einen kann es mitunter einfacher sein, Spezialaufgaben wie einen einfachen Viterbi-Decoder und einen Turbo-Decoder getrennt zu entwerfen und nicht eine allgemeine Funktion, die abhängig von Parametern entweder auf die eine oder andere Weise decodiert. Zum anderen besteht immer die Gefahr, dass ein allgemein gültiges Softwaremodul bezüglich seiner Laufzeit weit weniger effizient ist als getrennte Spezialmodule für beide Decodieraufgaben. Ein weiterer Schwachpunkt besteht darin, dass man bei der Erweiterung auf ein neues Codierverfahren gegebenenfalls gezwungen wird, das Modul von Grund auf neu zu entwickeln, und zwar ohne Wiederverwendung existierender Softwaremodule. Beispielsweise ist es nicht möglich, ein auf Binärcodes ausgelegtes Modul zur Erzeugung von Symbolcodes (Reed-Solomon-Codes) zu nutzen. Das umgekehrte Vorgehen ist ohne Weiteres vorstellbar, aber es erfordert den völlig neuen Entwurf eines Reed-Solomon-Encoders, der parameterabhängig auch Binärcodes beherrscht. Zudem ist die Wahrscheinlichkeit hoch, dass dieser neue Encoder für

die einfacheren Binärcodes ineffizient arbeitet. Der Kompromiss aus Flexibilität und Effizienz, der schon mit Blick auf die Hardware von Software Radios diskutiert wurde, setzt sich bei PaC-SDR also in die Welt der Software fort. Und schließlich betreffen die quantitativen Ergebnisse [114] wieder die klassische Nachrichtentechnik in Form von BER = f (SNR).

1.4 Ziel und Aufbau der vorliegenden Arbeit

Mod-SDR steht keinesfalls im Widerspruch zu den Ansätzen von PaC-SDR, sondern kann als Verallgemeinerung und systemtheoretisches Komplement betrachtet werden. Die Tatsache, dass die Modullaufzeiten im Vordergrund stehen, schließt weder aus, dass diese Module parametergesteuert sind, noch dass es sich um Spezialmodule handelt oder sogar um Spezialhardware. Wo PaC-SDR aber an seine Grenzen stößt (Komplexität und Laufzeitineffizienz allgemein gültiger Softwaremodule), sind stets Speziallösungen für nachrichtentechnische Aufgaben zu finden und in ein offenes, modulares System zu integrieren. Jedes allgemeine wie spezielle Softwaremodul hat eine Laufzeit, ja sogar jede Form von Spezialhardware (also Funktionen in ASICs, Coprozessoren oder auf spezialisierte Beschleuniger-Hardware ausgelagerte Funktionen), so dass Mod-SDR heterogene Systeme oder eine Mischung aus Hardware- und Softwarelösungen nicht von vornherein ausschließt.

Im Gegensatz zu PaC-SDR ist Mod-SDR eher strukturorientiert. Von den Algorithmen der Nachrichtentechnik wird so weit abstrahiert, dass lediglich die Laufzeit von Softwaremodulen, ihre Beziehungen untereinander und die Speicheranforderungen für Ein- und Ausgangsdaten erhalten bleiben. Wenn es in dieser Arbeit um Algorithmen geht, so sind nicht die Algorithmen der Signalverarbeitung gemeint, sondern diejenigen Algorithmen, die der Organisation des Softwareablaufs in einem Multiprozessorsystem dienen. Modulares Software Defined Radio verlässt damit naturgemäß die klassische Nachrichtentechnik und liegt eher an der Schnittstelle zwischen Informatik, Nachrichtentechnik und den Techniken des automatischen Entwurfs elektronischer Schaltungen. Daher sind an keiner Stelle dieser Arbeit Ergebnisse in der Form BER = f (SNR) zu finden. Nichts desto trotz werden sowohl die Modellvorstellungen als auch die Interpretation der Ergebnisse für die Anwendung in der Nachrichtentechnik entwickelt.

Das Ziel der vorliegenden Arbeit besteht darin, grundlegende und allgemein gültige

Richtlinien herzuleiten, die zum optimalen Betrieb von SDR-Systemen (parametrisiert oder nicht, homogen oder heterogen) nötig sind. Diese Richtlinien sollen so weit wie möglich von konkreten Hardware-Systemen und von bekannten Mobilfunkstandards unabhängig sein.

Wegen der Neuartigkeit der Sichtweise beschäftigt sich Kapitel 2 zunächst mit der Modellbildung für Hardware und Software unter den geschilderten Randbedingungen. Kapitel 3 führt dann eine präzise mathematische Formulierung der SDR-Entwurfsaufgabe ein. Diese Betrachtung zeigt jedoch recht eindeutig, dass diese Formulierung die Aufgabe zwar klassischen Lösungsverfahren zugänglich macht, die Anwendung der Verfahren im Betriebssystem eines Mod-SDRs aber nicht praktikabel ist. Alternative Lösungsverfahren werden sodann in Kapitel 4 beschrieben. Dort geht es um die Partitionierung von Signalverarbeitungssoftware und um das Erstellen von Ablaufplänen. Das Kapitel 5 zeigt die Ergebnisse, die mit den beschriebenen Verfahren zu erzielen sind und vergleicht ihre Eigenschaften. Diese Ergebnisse werden im Sinne der Nachrichtentechnik interpretiert und führen letztlich auf die grundlegenden Entwurfsrichtlinien für Modulares Software Defined Radio.

2 Modellbildung

Um die Ziele dieser Arbeit verfolgen zu können, müssen zunächst geeignete Modelle für Hardware und Software in einem Modularen Software Defined Radio entwickelt werden. Eine besondere Schwierigkeit besteht darin, die Flexibilität von Software Defined Radio in die Modellbildung zu übernehmen. Die Eigenschaften von Hardware und Software sind einerseits möglichst allgemein zu beschreiben, andererseits sollen mit Hilfe der Modelle aber auch Simulationen betrieben und konkrete Ergebnisse abgeleitet werden.

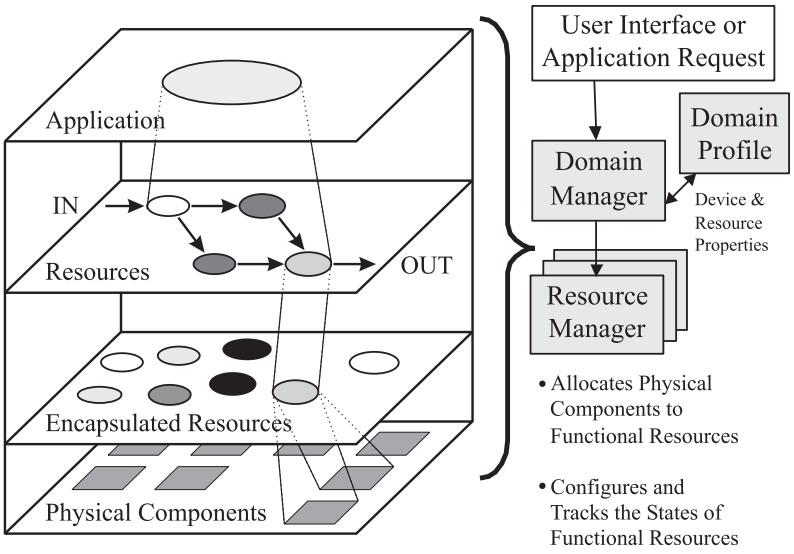


Bild 2.1 Frühe Modellvorstellung eines SDRs, Quelle: SDRF [92]

Das Bild 2.1 zeigt die Darstellung eines SDRs wie sie bereits Ende 1999 vom SDRF favorisiert wurde [92]. Es ist deutlich zu erkennen, dass eine Anwendungssoftware (*engl.* application) in einem SDR nicht als ein einziges großes Maschinenprogramm zu sehen ist, sondern sich naturgemäß aus verschiedenen nachrichtentechnischen Funktionen (*engl.* "functional resources" [92]) zusammensetzt, die alle zusammen Eingangsdaten verarbeiten und Ausgangsdaten produzieren. Zwischen den Funktionen bestehen logische Abhängigkeiten, und diese Funktionen sind in

irgendeiner Weise (*engl.* ”encapsulated resources” [92]) auch auf einer niedrigeren Hierarchieebene wiederzufinden. Desweiteren wird eine Abbildung von Software auf Hardware durchgeführt, und zwar so, dass ”encapsulated resources” unter Umständen auf mehr als einer physikalischen Komponente ablaufen. Bei der Hardware handelt es sich offensichtlich um ein System aus mehreren Prozessoren. Der Verbund aus Signalverarbeitungssoftware und Hardware wird von einem Betriebssystem verwaltet, das sich nach Bild 2.1 aus einem ”Domain Manager” und mehreren ”Resource Managern” zusammensetzt und das unabhängig vom betrachteten Mobilfunkstandard als Firmware in dem Software Defined Radio existiert. Bemerkenswert ist weiterhin, dass das Betriebssystem Anfragen des Nutzers oder einer laufenden Anwendung entgegennimmt.

Was sich genau hinter den verschiedenen Begriffen verbirgt und welche Verfahren für die Abbildung von Software auf Hardware zu wählen sind, bleibt in [92] offen. Versteht man jedoch unter ”functional resources” die nachrichtentechnischen Funktionen im Sinne von Kapitel 1, die hierarchisch aus anderen Funktionen und einfachen Softwaremodulen aufgebaut sind, so trifft das Bild bereits den Kern des in dieser Arbeit beschriebenen realistischen Modells für modulare, aus Hardware und Software bestehende Systeme: Es besteht die Notwendigkeit, Hardware und Software vermittels eines Betriebssystems zu kontrollieren, den Ablauf von Modulen auf einem Multiprozessorsystem so gut wie möglich zu koordinieren und auf Nutzeranfragen zu reagieren.

Allein aus Bild 2.1 wird in keiner Weise klar, wie Hardware und Software zu beschreiben sind (*engl.* ”domain profile” [92]) und wie die Hardware-Komponenten miteinander in Verbindung stehen. Eine entsprechende Verfeinerung dieser Aspekte zur Anwendung auf Modulares Software Defined Radio ist Gegenstand dieses Kapitels.

2.1 Hardware-Modell

Die allgemeine Vorstellung von einem idealen Software Radio ist im Zusammenhang mit Mod-SDR nicht hilfreich, denn dieses Ideal schließt den Begriff des Multiprozessors gar nicht ein, sondern basiert auf einem einzelnen Zentralprozessor (*engl.* CPU), der alle Signalverarbeitungsaufgaben auf allen Schichten des OSI-Modells erledigt. Sollte es jemals möglich sein, eine solche Ideallösung kosten- und leistungsgünstig herzustellen, besteht die in dieser Arbeit geschilderte Ent-

wurfsaufgabe nicht mehr. Allerdings ist aus der Diskussion der technischen und wirtschaftlichen Randbedingungen in Kapitel 1 zu schließen, dass dieser Fall mit sehr hoher Wahrscheinlichkeit auf absehbare Zeit nicht eintreten wird. Es gibt derzeit sogar eine ganze Reihe von Hinweisen [18, 31, 98, 101, 104, 106], dass sich die Signalverarbeitung in der mobilen Telekommunikation eher auf Strukturen mit verteilten Prozessorelementen stützen wird. Folglich braucht auch Modulares Software Defined Radio ein entsprechendes Hardware-Modell.

2.1.1 Symmetrischer Multiprozessor

Als Prototyp des Mehrprozessorsystems wird in dieser Arbeit ein symmetrischer Multiprozessor mit L identischen Prozessoren und B unabhängigen Bussen zur Inter-Prozessor-Kommunikation betrachtet.

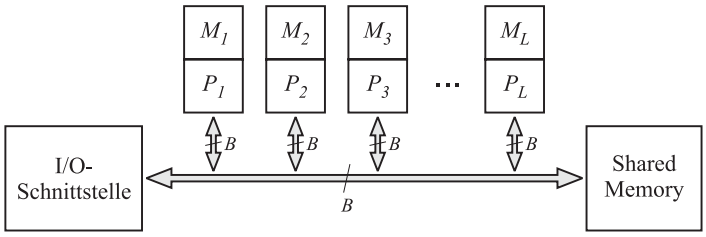


Bild 2.2 Symmetrischer Multiprozessor

Bild 2.2 zeigt das System mit einer Schnittstelle zur Ein- und Ausgabe von Daten (*engl.* I/O interface) und einem gemeinsamen Speicherbereich (*engl.* shared memory), auf den alle Prozessoren beliebig lesend und schreibend zugreifen können. Jedem Prozessor P_k, $1 \leq k \leq L$ ist ein eigener Datenspeicherbereich M_k zugeteilt (*engl.* distributed memory), auf den von P_k aus jederzeit uneingeschränkt zugegriffen werden kann. Es soll angenommen werden, dass Prozessoren aktiv an allen Buszugriffen beteiligt sind, d.h. ein Prozessor kann parallel zur Buskommunikation keine anderen Softwaremodule ausführen. Sollen Zwischenergebnisse zwischen zwei Prozessoren ausgetauscht werden, so schreibt der Quellprozessor die Daten über einen Bus in das Shared Memory, und der Zielprozessor holt die Daten dort später mit einem weiteren Zugriff auf den Bus ab. Der Vorteil eines solchen asynchronen Datenaustausches über das Shared Memory liegt darin, dass nicht beide Prozessoren gleichzeitig verfügbar sein müssen. Der Zielprozessor kann nach wie vor Rechnungen im Rahmen der Signalverarbeitung ausführen, während der Quell-

prozessor bereits Daten schreibt, und der Quellprozessor muss später die Ausführung von Modulen nicht mehr unterbrechen, wenn der Zielprozessor die für ihn bestimmten Daten aus dem Shared Memory liest.

Sicherlich bieten moderne Prozessoren Kommunikationsschnittstellen an, die unabhängig von der Aktivität des Prozessorkerns Daten versenden oder empfangen können, vorausgesetzt ein Softwaremodul wird ausdrücklich dahingehend programmiert. Zum Entwurfszeitpunkt eines Moduls ist im Allgemeinen aber unbekannt, ob parallel zur Signalverarbeitung ein Austausch von Zwischenergebnissen stattfinden soll. Das wird erst vom Betriebssystem zum Einsatzzeitpunkt geplant. Gerade das konkrete Zusammenspiel eines Moduls mit anderen Modulen entscheidet ja darüber, ob Inter-Prozessor-Kommunikation nötig ist oder nicht. Ferner kann ein nicht-präemptives Betriebssystem auch keine speziellen Maschinenbefehle in ein Modul einfügen, weil Module nach Voraussetzung in ihrer Ausführung nicht unterbrochen werden können. Demnach ist die Wahrscheinlichkeit gering, dass spezielle Kommunikationsschnittstellen immer genau so genutzt werden wie vom Prozessorhersteller angegeben. In dieser Hinsicht kann man also volle Prozessoraktivität bei Buskommunikation als pessimistische, aber nicht unwahrscheinliche Annahme werten.

Buszugriffe sollen exklusiv sein, d.h. während der Zugriff durch einen Prozessor stattfindet, kann ein zweiter Prozessor nicht gleichzeitig auf den gleichen Bus zugreifen. Sollte dieser Fall auftreten, spricht man von einem Buskonflikt. Im Rahmen des beschriebenen, möglichst einfachen Betriebssystems soll ein solcher Buskonflikt aufgelöst werden, indem der zweite Prozessor solange angehalten wird (”`PHYSICAL_WAIT_IDLE` condition” [80]), bis der Bus vom ersten Prozessor wieder freigegeben ist. Auf diese Weise können Leerlaufzeiten (*engl.* idle times) entstehen, in denen weder sinnvolle Rechnungen durchgeführt noch Zwischenergebnisse an andere Prozessoren verschickt werden. Es ist unmittelbar einsichtig, dass diese Leerlaufzeiten vermieden werden sollten, um die vorhandenen Prozessoren möglichst gut auszunutzen.

Ein symmetrischer Multiprozessor ist leicht erweiterbar, indem weitere Prozessoren gleicher Bauart an das Bussystem angeschlossen werden. Ein Bus kann beliebige Prozessoren miteinander verbinden, unterliegt also im Prinzip keinen Einschränkungen bezüglich der logischen Hardware-Struktur. Da physikalisch aber nur eine einzige Verbindung vorhanden ist, erlaubt ein einzelnes Bussystem die Vollvermaschung lediglich im zeitlichen Multiplex. Die Anzahl B der nötigen unabhängigen Busse ist eine kritische Größe beim Hardware-Entwurf, weil die Komplexität und

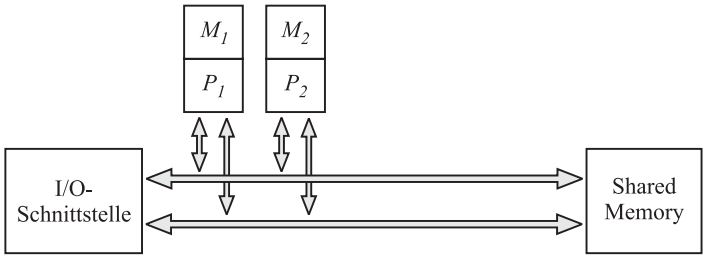


Bild 2.3 Symmetrischer Multiprozessor, $L = 2$ Prozessoren, $B = 2$ Busse

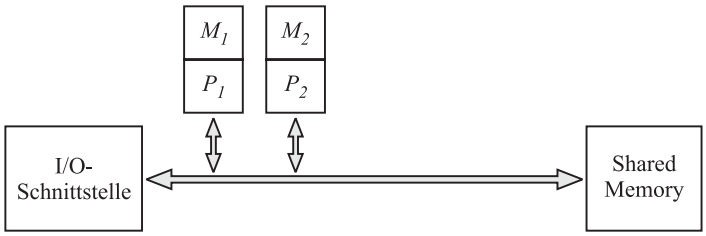


Bild 2.4 Zwei-Prozessor-System, $B = 1$ Bus

damit die Kosten sicherlich mit wachsendem B steigen. Bereits an dieser Stelle wird klar, dass der symmetrische Multiprozessor zwar einige Anforderungen erfüllt, die man von einem allgemeinen Hardware-Strukturmodell für Software Defined Radio erwartet, aber auch grundlegende Eigenschaften besitzt, die die Allgemeinheit einschränken.

Wenn sich der allgemeine Fall mit L Prozessoren als zu komplex erweist, erscheint es zunächst plausibel, die Entwurfsprinzipien für $L = 2$ Prozessoren zu untersuchen. Mehr als $B = 2$ Busse sind dann nicht sinnvoll, denn mehr als zwei gleichzeitige Zugriffe auf das Shared Memory oder die I/O-Schnittstelle kann es bei $L = 2$ Prozessoren nicht geben. Bild 2.3 zeigt diesen Sonderfall.

Bei dieser besonderen Prozessorkonfiguration besteht die Frage, ob sich das Software Defined Radio nicht auch genau so gut mit einem einzigen Bus betreiben lässt (siehe Bild 2.4). Die in Kapitel 4 diskutierten Algorithmen und die meisten Simulationsergebnisse in Kapitel 5 beziehen sich auf diese beiden Hardware-Konfigurationen.

2.1.2 Einsatz eines Coprozessors

Selbst wenn der Betrieb von 2 Bussen das Auftreten von Buskonflikten unterbindet, kann es sein, dass $L = 2$ Prozessoren nicht ausreichen, um die für einen bestimmten Mobilfunkstandard nötige Rechenleistung aufzubringen. In diesem Fall sind die Grenzen der Realisierbarkeit einfach überschritten, und es muss entschieden werden, dass dieser Standard mit dem gegebenen Software Defined Radio nicht betrieben werden kann.

Ein Lösungsansatz, der immer wieder im Zusammenhang mit SDR-Hardware auftaucht, ist der Einsatz eines Coprozessors als Beschleuniger für bestimmte Signalverarbeitungsaufgaben. Bild 2.5 zeigt diese Konfiguration.

Ausgehend von einem Zwei-Prozessor-System wird also ein dritter Prozessor an die Busse angeschlossen. Zunächst sollen keine Annahmen über die Spezialisierung des Coprozessors gemacht werden, weil man in der Phase des Systementwurfs ja noch entscheiden kann, welche Module auf den Beschleuniger ausgelagert werden. Dennoch ist zu prüfen, ob selbst unter idealen Bedingungen die Realisierungsgrenzen überschritten werden oder ob der Betrieb des Standards durch Einsatz des Coprozessors nun möglich ist. Unter idealen Bedingungen soll dabei ein unendlich schneller Coprozessor verstanden werden.

Im Rahmen der Untersuchungen zum effizienten Betrieb eines Coprozessors wird das Zwei-Prozessor-System mit $B = 2$ Bussen aus Bild 2.3 als hypothetisches Coprozessorsystem aufgefasst, das aus Bild 2.5 hervorgeht, wenn ideale Bedingungen für eine Beschleunigung der Rechnungen angenommen werden. Die Laufzeiten der ausgelagerten Module werden damit zu Null, die Ein- und Ausgangsdaten der betreffenden Module werden jedoch nach wie vor über das Bussystem mit den beiden anderen Prozessoren ausgetauscht. Eine Alternative zu diesem hypothetischen System besteht darin, das System aus Bild 2.5 gedanklich in zwei Subsysteme zu zerlegen: Ein tatsächlich realisierbares Coprozessor-Subsystem mit einem Bus (siehe Bild 2.6) und ein Restsystem nach Bild 2.4, das die beiden ursprünglichen Prozessoren und ebenfalls einen Bus enthält. Diese beiden Subsysteme können vom Betriebssystem des Mod-SDR koordiniert betrieben werden und zeigen den Vorteil, zur gleichen Zeit unterschiedliche Daten verarbeiten zu können.

Am Beispiel des Coprozessorseinsatzes wird deutlich, dass selbst der Sonderfall mit $L = 2$ Prozessoren und $B = 2$ Bussen bereits verschiedene Möglichkeiten zum Betrieb eines Software Defined Radios eröffnet und dass die Koordination zwischen Signalverarbeitungssoftware und Hardware durch das Betriebssystem von

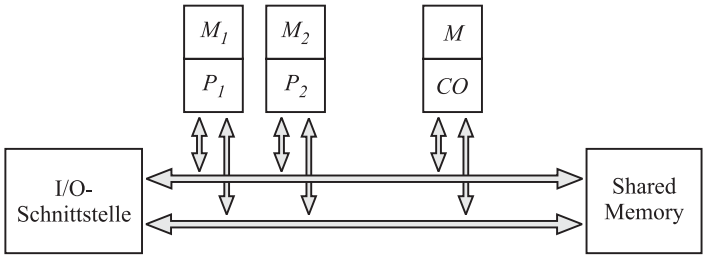


Bild 2.5 Zwei-Prozessor-System mit Coprozessor (*CO*), $B = 2$ Busse

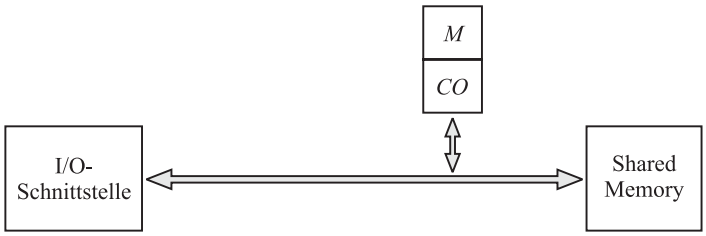


Bild 2.6 Coprozessor-Subsystem mit einem Bus

entscheidender Bedeutung sein wird. Bevor allgemeine Richtlinien für die Art und Weise dieser Koordination gefunden werden können, müssen noch Modelle für modulare Software eingeführt werden.

2.2 Software-Modelle

Der Begriff der Software wird in der Literatur häufig für alle möglichen Arten von ausführbaren Maschinenbefehlen benutzt. Auch verstehen Netzbetreiber unter Software zuweilen etwas anderes als die Hersteller von Endgeräten, und es muss bei Diskussionen viel Zeit darauf verwendet werden, sich auf eine gemeinsame Vorstellung von den Begriffen zu einigen. Leider bleibt der Softwarebegriff häufig voll und ganz unerklärt, oder er dient als intuitiver Sammelbegriff für alle möglichen Arten von Software auf allen Schichten des OSI-Modells. Der Grund liegt möglicherweise darin, dass das Konzept von Software Radio ja gerade dazu gedacht ist, sich konkreten Entwurfsentscheidungen so lange wie möglich zu enthalten und – übertrieben gesprochen – ”alles möglich” zu machen. Das ist natürlich für quan-

titative Untersuchungen nicht haltbar, denn die grundlegenden Vorstellungen von Softwareeigenschaften sollen klar nachvollziehbar und Randbedingungen reproduzierbar sein. Daher werden in diesem Abschnitt Modelle für Mod-SDR entwickelt, die sowohl die Flexibilität von Software Radio widerspiegeln, als auch mathematisch präzise sind, so dass sie später als Grundlage für Rechnersimulationen dienen können.

Die einzige verfügbare Spezifikation, welche eine allgemein gültige Beschreibung der Organisation und sinnvoller Schnittstellen für Software Defined Radios anstrebt, ist die Software Communications Architecture (SCA) [89]. Das Dokument befindet sich derzeit noch im Standardisierungsprozess und wird laufend erweitert. Die SCA entspringt einer Initiative des amerikanischen Verteidigungsministeriums, das mit Hilfe von SDR-Techniken bestehende Systeme interoperabel machen und langfristig die Vielfalt der Gerätschaften zur Funkkommunikation in den amerikanischen Streitkräften reduzieren soll. Die Arbeit an der SCA wird mit einem großen finanziellen und personellen Aufwand vorangetrieben, und da es keine alternativen Vorschläge aus Europa oder Asien gibt, stellt die SCA derzeit faktisch den Standard für Software Defined Radio dar, an dem sich Gerätehersteller weltweit orientieren.

Die vorliegende Arbeit soll keinesfalls auf militärisches SDR beschränkt sein, sondern allgemeine Entwurfsrichtlinien herleiten. Dennoch ist es sinnvoll, ein Software-Modell zu wählen, das mit der SCA kompatibel ist. Desweiteren sind die technischen Anforderungen an zivil genutzte Endgeräte oft eine Untermenge der Anforderungen aus dem militärischen Bereich. Die militärischen Randbedingungen (nicht-kooperatives Umfeld, fehlende Infrastruktur, verschiedenste Reichweiten, Notwendigkeit zur Verschlüsselung) werden lediglich durch andere ersetzt (Absatz auf einem Massenmarkt, niedrige Produktionskosten, Notwendigkeit zur Produktdifferenzierung). Daher muss es möglich sein, die Arbeit in den Rahmen der SCA einzuordnen.

Bild 2.7 zeigt einen Überblick über die prinzipielle Organisation eines Software Defined Radios nach der SCA. Die gesamte Software aus allen Schichten des OSI-Modells wird von einem Betriebssystem (*engl.* ”core framework” [89]) verwaltet und soll letztlich aus Kostengründen auf einem System ablaufen, das aus handelsüblichen Hardware-Bausteinen (*engl.* ”COTS hardware” [89]) besteht. Hardware und Betriebssystem bilden zusammen die Betriebsumgebung (*engl.* ”operating environment” [89]), in der die gesamte Software eines SDRs ihren Dienst tun soll.

Die vorliegende Arbeit beschränkt sich auf die physikalische Schicht (*engl.* ”mo-

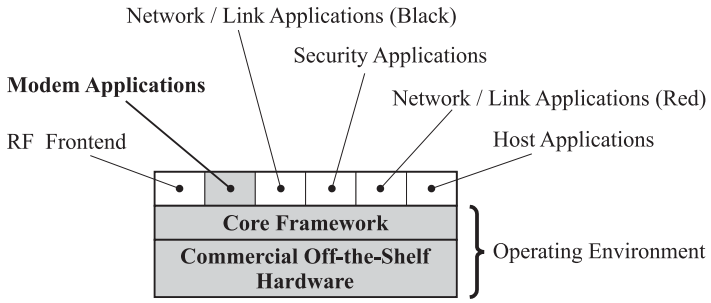


Bild 2.7 Signalverarbeitungssoftware und Betriebssystem, Quelle: JTRS JPO [89]

dem applications" [89]), wobei die beiden Schichten 1 und 2 des OSI-Modells im Folgenden unter diesem Begriff zusammengefasst werden. Das Bild 2.8 zeigt etwas detaillierter, wie Signalverarbeitungssoftware mit dem Betriebssystem interagiert und wo die Software der physikalischen Schicht einzuordnen ist.

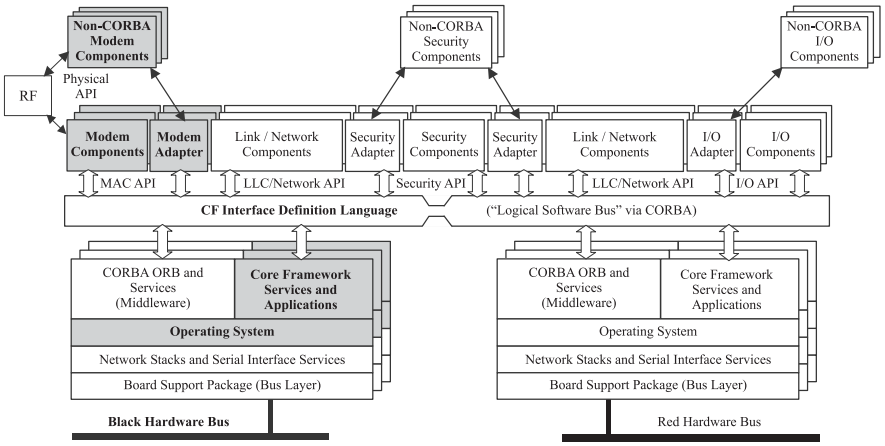


Bild 2.8 Software Communications Architecture, Quelle: JTRS JPO [89]

Im Mobilfunk wird für die physikalische Schicht auch oft der Begriff der Luftschnittstelle (*engl.* air interface) gebraucht. Darunter versteht man die Gesamtheit aller Signalverarbeitungsschritte, die notwendig sind, um Binärinformationen (Bits) auf die Funkübertragung über einen physikalischen Kanal vorzubereiten.

Darunter fällt auch die Sicherung der Binärinformationen durch eine auf die Eigenschaften des Funkkanals angepasste Kanalcodierung.

Es wird angenommen, dass die Signalverarbeitung für die Luftschnittstelle eines Mobilfunkstandards vollständig in Form von Softwaremodulen realisiert ist. Diese Software verarbeitet Signale zwischen dem Dienstzugangspunkt (*engl.* "SAP" [41], "interface" [40]) des analogen Hochfrequenzteils (*engl.* "RF frontend" [89]) für die physikalische Schicht einerseits und dem Dienstzugangspunkt der physikalischen Schicht für die nächsthöhere Protokollschicht (*engl.* "network/link applications" [89], "network layer" [41], "internet module" [40]) andererseits. Bei der Betrachtung der physikalischen Schicht geht es nicht um den gleichzeitigen Betrieb von mehreren Mobilfunkstandards, sondern allein um die optimale Abbildung der in Software realisierten Luftschnittstelle **eines beliebigen** Standards auf die zur Verfügung stehende Hardware. Der Nutzer wählt also aus vielen Alternativen genau eine drahtlose Zugangsmöglichkeit zum Festnetz.

Im Hinblick auf ein Inter-System-Handover [114] wird hier davon ausgegangen, dass alle Softwaremodule für die beiden am Handover beteiligten Luftschnittstellen bereits im Programmspeicher des Mobilgerätes stehen und dass ein hartes Umschalten zwischen den Standards möglich ist. Noch schneller als das Umladen von Parametersätzen in PaC-SDR ist sicherlich das Ändern der Befehlszeiger (*engl.* instruction pointer) von Prozessoren auf einen neuen Wert, so dass die Module des neuen Standards anstelle der Module des alten Standards abgearbeitet werden. Demnach entsteht dem Ansatz Mod-SDR auch in dieser Hinsicht kein Nachteil gegenüber PaC-SDR. Die Anzahl der Softwaremodule in einem Mod-SDR ist wahrscheinlich gröSSer als bei PaC-SDR, dafür können spezialisierte Module aber auch kompakter mit Maschinenbefehlen dargestellt werden. Generell soll in dieser Arbeit angenommen werden, dass der Programmspeicher keine kritische Ressource des Endgerätes darstellt, sondern nur der Datenspeicher. Desweiteren ist der Programmspeicherbedarf abhängig von vielen, im Allgemeinen unbekannten Details im Laufe des Entwurfsprozesses (*engl.* design flow), wohingegen der Datenspeicherbedarf eng an die Vorgaben eines Mobilfunkstandards gebunden ist. Auch aus diesem Grunde wird im Folgenden neben den Laufzeiten hauptsächlich der Datenspeicher berücksichtigt.

2.2.1 Echtzeitanforderungen im Mobilfunk

Das erklärte Entwurfsziel ist also die physikalische Schicht eines Modularen Software Defined Radios. Selbst unter dieser Einschränkung wird zuweilen davon gesprochen, dass eine unvorhersehbare Mischung aus unterschiedlichsten Diensten vorliegt und deshalb der Bedarf an Rechenleistung zufällig ist [62]. Der Grund liegt möglicherweise in der Auffassung, dass die besagte Mischung an Diensten zum Entwurfszeitpunkt unbekannt ist und ein stochastisches Verhalten in der Software nach sich zieht, das mit demjenigen von Zugriffen vieler Nutzer auf ein gemeinsames Medium vergleichbar ist. In diesem Abschnitt wird gezeigt, warum diese Sichtweise nicht haltbar ist, und zwar für keine wohldefinierte physikalische Schicht eines mobilen Kommunikationsgerätes.

Aus der Theorie der Kommunikation [34, 99] ist bekannt, dass sich zwei Kommunikationspartner zunächst auf einen gemeinsamen Protokollstapel einigen müssen, bevor erfolgreich Nutzdaten ausgetauscht werden können. Für den Mobilfunk bedeutet das: Zur drahtlosen Anbindung an ein Festnetz müssen sich Basisstation und Mobilgerät zunächst auf eine gemeinsame Luftschnittstelle einigen. Zwar können beim Verbindungsaufbau durchaus Parameter, die die Verbindungsqualität betreffen, zwischen den Kommunikationspartnern verhandelt werden, danach aber liegt die Signalverarbeitung auf beiden Seiten der Funkstrecke eindeutig fest, und zwar insbesondere deshalb, weil bei der Funkübertragung harte Echtzeitbedingungen einzuhalten sind. Nicht die ablaufende Software, sondern die in der Luftschnittstelle festgelegten Zeitbezüge der Funksignale (Rahmendauern, Wiederholraten usw.) bestimmen die notwendige Rechenleistung im Endgerät. In einem Software Defined Radio mag die Signalverarbeitung zwar beliebig sein, aber fest für jede Einzelverbindung. In dieser Hinsicht unterscheidet sich ein Mod-SDR nicht von anderen Mobilfunkgeräten.

Leitungsvermittelte Dienste

Im Gegensatz zu Endgeräten in leitungsgebundenen Netzen haben mobile Geräte nicht beliebig viel Datenspeicher zur Verfügung. Weiterhin sind viele Mobilfunkstandards nach wie vor auf leitungsvermittelte Übertragung ausgelegt, hauptsächlich aus Gründen der Kompatibilität zu Sprachdiensten, so z.B. auch die DCHs von UMTS [24].

Ein leitungsvermittelter Dienst stellt klare Qualitätsanforderungen in Form von harten Echtzeitbedingungen über alle Schichten des OSI-Modells hinweg. Wenn

man lediglich die physikalische Schicht betrachtet, so sind Qualitätsanforderungen (*engl.* QoS) in diesem Kontext wohldefiniert. Im Sender müssen die Binärdaten von höheren Schichten (*engl.* PDU) in regelmäßigen Abständen übernommen und nach der Signalverarbeitung innerhalb einer festen Zeitdauer ΔT an den analogen Hochfrequenzteil zur Übertragung weitergegeben werden. Auf ähnliche Weise agiert der Empfänger: Dort werden Abtastwerte vom analogen Hochfrequenzteil in regelmäßigen Abständen übernommen und müssen innerhalb der Zeitdauer ΔT in PDUs für höhere Schichten umgewandelt sein.

Sollte die physikalische Schicht unfähig sein, die betreffenden Daten innerhalb von ΔT zu verarbeiten, würde das Endgerät im günstigsten Falle spektrale Ressourcen verschwenden: Bei Vielfachzugriff nach dem TDMA-Prinzip würden Zeitschlitze ungenutzt bleiben, im Falle des FDMA-Prinzips wäre auf dem zugewiesenen Band keine kontinuierliche Übertragung möglich, und im Falle von CDMA reserviert das Endgerät einen Spreizcode, ohne ihn kontinuierlich zu nutzen. In allen diesen Fehlerfällen würde das mobile Endgerät zum Engpass im Mobilfunksystem werden, und nicht wie üblich der Mobilfunkkanal.

Paketvermittelte Dienste

Da paketvermittelte Dienste für die Betreiber von Mobilfunknetzen immer wichtiger werden, dürfen paketorientierte Signalverarbeitung und Zugriff auf ein gemeinsam genutztes Medium in der Argumentationskette nicht ausgelassen werden.

In diesem Falle denke man sich die Forderung nach Übertragungsqualität derart gelockert, dass lediglich im Mittel ein konstanter Nutzdatenfluß durch die physikalische Schicht des Endgerätes aufrechterhalten werden muss. Wenn dieser mittlere Durchsatz in der gleichen Größenordnung liegen soll wie der einer leitungsvermittelten Übertragung, gelangt man schnell zu derselben Schlußfolgerung wie vorher: Ein gemeinsam genutztes Medium erlaubt zwar Paketverzögerungen, ohne notwendigerweise Frequenzressourcen zu verschwenden, aber der Datenspeicher in einem Mobilgerät ist nach wie vor begrenzt, und allein diese Tatsache begrenzt das Zeitfenster zur Mittelung des Datendurchsatzes. Wenn höhere Protokollschichten die physikalische Schicht übermäßig mit Anfragen zur Paketübertragung belasten, beginnt der Speicher mit unverarbeiteten Daten solange vollzulaufen, bis die Signalverarbeitung komplett zusammenbricht. Wiederum würde das mobile Endgerät anstelle des Mobilfunkkanals zum Engpass im Mobilfunksystem werden.

Aus diesen Überlegungen folgt, dass die Signalverarbeitung für alle Dienstklassen

mehr oder weniger harten Echtzeitbedingungen unterworfen ist, und zwar unabhängig davon, ob die physikalische Schicht in Software realisiert ist oder nicht. Die Schritte zur Verarbeitung von Datenrahmen können von Standard zu Standard völlig verschieden sein, sie sind aber innerhalb eines Standards von Rahmen zu Rahmen identisch. Benachbarte Rahmen sind alle gleich aufgebaut und enthalten lediglich andere Binärinformationen. Das Senden und Empfangen von Rahmen über eine Luftschnittstelle (und damit auch die Rahmenverarbeitung) erfolgt in regelmässigen Zeitabständen ΔT, entweder strikt (leitungsvermittelt) oder im Mittel (paketvermittelt).

2.2.2 Gerichtete Graphen und diskrete Laufzeiten

In einer Umgebung, in der ein Software Defined Radio zum Einsatz kommt, sind also viele verschiedene Luftschnittstellen vorstellbar, die aus unterschiedlichsten nachrichtentechnischen Funktionen bestehen. Unabhängig von dieser Vielfalt sind alle diese Funktionen letztlich aus Softwaremodulen aufgebaut. Allen Modulen ist gemeinsam, dass sie irgendwann vom Betriebssystem zur Abarbeitung auf einem Prozessor P_k vorgesehen sind.

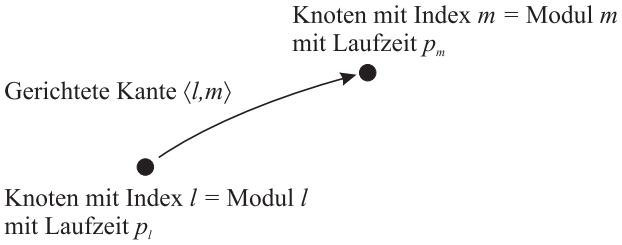


Bild 2.9 Grundlegende Bestandteile gerichteter azyklischer Graphen

Sobald alle von einem Modul m benötigten Eingangsdaten im lokalen Speicher M_k des Prozessors P_k verfügbar sind, kann die Ausführung beginnen. Eingangsdaten werden verarbeitet, die Laufzeit p_m vergeht, und Ausgangsdaten werden erzeugt. Diese Ausgangsdaten sind im Allgemeinen Zwischenergebnisse, die von einem oder mehreren anderen, nachfolgenden Modulen wieder als Eingangsdaten benötigt werden, und das gilt für alle Module zwischen den beiden Dienstzugangspunkten an den Grenzen der physikalischen Schicht. Derartige Datenflüsse können auf abstrakte Art und Weise mit gerichteten azyklischen Graphen (*engl.* DAG) dargestellt werden. Das Bild 2.9 zeigt die grundlegenden Bestandteile solcher Graphen.

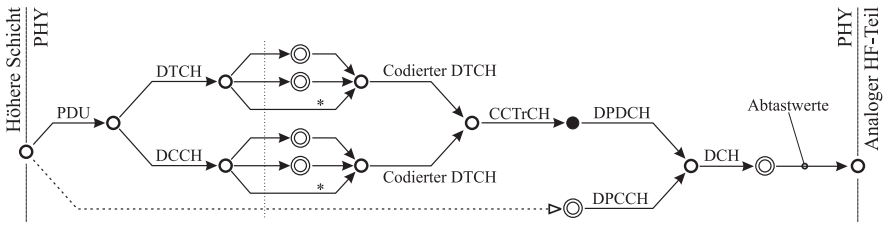


Bild 2.10 Graph mit verschiedenen Knotentypen, Beispiel, Sender eines MMRs

Gefüllte Kreise stehen für eindeutig nummerierte Module. Die Laufzeit $p_m \in \mathbb{N}$ sei zunächst ein ganzzahliges Vielfaches einer bestimmten Zeitauflösung Δt. Zwischenergebnisse fließen zwischen Modulen entlang durchgezogener, gerichteter Kanten $\langle l,m \rangle$. Im Allgemeinen definieren gerichtete Kanten logische Vorgänger-Nachfolger-Beziehungen zwischen Modulen. Kanten, die nicht unmittelbar mit einem Datenfluss verknüpft sind, sollen gestrichelt gezeichnet werden. Die Gesamtheit aller Kanten bildet die Kantenmenge $\mathbb{K}$.

Der Graph in Bild 2.10 entstammt der Darstellung eines Multi Mode Radios [114] auf der obersten Hierarchieebene. Knoten einer hierarchischen Graphendarstellung können nicht nur Module sein, sondern auch Funktionen, die als Doppelkreise dargestellt werden. Knoten mit sehr einfachen Funktionen, also z.B. Multiplexer, Demultiplexer oder SAPs, werden als nicht gefüllte, einfache Kreise dargestellt. Solchen Knoten ist die Laufzeit Null zugeordnet. Der Signalfluss in den hier betrachteten gerichteten Graphen verläuft stets von links nach rechts. Alle Rechnungen gehen von einem einzelnen Knoten, dem Startknoten (dem äußersten Knoten links in Bild 2.10), aus und laufen in einem einzelnen Knoten, dem Zielknoten (dem äußersten Knoten rechts in Bild 2.10), zusammen. Start- und Zielknoten können als Dienstzugangspunkte der Schichten interpretiert werden.

Der Graph in Bild 2.10 ist azyklisch, weil er keine Zyklen enthält. Insbesondere sind in solchen Graphendarstellungen also weder Programmschleifen noch Rekursionen erlaubt. Diese Aussage mag zunächst als eine starke Einschränkung erscheinen, weil viele bekannte nachrichtentechnische Funktionen wie z.B. entscheidungsrückgekoppelte Entzerrer, Träger- und Abtastratenregelung sowie Turbo-Decodierung traditionell Schleifenstrukturen enthalten. Bei genauer Betrachtung stellt sich jedoch heraus, dass die Schleifen lediglich zur kompakten Darstellung von Datenabhängigkeiten zwischen Iterationen dienen, während die Abfolge von Maschinenbefehlen innerhalb einer jeden Iteration eindeutig feststeht. Das muss auch so sein,

denn aus Sicht des Prozessors ist zu jedem Zeitpunkt genau klar, welcher Befehl als nächster ausgeführt und mit welchen Daten gearbeitet wird. In der praktischen Realisierung von Schleifen entkoppeln dabei für gewöhnlich Datenspeicher die zyklischen Abhängigkeiten zwischen den Iterationen, und es ist prinzipiell immer möglich, eine Schleife zu entrollen und rein sequentiell darzustellen. Da die gerichteten Kanten in Graphen dafür vorgesehen sind, logische Abfolgen von Modulen darzustellen, können Schleifen im Rahmen des gewählten Software-Modells sogar nur in nicht-iterativer, rein sequentieller Form auftreten. Im Hinblick auf Schleifen und Rekursionen wird in [62] gezeigt, dass beide aus Stabilitätsgründen eine endliche Laufzeit haben müssen. Demnach können in einem Modularen Software Defined Radio alle sinnvollen Rekursionen genau so wie alle sinnvollen Schleifen in rein sequentieller Form dargestellt werden. Folglich stellt die Wahl gerichteter azyklischer Graphen keine prinzipielle Einschränkung dar für das Ziel, ein SDR-Kommunikationsgerät zu entwerfen.

2.2.3 Stochastisches Laufzeit-Modell

Die Modellierung diskreter Laufzeiten im vorangegangenen Abschnitt entspringt im Wesentlichen der Vorstellung von der Taktung digitaler Rechensysteme. Die Zeitauflösung kann daher im Prinzip so hoch sein, dass Δt genau einem Prozessortakt entspricht. Bestehen aber selbst die kleinsten Softwaremodule aus relativ vielen Maschinenbefehlen, so wird die Quantisierung der Zeitachse dadurch sehr fein, und die Laufzeiten können als quasi-kontinuierlich angesehen werden. Für die Simulation im Rechner wird man daher für die Diskretisierung von Laufzeiten eine weit geringere Auflösung als den Prozessortakt wählen oder gleich mit reellwertigen Laufzeiten arbeiten.

Bereits mehrfach wurde darauf hingewiesen, dass Laufzeiten in einer SDR-Umgebung beliebig aber fest sind, und das nicht nur wegen der wachsenden Zahl an verfügbaren Mobilfunkstandards: Die Kombination der technischen Fähigkeiten von Netzbetreibern, Programmierteams, Geräteherstellern und Softwareanbietern, die Existenz vielfältiger Produktvarianten und verschiedenster Prozessoren, viele verschiedene Anwendungssituationen und letztlich das Verhalten einer großen Menge unabhängiger Nutzer trägt nur dazu bei, dass das Betriebssystem der physikalischen Schicht eines Modularen Software Defined Radios potentiell mit einer ungeheuren Vielfalt von Softwaremodulen und Laufzeiten konfrontiert wird. Ein entscheidender Beitrag der vorliegenden Arbeit ist es, diese potentielle Vielfalt als Zufallspro-

zess zu verstehen. Laufzeiten werden zu Zufallsvariablen, und konkrete Realisierungen von Mobilfunkstandards beinhalten nichts weiter als Realisierungen dieser Zufallsvariablen, die über Graphen in logischen Beziehungen zueinander stehen.

Jede Modullaufzeit ist eine von anderen Modulen unabhängige, reellwertige Zufallsvariable, und für alle Laufzeiten soll die gleiche Wahrscheinlichkeitsdichte $f_P(p)$ verwendet werden. Diese Dichte ist nur in einem Intervall $[p_{min}; p_{max}]$ von Null verschieden, also rechteckig gefenstert. Bild 2.11a zeigt ein Beispiel aus der Familie von Exponentialdichten mit dem Parameter λ, wobei in diesem Bild $\lambda > 0$.

$$\begin{aligned} f_P(p) &= \begin{cases} r_e \cdot e^{-\lambda(p-p_{min})} & ; p_{min} \leq p \leq p_{max} \\ 0 & ; \text{sonst} \end{cases} \\ r_e &= \frac{\lambda}{1 - e^{-\lambda(p_{max}-p_{min})}} \end{aligned} \tag{2.1}$$

Dabei ist $r_e \in \mathbb{R}^+$ diejenige Konstante, die das Integral von p_{min} bis p_{max} über $f_P(p)$ auf Eins normiert.

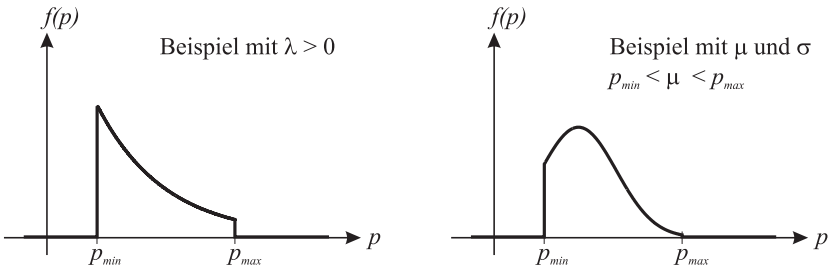
Neben einparametrischen Dichten wie den Exponentialdichten aus Gleichung (2.1) kann man auch Familien einer zweiparametrischen Dichte wählen, beispielsweise gefensterte Gaußdichten (siehe Bild 2.11b) mit den beiden Parametern μ und σ:

$$\begin{aligned} f_P(p) &= \begin{cases} r_g \cdot e^{-\frac{(p-\mu)^2}{2\sigma^2}} & ; p_{min} \leq p \leq p_{max} \\ 0 & ; \text{sonst} \end{cases} \\ r_g &= 2 \cdot \left(\text{erf}\left[\frac{p_{max} - \mu}{\sqrt{2}\sigma}\right] - \text{erf}\left[\frac{p_{min} - \mu}{\sqrt{2}\sigma}\right] \right)^{-1} \end{aligned} \tag{2.2}$$

$$\text{erf}\,[\gamma] = 2 \cdot \int_0^{\gamma} \frac{1}{\sqrt{\pi}} e^{-x^2} dx \tag{2.3}$$

Die Konstante $r_g \in \mathbb{R}^+$ ist so gewählt, dass sie das Integral von p_{min} bis p_{max} über $f_P(p)$ auf Eins normiert.

Zu einer Simulation von Mod-SDR gehört bisher die Angabe eines Graphen, der aus Funktionen und Modulen zusammengesetzt ist. Eine solche hierarchische Darstellung ist zwar gut geeignet, um Beziehungen zwischen nachrichtentechnischen Funktionen auf einer bestimmten Ebene kompakt darzustellen, aber völlig ungeeignet zur Bearbeitung durch das Betriebssystem eines Mod-SDRs. Ausgehend von



(a) Exponentialdichte **(b)** Gaußdichte

Bild 2.11 Gefensterte Dichtefunktionen, Beispiele

einer beliebigen Hierarchieebene können alle Funktionen rekursiv durch die beschreibenden Graphen niedrigerer Ebenen ersetzt werden, und das ist so lange möglich, bis der Graph nur noch Module als Knoten besitzt. Dieser Sonderfall soll im Folgenden flacher Graph genannt werden. Er stellt die Signalverarbeitungssoftware immer noch abstrakt dar, aber so nah wie möglich am Ablauf von Maschinenbefehlen, da Module ja die kleinsten Einheiten von ausführbaren Maschinenbefehlen sind, die vom Betriebssystem nicht unterbrochen werden können. Die Eingangsdatensätze für Systemsimulationen bestehen also aus Realisierungen solcher flacher Graphen, wobei jede einzelne Modullaufzeit die Realisierung eines Zufallsexperimentes nach Maßgabe einer Dichtefunktion ist.

Bei der Steuerung von beliebigen Entwurfsprozessen, die komplizierte Gesamtaufgaben in einzelne Funktionen und schließlich in Module zerlegen, taucht immer wieder die Frage nach einem methodischen Vorgehen auf, das dafür sorgt, dass die Lösung der Gesamtaufgabe besonders gut gelingt oder die Module für die Lösung weiterer Aufgaben möglichst gut wiederzuverwenden sind. Im Hinblick auf den Entwurf von Mod-SDRs liefert die Angabe einer Dichtefunktion keine unmittelbare Antwort auf diese Frage, jedoch kann untersucht werden, welche Dichte besonders gute Eigenschaften nach sich zieht, z.B. schnelle Berechnung von Funkrahmen oder gute Vorhersagbarkeit der Berechnungsgeschwindigkeit in einer SDR-Umgebung. Wie eine bestimmte Dichtefunktion zu erreichen ist, wird nicht gezeigt, sondern wie sich Mod-SDRs verhalten, wenn eine bestimmte Dichte der Modullaufzeiten vorliegt. Die Untersuchungen unter Annahme bestimmter Dichten beziehen sich also nicht auf eine Zerlegungsmethode, sondern auf das mögliche Ziel einer solchen Methode.

2.2.4 Erweitertes Ressourcen-Laufzeit-Modell

Der vorangegangene Abschnitt beschreibt sicherlich ein einfaches Modell für Software in einem Modularen Software Defined Radio, weil die Größe des Datenspeichers, den ein Modul zur Speicherung seiner Ausgangsdaten benötigt, noch nicht berücksichtigt wird. Desweiteren ist bereits im Abschnitt 2.1 beschrieben worden, dass der Austausch von Ergebnissen zwischen verschiedenen Prozessoren mit endlicher Geschwindigkeit über einen Bus erfolgt und damit ebenso Zeit kostet wie das Berechnen von Ergebnissen selbst. Auf der einen Seite sind Datenspeicherbedarf und Buskommunikation also deterministisch, auf der anderen Seite sollen die Modullaufzeiten Realisierungen von Zufallsexperimenten sein. Um beide Aspekte in einem erweiterten Software-Modell zu berücksichtigen, wird hier folgende lineare Abhängigkeit zwischen der Laufzeit p_m und dem Datenspeicherbedarf r_m eingeführt:

$$p_m = \alpha \cdot c \cdot r_m \tag{2.4}$$

Dabei ist α ein absolutes Maß für die Prozessorgeschwindigkeit und c die Realisierung einer reellwertigen Zufallsvariablen C, die mit einer Wahrscheinlichkeitsdichte $f_C(c)$ vollständig beschrieben ist. Als Dichte $f_C(c)$ kommen Funktionen der gleichen Gestalt wie beim einfachen stochastischen Laufzeit-Modell in Betracht, da die Zufallsvariable Laufzeit nur eine mit den reellen Faktoren α und r_m skalierte Version der Zufallsvariablen C ist. Formal ist das α in Gleichung (2.4) eine Proportionalitätskonstante, die zwischen dem Speicherbedarf r_m und der Modullaufzeit p_m vermittelt. Die Konstante α hat folglich die Einheit $[\alpha] = s/bit$, wohingegen die Realisierung c der Zufallsvariablen C einheitenlos ist. Die Idee der linearen Abhängigkeit stammt im Wesentlichen von der Vorstellung, dass in vielen Fällen um so mehr Zeit für Berechnungen benötigt wird, je mehr Ausgangsdaten ein Algorithmus erzeugen soll. Die Realisierung c der Zufallsvariablen C sorgt nach Maßgabe von $f_C(c)$ dafür, dass dieser Zusammenhang zwischen p_m und r_m nicht strikt linear, also deterministisch ist, sondern p_m wie in Abschnitt 2.2.3 eine Zufallsvariable bleibt, obwohl der Speicherbedarf r_m deterministisch ist.

Eine Algorithmenklasse, von der man zunächst annehmen kann, dass sie nicht vollständig von einem linearen Modell erfaßt wird, ist die der iterativen Algorithmen (z.B. adaptive Filter [36]). Abhängig von den verarbeiteten Signalen ist schnelle oder langsame Konvergenz möglich. Beispielsweise könnte ein Toleranzmaß, das als Abbruchkriterium dient, im Einzelfall sehr viel früher unterschritten werden als

im Mittel, obwohl zur Berechnung eines Datenrahmens die gleichen Softwaremodule ausgeführt werden. Allerdings besteht bei iterativen Algorithmen auch immer die Gefahr der langsamen Konvergenz. Daher ist neben einem datenabhängigen Abbruchkriterium stets ein unabhängiges Kriterium vorzusehen, das den Algorithmus nach einer bestimmten maximalen Anzahl von Iterationen kontrolliert beendet. Nur so ist eine Maximallaufzeit zu garantieren und damit Stabilität der Software nach [62]. Das hier vorgestellte lineare Ressourcen-Laufzeit-Modell zieht lediglich die garantierte Maximallaufzeit in Betracht. Diese hängt linear von der der vorgesehenen maximalen Anzahl von Iterationen ab. Die Laufzeit für jede Iteration wiederum hängt von der verarbeiteten Datenmenge ab, und zwar im Prinzip nicht anders als das bei jedem nicht-iterativen Algorithmus der Fall ist. Zur Ausführung eines iterativ arbeitenden Moduls wird die Ablaufplanung des Betriebssystems einen verfügbaren Prozessor genau für die Maximallaufzeit reservieren. Sollte das Modul vorzeitig zum Ausführungeende kommen, ist die verbleibende Zeit bis zur garantierten Maximallaufzeit nicht anderweitig nutzbar. Die Ablaufplanung ist demnach statisch, d.h. sie wird vor der Aufnahme des Datenaustausches über eine Luftschnittstelle einmal festgelegt und ändert sich während der Übertragung von Daten nicht. Alle aufeinanderfolgenden Datenrahmen werden auf gleiche Weise berechnet und tragen lediglich andere Nutzerinformationen. Obwohl die Nutzung der Prozessoren bei iterativen Algorithmen also niemals besser wird als im (schlechtesten) Extremfall, ist der Einsatz dieser Algorithmenklasse in einem Modularen Software Defined Radio durchaus möglich.

In dieser Arbeit wird davon ausgegangen, dass die Software im Sinne eines Mobilfunkstandards fehlerfrei und stabil programmiert ist. Weiterhin wird angenommen, dass die Abhängigkeit der Laufzeit von den verarbeiteten Datenmenge im Wesentlichen linear ist. Alle Abweichungen von dem strikt linearen Zusammenhang werden von der Zufallsvariablen C modelliert. Ihr Wert c kann also so interpretiert werden, dass er nicht nur den Zufall abdeckt, der sich aus den Eigenschaften der SDR-Umgebung ergibt, sondern auch alle Unzulänglichkeiten eines strikt linearen Modells.

Bei der Einführung des einfachen Laufzeitmodells in Abschnitt 2.2.3 wird strikt unterschieden zwischen Modulen und Funktionen. Funktionen müssen erst vollständig in Module aufgelöst werden, bevor der so entstandene flache Graph an das Betriebssystem übergeben wird, das seinerseits die Abbildung der Module auf Hardware-Bausteine vollzieht. Auf die Forderung nach Erzeugung eines flachen Graphen soll im erweiterten Ressourcen-Laufzeit-Modell aber verzichtet werden, und zwar aus folgendem Grund: Wenn das Laufzeitmodell so allgemein wie mög-

lich gehalten werden soll, so besteht überhaupt keine Möglichkeit, Funktionen in Module zu zerlegen, also den flachen Graphen herzustellen, ohne Einschränkungen bezüglich der inneren Struktur aller Funktionen zu machen. Einschränkungen zu machen ist aber genau **nicht** das Ziel der Modellbildung für Mod-SDR. Außerdem kann auch der Fall eintreten, dass ganze Funktionen selbst im konkreten Anwendungsfall nicht weiter zerlegbar sind. Im Kapitel 1 ist bereits geschildert worden, dass nachrichtentechnische Funktionen von einem Hersteller zuweilen nicht selbst entwickelt, sondern von spezialisierten Anbietern in Form von **monolithischen** Funktionskernen (*engl.* IP cores) eingekauft und in ein Modulares Software Defined Radio integriert werden könnten. Alle Mobilfunkstandards (existierende wie zukünftige) sind stets nach der Beschreibung nachrichtentechnisch relevanter **Funktionen** geordnet, und man kann annehmen, dass sich auch modulare Lösungen für Sender und Empfänger stets an diesen Funktionen orientieren werden. Schnittstellen zur Übergabe von Zwischenergebnissen treten stets zwischen Funktionen auf.

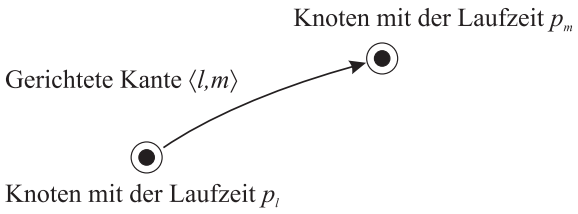


Bild 2.12 Knotendarstellung für das erweiterte Ressourcen-Laufzeit-Modell

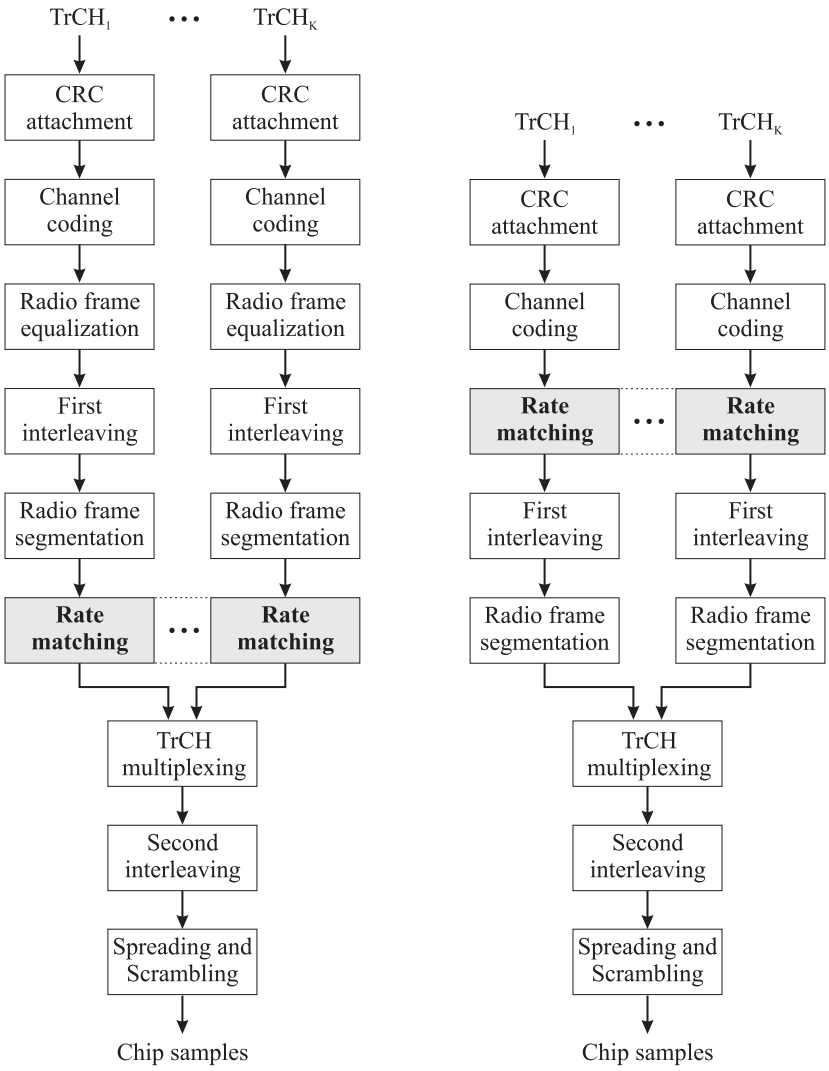
Das graphische Symbol für diejenigen Knoten eines Graphen, auf die in den Systemsimulationen das erweiterte Ressourcen-Laufzeit-Modell angewendet wird, ist ein Doppelkreis, dessen innerer Kreis gefüllt ist (siehe Bild 2.12). Mit der Gleichung (2.4) liegt ein Zusammenhang vor, der abhängig von den deterministischen Forderungen eines Standards nach Speicher (aber unabhängig von einer konkreten Lösung in Software oder in Hardware) zufällige Laufzeiten generiert, und das ohne Kenntnis der inneren Struktur von Funktionen. An die Stelle der Modullaufzeit tritt im erweiterten Ressourcen-Laufzeit-Modell der allgemeinere Begriff der Knotenlaufzeit.

2.2.5 Familien von UMTS-Graphen

Simulationen, die eines der oben diskutierten Software-Modelle nutzen, bestimmen zunächst nur die Laufzeiten von Knoten aus Zufallsexperimenten. Damit läßt sich bereits ein wichtiger Aspekt der SDR-Umgebung simulieren, denn das Betriebssystem eines Mod-SDRs muss in einer solchen Umgebung mit sehr vielen Laufzeitkombinationen möglichst gut zurechtkommen, d.h. eine möglichst gute Abbildung der Software auf die Hardware-Komponenten finden, so dass die Nutzung der Prozessoren optimal wird. Allerdings bleibt die Struktur des Graphen bislang unverändert.

Ein erster Schritt, den Raum der untersuchten Graphen zu erweitern, besteht darin, alle diejenigen Graphen zu erzeugen, die in UMTS möglich sind [24], und zwar abhängig von den Bedingungen, die der Nutzer festlegen kann: Die Anzahl der leitungsvermittelten Kanäle (DCHs) und für jeden Kanal die Nutzdatenrate, das Transmission Time Interval (TTI), den äußeren (CRC) und inneren Code (keine weitere Kanalcodierung, Einsatz eines Faltungscodes der Raten $R_c = 1/2$ oder $R_c = 1/3$ oder Einsatz eines Turbo-Codes der Rate $R_c = 1/3$, gemäß [26]) sowie den zulässigen Grad der Punktierung (*engl.* puncturing limit) und ein Punktierungsgewicht (*engl.* rate matching attribute). Außerdem muss bestimmt werden, ob der Uplink oder der Downlink simuliert werden soll, weil die Signalverarbeitung im UTRA FDD Downlink einen Sonderfall darstellt: Die Zerlegung in Rahmensegmente (*engl.* frame segmentation) findet beim FDD Downlink **nach** der Ratenanpassung statt (Bild 2.13b), in allen anderen Modi jedoch vor der Ratenanpassung (vgl. Bild 2.13a).

Die Ratenanpassung stellt in UTRA die wichtigste Funktion zwischen Nutzeranforderungen und den technischen Fähigkeiten der Luftschnittstelle dar. Einerseits verspricht UMTS dem Nutzer die Möglichkeit, auf mehreren unabhängigen logischen Transportkanälen (TrCHs) beliebige Datenraten zu übertragen, andererseits stellt die Luftschnittstelle auf einem physikalischen Datenkanal (DPDCH), der durch einen bestimmten OVSF Code (*engl.* channelization code) gekennzeichnet ist, nur einen Satz diskreter Datenraten zur Verfügung. Die physikalischen Datenraten sind diskret, weil in UTRA die Chiprate unabhängig von der Datenrate konstant bleibt. Die Ratenanpassung ist nicht nur Vermittler zwischen den Datenraten, sondern bereitet auch die Zusammenführung von Bitblöcken aus unterschiedlichen logischen Kanälen vor. Dazu müssen in jeder Ratenanpassungsfunktion, die die Daten eines einzelnen Kanals verarbeitet, die Parameter **aller** Kanäle bekannt sein. Die Zusam-



(a) TDD, FDD Uplink **(b)** Sonderfall FDD Downlink

Bild 2.13 UTRA-Signalverarbeitung für Transportkanäle im Sender, Quelle: [26]

menführung von Bitinformationen aus verschiedenen logischen Kanälen zu einem CCTrCH findet im Sender dann unmittelbar nach der Ratenanpassung statt. Nur im Sonderfall UTRA FDD Downlink werden die angepassten Datenblöcke vor der Zusammenführung noch einem Interleaving unterworfen und in Segmente zerlegt.

Durch das vorliegende Modell für UMTS-Graphen werden bereits viele der möglichen logischen Graphenstrukturen zur Erzeugung von DCHs erfasst, allerdings gibt es auch einige praktische Einschränkungen: Es wird von kontinuierlicher Übertragung mit 15 Zeitschlitzen pro Rahmen ausgegangen, und es ist keine Multicode-Übertragung [55] vorgesehen, d.h. ein Nutzer erhält nur einen OVSF Code und belegt damit auch nur einen physikalischen Datenkanal. Eine weitere Einschränkung betrifft die zur Übertragung vorgesehenen Datenblöcke (*engl.* transport blocks) und den Zeitbezug der TTIs. Für die Simulation wird angenommen, dass ein kontinuierlicher Datenstrom gesendet werden soll, d.h. in jedem TTI steht genau ein Datenblock zur Verarbeitung an. Die gesamte Signalverarbeitung orientiert sich am maximalen TTI, das der Nutzer anfordert. Das Bild 2.14 zeigt dies anhand eines Beispiels mit drei logischen Kanälen und TTI={10 ms, 20 ms, 40 ms}. Hier ist der Sender eines Mod-SDRs dargestellt, der Daten über einen UTRA FDD Uplink verschickt.

Zwischen dem SAP der physikalischen Schicht und der Ratenanpassung ist die logische Abfolge der nachrichtentechnischen Funktionen zunächst für jeden Kanal identisch. Die unterschiedlichen Funktionen entlang einer Signalverarbeitungskette arbeiten lediglich auf anderen Datenblöcken. Es fällt sofort auf, dass es bis zur Ratenanpassung für drei logische Kanäle nicht drei Ketten gibt, sondern sieben. Nach der Ratenanpassung sorgen die Zerlegung in Rahmensegmente und Multiplexer dafür, dass es vier Signalverarbeitungsketten gibt, bevor die Chipwerte an den analogen Hochfrequenzteil des Senders zur Übertragung übergeben werden. Der Grund für diese Struktur liegt in der Ausrichtung der Signalverarbeitung am maximalen TTI des Nutzers. Ein TTI von 40 ms bedeutet unmittelbar, dass Daten aus dem zugehörigen Transportblock bei der Übertragung auf genau 4 Funkrahmen verteilt werden, die stets die Zeitdauer 10 ms haben. Entsprechendes gilt für kleinere TTIs. Auch ein TTI = 80 ms ist möglich, so dass die Daten aus einem Transportblock auf genau 8 Rahmen verteilt werden. Im Beispiel ergibt sich die Anzahl der Signalverarbeitungsketten rechts von der Ratenanpassung also **zwangsläufig** aus der Forderung nach max(TTI) = 40 ms und aus den Eigenschaften der Luftschnittstelle. Die Strukturen links der Ratenanpassung müssen so angepasst werden, dass sie die Regeln der Luftschnittstelle erfüllen, und das heißt, es kommen zwei Kanäle

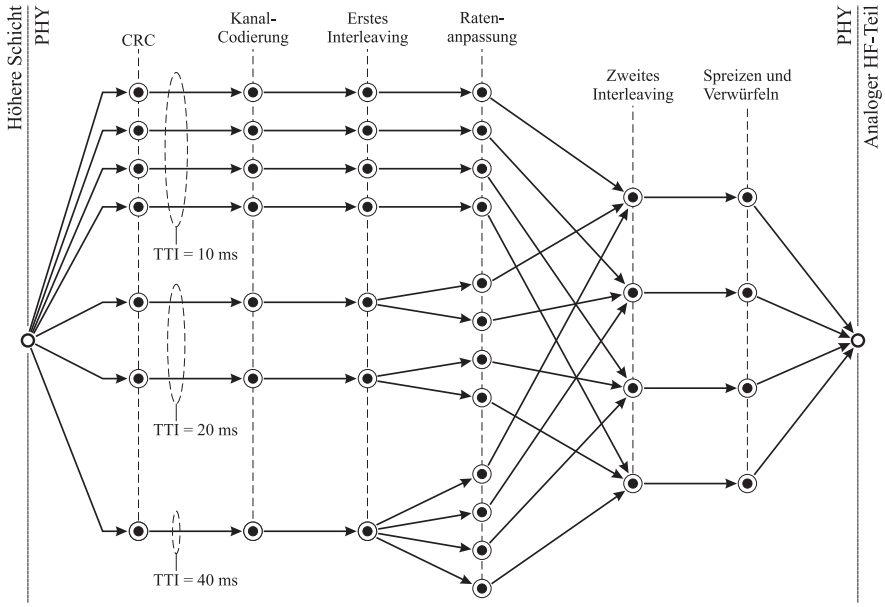


Bild 2.14 Graph für UTRA FDD Uplink, Signalverarbeitung im Sender

mit TTI = 20 ms und vier Kanäle mit TTI = 10 ms auf insgesamt 40 ms Funkdauer. Aus dieser Überlegung wird klar, warum von den logischen Kanälen mit kleinerem TTI als 40 ms links von der Ratenanpassung identische Kopien erzeugt werden. Diese Identität bezieht sich auf die Art der Funktionen, nicht auf den Inhalt oder Umfang der verarbeiteten Datenblöcke. Die Tatsache, dass alle sieben Pfade von einem einzigen Knoten (dem SAP der physikalischen Schicht) ausgehen, zeigt an, dass **alle** Eingangsdaten, die letztlich auf 40 ms Funkdauer abgebildet werden, zu Beginn der Signalverarbeitung verfügbar sind: Vier Transportblöcke des ersten Kanals, zwei Transportblöcke des zweiten Kanals und ein Transportblock des dritten Kanals. Der Ablaufplan für alle Funktionen des Graphen aus Bild 2.14 ist statisch und wird sich alle 40ms periodisch wiederholen.

In der Simulation ist zu beachten, dass diejenigen Funktionen, die in Bild 2.14 untereinander angeordnet sind, von gleicher Art sind. Beispielsweise sorgen alle Knoten der äußeren Kanalcodierung dafür, dass Transportblöcke mit einem CRC versehen werden. Abhängig vom logischen Kanal kann eine Prüfsumme aus der

Menge {CRC-8, CRC-12, CRC-16, CRC-24} gewählt werden. Für mehrere Kopien eines logischen Kanals werden die Laufzeiten der Funktionen notwendigerweise identisch sein, da das Betriebsystem lediglich mehrere Instanzen ein und derselben Softwarerealisierung erzeugt, die auf den verschiedenen Datenblöcken arbeiten. Weiterhin lassen sich viele Funktionen generisch programmieren und nach dem Ansatz des PaC-SDR [43, 114] in allen Signalverarbeitungsketten betreiben, lediglich unterschiedlich parametrisiert. In diesem Fall hängen nicht mehr die einzelnen Laufzeiten, sondern die Laufzeit der verschiedenen Arten von Funktionen vom Zufall ab. Dies wird in Simulationen berücksichtigt, indem für Gruppen von Funktionen derselben Art nur eine einzige Realisierung c der Zufallsvariablen C in Gleichung (2.4) eingesetzt wird.

UMTS eröffnet allein innerhalb eines Mobilfunkstandards zahllose Möglichkeiten des Betriebs der physikalischen Schicht, abhängig von allen genannten Parametern, die die logischen Transportkanäle beschreiben. Eine Familie von UMTS-Graphen ergibt sich, wenn einer dieser Paramter variabel gemacht wird und einen bestimmten Wertebereich durchläuft. Diese Familien von UMTS-Graphen sind zwar immer noch nicht allgemein, erlauben es aber im Prinzip, verschiedene Graphen mit unterschiedlicher Struktur miteinander zu vergleichen. Die Einschränkung auf einen bestimmten Standard erlaubt es auch, mehr Details in das Graphenmodell zu integrieren, so z.B. die oben erwähnten Gruppen von Funktionen einer bestimmten Art. Insgesamt ist das UMTS-Graphenmodell nur ein Schritt auf dem Weg zur Verallgemeinerung von Graphen, die in einem Modularen Software Defined Radio auftreten können.

2.2.6 Zufallsgraphen

Dem allgemeinsten Modell von Mod-SDR liegen Zufallsgraphen zugrunde, allerdings nur solche, die die besonderen Eigenschaften aus der allgemeinen Vorstellung von einem Software Defined Radio erfassen. Für die Systemsimulationen werden unregelmäßige, zusammenhängende, gerichtete, azyklische Graphen mit einer konstanten Anzahl M von Knoten erzeugt. Der Prozess zur Erzeugung der Zufallsgraphen beginnt mit einer Kette von zwei Knoten zwischen den Dienstzugangspunkten der physikalischen Schicht und des analogen Hochfrequenzteils. Weitere Knoten werden dann dem jeweils existierenden Graphen iterativ hinzugefügt. Dafür wird je ein Vorgänger und ein Nachfolger zufällig ausgewählt, und zwar mit gleicher Wahrscheinlichkeit unter den bereits bestehenden Knoten. Sollte ein neuer Knoten

die Kante zwischen zwei existierenden Nachbarknoten überbrücken, dann wird die überbrückte Kante mit der Wahrscheinlichkeit 0.5 entfernt. Damit der Graph insgesamt zusammenhängend bleibt, wird einfach der SAP der physikalischen Schicht von der Auswahlprozedur für Knoten ausgenommen. Die Eigenschaft der Azyklizität muss hingegen in jedem Schritt des erzeugenden Prozesses sichergestellt werden.

Obwohl mit diesem Software-Modell nicht nur die Laufzeiten der Knoten sondern auch die Struktur der Graphen einem Zufallsexperiment entstammen, müssen hier Annahmen getroffen werden, die die Allgemeingültigkeit geringfügig einschränken: Das Verhältnis von Ausgangs- zu Eingangsdatenblocklängen aller Knoten sei konstant 1.0, und Datenblöcke werden in voller Länge an alle Nachfolger eines Knotens weitergegeben. Bei den Simulationen, die diese Zufallsgraphen nutzen, kommt es also weniger auf die Laufzeitbeziehungen der Knoten untereinander an, sondern auf die Struktureigenschaften, die aus den unterschiedlichsten Vorgänger-Nachfolger-Beziehungen zwischen Knoten hervorgehen.

2.3 Zusammenfassung

Im Prinzip soll es mit Modularem Software Defined Radio möglich sein, dass irgendjemand aus der sehr großen, weltweit verfügbaren Bibliothek von Signalverarbeitungssoftware einen Satz von Modulen herausgreift, nachrichtentechnische Funktionen daraus konstruiert und so einen neuen Mobilfunkstandard schafft. Irgendein Modul m kann an irgendeiner Stelle eines Graphen auftreten, mit einem beliebigen Nachfolgemodul verknüpft werden und Daten austauschen, vorausgesetzt die Module sind im Hinblick auf die Datentypen kompatibel. Natürlich soll das Betriebssystem auch mit einem beliebigen neu konstruierten Standard zurechtkommen. Die genaue Kenntnis der Algorithmen zur Signalverarbeitung, die hinter den einzelnen Modulen stecken, ist dabei nicht notwendig, sondern in einem ersten Schritt lediglich die Struktur des Graphen und die Laufzeiten der Module. Im erweiterten Ressourcen-Laufzeit-Modell kommt noch der Datenspeicherbedarf und die Verallgemeinerung von Modullaufzeiten auf Knotenlaufzeiten hinzu.

Diese ganz allgemeine Modellvorstellung ist neuartig für den Bereich Software Defined Radio. Es kommt nicht mehr darauf an, eine einzelne nachrichtentechnische Funktion herauszugreifen, algorithmische Details zu analysieren und isoliert von anderen Funktionen über eine individuelle Laufzeit oder die Realisierung auf ei-

nem bestimmten Prozessor zu diskutieren. Das stochastische Laufzeit-Modell im Zusammenspiel mit Graphen ermöglicht vielmehr eine systemtheoretische Sichtweise, die darauf abzielt, systematische Eigenschaften von Software Defined Radio zu identifizieren und die Wahrscheinlichkeit für ein gutes Zusammenspiel von Software, Betriebssystem und Hardware zu maximieren.

Die Grundidee, die darin besteht, die Reaktion deterministischer Vorschriften eines Betriebssystems auf zufällige Graphenkonstellationen zu untersuchen, mag zunächst zweifelhaft erscheinen. Allerdings zeigt ein Vergleich mit einem etablierten Simulationsverfahren, dass diese Grundidee als Übertragung der klassischen Systemsimulation in der Nachrichtentechnik auf den Bereich Modulares Software Defined Radio interpretiert werden kann: Ganz allgemein werden Zufallsmodelle in der Nachrichtentechnik immer genau dann genutzt, wenn in einer Umgebung, in der das Nachrichtensystem eingesetzt werden soll, Effekte in solcher Vielfalt auftreten, dass sie nicht mehr im einzelnen zu erfassen sind, sondern nur noch auf integrale Weise, also z.B. durch Angabe einer Dichtefunktion. Als Beispiel kann hier das Verhalten vieler unabhängiger Nutzer in einem leitungsgebundenen Rechnernetz dienen. Netzsimulationen mit statistischen Verkehrsmodellen gehören zu den Standardverfahren der Nachrichtentechnik, die darauf abzielen, das Verhalten des Netzes beim Betrieb mit deterministischen Regeln zu verstehen. Im Mobilfunk treten Zufallsmodelle auf, weil die Eigenschaften des Funkkanals nicht für alle möglichen Kommunikationssituationen aller mobilen Nutzer detailliert erfasst werden können. Der Ausweg besteht in Systemsimulationen mit den üblichen zeitvarianten, frequenzselektiven WSSUS-Kanalmodellen. Die Ergebnisse sind stets **mittlere Bitfehlerraten**, abhängig vom **mittleren SNR** pro Bit. Über die Qualität der Übertragung für eine beliebige konkrete Einzelrealisierung eines Übertragungskanals wird dadurch nichts ausgesagt. Nichts desto trotz sind die Regeln, mit denen der Empfänger auf eine aktuelle Realisierung des Kanals reagiert, z.B. in Form eines Entzerrers, **deterministisch**. In gleicher Weise sind die Regeln der Partitionierung und der Ablaufplanung in einem Mod-SDR deterministisch, während die Graphen Realisierungen aus Zufallsexperimenten sind. Die Entwurfsqualität eines Mod-SDRs kann also mit Hilfe der geschilderten Software-Modelle im Prinzip in der gleichen Weise beurteilt werden wie die Qualität eines Entzererentwurfs in der klassischen Nachrichtentechnik mit den üblichen Kanalmodellen.

An dieser Stelle sei nochmals ausdrücklich darauf hingewiesen, dass stochastische Laufzeit-Modelle nichts mit zufälligem Bedarf an Rechenleistung zu tun haben. In den gewählten Software-Modellen wird die Art und Weise, eine große Population

von Mod-SDR-Geräten zu bauen und zu betreiben, als ein Zufallsprozess interpretiert. In jedem Einzelfall aber, also für jede Realisierung eines Mobilfunkstandards, muss das Betriebssystem mit deterministischen Laufzeiten und Graphen umgehen. Die bereits am Ende von Kapitel 1 erwähnten Aufgaben der Partitionierung und der Ablaufplanung für Softwaremodule beziehen sich in jedem Einzelfall auf einen deterministischen Graphen, und so ist auch der Bedarf an Rechenleistung in jedem Einzelfall deterministisch.

Sicherlich schließen zufällige Kombinationen von Knotenlaufzeiten und zufällige Kanten zwischen Funktionen mehr Fälle ein als nur die nachrichtentechnisch sinnvollen. Mit einem stochastischen Ressourcen-Laufzeit-Modell wird der Umfang der Entwurfsaufgabe also einerseits vergrößert. Andererseits wird eine kompakte Beschreibung des Modells für Simulationen ermöglicht. Sollte es gelingen, unter diesen Bedingungen allgemeine Richtlinien für den Entwurf von Modularen Software Defined Radios herzuleiten, so gelten diese Richtlinien mit hoher Wahrscheinlichkeit auch für diejenigen Realisierungen, auf die das Betriebssystem tatsächlich in der Mobilfunkumgebung stößt. Auch in dieser Hinsicht steht die Simulation von Mod-SDR-Software der klassischen Systemsimulation in der Nachrichtentechnik in nichts nach.

Die Gesamtheit der Softwaremodule und ihrer logischen Abhängigkeiten soll im Folgenden zusammengefasst werden unter dem Begriff der ”logischen Strukturen” eines Mod-SDRs. Unter dem Begriff der ”physikalischen Strukturen” hingegen soll die Gesamtheit aller Hardware-Bausteine und ihrer physikalischen Verbindungen verstanden werden. In den beiden Abschnitten über Hardware und Software sind Modelle beschrieben, die stets den allgemeinen Vorstellungen von einem Modularen Software Defined Radio entsprechen, zu den Modellen des SDRF und zur SCA kompatibel sind und jeweils dort mehr Details einfügen, wo die Systemsimulation eine bestimmte Fragestellung klären soll. Diejenigen deterministischen Verfahren, die das Betriebssystem eines Mod-SDRs benötigt, um beliebige Software auf Hardware abzubilden, also die Algorithmen der Partitionierung und der Ablaufplanung, sind Gegenstand der in der vorliegenden Arbeit dargestellten Untersuchungen.

3 Die Entwurfsaufgabe

Das Betriebssystem der physikalischen Schicht eines Modularen Software Defined Radios muss darauf ausgelegt sein, Anfragen nach der Realisierung einer beliebigen Luftschnittstelle entgegenzunehmen und eine Entscheidung über die Machbarkeit auf der vorhandenen Hardware zu fällen. Die Anfragen können vom Nutzer des Engerätes selbst oder ohne bewussten Nutzereingriff von einer auf dem Gerät laufenden Anwendung ausgehen. Vorstellbar ist in diesem Rahmen auch, dass ein Netzbetreiber per Funk Anfragen an das SDR-Gerät richtet, die den Zustand des Multiprozessorsystems (z.B. verfügbare Prozessortypen, struktureller Aufbau, freier Speicher, oder Auslastung der Prozessoren) betreffen, um Kenntnis über die Leistungsfähigkeit des Engerätes zu erhalten und um dem Nutzer entsprechend seiner verbleibenden Rechenleistung Dienste (basierend auf einer oder mehreren Luftschnittstellen) gezielt anbieten zu können. In dieser Arbeit geht es jedoch allein um die grundlegenden Abbildungsprinzipien von logischen Strukturen einer **einzigen** Luftschnittstelle auf die physikalischen Strukturen eines Endgerätes.

Eine Anfrage an das Betriebssystem umfasst mindestens eine konkrete Beschreibung der Luftschnittstelle in Form eines Graphen, der alle benötigten Module und die Abhängigkeiten zwischen Modulen enthält. Weiterhin müssen alle Module mit denjenigen Parametern versorgt sein, die die Nutzeranforderungen detailliert erfassen. Dann können Speicherbedarf und Modullaufzeiten für jedes einzelne Modul auf den vorhandenen Prozessoren ausgetestet werden. Aus der Definition der Luftschnittstelle ist eine bestimmte Rahmendauer bekannt und damit auch die Echtzeitperiode ΔT. Die Entwurfsaufgabe besteht nun darin, die Softwaremodule so auf die Hardware abzubilden, dass ΔT eingehalten wird. Sollte das Betriebssystem feststellen, dass das nicht machbar ist, so muss eine Rückweisung der Anfrage erfolgen. Wie die Entwurfsaufgabe mathematisch zu beschreiben ist und mit welchen Methoden die Aufgabe gelöst werden kann, wird im Folgenden gezeigt.

3.1 Formulierung als Optimierungsaufgabe

Es gilt zunächst das allgemeine symmetrische Multiprozessormodell aus Abschnitt 2.1.1. Ohne Einschränkung der Allgemeinheit wird der Startzeitpunkt T_0 der Echt-

zeitperiode zu $t = T_0 = 0$ gesetzt. Der Endzeitpunkt liegt entsprechend bei $t = T_0 + \Delta T$, wobei $t \in \mathbb{R}$. Zur Formulierung der Entwurfsaufgabe ist es hier notwendig, die reelle Zeitachse in äquidistante Intervalle der Länge Δt zu zerlegen. Diese Intervalle Δt sind mit $n \in \mathbb{Z}$ indiziert. Die Echtzeitperiode ΔT ist mit Intervallen der Länge Δt abgedeckt, und zwar so, dass $n = 1$ der Zeitindex für das erste Intervall $[0; \Delta t]$ und $n = N$ der Zeitindex für das letzte Intervall $[(N-1)\cdot\Delta t; N\cdot\Delta t]$ der Echtzeitperiode $\Delta T = N \cdot \Delta t$ ist. Mit der Diskretisierung der Zeitachse geht auch eine Diskretisierung der Modullaufzeiten einher. Die Laufzeit p_m eines Moduls m ist nun ein ganzzahliges Vielfaches der Zeitauflösung Δt. Der flache Graph enthält insgesamt M Module, die eindeutig mit $m \in \mathbb{N}$, $m : 1 \leq m \leq M$ indiziert sind, und darin eingeschlossen sind der Startknoten des Graphen mit dem Index $m = 1$ und der Zielknoten des Graphen mit dem Index $m = M$.

Für jedes Modul werden N Triggervariablen $x_{m,n} \in \{0; 1\}$ eingeführt, die den Beginn der Ausführung des Moduls m im Zeitintervall $n = n_s$ durch den Wert $x_{m,n_s} = 1$ anzeigen:

$$x_{m,n} = \begin{cases} 1 & ; n = n_s(m) \ : \text{ Die Ausführung des Moduls } m \text{ beginnt.} \\ 0 & ; n \neq n_s(m) \ : \text{ sonst} \end{cases}$$

Man kann sich die Menge aller $x_{m,n} \; \forall \; n : 1 \leq n \leq N$ als eine diskrete Zeitachse für das Modul m vorstellen, wobei das Triggersignal bei einem bestimmten Zeitindex $n_s : 1 \leq n_s(m) \leq N$ auftritt. Da jedes Modul eine solche Menge an Variablen besitzt, gibt es insgesamt M derartige Zeitachsen. Aus Gründen der Vereinfachung in den folgenden Gleichungen seien $x_{m,n} \equiv 0 \; \forall \; n \notin \{1,2,\ldots,N\}$.

Einen Ablaufplan (*engl.* schedule) zu erzeugen bedeutet einzig und allein die Festlegung aller Startindizes $n_s(m)$. Das Ergebnis der Ablaufplanung kann als Vektor von Startindizes $\vec{n}_s \in \mathbb{N}^m$ zusammengefasst werden. Das Entwurfsziel besteht darin, den Startindex des Zielknotens zu minimieren:

$$\min n_s(M) \ : \ \sum_{n=1}^{N} n \cdot x_{M,n} = n_s(M) \overset{!}{\leq} N \tag{3.1}$$

Natürlich darf der Startindex $n_s(M)$ in einem Echtzeitsystem den Zeitindex N nicht überschreiten. Darüber hinaus soll aber $n_s(M)$ auch minimiert werden, und das aus zwei guten Gründen. Erstens kann bei konstantem Systemtakt die Rechenzeit, die nicht auf die Signalverarbeitung in der physikalischen Schicht verwendet

wird, höheren Schichten für andere Aufgaben zur Verfügung gestellt werden. Zweitens kann Verlustleistung eingespart werden, wenn das Betriebssystem die Möglichkeit hat, den Systemtakt zu regeln. Die mittlere Schaltfrequenz von CMOS-Schaltungen hat unmittelbar Einfluss auf die dynamische Verlustleistung [87, 112] des mobilen Endgeräts. Abhängig vom unterstützten Mobilfunkstandard könnte der Systemtakt so weit gesenkt werden, dass der Zielknoten die Echtzeitbedingung $n_s(M) \leq N$ gerade noch erfüllt. Mit der Minimierung von $n_s(M)$ wachsen also die Möglichkeiten, den Systemtakt und damit die elektrische Verlustleistung zu senken.

Nachdem nun das Entwurfsziel mit Gleichung (3.1) beschrieben ist, müssen noch die Randbedingungen genau festgelegt werden. Ein Ablaufplan ist nur zulässig, wenn die Triggervariablen $x_{m,n}$ alle Randbedingungen erfüllen. Dazu wird angenommen, dass jedes Modul im flachen Graphen innerhalb einer Echtzeitperiode nur genau einmal ausgeführt wird. Dennoch können die Längen der Ein- und Ausgangsdatenblöcke beliebig sein. Die Randbedingung der einmaligen Ausführung pro Zeitintervall ΔT lässt sich folgendermaßen erfassen:

$$\sum_{n=1}^{N} x_{m,n} = 1 \qquad ; \forall\, m : 1 \leq m \leq M \tag{3.2}$$

Die Darstellung von SDR-Software mit Hilfe eines flachen Graphen legt eindeutige Vorgänger-Nachfolger-Beziehungen zwischen allen Modulen fest. Diese Beziehungen lassen sich ebenfalls mit Triggervariablen als Randbedingungen ausdrücken. Unter Kenntnis der Kantenrichtungen soll sichergestellt sein, dass das Modul m als Nachfolger von Modul l nicht gestartet wird, bevor die Ausführung von l zu Ende ist. Mit Hilfe der Triggervariablen ausgedrückt: Die Startindizes $n_s(l)$ und $n_s(m)$ dürfen niemals zusammen in einem Zeitfenster der Länge $p_l \cdot \Delta t$ den Wert 1 annehmen, wobei p_l das ganzzahlige Äquivalent der Laufzeit von Modul l ist. Ganz allgemein muss diese Fensterbedingung für alle Elemente der Kantenmenge $\mathbb{K}$ und für jeden zu prüfenden Zeitindex ν innerhalb der Echtzeitperiode ΔT gelten:

$$\sum_{n=\nu}^{\nu+p_l-1} (x_{l,n} + x_{m,n}) \leq 1 \qquad ; \forall\, \nu : 1 \leq \nu \leq N \;\wedge\; \forall\, \langle l,m \rangle \in \mathbb{K}$$

Diese Formulierung der Nebenbedingung ist schwach [15], weil sie die Kantenrichtungen nicht erfasst. Das ist am folgenden Beispiel leicht nachzuvollziehen: Zwei

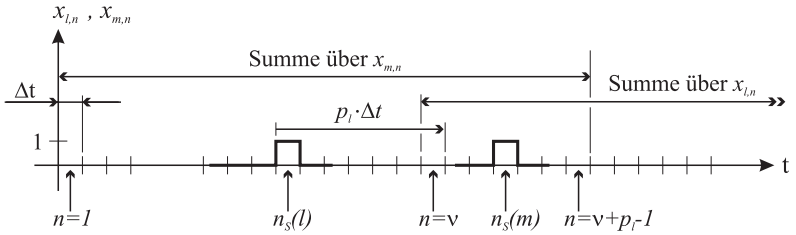


Bild 3.1 Vorgänger-Nachfolger-Beziehung nach Gleichung (3.3)

Triggervariablen $x_{l,n}$ und $x_{m,n}$ könnten so gesetzt werden, dass für die Startindizes $n_s(m) < [n_s(l) - p_l]$ gilt. Obwohl sie also nicht gemeinsam in einem Zeitfenster der Länge p_l liegen (und die Triggervariablen damit die obige Gleichung erfüllen), wird durch diese Konstruktion eindeutig die Vorgänger-Nachfolger-Beziehung zwischen den Modulen l und m verletzt. Deshalb ist für die Formulierung der SDR-Entwurfsaufgabe die folgende, starke Formulierung vorzuziehen:

$$\sum_{n=1}^{\nu+p_l-1} x_{m,n} + \sum_{n=\nu}^{N} x_{l,n} \leq 1 \qquad ; \forall\, \nu : 1 \leq \nu \leq N \ \wedge\ \forall\, \langle l,m \rangle \in \mathbb{K} \qquad (3.3)$$

Die erste Summe erstreckt sich über alle Zeitindizes von $n = 1$ bis $n = \nu + p_l - 1$, die zweite über Indizes vom beliebigen $n = \nu$ bis $n = N$. Folglich bezieht Gleichung (3.3) alle Zeitindizes der Echtzeitperiode ein und stellt dennoch die Fensterbedingung sicher. Bild 3.1 zeigt einen zulässigen Ablaufplan für die Kante $\langle l,m \rangle$ und einen willkürlichen Testzeitindex ν. Die diskrete Zeitachse in diesem Bild ist die Überlagerung der zwei Zeitachsen von Modul l und Modul m, die hier in der richtigen Reihenfolge gestartet werden. Offensichtlich wird die Gleichung (3.3) für dieses ν erfüllt. Anhand derartiger zeitlicher Darstellungen für alle Testzeitindizes ν kann man sich klarmachen, dass die Gleichung (3.3) korrekte Beziehungen zwischen Modulen erzwingt, und zwar sowohl im Hinblick auf die Fensterbedingung für Laufzeiten als auch auf die allgemeine zeitliche Anordnung von Vorgängern und Nachfolgern. Da diese starke Formulierung also Kantenrichtungen implizit erfasst, ist sie für die Beschreibung der Entwurfsaufgabe eindeutig vorzuziehen.

Neben der Graphendarstellung umfasst die Modellbildung für Modulares Software Defined Radio auch Prozessoren und Datenspeicher. Da die Anzahl der Prozessoren ebenso begrenzt ist wie der Gesamtumfang des Datenspeichers, müssen auch diese Randbedingungen als Gleichungen in $x_{m,n}$ ausgedrückt werden.

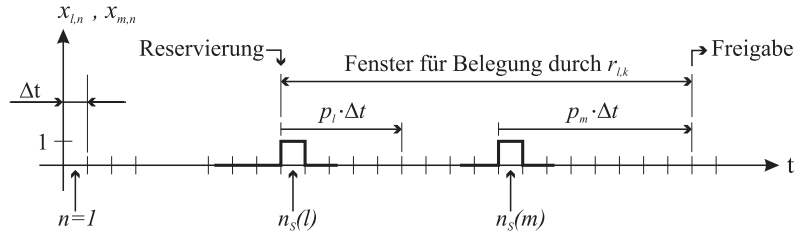


Bild 3.2 Belegung der Speicherressource R_k für eine gerichtete Kante $\langle l,m \rangle$

Zuerst sollen die Speicheranforderungen der Module betrachtet werden. Bei einem verteilten Prozessormodell seien insgesamt K verschiedene Ressourcen R_k vorhanden, die mit $k \in \mathbb{N} : 1 \leq k \leq K$ indiziert sind. Unter R_k können beispielsweise die Speicherbereiche M_k verstanden werden, die den Prozessoren aus Bild 2.2 zugeordnet sind, aber auch das Shared Memory oder der Bereich der I/O-Schnittstelle. Selbst Hardware-Architekturen, die nur über einen einzigen Speicher für alle Prozessoren verfügen, lassen sich modellieren, indem man $K = 1$ setzt.

Zu Beginn der Ausführung eines Moduls l reserviert das Betriebssystem von der Ressource R_k den angeforderten Datenspeicher $r_{l,k}$ für Ausgangsdaten. Wenn entlang der gerichteten Kante $\langle l,m \rangle$ Daten fließen, teilen sich die Module l und m den bereits reservierten Speicherbereich $r_{l,k}$. Dieser Bereich wird von beiden Modulen genutzt, von l als Ausgangsspeicher, von m als Eingangsspeicher. Erst am Ende der Ausführung von Modul m wird $r_{l,k}$ vom Betriebssystem wieder freigegeben. Bild 3.2 zeigt diesen Vorgang mit Hilfe von zwei überlagerten diskreten Zeitachsen der Module l und m, die hier nach wie vor in der richtigen Reihenfolge gestartet werden. Es ist zu erkennen, dass das Fenster, in dem der Teil $r_{l,k}$ der Ressource R_k reserviert ist, die Zeitindizes $n_s(l)$ bis $n_s(m) + p_m - 1$ umfasst. Ganz allgemein darf die Summe über alle Reservierungen in keinem Zeitintervall ν die verfügbare Kapazität der Ressource R_k übersteigen, und zwar für alle Kanten und alle Speicherressourcen $1 \leq k \leq K$:

$$\sum_{m=1}^{M} \left[\sum_{l} r_{l,k} \left(\sum_{n=1}^{\nu} x_{l,n} - \sum_{n=1}^{\nu - p_m} x_{m,n} \right) \right] \leq R_k \tag{3.4}$$

$$\forall\, \nu : 1 \leq \nu \leq N \;\wedge\; \forall \langle l,m \rangle \in \mathbb{K} \;\wedge\; \forall\, k : 1 \leq k \leq K$$

Der Term in runden Klammern stellt dabei sicher, dass die Länge des Reservierungsfensters nach Bild 3.2 richtig berücksichtigt wird.

Schließlich ist noch die begrenzte Prozessoranzahl zu erfassen. Formal bedeutet das lediglich, dass zu keinem Testzeitindex ν die Zahl der aktiven Prozessoren die Gesamtzahl der verfügbaren Prozessoren übersteigen darf:

$$\sum_{m=1}^{M} \sum_{n=\nu-p_m+1}^{\nu} x_{m,n} \leq L \qquad ; \forall\, \nu : 1 \leq \nu \leq N \tag{3.5}$$

In dieser Gleichung stellt die innere Summe sicher, dass die Laufzeiten richtig erfasst sind, während die äußere Summe über alle Module $m : 1 \leq m \leq M$ läuft und alle diejenigen Module aufaddiert, die zum Testzeitindex ν gerade aktiv Daten verarbeiten. Es ist unmittelbar klar, dass diese Summe die Gesamtzahl der verfügbaren Prozessoren nicht übersteigen darf.

Das System der Gleichungen (3.1)-(3.5) konstituiert eine Aufgabe der linearen ganzzahligen Optimierung (*engl.* ILP), die sogar binär ist, weil $x_{m,n} \in \{0; 1\}$. Die Anzahl der Gleichungen beläuft sich auf $M + (K+1) \cdot N \cdot |\mathbb{K}| + N$. Dieser Aufgabentyp ist als NP-schwer bekannt [69], und die optimale Lösung kann nur mit Branch-and-Bound-Methoden [95, 109] gefunden werden. Lediglich für spezielle Klassen von Vorgänger-Nachfolger-Beziehungen und für besondere Laufzeiten (siehe [107] and Referenzen dort) existieren Lösungsalgorithmen mit polynomialem Aufwand.

Die Entwurfsaufgabe für Modulares Software Defined Radio ist mit den Gleichungen (3.1)-(3.5) auf eine eindeutige, abstrakte, mathematische Weise beschrieben. Bisher wurden die technischen Möglichkeiten im Bereich Software Radio häufig überinterpretiert, wohingegen die konkreten Herausforderungen heruntergespielt oder nur isolierte Teile der Entwurfsaufgabe betrachtet wurden. Die hier hergeleitete Formulierung ersetzt frühere, subjektiv gefärbte, verbale Beschreibungen. Sie schließt bisherige Entwurfsansätze (z.B. PaC-SDR) ein und bezieht sich voll und ganz auf das Systemkonzept von SDR, nicht auf isolierte Teilaufgaben. Ein weiterer Vorteil liegt darin, dass der Entwurf von SDR als Optimierungsaufgabe existierenden allgemein gültigen Lösungsverfahren [8, 50] zugänglich gemacht wird. Das Betriebssystem eines Mod-SDRs könnte im Prinzip ein solches Verfahren als Kernbestandteil enthalten und anhand des $n_s(M)$ die optimale Entscheidung über die Machbarkeit oder die Rückweisung einer Realisierungsanforderung treffen.

Allein die Erkenntnis, dass die Entwurfsaufgabe einer bekannten Klasse von Aufgaben zuzuordnen ist und Lösungsverfahren existieren, die von der konkreten Anwendung auf SDR unabhängig sind, zeigt deutlich, wie sehr die technischen Mög-

lichkeiten auf der physikalischen Schicht durch klassische Randbedingungen begrenzt sind. Eine Realisierung in Software bringt ohne Zweifel den Vorteil der Flexibilität mit sich, setzt aber weder die Regeln der Nachrichtentechnik noch die der Echtzeitsysteme ausser Kraft.

Der wesentliche Nachteil der Formulierung als Aufgabe der linearen ganzzahligen Optimierung liegt im Umfang des Gleichungssystems (3.1)-(3.5), der mit Verbesserung der Zeitauflösung wächst. Die Anzahl der Gleichungen ist um so größer, je kleiner das Zeitintervall Δt gewählt wird. Einerseits werden dann die Fehler zwischen den tatsächlichen Modullaufzeiten und den quantisierten Versionen kleiner und Startzeitpunkte für die Module können genauer festgelegt werden, andererseits dauert die Lösung des Gleichungssystems sehr viel länger als mit einer groben Zeitauflösung. Gerade bei NP-schweren Aufgaben wächst die Rechenzeit bis zur Lösung im schlimmsten Fall exponentiell. Die Entscheidung über die Machbarkeit oder die Rückweisung einer Realisierungsanforderung soll im mobilen Endgerät aber so schnell wie möglich getroffen werden. Daher werden im Folgenden Algorithmen zur Ablaufplanung betrachtet, die schnell brauchbare, aber nicht unbedingt optimale Ergebnisse liefern.

3.2 Suboptimale Ablaufplanung und Speedup

Wenn nicht sichergestellt ist, dass das Betriebssystem aufgrund der optimalen Lösung $n_s(M)|_{opt}$ entscheidet, muss die Qualität eines Ablaufplans auf andere Weise gemessen werden. Ferner ist es im Sinne einer allgemein gültigen Entwurfstheorie für Mod-SDR sinnvoll, sich von konkreten Echtzeitanforderungen bestimmter Standards in Form von ΔT oder N zu lösen. Aus diesen Gründen wird von nun an mit der relativen Beschleunigung (*engl.* speedup) der Signalverarbeitung durch ein verteiltes Prozessorsystem gearbeitet. Dabei ist der Speedup s definiert als das Verhältnis der Gesamtlaufzeit ΔT_{mono} der physikalischen Schicht auf einem Einzelprozessor zur Gesamtlaufzeit ΔT_{multi} der gleichen physikalischen Schicht auf einem Multiprozessorsystem.

In [78] wird die lineare ganzzahlige Optimierungsaufgabe zunächst vereinfacht, indem die Ungleichung (3.4) fallengelassen wird. Damit geht die Annahme einher, dass die verteilten Speicher stets groß genug sind, um alle Zwischenergebnisse aufzunehmen, so dass die Ungleichung keine aktive Beschränkung mehr darstellt. Ein Nebeneffekt dieser Annahme ist, dass alle Fragen der Zuordnung von Modulen zu

Prozessoren und von Ausgangsspeicherbereichen zu verteilten Speicherressourcen R_k wegfallen. Weiterhin modelliert das Gleichungssystem (3.1)-(3.5) keine Buszugriffe. Der potentielle Austausch von Zwischenergebnissen zwischen Prozessoren sowie die Ein- und Ausgabe von Daten über die I/O-Schnittstelle erfolgen also in Nullzeit und ohne Buskonflikte. Insgesamt bedeutet das Fallenlassen der Ungleichung (3.4) folglich eine Reduktion auf eine Aufgabe der reinen Ablaufplanung für L Prozessoren. Aus dem Bereich der Wirtschaftstheorie ist ein einfacher Algorithmus bekannt, der solche Aufgaben auf Baumgraphen mit Einheitslaufzeiten optimal löst.

Der Algorithmus nach Hu [38] bewertet in einer ersten Phase alle Knoten eines Graphen mit ihrem Abstand zum Zielknoten, wobei dieser Abstand der Summe aller Modullaufzeiten entlang des längsten Pfades zum Zielknoten entspricht. Das Abstandsmaß wird in der Literatur auch Hu Level [53, 78], B Level (Bottom Level), Static Level [51] oder Height [54] genannt. In der zweiten Phase arbeitet der Hu-Algorithmus mit der Menge der noch nicht ausgeführten Module und einer Kandidatenliste, die alle diejenigen Module nach ihrem Hu Level geordnet enthält, deren Vorgänger ohne Ausnahme bereits ausgeführt worden sind. Für jeden Prozessor, der gerade frei und damit für die Ausführung eines weiteren Moduls wieder verfügbar geworden ist, entfernt der Algorithmus ein Modul an der Spitze der Kandidatenliste und übernimmt Startzeitpunkt und Prozessornummer in den Ablaufplan. Module stehen nur einmal in der Kandidatenliste und werden folglich genau einmal zur Ausführung gebracht. Da nur freien Prozessoren neue Module aus der Kandidatenliste zugewiesen werden, sind beide Randbedingungen nach Gleichung (3.2) und (3.5) erfüllt. Der Hu Level enthält indirekt Informationen über die Struktur des Graphen, so dass der Hu-Algorithmus automatisch alle Vorgänger-Nachfolger-Beziehungen erfüllt.

Mit Blick auf die Voraussetzungen lässt sich jedoch keine optimale Lösung der Planungsaufgabe erwarten. Zwar ist bei Untersuchungen zu existierenden Mobilfunkstandards beobachtet worden, dass Bäume in den Signalverarbeitungsstrukturen von physikalischen Schichten dominieren [78], aber das einfache stochastische Laufzeitmodell liefert keine Einheitslaufzeiten, sondern beliebige reellwertige Laufzeiten zwischen p_{min} und p_{max}. In der Behandlung von Multiraten-Graphen durch Lee und Messerschmidt [53] wird der Hu-Algorithmus ebenfalls erwähnt, allerdings abgewandelt für beliebige ganzzahlige Laufzeiten. In [78] wird der Hu-Algorithmus für die reellwertigen Laufzeiten aus dem stochastischen Mod-SDR-Modell erweitert. In diesem Fall ist die Aufgabe der Ablaufplanung bereits NP-

schwer [5], und es kann nicht mehr davon ausgegangen werden, dass der Hu-Algorithmus optimale Lösungen liefert. Dennoch ist der Algorithmus lauffähig und liefert einen brauchbaren Ablaufplan für die physikalische Schicht eines Mod-SDRs, und zwar viel einfacher und schneller als eine vollständige Suche nach der optimalen Lösung mit Hilfe von Branch-and-Bound-Methoden. Der auf reelle Laufzeiten erweiterte Hu-Algorithmus ist demnach eine erste pragmatische Wahl für den Kern des Betriebssystems. Ferner kann auch eine suboptimale Lösung nach Hu später als Ausgangspunkt für die Verbesserung der Lösung mit Hilfe von iterativen Suchverfahren dienen.

Bevor der Hu-Algorithmus auf die komplizierten logischen Strukturen eines Mod-SDR angewendet wird, sollen grundlegende Eigenschaften einfacher Graphen festgestellt werden. Dazu wird ein erzeugender Graph aus drei Funktionen eingeführt, aus dem Baumstrukturen rekursiv konstruiert werden können. Aus der Erfahrung im Umgang mit einer Vielzahl von existierenden Mobilfunkstandards im Rahmen des PaC-SDR-Ansatzes [74, 76, 114] ist bekannt, dass Baumstrukturen in der Funksignalverarbeitung typischerweise vorkommen. Die Untersuchung des erzeugenden Graphenelementes wird von der Erwartung motiviert, dass Eigenschaften, die dort beobachtet werden, unter rekursiver Konstruktion komplizierterer logischer Strukturen erhalten bleiben.

3.3 Ergebnisse mit einfachen Graphen

Das Bild 3.3 zeigt einen Graphen, der aus einem Demultiplexer, zwei parallelen Zweigen mit je einer Funktion, einem Multiplexer und einer weiteren Funktion in Serie besteht. Wie bereits in Abschnitt 2.2.2 beschrieben, ist sehr einfachen Modulen wie Multiplexern und Demultiplexern die Laufzeit Null zugeordnet. Lediglich die Funktionen tragen in diesem Graphen effektive Laufzeiten, und sie sind in eine Baumstruktur (rechts der senkrechten gepunkteten Linie) eingebunden.

Dieser einfache Graph kann als erzeugendes Element interpretiert werden, der die

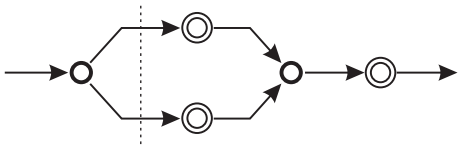


Bild 3.3 Erzeugendes Element für Baumgraphen

Konstruktion aller möglicher Baumgraphen erlaubt, indem die Funktionen in den Parallelzweigen (und **nur** diese) rekursiv wieder durch erzeugende Elemente ersetzt werden. Bevor das Betriebsystem zur Ablaufplanung übergehen kann, muss noch der flache Graph konstruiert werden. Dazu muss bestimmt werden, welche logischen Strukturen innerhalb der Funktionen vorzufinden sind.

Im Zusammenhang mit dem hierarchischen Aufbau von komplizierten Gebilden aus Einzelmodulen taucht häufig der Begriff der Granularität auf. Intuitiv kann man darunter die Anzahl der Module verstehen, aus denen eine bestimmte Funktion aufgebaut ist, also lediglich: Granularität $G_1 \in \mathbb{N}$. Es ist bereits dargestellt geworden, dass die Laufzeiten aller Module $p_m : 1 \leq m \leq M$ von besonderer Wichtigkeit sind, aber Laufzeiten fließen in G_1 nicht ein. Die bewusste Berücksichtigung von Laufzeiten führt hingegen auf eine allgemeinere Definition für die Granularität eines Graphen, der viele verschiedene Funktionen und Module umfassen kann. Als Granularität G wird das Verhältnis der Gesamtanzahl aller Module zu ihrer mittleren Laufzeit definiert:

$$G := \frac{M}{\overline{p}} = \frac{M}{\frac{1}{M}\sum_{m=1}^{M} p_m} \tag{3.6}$$

Beide Granularitätsbegriffe werden in der vorliegenden Arbeit benutzt, jedoch sollte bereits an dieser Stelle deutlich werden, dass sowohl G_1 als auch G unzureichend sind, um Aufbau und Eigenschaften eines SDR vernünftig zu beschreiben: G_1 erfasst weder die einzelnen Modullaufzeiten noch die Struktur der Funktion. G ist wegen der Mittelwertbildung ein integrales Maß für die gesamte Software innerhalb einer betrachteten OSI-Schicht und erfasst die Struktur dieser Schicht ebenso wenig wie G_1 die Struktur einer Funktion. Gerade der Begriff der Struktur aber ist es, der im Bereich des Modularen Software Defined Radio wichtige Fragen aufwirft. Granularität als Softwareeigenschaft ist also nur in Sonderfällen einsetzbar, und zwar genau dann, wenn über die logischen Strukturen Zusatzaussagen getroffen werden.

Im Falle der rekursiven Konstruktion mit erzeugenden Elementen soll die Baumeigenschaft auch im flachen Graphen erhalten bleiben. Folglich dürfen die Funktionen nur durch serielle Ketten von Modulen ersetzt werden. Die Anzahl der Module pro Funktion, also G_1, soll für alle Funktionen gleich sein. Da weiterhin die Dichtefunktion zur Erzeugung von Laufzeiten für alle Module identisch ist, ist auch der Mittelwert aller Laufzeiten konstant, und G_1 stellt lediglich eine skalierte Version der allgemeineren Granularität G dar.

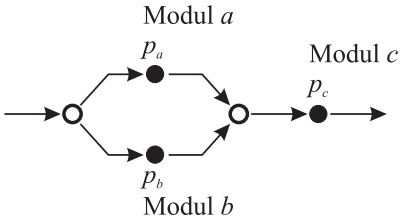


Bild 3.4 Erzeugendes Element, $G_1 = 1$

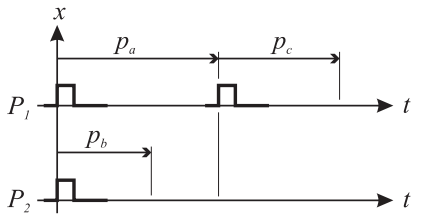


Bild 3.5 Typischer Ablaufplan

Die einfachste Realisierung eines Baumes besteht in dem erzeugenden Element selbst, wenn $G_1 = 1$ ist. Das ist derjenige Graph, bei dem alle Funktionen aus Bild 3.3 durch je ein Modul ersetzt sind (siehe Bild 3.4). Zunächst soll diese Realisierung bezüglich des zu erwartenden Speedups untersucht werden. Dafür wird zunächst die Familie der gefensterten Exponentialfunktionen zur Erzeugung von Modullaufzeiten herangezogen. Diese Wahl wird ebenfalls aus der Erfahrung mit PaC-SDR motiviert: Bei der Entwicklung von Softwaremodulen durch intuitives Aufteilen von klassischen nachrichtentechnischen Funktionen wurden relativ viele Module mit kleiner Laufzeit generiert. Nur wenige Module mit großen Laufzeiten wurden entworfen. Diese herkömmliche Art und Weise, Software für SDR zu programmieren, ist in einem Modell mit Exponentialdichte näherungsweise enthalten, wenn $\lambda > 0$. Aber auch eine Gleichverteilung der Modullaufzeiten kann mit $\lambda = 0$ modelliert werden, und es stellt sich die Frage, ob eine Dichte mit $\lambda < 0$ nicht höheren Speedup oder bessere Vorhersagbarkeit des Speedup über eine große Anzahl von Realisierungen erzielt. Indem der Dichteparameter λ zwischen negativen und positiven Werten variiert wird, kann ein Kontinuum an Realisierungsvoraussetzungen untersucht und mit den Voraussetzungen des herkömmlichen Softwareentwurfs verglichen werden.

Nach Bild 3.4 liegen nur die Module a und b in parallelen Zweigen, und es ist leicht einzusehen, dass der Einsatz von mehr als zwei Prozessoren bei dieser Struktur nicht sinnvoll ist. Aus diesem Grund wird im Folgenden $L = 2$ gesetzt. Das Bild 3.5 zeigt einen typischen Ablaufplan unter der Bedingung, dass $p_a > p_b$. Der Speedup kann dann wie folgt ausgedrückt werden:

$$s = \frac{p_a + p_b + p_c}{p_a + p_c} = 1 + \eta \qquad \text{, wobei} \quad \eta := \frac{p_b}{p_a + p_c} = \frac{p_b}{p_\tau} \tag{3.7}$$

Die Modellvorstellung von Mod-SDR erlaubt es, dass für den Betrieb der physika-

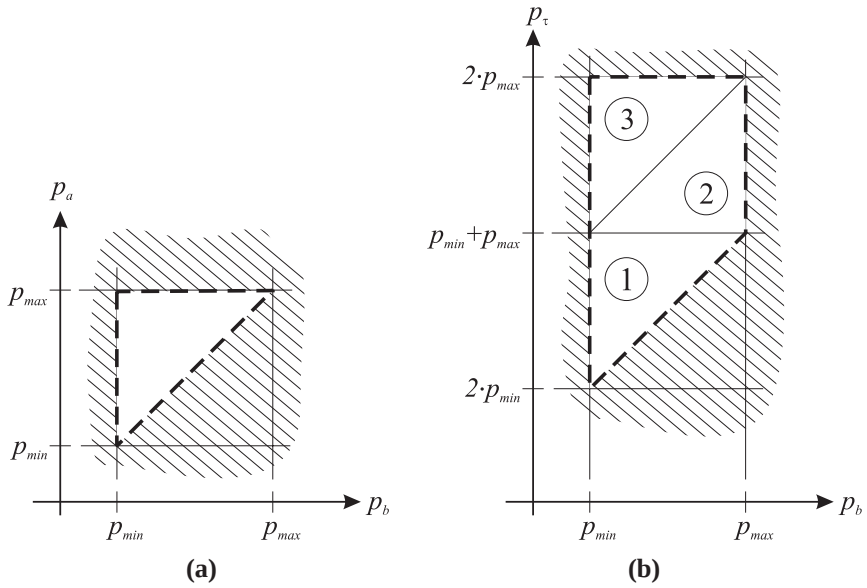


Bild 3.6 Gebiete in der $p_a p_b$-Ebene und in der $p_\tau p_b$-Ebene

lischen Schicht kompatible Module an einem beliebigen Knoten des Graphen zum Einsatz kommen. Daher stammen die Laufzeiten der Module a, b und c aus unabhängigen Zufallsexperimenten mit der gleichen Wahrscheinlichkeitsdichte. Aus der Dichtefunktion kann im Prinzip auf die Dichte des Speedups $f_S(s)$ geschlossen werden. Eine zu $f_S(s)$ äquivalente Beschreibung liefert die Verteilungsfunktion $F_H(\eta)$, die hier hergeleitet werden soll. Um Gleichung (3.7) nutzen zu können, muss die Beziehung $p_a > p_b$ sichergestellt sein. Das Bild 3.6(a) zeigt das dreieckige Gebiet in der $p_a p_b$-Ebene, auf dem die bedingte Dichte $f(p_a, p_b \mid p_a > p_b)$ definiert ist.

$$f\left(p_a, p_b \mid p_a > p_b\right) = \frac{f\left(p_a, p_b\right)}{P(p_a > p_b)} \tag{3.8}$$

Da die Modullaufzeiten p_a und p_b voneinander unabhängig sind, gilt $f(p_a,p_b) = f(p_a) \cdot f(p_b)$. Die Bedingung $p_a > p_b$ wird bereits durch die Einschränkung des Definitionsbereiches auf das dreieckige Gebiet in der $p_a p_b$-Ebene sichergestellt. Aus Symmetriegründen ist die Auftretenswahrscheinlichkeit $P(p_a > p_b) = 0{,}5$. Ferner ist es bezüglich der Variable η hinreichend, den Fall $p_a > p_b$ zu betrachten,

da der Fall $p_a < p_b$ zu identischen Ergebnissen führt, wenn nur die Rollen der Module a und b vertauscht werden. Folglich gilt $F_H(\eta) = F_H(\eta \mid p_a > p_b)$.

Der Nenner des Speedup-Ausdrucks in Gleichung (3.7) enthält die Summe der unabhängigen Variablen p_a und p_c, bzw. die neue Zufallsvariable $p_\tau := p_a + p_c$. Die gemeinsame Dichte von p_τ und p_b erhält man durch Faltung von $f(p_a, p_b \mid p_a > p_b)$ mit einer weiteren gefensterten Exponentialdichte in p_a-Richtung. Das Bild 3.6(b) zeigt das entsprechende Gebiet in der $p_\tau p_b$-Ebene, in dem die Verbunddichte $f(p_\tau, p_b \mid p_a > p_b)$ nicht verschwindet:

$$f(p_\tau, p_b \mid p_a > p_b) = 2 \cdot {r_e}^3 \cdot e^{-\lambda(p_b - p_{min})}$$

$$\cdot e^{-\lambda(p_\tau - 2 \cdot p_{min})} \cdot \begin{cases} p_\tau - (p_{min} + p_b) & ; \text{ Teilgebiet } (1) \\ p_{max} - p_b & ; \text{ Teilgebiet } (2) \\ 2 \cdot p_{max} - p_\tau & ; \text{ Teilgebiet } (3) \\ 0 & ; \text{ sonst } \end{cases} \tag{3.9}$$

Die gesuchte Verteilungsfunktion $F_H(\eta)$ ergibt sich, wenn man $f(p_\tau, p_b \mid p_a > p_b)$ über das Teilgebiet integriert, für das $p_b < \eta \cdot p_\tau$ gilt [70]. Die Gleichung $p_b = \eta \cdot p_\tau$ definiert eine Ursprungsgerade, deren Steigung von η abhängt, und das Integrationsgebiet liegt oberhalb dieser Geraden. So ist auch leicht zu sehen, dass

$$F_H(\eta) = 0 \quad \text{für} \quad \eta \leq \eta_{min} = \frac{p_{min}}{2 \cdot p_{max}} \tag{3.10}$$

$$\text{und} \quad F_H(\eta) = 1 \quad \text{für} \quad \eta \geq \eta_{max} = \frac{p_{max}}{p_{min} + p_{max}} \tag{3.11}$$

Die Konstanten $s_{min} = 1 + \eta_{min}$ und $s_{max} = 1 + \eta_{max}$ sind die untere und obere Schranke für den Speedup. Bild 3.7 zeigt das Integral in Abhängigkeit von Speedup $s = 1 + \eta$ für einige ausgewählte λ im Intervall $[-3; 3]$. Es ist festzustellen, dass die Verteilung $F_H(\lambda,s)$ nicht symmetrisch bezüglich $\lambda = 0$ ist, obwohl die Gestalt der Exponentialdichten für Parameterpaare $\pm\lambda$ durchaus symmetrisch bezüglich der Gleichverteilung ist.

Die Berechnung von $F_H(\lambda,s)$ auf Graphen mit einer größeren Knotenzahl zu erweitern, ist äußerst schwierig. Daher werden Verteilungsfunktionen für den Speedup im Folgenden aus Rechnersimulationen geschätzt. Dabei wird das Ergebnis nicht allein von der Dichtefunktion $f_P(p)$ sowie den angetroffenen logischen und

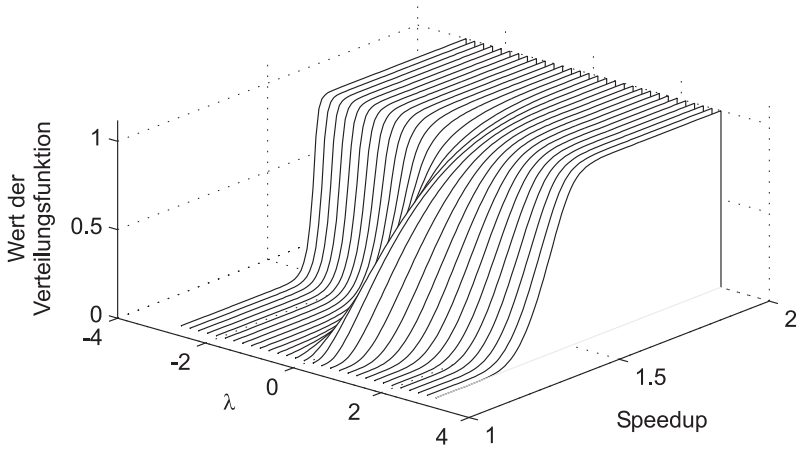


Bild 3.7 Verteilungsfunktion $F_H(\lambda,s)$

physikalischen Strukturen beeinflusst, sondern auch wesentlich von den Verfahren der Partitionierung und der Ablaufplanung. Jede Simulation zeigt also ein Gesamtergebnis in Form von Speedup, ohne dass der Beitrag einzelner Einflussgrößen unmittelbar sichtbar ist. Nichts desto trotz handelt es sich beim Speedup um das wesentliche Qualitätsmaß eines Multiprozessor-Echtzeitsystems, und der Speedup wird meist in Abhängigkeit eines einzelnen variablen Parameters gezeigt, der einen bestimmten Aspekt des Mod-SDR-Entwurfs betrifft. Einmal mehr zeigen sich hier Parallelen zur klassischen Systemsimulation in der Nachrichtentechnik.

In diesem Abschnitt wird nach wie vor die Aufgabe der reinen Ablaufplanung und der Hu-Algorithmus für reelle Laufzeiten betrachtet. Die Simulationen dienen zum Einen dazu, die theoretisch hergeleiteten Ergebnisse zu sichern, zum Anderen soll festgestellt werden, ob das für den einfachsten Baumgraphen vorhergesagte Speedup-Verhalten auch bei größeren Graphen auftritt.

Das Bild 3.8 zeigt die Ergebnisse für das erzeugende Element mit $G_1 = 1$ und einem Stichprobenumfang von 1000 Realisierung des Graphen für jeden Parameterwert λ, wobei hier $\lambda : -9 \leq \lambda \leq 9$. Jeder Punkt in dem Bild repräsentiert die Speedup-Messung für eine Realisierung. Die Systemauslastung ist nur ein anderer Begriff für Speedup und wird in Prozent auf der rechten vertikalen Achse von Bild 3.8 angegeben, während der absolute Speedup $s : 1 \leq s \leq 2$ auf der linken vertikalen Achse aufgetragen ist. Die Kreise repräsentieren den zu erwartenden Speedup

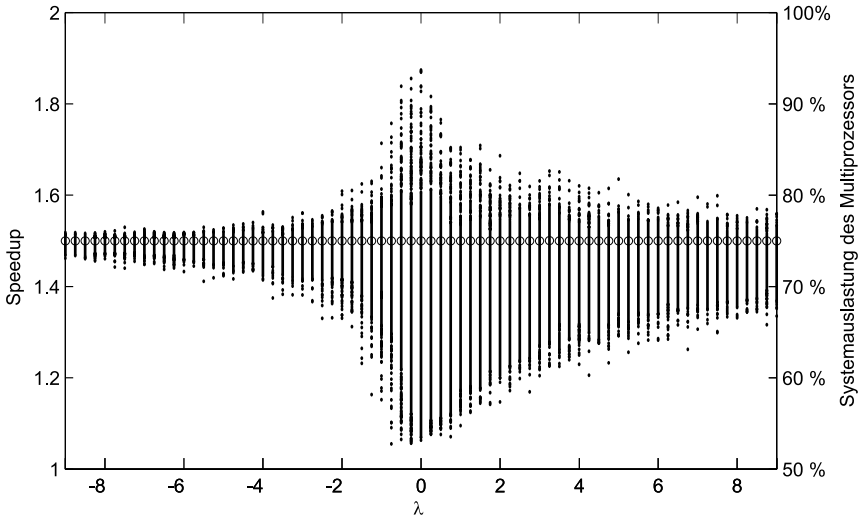


Bild 3.8 Speedup in Abhängigkeit vom Dichteparameter λ, erzeugendes Element mit $G_1 = 1$, $L = 2$, Stichprobenumfang: 1000 Realisierungen pro λ

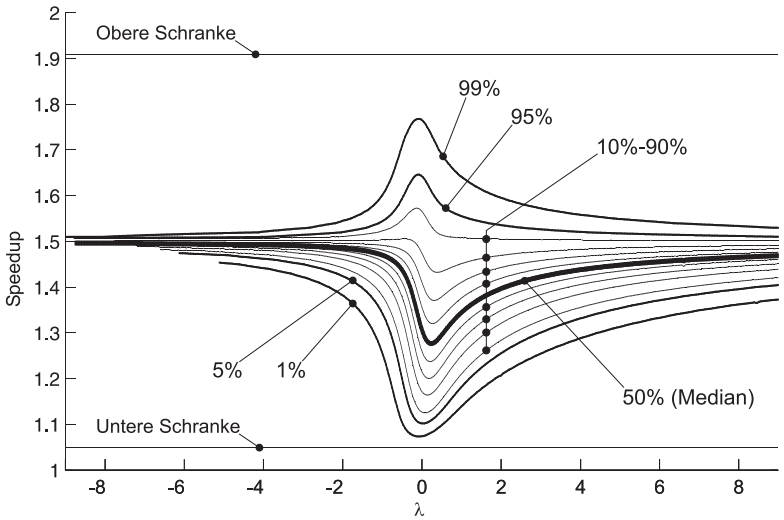
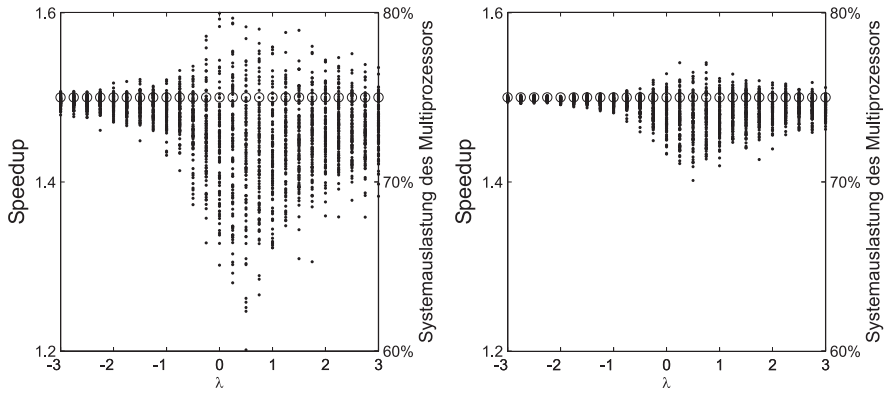


Bild 3.9 Höhenlinien von $F_H(\lambda,s)$, erzeugendes Element mit $G_1 = 1$, $L = 2$

mit Einheitslaufzeiten als Referenzwert. Bei insgesamt nur drei Modulen ist dieser Wert mit zwei parallelen Zweigen natürlich $s = 1{,}5$, und zwar unabhängig von λ. In der Tat könnte man anstelle der Einheitslaufzeit auch eine beliebige Konstante als Näherung für alle Laufzeiten wählen, eingeschlossen p_{min}, p_{max} oder auch den aus der Dichte resultierenden Mittelwert $\overline{p}$, und trotzdem würde man keinen anderen Wert als $s = 1{,}5$ erhalten. Diese Überlegung zeigt, dass keine der drei Größen allein — insbesondere nicht $\overline{p}$ — ausreicht, um das Speedup-Verhalten in einer SDR-Umgebung treffend vorherzusagen. Wenn $\lambda \to \infty$ oder $\lambda \to -\infty$ erzeugen die Zufallsexperimente zwangsläufig Laufzeiten in der Nähe von p_{min} oder p_{max}, und der Speedup muss für beide Extremfälle gegen $s = 1{,}5$ gehen. Das Bild 3.8 zeigt aber auch, dass die Streuung des Speedup in der Nähe von $\lambda = 0$ am größten ist und dass mit relativ hoher Wahrscheinlichkeit Speedups auftreten, die deutlich größer oder kleiner als $1{,}5$ sein können.

Als Vergleich zeigt das Bild 3.9 die theoretisch vorhergesagten Speedup-Ergebnisse in Form von Höhenlinien der Funktion $F_H(\lambda,s)$. Für ein gegebenes Quantil von Realisierungen gibt eine Höhenlinie an, welcher Speedup maximal erreichbar ist, abhängig vom Dichteparameter λ. Es sind Höhenlinien für eine ausgewählte Menge von Quantilen dargestellt, darunter auch Linien für extreme Werte wie das 1%- und das 99%-Quantil. Obere und untere Grenzen sind die Höhenlinien für das 0%- und das 100%-Quantil. Es ist offensichtlich, dass diese Speedup-Grenzen viel zu weit von der Mehrheit der Realisierungen entfernt liegen und folglich für eine treffende Vorhersage des Speedup-Verhaltens ebenfalls ungeeignet sind. Die Höhenlinien in Bild 3.9 decken sich jedoch hervorragend mit den experimentell ermittelten Speedup-Ergebnissen und zeigen sogar noch weitere Eigenschaften, die allein aus der Simulation nicht direkt abzulesen sind. Die Medianlinie bleibt für $\lambda < 2$ nah an $s = 1{,}5$, fällt bei $\lambda = 0$ schnell auf Werte im Bereich $s = 1{,}3$ und konvergiert für positive λ nur langsam wieder gegen $s = 1{,}5$. Demnach liefern mehr als die Hälfte aller SDR-Realisierungen einen niedrigen Speedup, wenn die Laufzeiten aus einer Gleichverteilung stammen. Ähnlich schlecht verhält sich die Streuung des Speedups: Sie liegt für 80% der Realisierungen im Bereich $[1{,}15; 1{,}6]$, und fast 90% davon erreichen den Referenzwert $s = 1{,}5$ nicht. Folglich trägt jede Änderung des Parameters λ, der von der Gleichverteilung wegführt, nur dazu bei, dass der Speedup des 50%-Quantils aller Realisierungen ebenso steigt wie die Vorhersagbarkeit des Speedups. Beides sind wünschenswerte Eigenschaften beim Entwurf von Mod-SDRs.

Auch der Einfluß der Granularität kann mit Hilfe von Simulationen untersucht wer-



(a) $G_1 = 10$ **(b)** $G_1 = 100$

Bild 3.10 Speedup für das erzeugende Element mit Granularität G_1, $L = 2$

den. Während man für das erzeugende Element mit $G_1 = 1$ streng genommen noch keine anspruchsvolle Ablaufplanung benötigt und man die Höhenlinien von $F_H(\lambda,s)$ berechnen kann, kommt im Folgenden der Hu-Algorithmus mit reellen Laufzeiten zum ersten Mal voll zum Einsatz. Die Bilder 3.10(a) und 3.10(b) zeigen die Speedup-Ergebnisse für $G_1 = 10$ und $G_1 = 100$, einen Stichprobenumfang von jeweils 100 pro λ und einige ausgewählte $\lambda : -3 \leq \lambda \leq 3$. In diesen Bildern wird lediglich der Speedup-Bereich $[1{,}2; 1{,}6]$ dargestellt. Es ist zu beobachten, dass sich die Streuung des Speedups verringert, wenn die Granularität steigt, und dass die Werte gegen $s = 1{,}5$ streben. Im Hinblick auf die Vorhersagbarkeit des Speedup ist eine hohe Granularität also für das erzeugende Element von Vorteil.

Zuletzt wird der Hu-Algorithmus auf größere Graphen angewendet, die noch schwerer analytisch zu berechnen sind als das erzeugende Element. Nichts desto trotz sind viele Graphen in der Signalverarbeitung für den Mobilfunk Baumgraphen, die sich rekursiv aus dem erzeugenden Element konstruieren oder auf diese Weise nähern lassen. Das Bild 3.11 zeigt eine solche Näherung, die im Folgenden mit $\mathcal{G}_A$ bezeichnet wird. $\mathcal{G}_A$ nähert den Graphen $\mathcal{G}_B$ aus Bild 2.10. Eine Eigenschaft von $\mathcal{G}_A$ besteht in der Unvollständigkeit des Baumes: Nur der obere Parallelzweig des erzeugenden Elementes wird vollständig rekursiv ersetzt. Beim Vergleich der beiden Graphen fallen die folgenden Unterschiede auf: Erstens enthält $\mathcal{G}_B$ zwei besondere Kanten, die in Bild 2.10 mit einem * gekennzeichnet sind und die Funktionen der

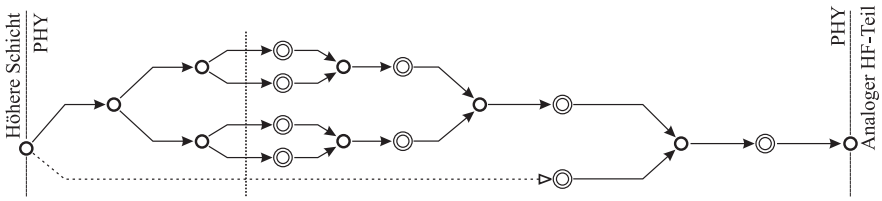
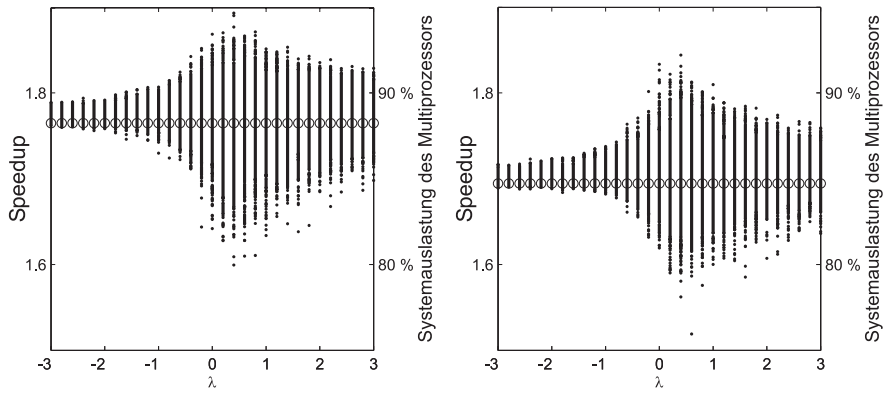


Bild 3.11 Näherung $\mathcal{G}_A$ für den Graphen $\mathcal{G}_B$ aus Bild 2.10

Kanalcodierung überbrücken. Zweitens fehlen $\mathcal{G}_B$ die seriellen Funktionen in den Zweigen des codierten DTCH und des codierten DTCH. Drittens transformiert in Bild 2.10 ein einzelnes Modul den CCTrCH in den DPDCH, wohingegen an der gleichen Stelle in Bild 3.11 eine ganze Funktion steht. Demnach ist die Baumstruktur von $\mathcal{G}_B$ nicht nur unvollständig, sondern die Rechenlast ist auch ungleichmäßig auf die Zweige verteilt. Entsprechend der Definition der Granularität sind alle Funktionen durch eine serielle Kette von G_1 Modulen zu ersetzen, um zu der flachen Version eines Graphen $\mathcal{G}$ zu kommen. Folglich ist die Anzahl der Module in $\mathcal{G}_A$ größer als in $\mathcal{G}_B$.

Die mit einem * gekennzeichneten überbrückenden Kanten in Bild 2.10 haben keinen Einfluß auf die Simulation, da die Gleichung (3.4) zur Beschreibung der Speicherbeschränkung fallengelassen wurde. Die Vorgänger-Nachfolger-Beziehungen werden für diese Kanten durch die parallel liegenden Funktionen implizit erfüllt. Demnach kann man diese Kanten ebensogut weglassen, ohne die Speedup-Ergebnisse der Simulation zu verändern. Die Bilder 3.12(a) und 3.12(b) zeigen die mit dem Hu-Algorithmus erzielten Ergebnisse für die Graphen $\mathcal{G}_A$ und $\mathcal{G}_B$, Granularität $G_1 = 10$, einen Stichprobenumfang von jeweils 1000 pro λ und einige ausgewählte $\lambda : -3 \leq \lambda \leq 3$. Erneut wird lediglich der Speedup-Bereich von Interesse dargestellt, in diesem Fall $[1{,}5; 1{,}9]$.

Wie zuvor zeigen die Kreise den zu erwartenden Speedup, wenn für alle Module des flachen Graphen Einheitslaufzeiten angenommen werden. Die Kreise für $\mathcal{G}_A$ fallen in die Nähe von $s = 1{,}8$, während die Kreise für $\mathcal{G}_B$ näher an $s = 1{,}7$ liegen. Offensichtlich hängt die Vorhersage des Speedup also nicht nur von der Grundstruktur aus Bild 3.11 ab, sondern auch von der Gesamtanzahl der Module. Diese Beobachtung ist ein weiterer Hinweis darauf, dass die Modulanzahl M in den allgemeineren Granularitätsbegriff zu übernehmen ist und dass eine hohe Granularität G möglicherweise auch im Allgemeinen für hohen Speedup sorgt. Eine ebenso



(a) Graph $\mathcal{G}_A$ **(b)** Graph $\mathcal{G}_B$

Bild 3.12 Speedup-Verhalten für verschiedene Graphen, $G_1 = 10$, $L = 2$

wichtige Beobachtung besteht jedoch darin, dass die Gestalt des Speedups in allen Simulationen praktisch unverändert bleibt und dem Verhalten des erzeugenden Elementes mit $G_1 = 1$ ähnlich ist.

Unter den gegebenen Bedingungen des Software-Modells lassen sich folgende Schlüsse ziehen [78]:

- Wenn ein Entwurfsprozess die Familie der Exponentialdichten als Zielfunktion hat, dann wird das Speedup-Verhalten des Mod-SDRs um so vorhersagbarer, je größer $|\lambda|$.
- Wenn der Dichteparameter λ durch einen iterativen Entwurfsprozess zu beeinflussen ist, dann sind negative Werte von λ zu bevorzugen, da die Streuung des Speedups bei $\lambda \to -\infty$ schneller sinkt.
- Diese Eigenschaften sind nicht nur für das erzeugende Element gültig, sondern auch für größere Baumgraphen.
- Die beobachtete Asymmetrie des Speedup bezüglich $\lambda = 0$ ist nicht mit der Unvollständigkeit von Baumstrukturen oder mit der Verteilung der Rechenlast zu begründen. Die Asymmetrie tritt auch bei vollständigen Bäumen und gleicher Lastverteilung auf alle Zweige auf.

- Je höher die Granularität, desto besser ist der Speedup einer Mod-SDR-Realisierung vorherzusagen.

Ursprünglich ist die Familie der Exponentialdichten eingeführt worden, um verschiedenste Grundvoraussetzungen für die Ablaufplanung in einem Mod-SDR zu untersuchen. Der Ausgangspunkt war dabei die Erfahrung mit dem PaC-SDR-Ansatz, die in den Untersuchungen mit $\lambda > 0$ eingeschlossen war. Aus den Schlussfolgerungen geht aber hervor, dass unter den genannten Bedingungen $\lambda < 0$ günstiger ist, weil es die Wahrscheinlichkeit für hohen Speedup über einem vergleichbaren λ-Bereich vergrößert.

Ein Nachteil der Simulation von Mod-SDR mit Exponentialdichten liegt darin, dass nur der Parameter λ zur Verfügung steht und dass sich die Schlussfolgerungen streng genommen nur auf Bibliotheken mit derart verteilten Modullaufzeiten anwenden lassen. Um das Speedup-Verhalten unter allgemeineren Bedingungen besser zu verstehen, wird nachfolgend die Familie der zweiparametrischen gefensterten Gaußdichten entsprechend Abschnitt 2.2.3 untersucht. Nach wie vor soll als Hardwaremodell das Zwei-Prozessor-System nach Bild 2.3 dienen.

Für das erzeugende Element mit $G_1 = 1$ ergibt sich nach der gleichen Herleitung, die zu Gleichung (3.9) geführt hat, unter Verwendung von Gaußdichten die folgende Verbunddichte für die Laufzeiten $p_\tau = p_a + p_c$ und p_b:

$$f(p_\tau, p_b \mid p_a > p_b) = 2 \cdot {r_g}^3 \cdot \sqrt{\pi}\sigma \cdot e^{-\frac{1}{2} \cdot \frac{(p_\tau - 2\mu)^2}{2\sigma^2}} \cdot e^{-\frac{(p_b - \mu)^2}{2\sigma^2}}$$

$$\cdot \begin{cases} \text{erf}\,[h(p_\tau - p_{min})] - \text{erf}\,[h(p_b)] & ;\ \text{Teilgebiet } (1) \\ \text{erf}\,[h(p_{max})] - \text{erf}\,[h(p_b)] & ;\ \text{Teilgebiet } (2) \\ \text{erf}\,[h(p_{max})] - \text{erf}\,[h(p_\tau - p_{max})] & ;\ \text{Teilgebiet } (3) \\ 0 & ;\ \text{sonst} \end{cases} \tag{3.12}$$

Dabei ist erf $[\gamma]$ die Fehlerfunktion nach Gleichung (2.3) und $h(x) := \frac{2x-\tau}{2\sigma}$. Nach wie vor gilt die Skizze in Bild 3.6(b). Die Verteilungsfunktion $F_H(\mu, \sigma, s)$ berechnet sich wiederum aus der Integration der Verbunddichte über das Gebiet oberhalb der Geraden $p_b = \eta \cdot p_b$ [70]. Das Bild 3.13 zeigt $F_H(\mu, \sigma, s)$ für einen Beispielfall, in dem das Fenster $[p_{min}; p_{max}] = [1; 10]$ ist und $\sigma = 1$ konstant. Offensichtlich ist $F_H(\mu,s)$ nicht symmetrisch zur Intervallmitte $\mu_0 = 5{,}5$, obwohl die Dichte mit

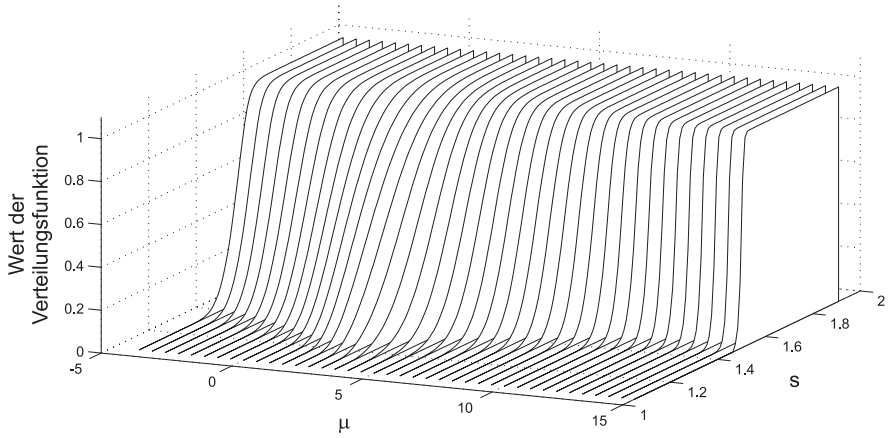


Bild 3.13 Verteilungsfunktion $F_H(\mu, \sigma, s)$ am Beispiel $\sigma = 1$

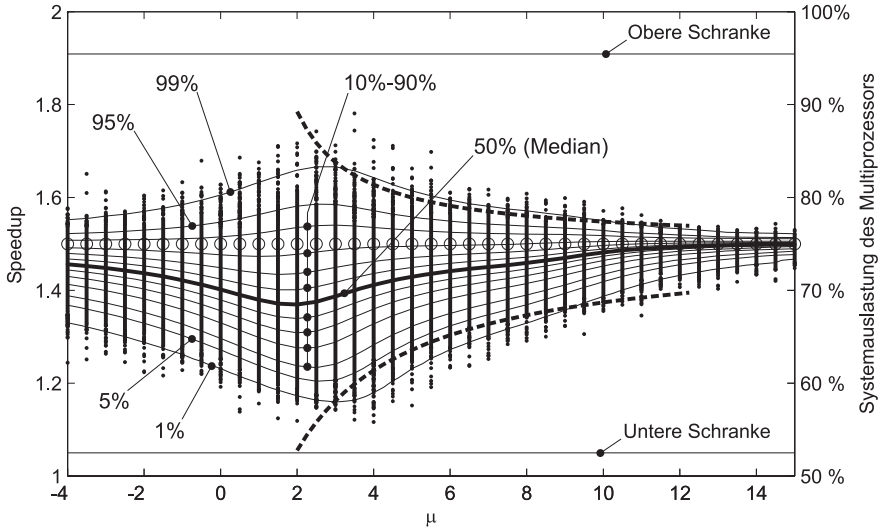


Bild 3.14 Speedup für das erzeugende Element, $G_1 = 1$, $L = 2$, $\sigma = 1$

$\mu = \mu_0$ symmetrisch im Intervall [1;10] liegt und die Gestalt der Paare gefensterter Gaußdichten für $\mu = \mu_0 \pm \Delta\mu$ spiegelsymmetrisch zu diesem Punkt sind. Eine ähnliche Beoachtung ist bereits im Zusammenhang mit den Exponentialdichten gemacht worden.

Das Bild 3.14 zeigt die Rechnersimulation für $G_1 = 1$ und überlagert die Höhenlinien von $F_H(\mu,s)$ für ausgewählte Quantile sowie die obere und untere Schranke für den Speedup [79]. Die beiden fett gestrichelten Linien sind Näherungen für die Höhenlinien des 1%- und des 99%-Quantils im Bereich $4 \leq \mu \leq 7$. Die Kreise zeigen nach wie vor die Vorhersage des Speedups $s = 1{,}5$ mit Einheitslaufzeiten an.

Der Verlauf der Medianlinie zeigt, dass die Mehrheit aller Realisierungen einen Speedup unter $s = 1{,}5$ liefert, es sei denn der Dichteparameter wird $\mu > 11$. Eine wichtige Beobachtung betrifft die Gestalt der Höhenlinien, die besonders gut an denen extremer Quantile sichtbar wird: Die Linie des 1%-Quantils ist streng konkav und die Linie des 99%-Quantils ist streng konvex im Intervall $4 \leq \mu \leq 7$. Diese Beobachtung dient als Ansatzpunkt für die nachfolgende Näherungsrechnung.

Auf den ersten Blick scheint die Verringerung der Speedup-Streuung im Intervall $[4;7]$ der Intuition zu widersprechen: Zuvor ist bereits festgestellt worden, dass der Speedup stets $s = 1{,}5$ beträgt, und zwar unabhängig davon, welche konstante Laufzeit als Einheitslaufzeit eingesetzt wird. Die Laufzeiten, die mit dem konstanten $\sigma = 1$ und $\mu \in [4;7]$ aus den Zufallsexperimenten erzeugt werden, sind zwar nicht konstant, aber liegen alle dicht an dem Mittelwert $p_m = \mu$, weil σ klein ist im Vergleich zu dem Intervall $[p_{min}; p_{max}] = [1;10]$. Daher ist zu erwarten, dass das Speedup-Verhalten sich kaum verändert, weil das Intervall $[\mu - 3\sigma; \mu + 3\sigma]$ vollständig in dem Intervall $[p_{min}; p_{max}]$ enthalten ist. Der effektive Mittelwert μ_{eff} (das erste Moment der gefensterten Gaußdichte) entspricht dann in etwa dem Dichteparameter μ, und die effektive Streuung σ_{eff} (das zweite Zentralmoment der gefensterten Gaußdichte) entspricht in etwa σ. Ohne Fensterung beschreiben μ und σ Gaußdichten, die im besagten Intervall als gute Näherung für die gefensterten Versionen angenommen werden können.

Diese Näherung wird im Folgenden weiter untersucht, um zu zeigen, dass die effektive relative Laufzeit-Streuung $\sigma_p = \sigma_{eff}/\mu_{eff}$ das beobachtete Speedup-Verhalten erzeugt. Die Höhenlinie für das 99%-Quantil gibt den maximal erreichbaren Speedup $\bar{s}$ in Abhängigkeit von μ an, wobei σ wie in der Simulation konstant bleibt. Hier wird die Wahrscheinlichkeit für beliebige Speedups $s \leq \bar{s}(\mu)$ aus der Wahr-

scheinlichkeit für Laufzeitkombinationen ermittelt, die $p_\tau \geq 2\mu - \sqrt{2}\, r_1\sigma$ und $p_b \leq \mu + r_1\sigma$ aufweisen, was in einer Obergrenze für den Speedup resultiert:

$$\bar{s}(\mu) = 1 + \eta = 1 + \frac{p_b}{p_\tau} = 1 + \frac{\mu + r_1\sigma}{2\mu - \sqrt{2}\, r_1\sigma} \tag{3.13}$$

Alle Laufzeiten sind voneinander unabhängig, also auch p_b und p_τ. Die Gleichung (3.13) gilt streng genommen nur unter der Bedingung $p_a > p_b$. Andernfalls tauschen die Module a und b ihre Rollen, aber $\bar{s}(\mu)$ bleibt gleich. Es ist von Vorteil, die Grenzen für die beide Laufzeiten p_τ und p_b so zu wählen, dass

$$P(p_\tau \geq 2\mu - \sqrt{2}\, r_1\sigma) = P(p_b \leq \mu + r_1\sigma) \tag{3.14}$$

weil dann die Wahrscheinlichkeit $P(s \leq \bar{s})$ mit $\bar{s} = \bar{s}(\mu)$ aus Gleichung (3.13) nur von **einer** unbekannten Konstanten $r_1 \in \mathbb{R}^+$ abhängt. Es sind **beide** Fälle $p_a > p_b$ und $p_a < p_b$ zu betrachten. Die Wahrscheinlichkeit $P(s \leq \bar{s})$ kann folgendermaßen ausgedrückt werden:

$$P(s \leq \bar{s}) = P\left[(\mathcal{E}_1 \vee \mathcal{E}_2) \wedge (p_\tau \geq 2\mu - \sqrt{2} r_1\sigma)\right] \tag{3.15}$$

wobei die Ereignisse $\mathcal{E}_1$ und $\mathcal{E}_2$ abkürzend stehen für

$$\mathcal{E}_1 \;:=\; (p_b \leq \mu + r_1\sigma \mid p_a > p_b) \tag{3.16}$$

$$\mathcal{E}_2 \;:=\; (p_a \leq \mu + r_1\sigma \mid p_a < p_b) \tag{3.17}$$

Die Anwendung der Bayes-Regel [49] auf die Wahrscheinlichkeit $P_1 = P(\mathcal{E}_1)$ liefert einen Ausdruck, dessen Terme mit Integralen über Teilgebieten der $p_a p_b$-Ebene identifiziert werden können:

$$P_1 = \frac{P(p_a > p_b \mid p_b \leq \mu + r_1\sigma) \cdot P(p_b \leq \mu + r_1\sigma)}{P(p_a > p_b)} \tag{3.18}$$

Ein Vergleich mit Bild 3.15(a) zeigt, dass $P(p_a > p_b) = \frac{1}{2}$ (Fläche des gestrichelten Dreiecks im Vergleich zur Fläche des Quadrats $[p_{min}; p_{max}] \times [p_{min}; p_{max}] \in \mathbb{R}^2$) und $P(p_a > p_b \mid p_b \leq \mu + r_1\sigma) \approx \frac{1}{2}$ (die dicht schraffierte Teilfläche entspricht fast der gesamten Dreiecksfläche), solange die Grenze $\mu + r_1\sigma$ nur groß genug gegenüber μ ist, und das ist der Fall, wenn wie hier die Höhenlinie des 99%-Quantils approximiert werden soll. Aus Gleichung (3.18) folgt die einfache Näherung

$$P_1 \approx P(p_b \leq \mu + r_1\sigma) =: \tilde{P}(r_1)$$

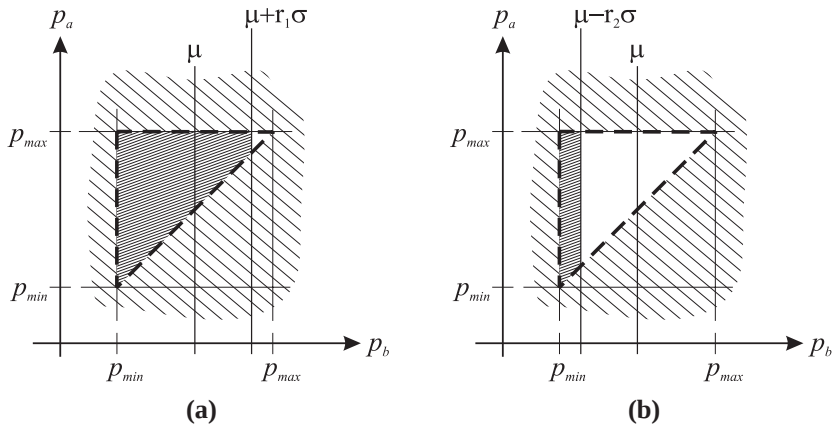


Bild 3.15 Gebiete in der $p_a p_b$-Ebene

Auf die gleiche Weise erhält man $P_2 \approx P(p_a \leq \mu + r_1\sigma) = \tilde{P}(r_1)$ aus $P(\mathcal{E}_2)$ nach Gleichung (3.17). Jetzt fordert man für die Approximation der Höhenlinie des 99%-Quantils:

$$\begin{aligned} P(s \leq \bar{s}) &= P\left[\mathcal{E}_1 \wedge (p_\tau \geq 2\mu - \sqrt{2}r_1\sigma)\right] + P\left[\mathcal{E}_2 \wedge (p_\tau \geq 2\mu - \sqrt{2}r_1\sigma)\right] \\ &\approx 2 \cdot \tilde{P}(r_1) \cdot P(p_\tau \geq 2\mu - \sqrt{2}\, r_1\sigma) \\ &= 2 \cdot [\tilde{P}(r_1)]^2 \qquad ; \text{ nach Gleichung (3.14)} \\ &\stackrel{!}{=} 0.99 \qquad (3.19) \end{aligned}$$

Daraus folgt $\tilde{P}(r_1) = \sqrt{0.5 \cdot 0.99}$ und mit einer Tabelle [12] für das Gaußsche Standardintegral $r_1 \approx 0.54$. Mit diesem Faktor r_1 und $\sigma = 1$ erhält man den Verlauf $\bar{s}(\mu)$ als konvexe, fett gestrichelte Linie in Bild 3.14.

Die Höhenlinie für das 1%-Quantil ist auf ähnliche Weise mit der noch unbekannten Konstanten $r_2 \in \mathbb{R}^+$ auszudrücken:

$$\underline{s}(\mu) = 1 + \frac{\mu - r_2\sigma}{2\mu + \sqrt{2}\, r_2\sigma} \qquad (3.20)$$

Ein Unterschied besteht jedoch in der Näherung für $P_3 = P(\mathcal{E}_3)$, wobei $\mathcal{E}_3$ abkürzend für das Ereignis $\mathcal{E}_3 := (p_b \leq \mu - r_2\sigma \mid p_a > p_b)$ steht. Nach Anwendung der

Bayes-Regel erhält man:

$$P_3 = \frac{P(p_a > p_b \mid p_b \leq \mu - r_2\sigma) \cdot P(p_b \leq \mu - r_2\sigma)}{P(p_a > p_b)} \tag{3.21}$$

Ein Vergleich mit Bild 3.15(b) zeigt, dass jetzt $P(p_a > p_b \mid p_b \leq \mu - r_2\sigma) \approx 1$ (die dicht schraffierte Teilfläche entspricht fast der Fläche des rechteckigen Gebietes $[p_{min}; (\mu - r_2\sigma)] \times [p_{min}; p_{max}] \in \mathbb{R}^2$), solange die Grenze $\mu - r_2\sigma$ nur klein genug gegenüber μ ist, und das ist der Fall, wenn wie hier die Höhenlinie des 1%-Quantils approximiert werden soll. Nach wie vor ist $P(p_a > p_b) = \frac{1}{2}$. Aus Gleichung (3.21) folgt damit die einfache Näherung

$$P_3 \approx 2 \cdot P(p_b \leq \mu - r_2\sigma) = 2 \cdot \tilde{P}(r_2)$$

Aus Symmetriegründen ist ebenfalls $P_4 = P(\mathcal{E}_4) \approx 2 \cdot \tilde{P}(r_2)$, wobei $\mathcal{E}_4$ abkürzend für das Ereignis $\mathcal{E}_4 := (p_a \leq \mu - r_2\sigma \mid p_a < p_b)$ steht. Daher wird hier analog zu Gleichung (3.19) gefordert:

$$P(s \leq \underline{s}(\mu) \approx 4 \cdot [\tilde{P}(r_2)]^2 \stackrel{!}{=} 0.01 \tag{3.22}$$

Aus der Tabelle für das Gaußsche Standardintegral [12] liest man $r_2 \approx 1{,}65$ ab. Das Bild 3.14 zeigt den Verlauf von $\underline{s}(\mu)$ mit dem ermittelten r_2 und $\sigma = 1$ als konkave, fett gestrichelte Linie.

Die Gaußsche Approximation der gefensterten Dichten verfügt in der Tat über eine relative Streuung σ/μ, die sich bei konstantem σ mit wachsendem μ **verändert**, und genau in diesem Punkt erweist sich die anfangs geschilderte Intuition als nicht zutreffend. Die Näherungen und die nummerisch berechneten Höhenlinien zeigen das gleiche Verhalten, und so liegt die Vermutung nahe, dass die effektive relative Laufzeit-Streuung $\sigma_p = \sigma_{eff}/\mu_{eff}$, die gerade im Intervall $[4; 7]$ dem Verhältnis σ/μ sehr nahe kommt, wesentlich für das beobachtete Speedup-Verhalten verantwortlich ist. Das Bild 3.16 zeigt σ_p in Abhängigkeit von μ zusammen mit dem nominalen Verhältnis σ/μ (gestrichelt). Ein Vergleich der durchgezogenen Kurve mit der Höhenlinie für das 99%-Quantil zeigt eine erstaunliche Ähnlichkeit über den gesamten untersuchten Bereich der Variablen μ hinweg.

Die Herleitung von $F_H(\mu, \sigma, s)$ aus Gleichung (3.12) ist für beliebige μ und σ gültig. Folglich lässt sich auch eine Funktion $\sigma(\mu)$ einsetzen, die so gewählt wird, dass σ_p für alle μ konstant ist. Um eine möglichst große Übereinstimmung von

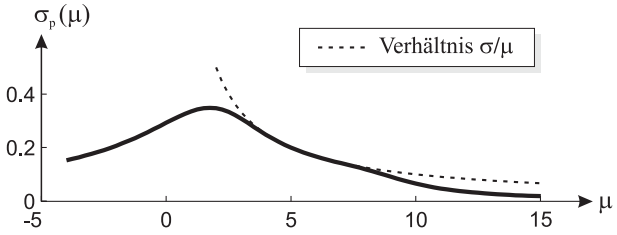


Bild 3.16 Effektive relative Laufzeit-Streuung in Abhängigkeit von μ

σ/μ mit σ_p in der Mitte des Intervalls $[p_{min}; p_{max}] = [1; 10]$ zu erzielen, wird für den Beispielfall mit $\sigma = 1$ die effektive relative Laufzeit-Streuung $\sigma_p = \mu_0^{-1}$ gewählt. Die Funktion $\sigma(\mu)$ kann nur bestimmt werden, indem man die Gleichung $\sigma_{eff}(\mu,\sigma) - \sigma_p \cdot \mu_{eff}(\mu,\sigma) = 0$ nummerisch nach $\sigma(\mu)$ auflöst. Das Bild 3.17 zeigt die Funktion, deren Werte für die nachfolgende Berechnung der Höhenlinien von $F_H(\mu, \sigma(\mu), s)$ bis auf einen Restfehler von $\epsilon(\sigma) < 10^{-4}$ (unabhängig von μ) bestimmt wurden. Auch die Simulation des erzeugenden Elementes mit $G_1 = 1$ kann ohne Weiteres unter Verwendung der Funktion $\sigma(\mu)$ durchgeführt werden.

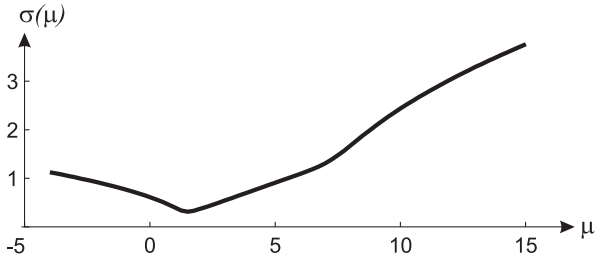


Bild 3.17 Nummerisch ermittelter Dichteparameter $\sigma(\mu)$ für konstantes σ_p

Das Bild 3.18 zeigt das Speedup-Verhalten unter der Bedingung eines konstanten $\sigma_p = \mu_0^{-1}$. Naturgemäß erfüllen die Höhenlinien im Interval $[4; 7]$ jetzt die Erwartungen an einen konstanten Verlauf. Dabei ist festzustellen, dass die Medianlinie den Referenzwert $s = 1{,}5$ nicht erreicht. Erstaunlicherweise erstreckt sich dieses Verhalten sogar auf das größere Intervall $[2; 7]$. Für $\mu < 1$ verlaufen die Höhenlinien ebenfalls konstant, und die Streuung des Speedups ist hier am geringsten. Im Gegensatz dazu steigt die Speedup-Streuung für $\mu > 7$ im beobachteten Parameterbereich stetig an. Diese Ergebnisse widerlegen die Vermutung, dass ein konstantes

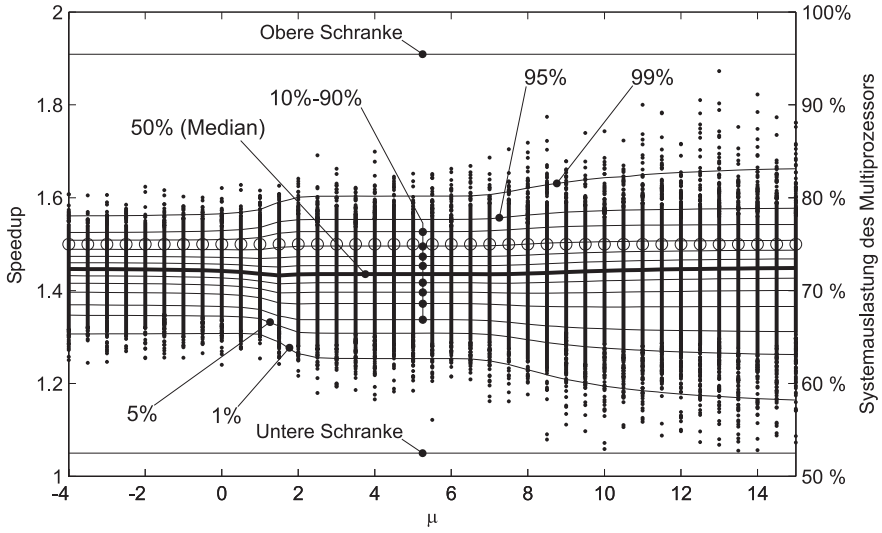


Bild 3.18 Speedup für das erzeugende Element, $G_1 = 1$, $L = 2$, σ_p konstant

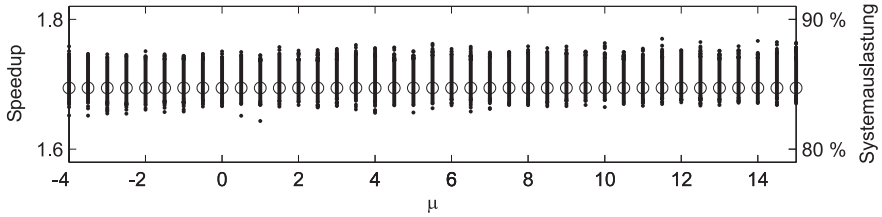


Bild 3.19 Simulation von Graph $\mathcal{G}_B$ mit $G_1 = 10$, $L = 2$, σ_p konstant

σ_p generell eine konstante Speedup-Streuung nach sich zieht. Zumindest im Falle des erzeugenden Elementes mit $G_1 = 1$ hat die Gestalt der Dichte $f_P(p)$ ebenfalls einen Einfluß.

Größere Graphen sind nach wie vor analytisch äußerst schwer zu behandeln. Daher folgt auch hier wieder die Simulation des Graphen aus Bild 2.10 für $G_1 = 10$ mit Hilfe des Hu-Algorithmus. Das Bild 3.19 zeigt als Ergebnis einen relativ hohen

Speedup, der unabhängig von μ kaum streut. Die Vorhersage mit Einheitslaufzeiten ist wie üblich mit Kreisen gekennzeichnet.

Aus den Untersuchungen mit der zweiparametrischen Gaußdichte werden folgende Schlussfolgerungen gezogen [79]:

- Die effektive relative Streuung der Laufzeiten, $\sigma_p = \sigma_{eff} / \mu_{eff}$ hat einen wesentlichen Einfluss auf die Speedup-Streuung bei Mod-SDR-Systemen.
- Die Abhängigkeit der Speedup-Streuung von der Gestalt der Laufzeitdichte, die beim erzeugenden Element zu beobachten ist, wird mit steigender Granularität G_1 schnell vernachlässigbar.

Wenn also die gegebene Laufzeitverteilung einer Softwarebibliothek durch einen bestimmten Entwurfsprozess beeinflusst wird, so sollte das Entwurfsziel nach den vorangegangenen Erkenntnissen gerade **nicht** die Formung der Dichtefunktion auf eine bestimmte Gestalt sein, sondern die Verminderung der effektiven relativen Laufzeit-Streuung σ_p.

3.4 Zusammenfassung

In diesem Kapitel ist die Entwurfsaufgabe für Modulares Software Defined Radio präzisiert und formal in das Gebiet der linearen ganzzahligen Optimierung eingeordnet worden. Derartige Aufgaben sind als NP-schwer bekannt, und damit ist eine optimale Ablaufplanung nur möglich, wenn das Betriebssystem eines Mod-SDRs Branch-and-Bound-Methoden anwendet. Die Entscheidung über die Machbarkeit oder Rückweisung einer Realisierungsanforderung soll aber im Endgerät möglichst schnell getroffen werden, und damit scheidet die potentiell langwierige Branch-and-Bound-Suche nach der optimalen Lösung aus. Als Alternative wurde der Hu-Algorithmus vorgestellt, der mit relativ einfachen Mitteln schnell suboptimale Ablaufpläne liefert.

Der Hu-Algorithmus ist in diesem Kapitel nur unter vereinfachten Randbedingungen zum Einsatz gekommen, denn weder der Speicherbedarf von Modulen noch die Inter-Prozessor-Kommunikation ist bei der suboptimalen Ablaufplanung bisher berücksichtigt worden. Weiterhin sind die Begriffe Granularität und Speedup eingeführt worden. Der Begriff der Granularität hat sich als unzureichend erwiesen, Software für eine Mod-SDR-Realisierung umfassend zu beschreiben. In gleicher

Weise wie abstrakte Graphenmodelle mit Zufallslaufzeiten es erlauben, sich von den konkreten logischen Strukturen existierender Mobilfunkstandards zu lösen, erlaubt es der Speedup als relatives Qualitätsmaß für den Betriebspunkt eines Mod-SDR, sich von konkreten Echtzeitanforderungen zu lösen. Je größer der erreichbare Speedup für ein Quantil von Realisierungen, desto besser die Ablaufplanung.

Unter diesen Bedingungen sind hier die Speedup-Eigenschaften einfacher Graphen und größerer Baumgraphen mit unterschiedlicher Verteilung der Rechenlast auf Zweige untersucht worden. Während einfache Graphen noch analytisch zu behandeln sind (und die Simulation praktisch keinen Algorithmus für die Ablaufplanung erfordert), ist die Untersuchung größerer logischer Strukturen nur noch per Simulation mit Hilfe des Hu-Algorithmus möglich. Der entscheidende Nachteil besteht dabei in der groben Vereinfachung der Randbedingungen. Im folgenden Kapitel werden deshalb Algorithmen zur Partitionierung und Ablaufplanung vorgestellt, die den Speicherbedarf von Modulen und Kosten für den Datenaustausch zwischen Prozessoren ausdrücklich berücksichtigen.

4 Partitionierung und Ablaufplanung

Unter Partitionierung versteht man den Prozess der Abbildung von logisch abhängigen Modulen auf Prozessoren, wohingegen die Ablaufplanung die Festlegung der Startzeiten auf den Prozessoren betrifft. Beide Aufgaben sind im Allgemeinen nicht unabhängig voneinander, und diese Aussage bezieht sich in erster Linie auf heterogene Multiprozessorsysteme, bei denen die Laufzeit eines Moduls davon abhängig ist, auf welchem Prozessor es gestartet wird. Als typisches Beispiel für ein solches heterogenes System kann eine Kombination aus FPGA und DSP dienen, wenn angenommen wird, dass das FPGA im Hinblick auf eine bestimmte Signalverarbeitungsaufgabe relativ unflexibel, aber schnell ist, der DSP hingegen sehr flexibel, aber vergleichsweise langsam. Die Entwurfsaufgabe besteht dann darin zu entscheiden, auf welchen der beiden Prozessoren die Aufgabe am besten abgebildet werden soll. Einerseits erfordert die Ablaufplanung bereits die Kenntnis einer Laufzeit und damit die Kenntnis des Partitionierungsergebnisses. Folglich sollte die Partitionierung vor der Ablaufplanung erfolgen. Andererseits erfordert der Vorgang der Partitionierung Kenntnisse über die Verfügbarkeit eines bestimmten Prozessors zu einem fraglichen Zeitpunkt. Diese Kenntnis über Zeitabläufe kann nur von der Ablaufplanung stammen, und folglich sollte die Ablaufplanung vor der Partitionierung erfolgen. Diese gegenseitige Abhängigkeit ist es, die den gemeinsamen Entwurf von Hardware und Software (*engl.* hardware/software co-design) in heterogenen Systemen äußerst schwierig macht.

Das Modell des symmetrischen Multiprozessors, das im Rahmen der vorliegenden Arbeit betrachtet wird, ist jedoch der Prototyp des homogenen Hardware-Systems, und genau diese Einschränkung erlaubt es, den Kreis der gegenseitigen Abhängigkeit von Partitionierung und Ablaufplanung zu durchbrechen.

4.1 Grundsätzliches Vorgehen

Da bei einem homogenen System unabhängig vom ausführenden Prozessor die gleiche Modullaufzeit vergeht, erfolgt die Abbildung auf Hardware in zwei ge-

trennten Schritten: Zunächst wird die in Form eines Graphen gegebene Software partitioniert, und danach wird ein Ablaufplan erstellt. Schließlich kann der Speedup gemessen werden, indem man das Verhältnis der Echtzeitperioden ΔT_{mono} des Einzelprozessors und ΔT_{multi} des Multiprozessors berechnet. Bevor der allgemeine Fall des Mehrprozessorsystems mit $L \in \mathbb{N}$ angegangen werden kann, sollen in dieser Arbeit die Möglichkeiten beim Betrieb von zwei Prozessoren zunächst vollauf verstanden werden. Aus diesem Grund wird im Folgenden $L = 2$ gesetzt. Die beiden Partitionen sind disjunkte Teilmengen der Knotenmenge $\mathbb{M}$ und werden mit $\mathbb{A}$ und $\mathbb{B}$ bezeichnet, wobei $\mathbb{A} \cup \mathbb{B} = \mathbb{M}$ gilt.

Der Graph besteht ursprünglich aus Knoten und gerichteten Kanten, wobei die Kanten entweder mit einem logischen Datenfluss verknüpft sind oder reine Vorgänger-Nachfolger-Beziehungen zwischen Knoten etablieren. Die Knoten stehen entweder für einfache Module oder ganze nachrichtentechnische Funktionen. In jedem Fall tragen sie ein eindeutiges Laufzeitattribut und sollen reguläre Knoten genannt werden, da sie unmittelbar zur Erfüllung der gewünschten Signalverarbeitungsaufgaben eines Mobilfunkstandards beitragen.

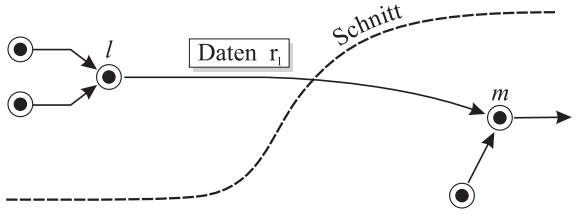


Bild 4.1 Schnitt einer gerichteten Kante des Graphen

Ein Partitionierungsverfahren liefert mit Angabe der Partitionen $\mathbb{A}$ und $\mathbb{B}$ auch immer einen Schnitt durch bestimmte Kanten des Graphen (siehe Bild 4.1). Dieser Schnitt trennt benachbarte Knoten, die auf verschiedene Prozessoren abgebildet werden. Fließen normalerweise Daten entlang einer solchen Kante, so verläuft der physikalische Datenaustausch nach dem Partitionieren auf asynchrone Weise unter Verwendung des Shared Memory (siehe Abschnitt 2.1.1). Der Quellprozessor schreibt einen Datenblock über den Bus in das Shared Memory, und der Zielprozessor liest diese Daten zu einem späteren Zeitpunkt mit einem weiteren Buszugriff aus dem Shared Memory aus. Das Bild 4.2 zeigt, wie dieser Bustransfer mit Graphelementen dargestellt werden kann.

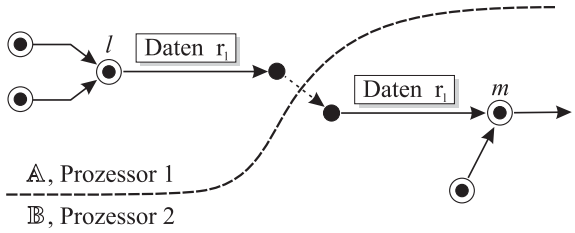


Bild 4.2 Erweiterung des Graphen um Bustransfer-Knoten

Zu beiden Seiten des Schnittes werden zusätzliche Knoten eingeführt, die den Schreib- und Lesezugriff auf das Shared Memory modellieren. Diese Paare von Bustransfer-Knoten werden erst beim Übergang vom logischen auf den physikalischen Datenfluß nötig. Sie erweitern den ursprünglichen Graphen stets um eine gerade Anzahl von Knoten. Im Gegensatz zu den regulären Knoten tragen Bustransfer-Knoten nicht unmittelbar zur Lösung von Signalverarbeitungsaufgaben im Sinne eines Mobilfunkstandards bei, sie sind aber bei der Abbildung auf beliebige Mehrprozessorsysteme unumgänglich. Im Fall $L = 2$ laufen auf beiden Prozessoren also zusätzliche Softwaremodule ab, deren einzige Aufgabe die physikalische Realisierung der Inter-Prozessor-Kommunikation ist. Durch Modellierung mit Knoten kann eine Ablaufplanung des Betriebssystems, die auf den Umgang mit Knoten ausgelegt ist, die regulären Knoten eines Graphen und Bustransfers gleich behandeln. Lediglich die Laufzeit, die Bustransfer-Knoten zugeordnet ist, muss anders bestimmt werden als die Laufzeit regulärer Knoten, die dem erweiterten Ressourcen-Laufzeit-Modell folgt.

In Abschnitt 2.2.4 ist die Proportionalitätskonstante α eingeführt worden, die als absolute Prozessorgeschwindigkeit bei der Erzeugung von Datenmengen am Ausgang von Softwaremodulen zu interpretieren war. Mittlerweile wird die Qualität eines Mod-SDR-Entwurfs aber in Form von Speedup gemessen. Dieses relative Maß ist eingeführt worden, um eine Loslösung der Entwurfsrichtlinien von konkreten Echtzeitanforderungen zu erreichen. Folglich ist der konkrete Wert von α ohne Bedeutung, denn er kann bei der Berechnung des Speedups über das Verhältnis von ΔT_{mono} und ΔT_{multi} gekürzt werden. Ein bestimmtes α sorgt lediglich für eine Streckung oder Stauchung der Zeitachse eines typischen Ablaufplanes, aber der Speedup bleibt davon unberührt. An dieser Stelle wird nochmals deutlich, wie wenig sinnvoll die Angabe absoluter Werte für die Rechenleistung eines betrachteten Hardware-Systems im Zusammenhang mit Software Defined Radios ist.

Um auch unter Einbeziehung eines Bussystems weiterhin allgemein gültige Erkenntnisse über Entwurf und Betrieb von Mod-SDRs gewinnen zu können, wird hier die relative Busgeschwindigkeit $\beta \in \mathbb{R}^+$ eingeführt. Diese Konstante gibt an, wieviel mal schneller ein Prozessor einen Datenblock über den Bus transferieren kann anstatt den gleichen Datenblock durch Signalverarbeitung als Zwischenergebnis am Ausgang eines Moduls zu erzeugen. Das Laufzeitmodell für jeden einzelnen Bustransfer-Knoten eine Paares lautet also:

$$p_{\langle l,m \rangle} = \alpha \cdot \beta^{-1} \cdot r_l \qquad \begin{array}{l} \text{für diejenigen Kanten } \langle l,m \rangle \subset \text{Kantenmenge } \mathbb{K}, \\ \text{die von dem Schnitt betroffen sind.} \end{array} \tag{4.1}$$

Im Gegensatz zu den Laufzeiten regulärer Knoten hängen die Laufzeiten der Bustransfer-Knoten nicht über die Zufallsvariable C von der hochflexiblen SDR-Umgebung ab, sondern deterministisch von der lokalen Partitionierungsentscheidung und der Leistungsfähigkeit des Bussystems. Die relative Busgeschwindigkeit β ist als ein Entwurfsparameter des Mod-SDR zu sehen, der die grundlegenden Hardware-Eigenschaften eines Endgerätes unmittelbar betrifft.

4.2 Ansätze zur Partitionierung

Indirekt ist durch die endliche Busgeschwindigkeit aber auch die Arbeitsweise des Betriebssystems betroffen, da in der Partitionierungsphase zunächst über die Abbildung von Knoten auf Prozessoren entschieden werden muss. Je mehr Kanten geschnitten werden, desto mehr Bustransfers sind nötig. Im gleichen Maße steigt der Anteil von Prozessorzeit, die auf die Ausführung von Bustransfer-Knoten verwendet wird, nicht auf reguläre Signalverarbeitung. Daher ist ein Ziel der Partitionierung sicherlich, die Summe der Laufzeiten aller Bustransfer-Knoten zu minimieren. Andererseits ist die gesamte Signalverarbeitung auf beide Prozessoren möglichst ausgewogen zu verteilen, um maximalen Speedup zu erreichen.

4.2.1 Implizite Partitionierung mit dem Hu-Algorithmus

Der Hu-Algorithmus ist ursprünglich als Methode zur Ablaufplanung unter einfachsten Randbedingungen entwickelt worden, nicht zur Partitionierung. Eine Bewertung der Datenmenge, die zwischen Prozessoren mit endlicher Geschwindigkeit

ausgetauscht werden muss, findet nicht statt. Dennoch wählt der Algorithmus aus einer Kandidatenliste ohne Weiteres Knoten aus und weist sie einem freien Prozessor zur Ausführung zu. Auf diese Weise wird eine implizite Partitionierung vorgenommen. Eine erste Entwurfsannahme könnte sein, dass auf diese Weise ausreichend hoher Speedup erzeugt wird, wenn die Busgeschwindigkeit nur hoch genug ist und die vereinfachten Randbedingungen nur näherungsweise erfüllt sind.

Hohe Busgeschwindigkeiten sind aber im Hinblick auf die Minimierung der dynamischen Verlustleistung in CMOS-Schaltungen [112] zu vermeiden. Das Bussystem stellt ein großes Teilsystem der Hardware dar, da es alle Prozessor- und Speicherkomponenten miteinander verbindet, und je schneller dieses Teilsystem im Vergleich zu den Prozessoren getaktet wird, desto größer die elektrische Verlustleistung.

Obwohl nicht zu erwarten ist, dass der Hu-Algorithmus unter realistischen Betriebsbedingungen sehr gute Speedup-Ergebnisse liefert, kann die implizite Partitionierung zumindest als Referenz für den Vergleich mit anderen Ansätzen dienen.

4.2.2 Kernighan-Lin-Algorithmus

Dieses Partitionierungsverfahren ist von Kernighan und Lin ursprünglich entwickelt worden, um elektronische Bauelemente auf gedruckte Leiterplatten so zu verteilen, dass die Anzahl der Bauelemente in beiden Partitionen ausgewogen und gleichzeitig die benötigte Anzahl elektrischer Verbindungen zwischen den Leiterplatten minimal ist [47]. Auch dort wird zunächst mit zwei Partitionen gearbeitet. Die potentiellen Kosten $w_{l,m} \in R_0^+$ für einzelne Verbindungen zwischen Bauelementen werden den Kanten des Graphen als Gewichte zugeordnet. Es gilt allgemein

$$w_{l,m} = w_{m,l} \tag{4.2}$$

$$w_{l,m} \neq 0 \quad \Leftrightarrow \quad \exists \langle l,m \rangle \in \mathbb{K} \tag{4.3}$$

und im Speziellen bei Kernighan und Lin $w_{m,m} = 0 \ \forall \ m : 1 \leq m \leq M$. Die Kantengewichte werden üblicherweise in einer symmetrischen Gewichtsmatrix $\mathbf{W} = [w_{l,m}], \ 1 \leq l,m \leq M$ zusammengefasst. Als externe Verbindungskosten E_a und interne Verbindungskosten I_a eines Knotens $a \in \mathbb{A}$ werden definiert:

$$E_a := \sum_{y \in \mathbb{B}} w_{a,y} \qquad \text{und} \qquad I_a := \sum_{x \in \mathbb{A}} w_{a,x} \tag{4.4}$$

Analog dazu lassen sich externe und interne Kosten für $b \in \mathbb{B}$ definieren [47].

Während die einzelnen Beiträge $w_{l,m}$ als potentielle Einzelkosten, die bei einem gegebenen Graphen entstehen **können**, zu interpretieren sind, ist die Summe aller externen Kosten ein abstraktes Maß für diejenigen Kosten, die in der Tat durch elektrische Verbindungen zwischen zwei Leiterplatten entstehen. Entsprechend fasst die Summe der internen Kosten alle diejenigen potentiellen Kosten zusammen, die durch eine bestimmte Partitionierung **vermieden** werden. Generell ist die Minimierung der externen Kosten äquivalent zur Maximierung der internen Kosten.

Das Ziel des KL-Algorithmus ist die Minimierung der externen Kosten. Dabei kommt ein lokales Suchverfahren zu Einsatz, das ausgehend von einer bereits ausgewogenen Startkonfiguration Knotenpaare zwischen den Partitionen austauscht, bis keine Verringerung der externen Verbindungskosten mehr zu erreichen ist. Zwei Aspekte des KL-Algorithmus sind von besonderer Bedeutung: Die individuelle Bewertung aller Knoten $m : 1 \leq m \leq M$ mit einem charakteristischen Wert D_m und die Auswahl von Knoten für die Paarbildung. Ursprünglich wird der charakteristische Wert aller Knoten eines Graphen als die Differenz $D_{link,m} = (E_m - I_m)$ zwischen externen und internen Kosten des jeweiligen Knotens angegeben. Der Wert $D_{link,m}$ hängt dabei nur von dem Knoten m selbst ab, und nicht etwa von der Paarung mit anderen Knoten. Dennoch steht D_a für $a \in \mathbb{A}$ unmittelbar im Zusammenhang mit dem potentiellen Gewinn, den man beim Austausch mit einem Knoten $b \in \mathbb{B}$ erzielen kann, und zwar über die Summe $\overline{g} = D_{link,a} + D_{link,b}$, die eine obere Grenze für den tatsächlichen Gewinn darstellt. Ein positiver Gewinn entsteht, wenn die Summe aller externen Kosten durch einen Knotenpaartausch reduziert werden kann.

Der Ansatz von Kernighan und Lin sieht zwei ineinander geschachtelte Schleifen vor. In der äußeren Schleife werden ganze Sätze von Knotenpaaren zwischen den Partitionen ausgetauscht, solange der Gesamtgewinn positiv ist. In der inneren Schleife wird jeweils ein Satz von Knotenpaaren zusammengestellt, der den Gesamtgewinn maximiert. Dazu sind in jedem Schritt genau diejenigen Kandidaten aus $\mathbb{A}$ und $\mathbb{B}$ auszuwählen, die tatsächlich beim nächsten Paartausch den größten Beitrag zum Gesamtgewinn eines Knotensatzes erzielen. In [47] wird gezeigt, dass es besonders vorteilhaft ist, die Knoten einer jeden Partition entsprechend ihrem charakteristischen Wert zu ordnen, denn damit steht eine Folge von korrespondierenden Knotenpaaren zur Verfügung, deren potentielle Gewinnbeiträge $\overline{g}$ lokal maximal sind und von Paar zu Paar monoton fallen. Nun wird das Paar mit dem maximalen tatsächlichen Gewinn g_{max} gesucht. Dazu wird für die Folgenglieder der

tatsächliche Gewinn g aus der oberen Grenze $\overline{g}$ und einem nicht-negativen Korrekturterm berechnet. Die Suche kann sofort abgebrochen werden, wenn eine obere Grenze angetroffen wird, die unter dem bereits gefundenen g_{max} liegt.

Mit Blick auf die Partitionierung in einem Mod-SDR scheint der KL-Algorithmus die analoge Aufgabe unmittelbar zu lösen: Das Betriebssystem muss Knoten auf Prozessoren ausgewogen verteilen und dabei gleichzeitig die Kosten für Inter-Prozessor-Kommunikation minimieren. Ein erheblicher Unterschied besteht aber darin, dass für die Echtzeitverarbeitung die Summe aller Laufzeiten einer Partition viel wichtiger ist als die reine Knotenanzahl und dass Kosten nicht nur durch den Datenaustausch zwischen Prozessoren entstehen können, sondern auch durch Leerlaufzeiten, wenn die Arbeitslast ungleich verteilt ist. Ein Prozessor führt dann reguläre Knoten bis zum Erreichen der Echtzeitperiode ΔT aus, während der zweite Prozessor bereits leerläuft. Daher werden die charakteristischen Werte der Knoten für den Einsatz des KL-Algorithmus bei Mod-SDR folgendermaßen verändert [80]:

$$D_a := (E_a - I_a) + (\Delta p + 2p_a) = D_{link,a} + D_{load,a} \tag{4.5}$$

$$D_b := (E_b - I_b) + (\Delta p - 2p_b) = D_{link,b} + D_{load,b} \tag{4.6}$$

wobei $D_{load,m} = (\Delta p + 2p_m)$, $a \in \mathbb{A}$ und $b \in \mathbb{B}$. Mit $\mathbb{A}$ wird hier immer die (bezüglich der Summe ihrer Laufzeiten) größere Partition bezeichnet, so dass die Differenz der Gesamtlaufzeiten Δp stets nicht-negativ ist:

$$\Delta p = \sum_{a \in \mathbb{A}} p_a - \sum_{b \in \mathbb{B}} p_b \geq 0 \tag{4.7}$$

Es ist hinreichend, diesen Fall zu betrachten, da sonst lediglich die Bezeichnungen a und b bzw. $\mathbb{A}$ und $\mathbb{B}$ zu vertauschen sind, um das gleiche algorithmische Verhalten zu erreichen.

Der ursprüngliche Anteil $D_{link,m}$ an einem charakteristischen Wert D_m wird unverändert aus [47] übernommen, wobei die Kantengewichte auf

$$w_{l,m} = 2 \cdot p_{\langle l,m \rangle} \tag{4.8}$$

gesetzt werden. Die potentiellen Verbindungskosten zwischen regulären Knoten treten in Form von Laufzeiten der Bustransfer-Knoten auf, und da Quell- und Zielprozessor jeweils einen solchen zugeteilt bekommen, betragen die Kosten für jede geschnittene Kante tatsächlich das Doppelte der Laufzeit $p_{\langle l,m \rangle}$ nach Gleichung

(4.1). Auch mit der erweiterten Definition der charakteristischen Werte gemäß der Gleichungen (4.5) und (4.6) wird der tatsächlich erzielbare Gewinn g beim Austausch von $a \in \mathbb{A}$ gegen $b \in \mathbb{B}$ von oben beschränkt:

$$\overline{g} = D_a + D_b = D_{link} + D_{load} \tag{4.9}$$

Der Term $D_{link} = D_{link,a} + D_{link,b}$ wird klassisch [47] weiterverarbeitet, wohingegen der neue Term $D_{load} = D_{load,a} + D_{load,b}$ als obere Schranke für den Gewinn G_{load} anders behandelt werden muss. Das Bild 4.3(a) zeigt D_{load} und G_{load} als Funktion der Laufzeitdifferenz $(p_a - p_b)$ der beiden ausgetauschten Knoten.

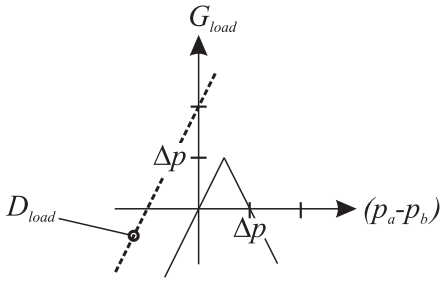
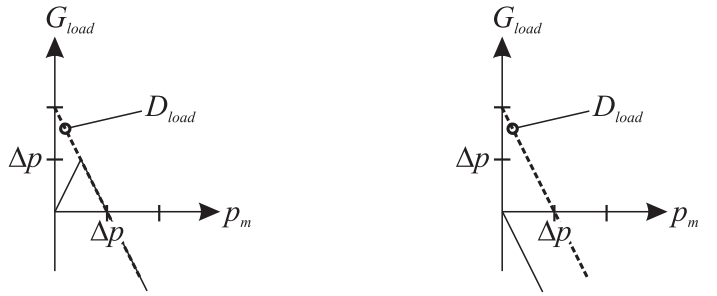


Bild 4.3 Obere Schranke D_{load} (gestrichelt) für den Gewinn G_{load}

Abhängig von der Bedingung $2(p_a - p_b) \lesseqgtr \Delta p$ ist entweder $2\Delta p$ oder $4(p_a - p_b)$ von der Grenze D_{load} zu subtrahieren, um den tatsächlichen Gewinn zu erhalten. Für die Anwendung des KL-Algorithmus ist es besonders wichtig, dass beide Korrekturterme nicht-negativ sind. Nur so ist sichergestellt, dass alle in [47] geschilderten Eigenschaften der charakteristischen Werte erhalten bleiben. Insbesondere kann die Bewertung, Sortierung und Auswahl von Knoten dann unverändert übernommen werden.

Kernighan und Lin schlagen in ihrer Arbeit auch eine Methode vor, um Knoten unterschiedlicher Größe zu behandeln. Es ist unmittelbar einleuchtend, diese ”Größe” mit der Laufzeit eines Knotens zu identifizieren. Reelle Laufzeiten sollten demnach in ganzzahlige Vielfache einer Laufzeitauflösung Δt quantisiert und mit Hilfe unendlich großer Kantengewichte zu Gruppen zusammengeschlossen werden. Das Dilemma dieses Ansatzes ist aber bereits aus dem Abschnitt 3.1 bekannt: Einerseits vergrößert eine gute Zeitauflösung den Gesamtumfang der Partitionierungsaufgabe, andererseits leidet eine Partitionierung mit geringer Zeitauflösung am starken



(a) $\sum_{a\in\mathbb{A}} p_a > \sum_{b\in\mathbb{B}} p_b$ **(b)** $\sum_{a\in\mathbb{A}} p_a < \sum_{b\in\mathbb{B}} p_b$

Bild 4.4 Obere Schranken $D_{load,m,1}$ (gestrichelt) für den Gewinn G_{load}

Quantisierungsrauschen der Lösung. Die in dieser Arbeit vorgestellte Erweiterung der charakteristischen Werte nach den Gleichungen (4.5) und (4.6) vermeidet dieses Dilemma voll und ganz.

Ein weiterer Unterschied zu den ursprünglichen Entwurfsbedingungen von Kernighan und Lin besteht darin, dass für die Startkonfiguration weder die gleiche Knotenzahl pro Partition noch eine ausgeglichene Gesamtlaufzeit angenommen werden kann. Aus diesem Grund wird es notwendig, neben dem Austausch von Knotenpaaren auch den Partitionswechsel einzelner Knoten zu beherrschen. Der charakteristische Wert eines Knotens wird dafür wie folgt definiert:

$$\overline{g}_1 = D_{m,1} := (E_m - I_m) + (2\Delta p - 2p_m) \tag{4.10}$$

Der klassische Term $D_{link,m,1} = E_m - I_m$ wird auch hier verwendet und um die obere Schranke $D_{load,m,1} = 2\Delta p - 2p_m$ für den Gewinn G_{load} ergänzt. Die Bilder 4.4(a) und 4.4(b) zeigen diese obere Schranke und den tatsächlich erreichbaren Gewinn in Abhängigkeit von $p_m > 0$ mit $m \in \mathbb{A}$ für beide Fälle $\sum_{a\in\mathbb{A}} p_a \lessgtr \sum_{b\in\mathbb{B}} p_b$, die getrennt betrachtet werden müssen. Im Fall von Bild 4.4(a) gilt $\sum_{a\in\mathbb{A}} p_a \geq \sum_{b\in\mathbb{B}} p_b$, und die obere Grenze ist um den Term $(4p_m - 2\Delta p) > 0$ zu korrigieren, wenn $(2p_m < \Delta p)$ gilt. Sonst ist bereits $D_{load,m,1} = G_{load,m,1}$. Im Fall von Bild 4.4(b) gilt $\sum_{a\in\mathbb{A}} p_a < \sum_{b\in\mathbb{B}} p_b$, und der Übergang eines Knotens $m \in \mathbb{A}$ nach $\mathbb{B}$ zieht sicherlich einen negativen Gewinn nach sich. Folglich muss $D_{load,m,1}$ dann um $2\Delta p$ nach unten korrigiert werden. Es ist zu beachten, dass auch beim Partitionswechsel einzelner Knoten alle Korrekturterme nicht-negativ und damit die Voraussetzungen für den Ansatz von Kernighan und Lin gegeben sind.

Der in dieser Arbeit erweiterte KL-Algorithmus hat den Vorteil, dass er reellwertige potentielle Kosten für den Datenaustausch zwischen Prozessoren und für unausgewogene Arbeitslast gleichzeitig berücksichtigt. Auf diese Weise können Kompromisse zwischen den beiden Kostenarten eingegangen werden, z.B. läßt der Algorithmus durchaus den Wechsel eines Knotens von der kleineren in die größere Partition zu, wenn der Gewinn durch Reduktion der Verbindungskosten den Verlust durch das erhöhte Laufzeitungleichgewicht übersteigt. Da es sich beim KL-Algorithmus aber um ein lokales Suchverfahren handelt, ist zu erwarten, dass das Ergebnis (und damit auch der erreichbare Speedup) von der Startkonfiguration abhängt. Es ist nicht sichergestellt, dass der Algorithmus die optimale Lösung für die Partitionierungsaufgabe liefert. Die Suche kann ohne Weiteres in einem lokalen Minimum der externen Kosten terminieren.

Als Alternative zur lokalen Suche bieten sich Verfahren an, die einen gegebenen Graphen in seiner Gesamtheit bewerten und auf diese Weise eine Partitionierung der Menge $\mathbb{M}$ gewinnen. Die im folgenden Abschnitt betrachtete Spektrale Partitionierung ist ein solches globales Verfahren.

4.2.3 Spektrale Partitionierung

Die spektrale Partitionierung beruht auf den Extremaleigenschaften quadratischer Formen $\vec{q}^{\,T}\mathbf{A}\vec{q}$ unter der Nebenbedingung $\vec{q}^{\,T}\mathbf{B}\vec{q} = 1$, wobei $\vec{q} \in \mathbb{R}^M$ ist und $\mathbf{A}$, $\mathbf{B}$ reelle, symmetrische $M \times M$ Matrizen. Es ist unmittelbar einzusehen, dass die Extremstellen der quadratischen Form unter der Nebenbedingung auch Extremstellen des Quotienten $(\vec{q}^{\,T}\mathbf{A}\vec{q})/(\vec{q}^{\,T}\mathbf{B}\vec{q})$ sind. Unter welcher notwendigen Bedingung die Extremwerte angenommen werden, kann mit Hilfe der Lagrange-Methode gezeigt werden:

$$\begin{aligned} & \tfrac{\partial}{\partial \vec{q}}\left[\vec{q}^{\,T}\mathbf{A}\vec{q} - \varrho \cdot (\vec{q}^{\,T}\mathbf{B}\vec{q} - 1)\right] &&= 0 \\ \Leftrightarrow \quad & 2 \cdot \mathbf{A}\vec{q} - 2\varrho \cdot \mathbf{B}\vec{q} &&= 0 \\ \Leftrightarrow \quad & \mathbf{A}\vec{q} &&= \varrho \cdot \mathbf{B}\vec{q} \end{aligned} \tag{4.11}$$

Die Gleichung (4.11) stellt eine verallgemeinerte symmetrische Eigenwertaufgabe dar, und die gesuchten Extremstellen müssen demnach Lösungen dieser Aufgabe sein. Alle Eigenwerte und zugehörigen Eigenvektoren $(\varrho_m, \vec{q}_m) : 1 \leq m \leq M$ seien so indiziert, dass gilt:

$$\varrho_1 \geq \varrho_2 \geq \ldots \geq \varrho_{M-1} \geq \varrho_M \tag{4.12}$$

Die verallgemeinerte symmetrische Eigenwertaufgabe kann durch Multiplikation der Gleichung (4.11) mit $\mathbf{B}^{-1}$ in eine gewöhnliche Eigenwertaufgabe

$$\max(\vec{q}^T\mathbf{P}\vec{q}) \quad \text{unter der Nebenbedingung} \quad \vec{q}^T\vec{q} = 1 \tag{4.13}$$

$$\Leftrightarrow \quad \mathbf{P}\vec{q} = \varrho \cdot \vec{q} \qquad \text{mit} \quad \mathbf{P} = \mathbf{B}^{-1}\mathbf{A} \tag{4.14}$$

umgeformt werden. Die geordnete Folge von Eigenwerten ϱ_m wird als Spektrum der Matrix $\mathbf{P}$ bezeichnet. Im Allgemeinen ist $\mathbf{P}$ jedoch **nicht symmetrisch**, so dass die besonderen Eigenschaften reeller, symmetrischer Matrizen für die Herleitung der Extremstellen und -werte von (4.11) nicht genutzt werden können.

Die verallgemeinerte symmetrische Eigenwertaufgabe zu lösen bedeutet, mittels einer Matrix $\mathbf{Q}$ eine simultane Hauptachsentransformation [115] der Matrizen $\mathbf{A}$ und $\mathbf{B}$ durchzuführen, so dass gilt:

$$\mathbf{Q}^T\mathbf{A}\mathbf{Q} \;=\; \mathbf{\Delta}_C = \text{diag}(\rho_m) \tag{4.15}$$

$$\mathbf{Q}^T\mathbf{B}\mathbf{Q} \;=\; \mathbf{I} \tag{4.16}$$

Denn mit diesen beiden speziellen Bedingungen ergibt sich

$$\mathbf{A}\mathbf{Q} = \mathbf{Q}^{-T}\mathbf{\Delta}_C = \mathbf{Q}^{-T}\mathbf{I}\,\mathbf{\Delta}_C = \mathbf{Q}^{-T}(\mathbf{Q}^T\mathbf{B}\mathbf{Q})\mathbf{\Delta}_C = \mathbf{B}\mathbf{Q}\mathbf{\Delta}_C \tag{4.17}$$

oder spaltenweise $\mathbf{A}\vec{q}_m \;=\; \varrho_m \cdot \mathbf{B}\vec{q}_m$, wobei $\vec{q}_m$ die m-te Spalte der Matrix $\mathbf{Q}$ ist. Offensichtlich enthält also die spezielle Matrix $\mathbf{Q} = [\vec{q}_m], 1 \leq m \leq M$ alle Eigenvektoren der Gleichung (4.11).

Die Konstruktion der Transformationsmatrix $\mathbf{Q}$ geschieht mit Hilfe der Substitution $\vec{q} = \mathbf{B}^{-\frac{1}{2}}\vec{x}$ und Multiplikation der Gleichung (4.11) von links mit $\mathbf{B}^{-\frac{1}{2}}$. Dadurch wird die verallgemeinerte symmetrische Eigenwertaufgabe in eine gewöhnliche **symmetrische** Eigenwertaufgabe transformiert.:

$$\mathbf{C}\vec{x} = \varrho \cdot \vec{x} \qquad \text{mit} \quad \mathbf{C} := \mathbf{B}^{-\frac{1}{2}}\mathbf{A}\mathbf{B}^{-\frac{1}{2}} \;\text{ reell und symmetrisch} \tag{4.18}$$

Die Matrix $\mathbf{B}^{-\frac{1}{2}} \;=\; \mathbf{U}_B\mathbf{\Delta}_B^{-\frac{1}{2}}\mathbf{U}_B^T$ ist leicht aus der Matrix $\mathbf{U}_B$ der orthogonalen Eigenvektoren von $\mathbf{B}$ und der Diagonalmatrix $\mathbf{\Delta}_B$ der zugehörigen Eigenwerte zu konstruieren, da $\mathbf{B}$ reell und symmetrisch ist. Die Matrix $\mathbf{B}^{-\frac{1}{2}}$ ist ebenfalls reell und symmetrisch.

Die Matrix $\mathbf{C}$ aus Gleichung (4.18) kann mit der orthogonalen Matrix ihrer Eigenvektoren $\mathbf{U}_C \;=\; [\vec{u}_{C,m}], 1 \leq m \leq M$ auf Diagonalgestalt $\mathbf{\Delta}_C$ transformiert

werden. Ferner ist einfach zu zeigen [115], dass die Extremwerte der zugehörigen quadratischen Form $\vec{x}^{\,T}\mathbf{C}\vec{x}$ unter der Nebenbedingung $\vec{x}^{\,T}\vec{x} = 1$ gerade bei den Eigenvektoren $\vec{x} = \vec{x}_1 = \vec{u}_{C,1}$ bzw. $\vec{x} = \vec{x}_M = \vec{u}_{C,M}$ angenommen werden und dem größten Eigenwert ϱ_1 bzw. dem kleinsten Eigenwert ϱ_M der Matrix $\mathbf{C}$ entsprechen. Folglich werden die gleichen Extremwerte angenommen, wenn $\vec{q}_1 = \mathbf{B}^{-\frac{1}{2}}\vec{x}_1$ bzw. $\vec{q}_M = \mathbf{B}^{-\frac{1}{2}}\vec{x}_M$ in (4.11) eingesetzt wird. Allgemein gilt für die Eigenvektoren der verallgemeinerten symmetrischen Eigenwertaufgabe $\mathbf{Q} = \mathbf{B}^{-\frac{1}{2}}\mathbf{U}_C$.

Wird an einen Vektor $\tilde{\vec{x}}$ eine beliebige Zusatzbedingung in Form einer linearen Gleichung $\vec{p}^{\,T}\tilde{\vec{x}} = r$ gestellt, so kann diese Bedingung (durch die Substitution $\vec{x} = \tilde{\vec{x}} - r \cdot \vec{e}_p$, wobei $\vec{e}_p$ der Einheitsvektor in p-Richtung ist) immer homogen formuliert werden:

$$\vec{p}^{\,T}\vec{x} = 0 \qquad \Leftrightarrow \qquad \vec{x} \perp \vec{p} \tag{4.19}$$

Das Courant-Fischer-Theorem [115] stellt in diesem Fall lediglich sicher, dass bei beliebiger Wahl von $\vec{p}$ für eine einzelne Zusatzbedinung das Maximum (Minimum) der quadratischen Form $\vec{x}^{\,T}\mathbf{C}\vec{x}$ unter der Nebenbedingung $\vec{x}^{\,T}\vec{x} = 1$ nach unten (oben) durch den Wert des zweitgrößten (zweikleinsten) Eigenwertes ϱ_2 (ϱ_{M-1}) beschränkt ist. Bei der speziellen Wahl $\vec{p} = \vec{u}_{C,1}$ ($\vec{u}_{C,M}$), also wenn die Zusatzbedingung gestellt wird, dass die Extremstellen senkrecht auf den größten (kleinsten) Eigenvektor der Matrix $\mathbf{C}$ stehen sollen, nehmen die ausgezeichneten Extremwerte die Werte der Schranke in der Tat an [115], und zwar an den Stellen $\vec{x} = \vec{x}_2 = \vec{u}_{C,2}$ bzw. $\vec{x} = \vec{x}_{M-1} = \vec{u}_{C,M-1}$. Im Hinblick auf die verallgemeinerte symmetrische Eigenwertaufgabe folgt, dass die gleichen ausgezeichneten Extremwerte an den Stellen $\vec{q}_2 = \mathbf{B}^{-\frac{1}{2}}\vec{u}_{C,2}$ bzw. $\vec{q}_{M-1} = \mathbf{B}^{-\frac{1}{2}}\vec{u}_{C,M-1}$ angenommen werden. Entsprechend der speziellen Wahl $\vec{x} \perp \vec{u}_{C,1}$ lautet die Zusatzbedingung für die verallgemeinerte symmetrische Eigenwertaufgabe:

$$\vec{q}^{\,T}\mathbf{B}\vec{q}_1 = 0 \qquad \Leftrightarrow \qquad \vec{q} \perp_{\mathbf{B}} \vec{q}_1 \tag{4.20}$$

Die Gesamtheit dieser Eigenschaften ist für die spektrale Partitionierung unmittelbar zu nutzen. Ding et al. [20] zeigen, dass bei Verwendung zweier besonderer reeller symmetrischer Matrizen $\mathbf{W}$ und $\mathbf{L}$ eine für die Partitionierung sinnvolle Zielfunktion als quadratische Form der Indikatorvektoren $\vec{a}$ und $\vec{b}$ auszudrücken ist:

$$\vec{a} = (1,\ldots,1,0,\ldots,0)^T \quad \text{wobei} \quad a_m = \begin{cases} 1 & : m \in \mathbb{A} \\ 0 & : m \in \mathbb{B} \end{cases} \tag{4.21}$$

$$\vec{b} \;=\; (0,\ldots,0,1,\ldots,1)^T \quad \text{wobei} \quad b_m = \begin{cases} 0 & : m \in \mathbb{A} \\ 1 & : m \in \mathbb{B} \end{cases} \tag{4.22}$$

Die Indizes der Vektorkomponenten entsprechen den Indizes der Knoten eines Graphen, und sie sind dabei so geordnet, dass sich die übersichtliche Form von $\vec{a}$ und $\vec{b}$ in den Gleichungen (4.21) und (4.22) ergibt. Die Komponentenwerte a,$b \in \{0;1\}$ zeigen die eindeutige Zugehörigkeit eines jeden Knotens zu einer Partition an. Da die Partitionen $\mathbb{A}$ und $\mathbb{B}$ disjunkt sind, gilt stets $\vec{a} \perp \vec{b}$.

Die Matrix $\mathbf{W}$ ist die in Abschnitt 4.2.2 eingeführte Gewichtsmatrix des Graphen, wobei für die spektrale Partitionierung jedoch $w_{m,m} = p_m \neq 0$ gesetzt wird. Die Matrix

$$\mathbf{L} \;=\; \mathbf{D} - \mathbf{W} \tag{4.23}$$

$$\text{mit} \quad \mathbf{D} \;=\; \text{diag}(d_l), \qquad d_l = \sum_{m=1}^{M} w_{l,m} \tag{4.24}$$

heißt Laplace-Matrix, weil $\vec{q}^{\,T}\mathbf{L}\vec{e}_m$, also die Anwendung auf den Knoten mit dem Index m, der Anwendung des diskreten Laplace-Operators auf diesen Knoten entspricht. Es existiert hier eine direkte Analogie zu Laplace-Operatoren in der Bildverarbeitung (siehe Anhang A).

Mit der Laplace-Matrix können interne und externe Kosten eines Schnittes folgendermaßen angegeben werden:

$$E \;=\; \sum_{m\in\mathbb{A}} E_m = \vec{a}^{\,T}\mathbf{L}\vec{a} = \sum_{m\in\mathbb{B}} E_m = \vec{b}^{\,T}\mathbf{L}\vec{b} \tag{4.25}$$

$$I_{\mathbb{A}} \;=\; \sum_{m\in\mathbb{A}} I_m = \vec{a}^{\,T}\mathbf{W}\vec{a} \tag{4.26}$$

$$I_{\mathbb{B}} \;=\; \sum_{m\in\mathbb{B}} I_m = \vec{b}^{\,T}\mathbf{W}\vec{b} \tag{4.27}$$

Die internen Kosten beinhalten hier sowohl die Summe aller durch eine bestimmte Partitionierung vermiedenen potentiellen Verbindungskosten aus $\mathbf{W}$ als auch die Summe aller Knotenlaufzeiten der entsprechenden Partitionen. Um die externen Kosten zu minimieren und gleichzeitig die internen Kosten zu maximieren, wird

folgende Zielfunktion minimiert:

$$Mcut = \frac{E}{I_{\mathbb{A}}} + \frac{E}{I_{\mathbb{B}}} = \frac{\vec{a}^{\,T}\mathbf{L}\vec{a}}{\vec{a}^{\,T}\mathbf{W}\vec{a}} + \frac{\vec{b}^{\,T}\mathbf{L}\vec{b}}{\vec{b}^{\,T}\mathbf{W}\vec{b}} \tag{4.28}$$

Nun ist aber bereits in Kapitel 3 gezeigt worden, dass das Auffinden der optimalen Vektoren $\vec{a}, \vec{b}$ mit $a_m, b_m \in \{0; 1\}$ NP-schwer ist. Demnach ist nicht zu erwarten, dass diese Lösung jetzt mit garantiert polynomialem Aufwand, etwa durch Matrixmanipulation, herbeizuführen ist, es sei denn, die Aufgabe wird in einer bestimmten Weise vereinfacht. Diese Vereinfachung besteht darin, dass anstelle der diskreten Vektorkomponenten nun Komponenten eines Vektors $\vec{q} \in \mathbb{R}^M$ mit $q_m \in [-1; 1]$ zugelassen werden. Die Zugehörigkeit eines Knotens m zu einer Partition wird dabei durch das Vorzeichen von q_m angezeigt.

Ding et al. zeigen [20], dass unter dieser Relaxierung die gleichzeitige Minimierung der beiden Summanden aus Gleichung (4.28) äquivalent ist mit

$$\min(\vec{q}^{\,T}\mathbf{L}\vec{q}) \quad \text{unter der Nebenbedingung} \quad \vec{q}^{\,T}\mathbf{D}\,\vec{q} = 1 \tag{4.29}$$

$$\text{und der Zusatzbedingung} \quad \vec{q}^{\,T} \perp_{\mathbf{D}} \vec{e} \tag{4.30}$$

Da $\mathbf{L} = \mathbf{D} - \mathbf{W}$ ist, kann an die Stelle der Minimierung auch die Maximierung der quadratischen Form mit $\mathbf{W}$ treten:

$$\max(\vec{q}^{\,T}\mathbf{W}\vec{q}) \quad \text{unter der Nebenbedingung} \quad \vec{q}^{\,T}\mathbf{D}\,\vec{q} = 1 \tag{4.31}$$

$$\text{und der Zusatzbedingung} \quad \vec{q}^{\,T} \perp_{\mathbf{D}} \vec{e} \tag{4.32}$$

Dies entspricht der am Anfang des Abschnitts geschilderten verallgemeinerten symmetrischen Eigenwertaufgabe mit den Matrizen $\mathbf{A} = \mathbf{W}$ und $\mathbf{B} = \mathbf{D}$. Der Vektor $\vec{e} := \vec{1}$ ist in mehrfacher Hinsicht ein spezieller Vektor in Bezug auf diese Matrizen. Erstens ist einfach nachzuprüfen, dass $\vec{e}$ Eigenvektor zum Eigenwert $\varrho = 1$ ist, weil $\mathbf{W}\vec{e} = \mathbf{D}\vec{e}$. Die Matrix $\mathbf{D}$ ist ja gerade aus der Zeilensumme von $\mathbf{W}$ konstruiert worden. Zweitens ist die Matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$ nach Gleichung (4.14) eine stochastische Matrix und besitzt damit den größten Eigenwert $\varrho_1 = 1$ mit der Vielfachheit 1 [93]. Folglich ist der Vektor $\vec{e} = \vec{q}_1$ auch gleichzeitig Eigenvektor zum **größten** Eigenwert der Aufgabe aus Gleichung (4.31). Drittens ist der Vektor $\vec{e} = \vec{q}_1$ sicherlich **keine** sinnvolle Lösung der Partitionierungsaufgabe, da alle Komponenten positives Vorzeichen haben und die Knoten des Graphen damit nur einer einzigen Partition zugeordnet werden würden. Der Gleichung (4.20) entsprechend muss die Erfüllung der Zusatzbedingung (4.32) gefordert werden. Das Maximum von (4.31)

wird dann mit dem Eigenvektor $\vec{q}_2$ erreicht, der dem zweitgrößten Eigenwert ϱ_2 zugeordnet ist.

Anstelle des Eigenvektors zum zweitgrößten Eigenwert von (4.31) kann auch der Eigenvektor zum zweitkleinsten Eigenwert der Aufgabe in (4.29) bestimmt werden. Infolge der Arbeiten von Fiedler [29, 30] wird dieser Vektor auch Fiedler-Vektor genannt, und der zugehörige Eigenwert ϱ_{M-1} heißt Fiedler-Wert. Die Simulationsergebnisse zur spektralen Partitionierung in Kapitel 5 basieren alle auf der Berechnung des Fiedler-Vektors für Gleichung (4.29).

Da der Fiedler-Vektor lediglich die relaxierte Entwurfsaufgabe optimal löst, muss noch eine Diskretisierung dieser Lösung erfolgen. Die einfachste Methode besteht darin, die Knoten des Graphen entsprechend den Vorzeichen der Komponenten des Lösungsvektors $\vec{q}_{M-1}$ den beiden Partitionen zuzuordnen. Der Fiedler-Wert stellt dann eine untere Schranke für die diskretisierte Lösung dar. Eine heuristische Erweiterung des Verfahrens besteht darin, nach einer ersten Partitionierungsphase alle Knoten mit einem Maß für die Stärke der Bindung zu seiner jeweiligen Partition (*engl.* "linkage difference" [20]) erneut zu bewerten. In einer zweiten Phase wird mit diesem Maß dann pro Partition eine Liste gebildet, in der die zugehörigen Knoten in aufsteigender Reihenfolge ihrer Partitionsbindung geordnet stehen. Knoten mit niedriger Bindung zur eigenen Partition sind erfahrungsgemäß offenbar gute Kandidaten für einen Partitionswechsel. In der so ermittelten Reihenfolge (*engl.* "linkage differential order" [20]) werden Partitionswechsel regelrecht ausprobiert, indem die Zielfunktion nach Gleichung (4.28) mit entsprechend geänderten Indikatorvektoren $\vec{a}'$ und $\vec{b}'$ berechnet wird. Tritt dabei ein Funktionswert auf, der kleiner ist als bei Vorzeichenentscheidung über die Komponenten des Fiedler-Vektors, so wird der Partitionswechsel für den betreffenden Knoten akzeptiert. Diese Heuristik wurde beispielsweise in [10] erfolgreich auf Mod-SDR angewendet.

4.2.4 Graphenverdoppelung

Ein ganz einfacher Ansatz zur Abbildung von Software auf Hardware besteht darin, einen gegebenen Graphen zu duplizieren und eine vollständige Kopie dem einen Prozessor und die andere vollständige Kopie dem anderen Prozessor zuzuordnen. Das ist ohne Weiteres möglich, weil die Nutzerdaten in unterschiedlichen Funkrahmen zwar auf identische Weise verarbeitet werden, sonst aber völlig unabhängig voneinander sind.

Es besteht bei Graphenverdoppelung keine Notwendigkeit zum Austausch von Zwischenergebnissen zwischen den Prozessoren, sondern lediglich zum Bustransfer von Ein- und Ausgangsdaten zwischen der I/O-Schnittstelle und den verteilten Speichern der Prozessoren. Auf ein separates Shared Memory kann verzichtet werden, was bezüglich der Hardware-Realisierung einen großen Vorteil bedeutet. Der Partitionierungsvorgang erfordert keinen Algorithmus, und die Partitionsgrößen sind per Konstruktion ausgewogen. Ein Nachteil der Graphenverdoppelung ist sicherlich die Verdoppelung von Programm- und Datenspeicherbedarf. Alle Softwaremodule existieren doppelt, und pro Prozessor ist der komplette Ein- und Ausgangsdatenspeicher für einen Funkrahmen zu reservieren.

Generell ist nicht nur der Speedup als Qualitätsmaß für den Entwurf eines Mod-SDRs zu sehen, sondern auch das Verzögerungsverhalten der Signalverarbeitung in Software. Beide Größen hängen wesentlich von der zeitlichen Abfolge der Ausführung von Softwaremodulen ab. Daher werden im Folgenden entsprechende Ansätze zur Ablaufplanung betrachtet.

4.3 Ansätze zur Ablaufplanung

Die Beurteilung der Betriebsart eines Mod-SDR ist nicht allein mit der Kenntnis der Partitionierungsstrategie möglich, sondern immer nur mit der Kombination aus Partitionierung und zeitlicher Ablaufplanung. Ein bestimmter Ansatz für die Ablaufplanung kann gegebenenfalls Einfluss auf die Randbedingungen der Partitionierung haben, so z.B. im Falle des Half-Frame Pipelining, das am Ende dieses Abschnittes diskutiert wird. Trotz dieser Bindung zwischen Partitionierung und Ablaufplanung werden beide aber nach wie vor in zwei aufeinanderfolgenden, getrennten Schritten ausgeführt. Für die Ablaufplanung kann deshalb angenommen werden, dass die Zuordnung aller Knoten zu den Partitionen bereits bekannt ist und der Graph durch Bustransfer-Knoten an den geschnittenen Kanten ergänzt wurde.

4.3.1 Hu-Scheduling ohne und mit Pipelining

Die einfachste Ablaufplanung besteht darin, den Hu-Algorithmus auf jede Partition getrennt anzuwenden: Unabhängig von der Partitionierung erhalten alle Knoten des um die Bustransfer-Knoten ergänzten Graphen einen Hu Level. In jeder Iteration ist ein Knoten zu bestimmen, der den aktuellen Zeitpunkt als Startzeitpunkt zugeteilt bekommt.

Pro Partition wird zu Beginn einer jeden Iteration eine Kandidatenliste erstellt, die alle unmittelbaren Nachfolger von bereits bearbeiteten Knoten der betrachteten Partition enthält. Logische Abhängigkeiten zu Vorgängern aus der anderen Partition bestehen nur noch über Bustransfer-Knoten. Der Fall eines antikausalen Bustransfers tritt ein, wenn auf einem Prozessor ein Bustransfer-Knoten zur Ausführung eingeplant wird, bevor die Ausführung des korrespondierenden Bustransfer-Knotens der anderen Partition beendet ist (siehe Bild 4.5). Dieser Fall ist zunächst strikt zu vermeiden. Folglich sind alle Knoten, die einen antikausalen Bustransfer nach sich ziehen würden, aus der Kandidatenliste zu streichen. Unter den verbleibenden Kandidaten wird derjenige mit dem größten Hu Level ausgewählt und zur Ausführung am aktuellen Zeitpunkt eingeplant. Unter einer Teilmenge von Knoten mit identischem Hu Level erhält der Knoten mit maximaler Laufzeit Vorrang bei der Ausführung. Zur Vorbereitung der nächsten Iteration wird der aktuelle Zeitpunkt um die Laufzeit des ausgewählten Knotens verschoben.

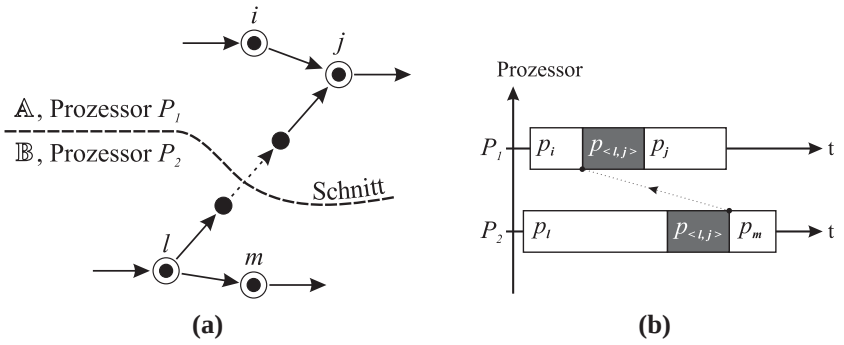


Bild 4.5 Antikausaler Bustransfer

Ist die Kandidatenliste zum aktuellen Zeitpunkt leer, so ist zufolge unerfüllter logischer Abhängigkeiten zur anderen Partition kein einziger Knoten der betrachteten Partition mehr lauffähig. Die Abhängigkeiten betreffen damit sicher Bustransfer-Knoten. Der aktuelle Zeitpunkt muss solange verschoben werden, bis wieder ein kausaler Transfer von Zwischenergebnissen über den Bus möglich ist. Auf diese Weise kommt es auf dem betrachteten Prozessor zum Leerlauf. Der Zustand des Prozessors wird in diesem Fall mit `LOGICAL_WAIT_IDLE` bezeichnet. Das Bild 4.6 zeigt die Situation mit zwei getrennten Zeitachsen für die beiden Prozessoren.

Trotz Erfüllung logischer Abhängigkeiten kann es vorkommen, dass der Startzeitpunkt eines Bustransfer-Knotens verschoben werden muss, weil der Bus zum ak-

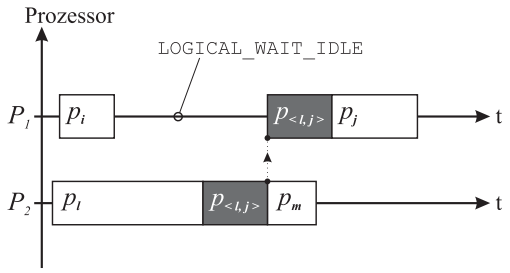


Bild 4.6 Leerlaufzeit auf Prozessor 1, kausaler Bustransfer

tuellen Zeitpunkt bereits für einen anderen Datentransfer genutzt wird. Um die Auftretenswahrscheinlichkeit für diesen Fall zu minimieren, wird parallel zur Kandidatenliste immer eine Alternativliste geführt, die nur reguläre Knoten enthält und ebenfalls nach Hu Level und Laufzeit geordnet ist. Die Ausführung eines Bustransfer-Knotens kann bei einem Buskonflikt verschoben werden, und an seine Stelle tritt der reguläre Knoten aus der Alternativliste. Ist die Alternativliste leer, so muss ein blockierter Bustransfer-Knoten zeitlich so lange verschoben werden, bis der Bus wieder frei ist. Auf diese Weise entsteht auf dem betrachteten Prozessor Leerlaufzeit, die allein auf die Belegung einer physikalischen Struktur des Mod-SDR zurückgeht. Der Zustand des Prozessors wird in diesem Fall mit `PHYSICAL_WAIT_IDLE` bezeichnet. Das Bild 4.7 zeigt die Situation mit zwei getrennten Zeitachsen für die beiden Prozessoren.

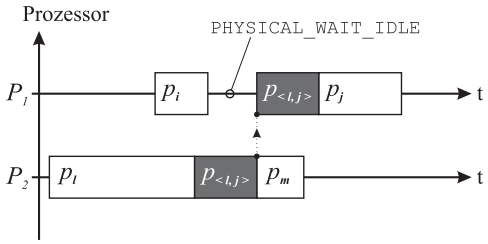


Bild 4.7 Leerlaufzeit auf Prozessor 1, kausaler Bustransfer

Eine Variante bei der Erstellung der Kandidatenliste besteht darin, nur Bustransfer-Knoten aufzunehmen und alle regulären Knoten in der Alternativliste zu halten. Auf diese Weise wird eine Priorisierung des Austauschs von Zwischenergebnis-

sen erreicht. Bustransfer-Knoten werden zur Ausführung eingeplant, obwohl ihr Hu Level nicht maximal ist. Die Motivation dafür besteht in der Erwartung, dass bei diesem Vorgehen die Wahrscheinlichkeit für den Zustand `PHYSICAL_WAIT_IDLE` sinkt: Je früher ein möglicher Bustransfer ausgeführt wird, desto geringer ist die Wahrscheinlichkeit, dass bei Auftreten eines Buskonflikts die Alternativliste mit regulären Knoten der betrachteten Partition leer ist. Gleichzeitig wird die Alternativliste der anderen Partition frühestmöglich gefüllt. Naturgemäß tritt der Zustand `PHYSICAL_WAIT_IDLE` bei $L = 2$ Prozessoren nur für $B = 1$ Bus auf. Der Hardware-Aufwand für $B = 2$ ist erheblich, nicht nur wegen des großen Platzbedarfs von zwei separaten Gruppen von Busleitungen, sondern auch wegen der zusätzlichen Anbindungen an die Prozessoren, das Shared Memory und die I/O-Schnittstelle. Aber selbst mit zwei Bussen kann es durchaus zu Leerlaufzeiten kommen, und zwar allein durch das Auftreten des Zustands `LOGICAL_WAIT_IDLE`.

Die Algorithmen der Partitionierung aus den Abschnitten 4.2.2 und 4.2.3 betrachten potentielle Verbindungskosten in Form von Laufzeiten der Bustransfer-Knoten mit dem Ziel, diese Kosten bei ausgewogener Größe der Partitionen zu minimieren. Im Hinblick auf das zeitliche Auftreten von Knoten werden jedoch keine besonderen Annahmen gemacht. Dadurch kann es in der Tat zu den beschriebenen Leerlauf-Situationen kommen. Jede Art von Leerlauf führt notwendigerweise zu einer Reduktion der Qualität des Mod-SDR-Entwurfs. Im Hinblick auf den zu erwartenden Speedup kann dieser Nachteil bei der Ablaufplanung mit dem Prinzip des Pipelining behoben werden, jedoch nur um den Preis einer größeren Verzögerung.

Pipelining beruht zum Einen auf dem Vorwissen um die sich periodisch wiederholende Ausführung aller Softwaremodule und zum Anderen auf der Unabhängigkeit der Nutzerdaten zwischen beliebigen Funkrahmen. Zu welchem Rahmen die Daten gehören, die innerhalb einer Echtzeitperiode ΔT von einem bestimmten Modul m verarbeitet werden, hat keinen Einfluss auf dessen Laufzeit p_m. Daher können alle Berechnungen, die zu einem bestimmten Funkrahmen gehören, auf der Zeitachse um ganzzahlige Vielfache von ΔT verschoben werden, ohne dass sich die Gestalt eines Ablaufplanes ändert. Die Verarbeitung von Daten eines bestimmten Funkrahmens erstreckt sich insgesamt über einen Zeitraum, der mehrere Echtzeitperioden ΔT umfasst. Diese Anzahl von Echtzeitperioden wird im Folgenden als Tiefe der Pipeline bezeichnet.

Insbesondere ist durch Anwendung des Pipelining-Prinzips die Kausalität von ursprünglich antikausalen Bustransfers wieder herzustellen. Folglich kann der Pro-

zessorzustand `LOGICAL_WAIT_IDLE` durch Pipelining vollständig eliminiert werden. Buskonflikte werden aufgelöst, indem der blockierte Knoten einfach den frühest möglichen konfliktfreien Startzeitpunkt innerhalb von ΔT zugewiesen bekommt. Im Allgemeinen ist der Datentransfer zwischen korrespondierenden Knotenpaaren dann antikausal und wird in einem zweiten Schritt ebenfalls kausalisiert. Das Grundgerüst des Hu-Algorithmus zur Ablaufplanung mit allen Optionen (Pipelining, Priorisierung von Bustransfers, Ein- oder Zwei-Bus-System) findet sich in Form eines Pseudo-Codes in [81].

Bei der praktischen Umsetzung des Pipelining-Prinzips wird zunächst einfach auf das Entfernen von antikausalen Bustransfers aus der Kandidatenliste verzichtet. Auf diese Weise lässt sich der Ablaufplan für beide Prozessoren dicht mit Knotenlaufzeiten ausfüllen. Leerlaufzeit entsteht lediglich am Ende von ΔT, wenn die Partitionierung es nicht vermocht hat, bezüglich ihrer Gesamtlaufzeit ausgewogene Partitionen zu erzeugen. Der Schritt der Kausalisierung wird in der zeitlichen Reihenfolge aller antikausalen Transfers vollzogen. Dabei zieht ein jeder solcher Transfer nicht unmittelbar eine Erhöhung der Pipeline-Tiefe um 1 nach sich. Es kann durchaus vorkommen, dass die Herstellung der Kausalität für einen bestimmten Bustransfer auch für zeitlich nachfolgende Transfers wirksam ist.

Betrachtet man eine bestimmte Echtzeitperiode ΔT, so werden beim Pipelining innerhalb dieses Zeitraums Berechnungen mit Daten verschiedener Rahmen geschachtelt. Zuweilen wird diese Art der Signalverarbeitung als Parallelverarbeitung bezeichnet, wobei ”Parallelität” rein zeitlich verstanden wird, also letztlich als Gleichzeitigkeit. Im Rahmen der vorliegenden Arbeit wird jedoch klar unterschieden: **Parallelisierung** bedeutet die Ausnutzung der inhärenten logischen Parallelität eines Graphen zur gleichzeitigen Ausführung von Softwaremodulen auf mehreren Prozessoren, die Daten **eines einzigen** Funkrahmens verarbeiten. Genau das ist aber beim Pipelining im Allgemeinen gerade nicht gegeben: **Pipelining** bedeutet die Ausnutzung der Periodizität in der Signalverarbeitung zur gleichzeitigen Ausführung von Softwaremodulen auf mehreren Prozessoren, die Daten **verschiedener** Funkrahmen verarbeiten.

Der wichtigste Vorteil von Pipelining besteht darin, dass die beiden Schritte Partitionierung und Ablaufplanung in der Tat unter den gleichen Annahmen arbeiten: Speedup-Verluste resultieren einzig und allein aus Bustransfers und unausgewogener Lastverteilung. Die Nachteile von Pipelining bestehen aber in erhöhtem Speicherbedarf und in zusätzlicher Verzögerung. Der Speicherbedarf ist gegenüber der Signalverarbeitung ohne Pipelining sicherlich erhöht, weil während der Echtzeitpe-

riode ΔT Zwischenergebnisse von mehreren verschiedenen Funkrahmen im Speicher gehalten werden müssen. Die Simulationsergebnisse in Kapitel 5 zeigen die zu erwartenden Verzögerungen für das Beispiel eines UMTS-Graphen als Tiefe der Pipeline. Spätestens dort entsteht der Wunsch nach anderen, einfacheren Varianten für die Anwendung des Pipelining-Prinzips.

4.3.2 Graph Duplication Pipelining

Der Ansatz des Graph Duplication Pipelining (GDP) bedient sich der Graphenverdoppelung für die Partitionierung und wurde ursprünglich als radikale Alternative zu anderen Kombinationen aus Partitionierung und Ablaufplanung betrachtet [9], und zwar zunächst für leitungsvermittelte Dienste.

Unter dieser Voraussetzung ist das Füllen und Leeren der Pipeline im Vergleich zur gesamten Verbindungsdauer unerheblich, und es kommt im Wesentlichen darauf an, die Echtzeitperiode ΔT im eingeschwungenen Zustand des Ablaufplanes möglichst dicht mit Knotenlaufzeiten zu füllen. Da bei der Anwendung des Pipelining-Prinzips ohnehin mit erhöhtem Speicherbedarf gerechnet werden muss, gilt es doch zumindest, die Verzögerungszeit zu minimieren, die allein durch die Art der Pipelining-Signalverarbeitung entsteht, und genau das ist mit GDP zu erreichen. Die Vorteile der Graphenverdoppelung sind bereits im Abschnitt 4.2.4 diskutiert worden: Reduktion von Bustransfers auf das absolut notwendige Minimum und Ausgewogenheit der Partitionslaufzeiten per Konstruktion. Lediglich der Austausch von Ein- und Ausgangsdaten zwischen der I/O-Schnittstelle und den einzelnen Prozessoren muss nun zeitlich geschickt geplant werden. Mit Blick auf die Realisierungskosten soll das Hardware-System nur über einen einzigen Bus verfügen.

Unabhängig von den logischen Strukturen eines Graphen sind alle Softwaremodule eines gegebenen Graphen zeitlich rein sequentiell auszuführen, d.h. eventuell vorhandene Parallelstrukturen sind in keiner Weise zur Erhöhung des Speedups zu nutzen, wenn ein Prozessor genau eine Kopie des Graphen zugewiesen bekommt. Folglich können die Laufzeiten aller regulären Knoten zur Laufzeit T_P akkumuliert werden. Ebenso wird die Summe aller Laufzeiten von Bustransfer-Knoten für Ein- und Ausgangsdaten, T_I bzw. T_O gebildet. Zunächst kann man auf die Idee kommen, alle Prozessoren so schnell wie möglich mit Eingangsdaten zu versorgen, so dass die Signalverarbeitung mit Daten von zwei benachbarten Funkrahmen quasi gleichzeitig ablaufen kann. Das Bild 4.8 zeigt den resultierenden Ablaufplan,

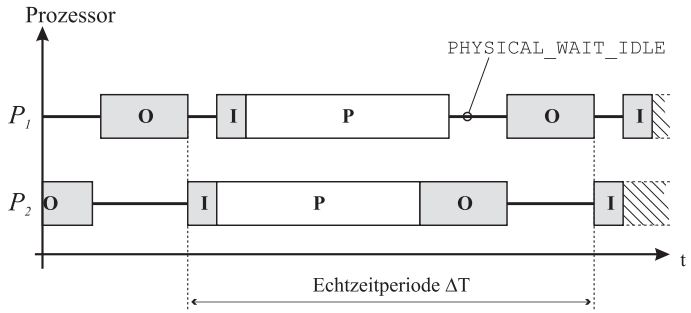


Bild 4.8 GDP-Ablauf mit Leerlaufzeiten, $L = 2$

der notwendigerweise zu Leerlaufzeiten führt. Es stellt sich schnell heraus, dass Ein- und Ausgabe von Daten über den Bus zeitlich unmittelbar aufeinander folgen müssen, um einen dichten Ablaufplan zu erzeugen. Dabei wird die natürliche Reihenfolge von Dateneingabe, -verarbeitung und -ausgabe nach wie vor aufrechterhalten. Das Bild 4.9 zeigt den resultierenden, maximal dichten Ablaufplan für GDP. Damit ist für dieses Beispiel gezeigt, dass GDP den maximalen Speedup von $\overline{s} = 2$ für $L = 2$ erreicht.

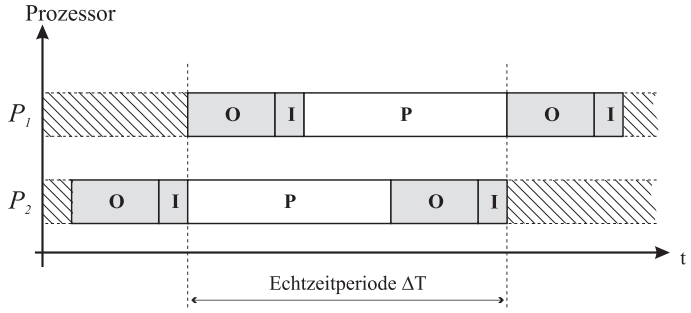


Bild 4.9 Dichter Ablaufplan für GDP, $L = 2$

Solange $T_P \geq T_{IO}$, wobei $T_{IO} := T_I + T_O$, kann man im allgemeinen Fall $L \in \mathbb{N}$, $\beta \in \mathbb{R}^+$ und $B = 1$ zur Herleitung einer oberen Grenze für den Speedup aus dem Fall der idealen Parallelisierung der Knoten eines Graphen wie folgt argumentieren: Die Bustransfers für Ein- und Ausgabe müssen sicherlich rein sequentiell ausgeführt werden, weil die L Prozessoren nur einen einzigen Bus zur Verfügung haben.

Dabei ändert sich die Zeitdauer T_{IO} nicht, wenn die verschiedenen Bustransfers von den Prozessoren tatsächlich individuell ausgeführt werden oder wenn in der Vorstellung genau ein Prozessor den Bus stellvertretend für alle anderen für die Dauer T_{IO} belegt. Im besten Falle können während dieser Zeit dann auf $(L-1)$ Prozessoren reguläre Knoten ausgeführt werden. Die verbleibende Laufzeit regulärer Knoten ist dann $T_P - (L-1) \cdot T_{IO}$, und diese Laufzeit kann wiederum im besten Falle auf alle L Prozessoren völlig ausgewogen verteilt werden. Insgesamt ergibt sich:

$$T_P \geq T_{IO} \; : \quad \Delta T_{multi,min} = T_{IO} + \frac{T_P - (L-1) \cdot T_{IO}}{L} \tag{4.33}$$

Die Laufzeit der Bustransfer-Knoten ist aber von der relativen Busgeschwindigkeit β abhängig, und es ist offensichtlich, dass Leerlaufzeiten im Ablaufplan entstehen, wenn das System von Bustransfers dominiert wird:

$$T_P < T_{IO} \; : \quad \Delta T_{multi,min} = T_{IO} \tag{4.34}$$

Mit $\Delta T_{mono} = T_P + T_{IO}$ lässt sich die folgende theoretische Obergrenze für den Speedup angeben:

$$\overline{s} = \frac{\Delta T_{mono}}{\Delta T_{multi,min}} = \begin{cases} L & ; \; T_P \geq T_{IO} \\ 1 + \frac{T_P}{T_{IO}} & ; \; T_P < T_{IO} \end{cases} \tag{4.35}$$

Der von einer Mod-SDR-Realisierung tatsächlich erreichte Speedup s kann relativ zu dieser theoretischen Grenze angegeben werden. Das Verhältnis $y := s/\overline{s}$ wird kurz relativer Speedup genannt.

Den zeitlichen Ablauf für GDP und den Sonderfall $L = 2$ in Bild 4.9 kann man leicht auf $L \in \mathbb{N}$ verallgemeinern. Ein dichter Ablaufplan entsteht bei L Kopien eines gegebenen Graphen genau dann, wenn $T_P \geq (L-1) \cdot T_{IO}$ ist. Folglich ist der Speedup dann $s = \overline{s} = L$. Andernfalls ist $\Delta T_{multi} = L \cdot T_{IO}$ und $\Delta T_{mono} = L \cdot (T_P + T_{IO})$. Die theoretische Obergrenze in Gleichung (4.35) wird von GDP also erreicht, allerdings wird auch der Zustand des vom Bus dominierten Systems im Vergleich zur idealen Parallelisierung schon bei einem $(L-1)$mal schnelleren Bus erreicht. Um also den Speedup der idealen Parallelisierung zu erreichen, benötigt ein mit GDP betriebenes Mod-SDR einen $(L-1)$mal schnelleren Bus, und das erzeugt in CMOS-Schaltungen die $(L-1)$fache dynamische Verlustleistung. Diesem Nachteil steht gegenüber, dass $\overline{s} = L$ auch tatsächlich erreicht

wird, und zwar unabhängig von den logischen Strukturen des Graphen, wohingegen ein Parallelisierungsansatz sehr stark von den strukturellen Eigenschaften des gegebenen Graphen abhängt [6] und $s = \overline{s} = L$ praktisch nicht zu erreichen ist. Die Argumentation zeigt, dass GDP im Fall $T_P \geq (L-1) \cdot T_{IO}$ hinsichtlich des Speedups für leitungsvermittelte Dienste optimal ist und für $L = 2$ sogar Gleichung (4.35) erfüllt.

Gerade die Verbindung von Anwendungen auf Laptop-Rechnern mit dem Mobilfunk macht eine paketorientierte drahtlose Anbindung mobiler Nutzer an das IP-basierte Festnetz interessant. Daher muss ein hochflexibler Systemansatz wie Mod-SDR auch auf seine Leistungsfähigkeit bei paketvermittelten Diensten untersucht werden. Ein OFDM-basierter Standard wie IEEE802.11a [1] kann hier als Beispiel dienen. Die physikalische Schicht transformiert IP-Pakete, die Nutzer- und Kontrolldaten aus höheren Schichten enthalten, in Pakete aus OFDM-Symbolen, die auf dem Mobilfunkkanal übertragen werden. Zusätzlich müssen MAC-Pakete für die Kontrolle der Zugriffe mehrerer Nutzer auf das physikalische Medium erzeugt werden. Alle Pakete, die auf dem Kanal übertragen werden, sind mit Kontrollinformationen der physikalischen Schicht zu versehen: Eine Präambel mit der Länge von zwei OFDM-Symbolen und das SIGNAL-Feld mit der Länge eines OFDM-Symbols. Die Modulationsart auf allen Unterträgern des OFDM-Verfahrens kann BPSK, QPSK, 16-QAM oder 64-QAM sein, abhängig vom momentanen SNR des Kanals. MAC-Pakete werden abhängig vom SIGNAL-Feld (immer BPSK) entweder mit BPSK, QPSK oder 16-QAM moduliert. Auf der Ebene der Algorithmen kann die benötigte Adaptivität für den Modulator (oder für andere **geeignete** nachrichtentechnische Funktionen) durchaus mit dem Ansatz PaC-SDR [43, 114] realisiert werden. Auf der Ebene der logischen und physikalischen Strukturen eines Mod-SDR ist aber nach wie vor zu entscheiden, wie modulare Software auf die verfügbare Hardware abzubilden ist. Die Signalverarbeitung von Daten, die in ein OFDM-Symbol einfließen, kann wie zuvor auf abstrakte Weise mit einem Graphen dargestellt werden. Das einzelne OFDM-Symbol aus dem Beispiel ist also mit dem allgemeineren Begriff des Funkrahmens zu identifizieren.

Die Verarbeitung von Paketen kann entweder Rahmen für Rahmen erfolgen oder unter Anwendung des Pipelining-Prinzips. Im ersten Fall unterscheidet sich der Speedup nicht von den Ergebnissen bei leitungsvermittelten Diensten, weil der momentane Speedup in jeder Echtzeitperiode gleich dem mittleren Speedup über die gesamte Verbindungsdauer ist. Bei der Paketverarbeitung mit GDP muss der Speedup aber über die gesamte Paketdauer berechnet werden. Gerade bei kurzen

IP-Paketen kann die Anzahl der Funkrahmen pro Paket so gering sein, dass das Füllen und das Leeren der Pipeline erheblichen Einfluss auf den Speedup nimmt. Für die Berechnung wird ein möglichst einfaches Betriebssystem angenommen, das beide Prozessoren ausschließlich für die Aufgaben der Basisbandsignalverarbeitung reserviert, sobald ein Paket anliegt. Die Gesamtzahl identisch zu behandelnder Funkrahmen pro Paket sei $N_F \in \mathbb{N}$ und für die Rechnung konstant.

Der Fall $N_F = 1$ ist für GDP mit einem Zwei-Prozessor-System nicht sinnvoll, da dann selbstverständlich ein Prozessor leerläuft, während der andere den Funkrahmen verarbeitet, was formal einem Speedup von $s = 0{,}5\,\overline{s}$ entspricht. Der Fall $N_F = 2$ ist ein Sonderfall, da der Ablaufplan nie in den eingeschwungenen Zustand übergeht, sondern nur aus dem abwechselnden Füllen und Leeren der Pipeline besteht. Unter der Bedingung, dass das System mit $B = 1$ nicht von Bustransfers dominiert wird, ist der Ausdruck für den relativen Speedup [9]:

$$y = 1 - \left[2 + \frac{T_I + T_P}{T_O}\right]^{-1} \quad ; \left(T_P \geq T_{IO}\right) \wedge \left(T_O \geq T_I\right), N_F = 2 \tag{4.36}$$

Wenn $(T_O < T_I)$ gilt, ist einfach $T_I \leftrightarrow T_O$ in der Gleichung (4.36) auszutauschen. Der Fall $N_F = 2$ kann innerhalb eines IEEE802.11a WLANs in der Tat auftreten, z.B. bei sehr kurzen TCP ACK-Paketen (20 Bytes), die in IP-Pakete (40 Bytes) eingepackt und bei der Modulationsart 64-QAM von der physikalischen Schicht in nur zwei OFDM-Symbole übersetzt werden.

Für alle $N_F > 2$ und $T_P \geq T_{IO}$ muss zwischen einer geraden und ungeraden Anzahl von Funkrahmen pro Paket unterschieden werden:

$$y_g = 1 - \left[1 + \frac{N_F}{2} \cdot \left(1 + \frac{T_P}{T_{IO}}\right)\right]^{-1} \quad ; N_F \text{ gerade} \tag{4.37}$$

$$y_u = \frac{N_F}{N_F + 1} \quad ; N_F \text{ ungerade} \tag{4.38}$$

Sobald das System im Wesentlichen von Bustransfers bestimmt wird, gilt für alle $N_F \geq 2$:

$$y = \left[1 - \frac{1}{N_F} \cdot \left(1 - \frac{T_X}{T_{IO}}\right)\right]^{-1} \quad ; T_P < T_{IO} \tag{4.39}$$

wobei $T_X = 2T_P$ für $(T_P > T_O > T_I)$ oder $(T_P > T_I > T_O)$, $T_X = T_P + T_O$ für $(T_O > T_P > T_I)$, und $T_X = T_P + T_I$ für $(T_I > T_P > T_O)$. Wenn $(T_O > T_I > T_P)$

oder $(T_I > T_O > T_P)$, dann wird der Speedup vollständig vom Bus dominiert, und es gilt $y = 1$. Allerdings ist dies kein sinnvoller Betriebspunkt für ein Mod-SDR.

Der wesentliche Nachteil von GDP besteht nach wie vor in der Verdopplung des Speicherbedarfs, der mit der Verdopplung des Graphen einhergeht. Daher wird im folgenden Abschnitt ein Ansatz zur Ablaufplanung eingeführt, der das Pipelining-Prinzip nutzt und mit einem einzigen Graphen auskommt.

4.3.3 Half-Frame Pipelining

Die Grundidee von Half-Frame Pipelining (HFP) besteht darin, einen gegebenen Graphen vertikal zu schneiden, so dass sich die Berechnung eines kompletten Funkrahmens in die Berechnung eines linken und eines rechten Halbrahmens aufteilen lässt. Es wird vereinbart, dass der linke Halbrahmen, der die Eingangsdaten verarbeitet, immer als **erster Halbrahmen** bezeichnet wird und der rechte Halbrahmen, der die Ausgangsdaten erzeugt, entsprechend als **zweiter Halbrahmen**. Die beiden Halbrahmen sollten hinsichtlich der Rechenzeit so ausgewogen wie möglich sein. Jedem Prozessor wird dann ein Halbrahmen zugewiesen, und die Ablaufplanung sieht vor, dass der eine Prozessor bereits reguläre Knoten des ersten Halbrahmens mit Index $(i + 1)$ ausführt, während der zweite Prozessor noch die zweite Hälfte des Vorgängerrahmens mit dem Index i verarbeitet. Es erfolgt also ein Pipelining zwischen den zwei Hälften **eines** Graphen.

Aufgrund des Schnittes wird der ursprüngliche Graph vor dem Prozess der Ablaufplanung um Bustransfer-Knoten erweitert. Da die Softwaremodule der Halbrahmen auf jedem Prozessor zeitlich rein sequentiell verarbeitet werden, ist es auch bei HFP nicht möglich, eventuell vorhandenen Parallelstrukturen zur Erhöhung des Speedups zu nutzen. Folglich können die Laufzeiten aller regulären Knoten wieder zur Laufzeit T_P akkumuliert werden. Die Laufzeiten von Bustransfer-Knoten zum Austausch von Zwischenergebnissen zwischen den Partitionen belaufen sich auf T_B. Die Summe aller Laufzeiten für Ein- und Ausgangsdaten wird mit T_I bzw. T_O bezeichnet. Zunächst kann man wie bei GDP auf die Idee kommen, den zweiten Prozessor so schnell wie möglich mit Zwischenergebnissen zu versorgen, nachdem der erste Prozessor neue Eingangsdaten gelesen hat, so dass die Signalverarbeitung mit den Daten der beiden Halbrahmen quasi gleichzeitig ablaufen kann. Das Bild 4.10 zeigt den resultierenden Ablaufplan, der notwendigerweise zu Leerlaufzeiten führt. Es stellt sich heraus, dass vielmehr der Ablaufplan nach Bild 4.11 die Echtzeitperiode dicht mit Laufzeiten ausfüllt und daher vorzuziehen ist.

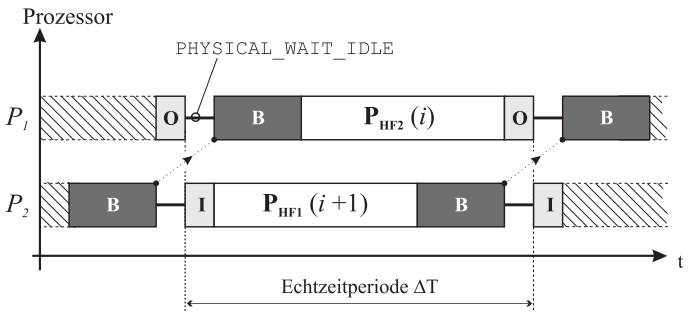


Bild 4.10 HFP-Ablauf mit Leerlaufzeiten, $L = 2$

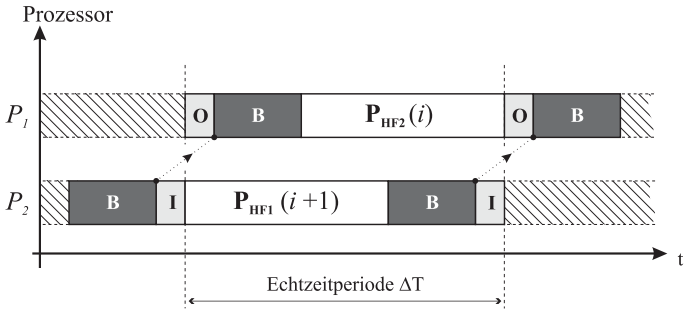


Bild 4.11 Dichter Ablaufplan für HFP, $L = 2$

Aus der Beschreibung der Grundidee wird bereits deutlich, dass die besondere Form des Ablaufplanes eine Partitionierung der Knotenmenge unter einer Nebenbedingung voraussetzt, die mit der Forderung nach dem senkrechten Schnitt gegeben ist. Trotzdem wird HFP in dieser Arbeit als Verfahren der Ablaufplanung gesehen, weil die zeitliche Koordination der Berechnungen im Vordergrund steht und die entsprechende Partitionierung keine neuartigen Ansätze erfordert, sondern lediglich Varianten der in Abschnitt 4.2 eingeführten Verfahren.

In [75] wird der Graph mit Hilfe einer Straffunktion auf die Partitionierung vorbereitet. Die potentiellen Verbindungskosten sind wie üblich den Kantengewichten $w_{l,m}$ zugeordnet. Die Straffunktion ist nur von den angetroffenen Knotenlaufzeiten und den logischen Strukturen des Graphen abhängig. Sie wird so konstruiert, dass ihr Betrag klein wird, wenn der Schnitt durch den Graphen senkrecht ist. Die

Straffunktion betrifft ausschließlich die Laufzeiten von Knoten, und der **Betrag** der Straffunktion wird einfach zu den Gewichten $w_{l,m}$ abgehender Kanten hinzuaddiert, um gleichzeitig die Schnittkosten zu minimieren. Die Diagonalelemente der Gewichtsmatrix $\mathbf{W}$ werden zu $w_{m,m} = 0 : 1 \leq m \leq M$ gesetzt. Mit dieser Matrix $\mathbf{W}$ und der Diagonalmatrix $\mathbf{D}$ nach Gleichung (4.24) kann eine modifizierte spektrale Partitionierung [75] durchgeführt werden. Leider führt diese Variante gelegentlich zu nicht-vertikalen Schnitten, so dass eine Nachverarbeitung notwendig wird, die auf alternierende Partitionszugehörigkeit entlang aller Pfade prüft und gegebenenfalls Knoten zwischen den beiden Partitionen austauscht. Dadurch wird ein zweiter Schritt der Nachverarbeitung notwendig, der die Ausgewogenheit der Partitionsgrößen wieder herstellt, und zwar durch den Transfer einzelner Knoten von der größeren zur kleineren Partition. Als Auswahlkriterium dient dabei der maximale Hu Level für den linken Halbrahmen bzw. der minimale Hu Level für den rechten Halbrahmen [75].

Dass ein senkrechter Schnitt mit Hilfe der Straffunktion nicht garantiert werden kann, ist der wesentliche Nachteil dieser Variante. Auch die Einführung einer weiteren Straffunktion für den gleichen Graphen, jedoch mit umgekehrt gerichteten Kanten, und Mittelwertbildung über beide Beiträge erzielt keine Verbesserung [83]. Weiterhin ist es einfach nachzuvollziehen, dass für einen Knotentransfer von der größereren zur kleineren Partition nicht nur ein Knoten mit dem größten bzw. kleinsten Hu Level in Frage kommt. Dieses Kriterium ist zwar hinreichend, um einen vertikalen Schnitt aufrechtzuerhalten, aber nicht notwendig. Wenn der linke Halbrahmen die größere Partition bildet, so kommen alle Knoten mit ausgehenden Kanten, die in keine andere als die kleinere Partition zeigen, für einen Knotentransfer in Frage. Wenn der rechte Halbrahmen die größere Partition bildet, wird die Liste der Kandidaten von allen Knoten gebildet, die ausschließlich über einlaufende Kanten aus der kleineren Partition verfügen. In beiden Fällen ist die Kandidatenliste niemals leer, denn mindestens der Knoten mit dem größten bzw. kleinsten Hu Level ist immer vorhanden. Mit einer solchen Kandidatenliste, die stets für die Erfüllung der Nebenbedingung sorgt, kann der KL-Algorithmus ohne Weiteres betrieben werden. Im Gegensatz zum Ausgleich der Partitionslaufzeiten in der Nachverarbeitung der spektralen Partitionierung schließt der KL-Algorithmus automatisch einen Kompromiss zwischen der Unausgewogenheit der Partitionen und den tatsächlichen Verbindungskosten. Als Startkonfiguration wird die triviale Partitionierung gewählt, bei der alle Knoten des Graphen bis auf den Zielknoten dem ersten Halbrahmen zugeordnet sind.

Für leitungsvermittelte Dienste ist bereits erkannt worden, dass GDP optimal hinsichtlich des Speedups ist. Weil HFP zusätzlich zu den Gesamtlaufzeiten T_P und T_{IO} noch Laufzeiten T_B von Bustransfer-Knoten aufweist, kann HFP nicht optimal sein. Allerdings entfällt bei HFP die Verdoppelung des Graphen, und das ist gerade in mobilen Geräten mit wenig Speicher von Vorteil. Ein Vergleich mit der oberen Schranke in Kapitel 5 wird zeigen, welcher Speedup-Verlust dafür beim Betrieb eines Mod-SDR mit HFP hinzunehmen ist.

Für paketvermittelte Dienste muss der Speedup $s = \Delta T_{mono}/\Delta T_{multi}$ wiederum über die gesamte Paketdauer berechnet werden. Die Echtzeitperiode des Einzelprozessors ist unverändert $\Delta T_{mono} = T_P + T_{IO}$. Das Zwei-Prozessor-System mit $B=1$ Bus wird von Bustransfers dominiert, wenn gilt:

$$(T_{IO} + 2T_B) > \max\left[(T_I + T_B + T_{P,1}),\ (T_O + T_B + T_{P,2})\right] \tag{4.40}$$

wobei $T_{P,1}$ und $T_{P,2}$ die Summe der Laufzeiten aller regulären Knoten im ersten bzw. zweiten Halbrahmen bezeichnen, und $T_P = T_{P,1} + T_{P,2}$. Bei Anwendung von HFP setzt sich die Echtzeitperiode dann wie folgt zusammen:

$$\Delta T_{multi} \;=\; N_F \cdot (T_{IO} + 2T_B) - 2T_B + 2T_P + \tau_1 + \tau_2 + \tau_B(N_F) \tag{4.41}$$

$$\tau_i \;:=\; \begin{cases} T_B - T_{P,i} & ;\ T_{P,i} < T_B \\ 0 & ;\ \text{sonst} \end{cases} \tag{4.42}$$

$$\tau_B(N_F) \;:=\; \begin{cases} T_B - T_{P,1} & ;\ (N_F = 2)\ \wedge\ (T_{P,2} > T_{P,1} > T_B) \\ T_B - T_{P,2} & ;\ (N_F = 2)\ \wedge\ (T_{P,1} > T_{P,2} > T_B) \\ 0 & ;\ \text{sonst} \end{cases} \tag{4.43}$$

Wenn Gleichung (4.40) nicht zutrifft, ergibt sich die folgende Echtzeitperiode, **wenn** $(T_{P,1} + T_I) > (T_{P,2} + T_O)$:

$$\Delta T_{multi} \;=\; N_F \cdot (T_I + T_{P,1} + T_B) + 2T_O + T_B + T_{P,2} + \tau_P(N_F) \tag{4.44}$$

$$\tau_P(N_F) \;=\; \begin{cases} T_O + T_{P,2} - T_{P,1} & ;\ (N_F > 2)\ \wedge\ (T_{P,1} < T_{P,2} + T_O) \\ 0 & ;\ \text{sonst} \end{cases} \tag{4.45}$$

Wenn hingegen $(T_{P,1} + T_I) \leq (T_{P,2} + T_O)$ gilt, so müssen in den Gleichungen (4.44) und (4.45) lediglich $T_I \leftrightarrow T_O$ und $T_{P,1} \leftrightarrow T_{P,2}$ ausgetauscht werden. Diese

Bedingungen und Gleichungen bilden die Grundlage zur Messung des Speedups bei HFP für paketvermittelte Dienste.

4.4 Zusammenfassung

In diesem Kapitel sind die grundlegenden Aufgaben der Partitionierung und der Ablaufplanung dargestellt worden. Für das Modell eines symmetrischen Multiprozessors mit $L = 2$ Prozessoren und maximal $B = 2$ Bussen liegen drei verschiedenartige Ansätze für die Partitionierung eines Graphen vor: Die implizite Partitionierung, der KL-Algorithmus und die spektrale Partitionierung. Zusätzlich ist die Graphenverdopplung mit ihren Vor- und Nachteilen in mehreren Abschnitten diskutiert worden. Für die Ablaufplanung steht auf der einen Seite eine Modifikation des Hu-Algorithmus zur Verfügung, der alle folgenden Wahlmöglichkeiten zu berücksichtigen erlaubt: Pipelining, Priorisierung von Bustransfers und bis zu zwei separate Busse im System. Auf der anderen Seite stehen GDP und HFP als Vorschriften zur zeitlichen Koordination von Rechnungen, die per Konstruktion gewisse Eigenschaften erfüllen und einfach zu realisieren sind, selbst wenn das Hardware-System nur über einen einzigen Bus verfügt. GDP nutzt die Graphenverdoppelung und benötigt deshalb keinen Algorithmus zur Partitionierung. HFP erfordert lediglich die Einhaltung einer Nebenbedingung bei der Partitionierung und kommt ohne die Graphenverdoppelung aus. Alle Kombinationen von Partitionierung und Ablaufplanung haben zunächst zum Ziel, die Wahrscheinlichkeit für hohen Speedup in einer Mod-SDR-Umgebung zu maximieren. Speicherbedarf, Verzögerungsverhalten und äquidistante Ergebnisausgabe sind jedoch ebenfalls wichtige Aspekte, die bei der Beurteilung eines Mod-SDR eine Rolle spielen. Das folgende Kapitel untersucht alle diese Aspekte und zeigt quantitative Ergebnisse für leitungsvermittelte ebenso wie für paketvermittelte Dienste.

5 Leistungsfähigkeit von Mod-SDR

In diesem Kapitel wird die Leistungsfähigkeit des Ansatzes Mod-SDR bewertet, wenn der Kern des Betriebssystems die Verfahren aus Kapitel 4 anwendet. Neben Rechnersimulationen werden in dem Abschnitt 5.1.6 auch weitere analytische Ergebnisse zum leistungseffizienten Betrieb eines Coprozessors hergeleitet.

Bei den hier vorgestellten Simulationen kommen die Hardware- und Software-Modelle aus Kapitel 2 zum Einsatz. Unter dem erweiterten Ressourcen-Laufzeit-Modell aus Abschnitt 2.2.4 werden die Knotenlaufzeiten nicht nur von der Dichte $f_C(c)$ beeinflusst, sondern auch von den deterministischen Speicheranforderungen. Bleibt der Speicherbedarf aller Knoten im Extremfall konstant (bei Zufallsgraphen nach Abschnitt 2.2.6 liegt er zumindest in der gleichen Größenordnung), ist die Dichte der Laufzeiten $f_P(p)$ lediglich eine skalierte Version der Dichte $f_C(c)$ und die Ergebnisse aus Kapitel 3 sind direkt anwendbar. Streuen die Werte des Speicherbedarfs hingegen stark, so wird die effektive relative Streuung der Laufzeiten dadurch viel mehr bestimmt als durch die Zufallsvariable C und deren Dichte $f_C(c)$. Im Allgemeinen gilt daher: Je größer die Streuung des Speicherbedarfs, desto unerheblicher die Gestalt von $f_C(c)$. Die wichtige Erkenntnis aus Kapitel 3, dass nicht die Gestalt der Dichtefunktion, sondern die effektive relative Laufzeit-Streuung σ_p wesentlich für das Speedup-Verhalten verantwortlich ist, erlaubt es, für die Rechnersimulationen in diesem Kapitel letztlich eine gefensterte Dichtefunktion $f_C(c)$ von beliebiger Gestalt festzulegen. Das Ressourcen-Laufzeit-Modell, das für alle Simulationen in diesem Kapitel zugrunde gelegt wird, benutzt als Dichte $f_C(c)$ eine Gaußdichte nach Gleichung (2.2) mit dem Fenster $[c_{min}; c_{max}] = [0{,}5; 1{,}5]$ und den Parametern $\mu_c = \mu_{c,\mathit{eff}} = 1{,}0$ und $\sigma_{c,\mathit{eff}} = 0{,}25$. Diese Wahl kann so interpretiert werden, dass die Realisierungen von Laufzeiten, die das Betriebssystem des Mod-SDR als Knotenlaufzeit antrifft, um bis zu 50% von der strikt linearen Beziehung zwischen Speicherbedarf und Laufzeit abweichen können.

Zunächst findet ein unmittelbarer Vergleich des Speedupverhaltens zwischen zwei Partitionierungsverfahren mit und ohne Bewertung der Verbindungskosten statt. Daraus sind bereits einige grundlegende Eigenschaften des Zusammenspiels von Partitionierung, Ablaufplanung und physikalischen Strukturen des Mod-SDR abzulesen. Es folgen Untersuchungen zu ausgewählten Fragestellungen, die im Wesent-

lichen die drei Größen Speedup, Verzögerung und Speicherbedarf betreffen. Die Signalverarbeitung für leitungsvermittelte Dienste kann damit umfassend bewertet werden. Einige Erkenntnisse lassen sich unmittelbar auf paketvermittelte Dienste übertragen, andere folgen erst aus weiteren Untersuchungen. Es wird gezeigt, dass die Ergebnisse für paketorientierte Signalverarbeitung durchaus für etablierte WLAN-Standards relevant sind. Die Zusammenfassung enthält eine abschließende Bewertung der verschiedenen Betriebsarten eines Mod-SDR.

5.1 Leitungsvermittelte Dienste

5.1.1 Grundlegende Mod-SDR-Systemparameter

In diesem Abschnitt wird zunächst der Hu-Algorithmus mit dem KL-Algorithmus verglichen. Beide dienen der Partitionierung eines Graphen, wobei der Hu-Algorithmus lediglich implizit partitioniert, der KL-Algorithmus hingegen die Laufzeiten von Bustransfer-Knoten ausdrücklich berücksichtigt. Als Software-Modell dient hier der in Bild 5.1 dargestellte Standard-UMTS-Graph [23, 80], der einen DTCH mit 64 kbps und einen DCCH mit $2{,}5$ kbps Nutzdatenrate beinhaltet. Das Hardware-Modell besteht zunächst in dem System mit $L = 2$ Prozessoren und $B = 1$ Bus. Die Ablaufplanung erfolgt mit dem modifizierten Hu-Algorithmus aus Abschnitt 4.3.1. Für die späteren Rechnersimulationen gilt es zunächst zu verstehen, welche Optionen der Ablaufplanung (Bus-Priorisierung, Pipelining, zweiter Bus) welches prinzipielle Systemverhalten nach sich ziehen.

Als erstes soll hier auf die Priorisierung von Bustransfers eingegangen werden. Das Bild 5.2 zeigt den absoluten Speedup (Kreise, linke Ordinate) und die Busaktivität (Punkte, rechte Ordinate) in Abhängigkeit von der relativen Busgeschwindigkeit β. Die Busaktivität ist das Verhältnis der Summe aller Buszugriffszeiten innerhalb von ΔT zur gesamten Echtzeitperiode ΔT. Der Stichprobenumfang beträgt 400 Realisierungen für jeden Parameterwert β. Ausgehend von Speedup-Werten um $s = 1$ bei einem langsamen Bus ($\beta < 1$) steigt der Speedup stetig an und nähert sich mit wachsendem β der oberen Grenze $\overline{s} = 2$. Ein langsamer Bus mit $\beta < 1$ beschreibt strenggenommen keinen sinnvollen Betriebspunkt für ein Mod-SDR, weil dann allein der Datentransfer über den Bus zum Engpass des Mobilfunksystems wird, nicht die Verarbeitung der Funksignale durch die Prozessoren.

Die Speedup-Streuung in Bild 5.2 steigt mit wachsendem β ebenfalls an und er-

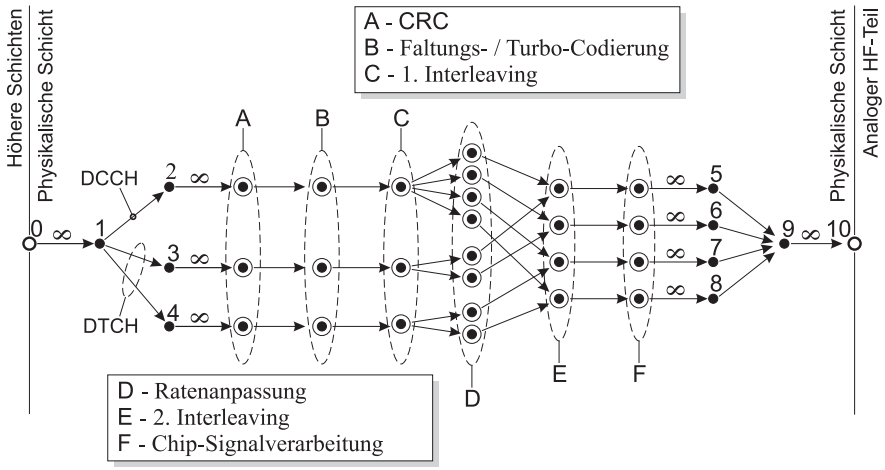


Bild 5.1 Gerichteter Graph, UTRA FDD (Sender), 64 kbps DTCH Uplink [23]

reicht bei einem sehr schnellen Bus ($\beta \gg 1$) Werte zwischen $s = 1{,}2$ und $\overline{s} = 2$. Diese Ergebnisse sind ohne die Priorisierung von Bustransfer-Knoten erzielt worden. Das Bild 5.3 zeigt hingegen die Ergebnisse mit Bus-Priorisierung. Bis auf einige Ausreißer bei $s = 1{,}5$ erzeugen jetzt die meisten Realisierungen einen Speedup zwischen $s = 1{,}8$ und $\overline{s} = 2$. Die Busaktivität in beiden Bildern zeigt kaum Unterschiede, und das ist bei einem Graphen mit fester logischer Struktur auch zu erwarten. Allerdings ist die Tatsache bemerkenswert, dass gerade bei einem schnellen Bus eine erhebliche Verbesserung des Speedupverhaltens erreicht wird, denn die Wahrscheinlichkeit von Leerlaufzeiten zufolge des Prozessorzustandes `PHYSICAL_WAIT_IDLE` ist hier äußerst gering. Daher kann geschlossen werden, dass die Priorisierung von Bustransfers eine einfache Methode ist, um den Prozessorzustand `LOGICAL_WAIT_IDLE` zu vermeiden und so für das Gesamtsystem einen höheren Speedup zu erzielen. Bus-Priorisierung wird im Folgenden immer dann eingesetzt, wenn die Ablaufplanung mit einem Bus und ohne Pipelining arbeitet.

Als nächstes wird der Vergleich von KL- und Hu-Algorithmus durchgeführt. Aufgrund der Tatsache, dass der KL-Algorithmus potentielle Verbindungskosten zwischen den Partitionen berücksichtigt und insgesamt aufwendiger ist als der Hu-Algorithmus, erwartet man zunächst auch ein besseres Speedup-Ergebnis für den

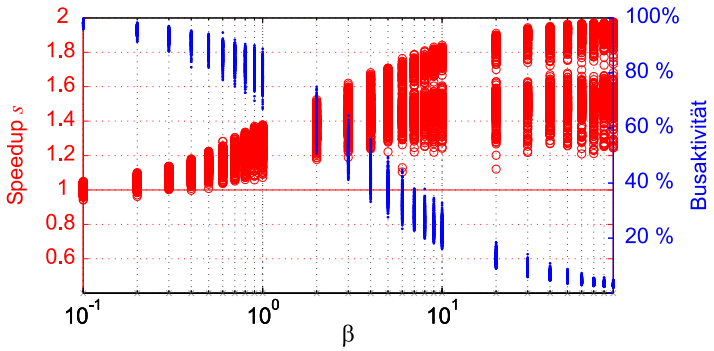


Bild 5.2 Hu-Algorithmus, Ablaufplan ohne Priorisierung von Bustransfers

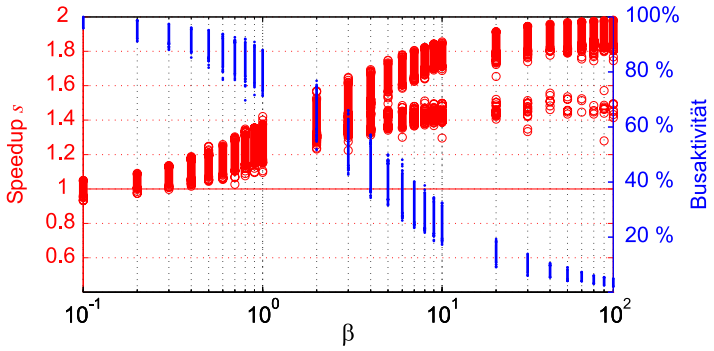


Bild 5.3 Hu-Algorithmus, Ablaufplan **mit** Priorisierung von Bustransfers

KL-Algorithmus. Das Bild 5.4 zeigt das Verhältnis der Speedups beider Verfahren für gleiche Realisierungen (Kreise, linke Ordinate) und die Busaktivität des KL-Algorithmus (Punkte, rechte Ordinate). Ein Verhältnis von 1,0 zeigt an, dass beide Algorithmen bei der gleichen Mod-SDR-Realisierung den gleichen Speedup erzielen. Bei größeren Werten erzielt der KL-Algorithmus den besseren Speedup, bei kleineren Werten der Hu-Algorithmus.

Entgegen der ersten Erwartung zeigt das Bild 5.4 keine systematische Überlegenheit des aufwendigeren KL-Algorithmus, und der Grund liegt darin, dass der Algorithmus zwar potentielle Verbindungskosten und die Ausgewogenheit der Partitionen modelliert, aber nicht die Prozessorzustände `LOGICAL_WAIT_IDLE` und `PHYSICAL_WAIT_IDLE`, denn diese entstehen erst im zeitlichen Ablauf, und der

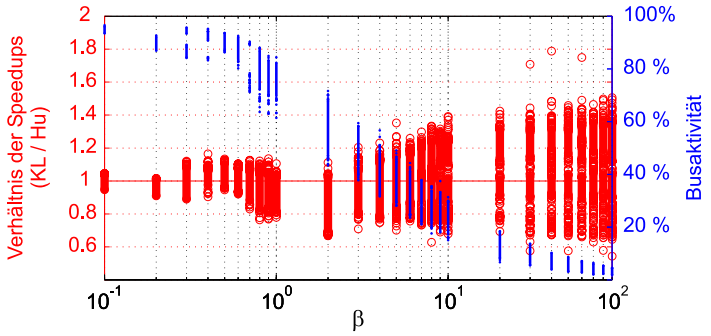


Bild 5.4 Vergleich von KL- und Hu-Algorithmus, $B = 1$, Ablaufplan ohne Pipelining

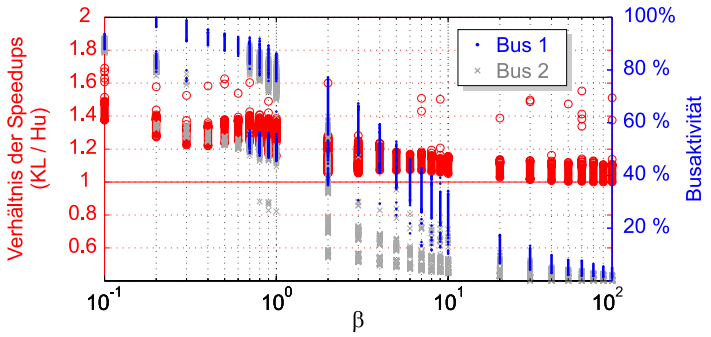


Bild 5.5 Vergleich von KL- und Hu-Algorithmus, $B = 2$, Ablaufplan mit Pipelining

ist während der Phase der Partitionierung noch unbekannt. Beide Prozessorzustände können aus einem Ablaufplan nur durch Hinzunahme eines zweiten Busses und zusätzliches Pipelining eliminiert werden. Das Bild 5.5 zeigt, dass unter diesen Bedingungen der KL-Algorithmus ohne Ausnahme dem Hu-Algorithmus überlegen ist. Ferner ist der Gewinn des KL-Algorithmus bei niedrigen Busgeschwindigkeiten größer als bei hohen Busgeschwindigkeiten. Dieser Effekt kann unmittelbar der Anwendung des Pipelining-Prinzips zugeschrieben werden: In dem Software-Modell nach Bild 5.1 gibt es Bustransfer-Knoten, zu denen kein Parallelzweig existiert. Ein Verfahren wie die implizite Partitionierung nach Hu, die ursprünglich auf eine möglichst gute Parallelisierung abzielt, geht bereits in der Phase der Partitionierung davon aus, dass solche Knoten nur seriell ausführbar sind. Pipelining hinge-

gen erlaubt es, durch das Schachteln der Berechnungen aus mehreren Funkrahmen künstlich parallele Zweige zu diesen Knoten zu schaffen und so Leerlaufzeiten zu vermeiden. Zudem ermöglicht es der zweite Bus, dass nicht nur reguläre Knoten, sondern auch Bustransfer-Knoten gleichzeitig ausgeführt werden können. Damit sind die Entwurfsbedingungen des KL-Algorithmus genau erfüllt, und der Algorithmus ist folglich systematisch überlegen. Die Aktivität beider Busse ist im Bild 5.5 mit verschiedenen Symbolen gekennzeichnet. Die hohe Aktivität, insbesondere bei $\beta < 1$, zeigt an, dass beide Busse in der Tat notwendig sind, um mit dem Verfahren von Kernighan und Lin zwischen 10% und 40% Gewinn gegenüber der Hu-Lösung zu erzielen. Mit wachsender Busgeschwindigkeit geht der Hu-Algorithmus mehr und mehr auf seine Entwurfsbedingungen zu und folglich verbessern sich so die Vergleichswerte. Es fällt auf, dass der Gewinn bei schnellen Bussen ($\beta \gg 1$) höchstens 10% ist und damit kleiner als die größten Gewinne in Bild 5.4. Diese Beobachtung kann damit erklärt werden, dass der Hu-Algorithmus in einem fairen Vergleich selbstverständlich ebenfalls **beide** Busse nutzen kann und so auch höhere Vergleichswerte beim Speedup erzeugt.

Da die Einführung eines zweiten Busses gerade in mobilen Endgeräten große Kosten nach sich zieht, soll der Vergleich zwischen den beiden Partitionierungsstrategien mit $B = 1$ Bus und Pipelining fortgesetzt werden. Damit arbeiten beide Verfahren unter Randbedingungen, für die sie streng genommen nicht entworfen wurden: Der Hu-Algorithmus modelliert weder die endliche Busgeschwindigkeit noch die Möglichkeiten von Pipelining, der KL-Algorithmus setzt zu jedem Zeitpunkt die volle Kommunikationsfähigkeit für beide Prozessoren voraus, die aber mit einem einzigen Bus nicht gegeben ist. Pipelining kann als reine Software-Methode interpretiert werden, weil es sich nur auf die zeitliche Koordination der Signalverarbeitung bezieht und keine zusätzlichen Hardware-Kosten entstehen, sondern allenfalls größere Verzögerungen als ohne Pipelining. Das Bild 5.6 zeigt das Verhältnis der erzielten Speedups mit $B = 1$ und Anwendung des Pipelining-Prinzips bei der Ablaufplanung. Als Startkonfiguration für den KL-Algorithmus wurde die Hu-Lösung gewählt.

Offensichtlich leiden die Ablaufpläne mit und ohne Pipelining (Bilder 5.4 und 5.6) am gleichen Nachteil: Es ist nicht vorherzusagen, ob der KL-Algorithmus (Verhältnis der Speedups > 1) oder der Hu-Algorithmus (Verhältnis der Speedups < 1) das bessere Ergebnis erzielt. Erst für $\beta > 2$ resultiert der KL-Algorithmus bei der Mehrzahl aller Realisierungen in einem größeren Speedup als der Hu-Algorithmus. Für $\beta > 10$ nähert sich der Bus den Entwurfsbedingungen des Hu-Algorithmus so

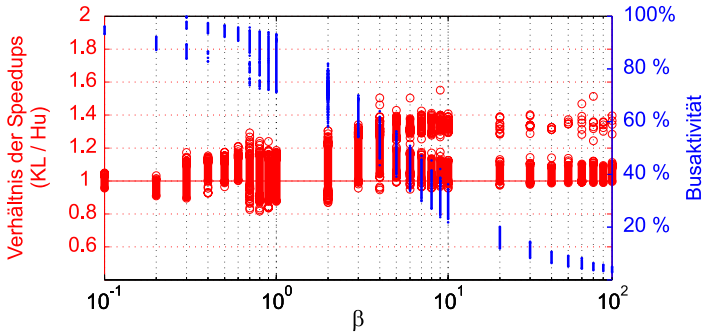


Bild 5.6 Startkonfiguration für KL: Hu-Lösung, $B = 1$, Ablaufplan mit Pipelining

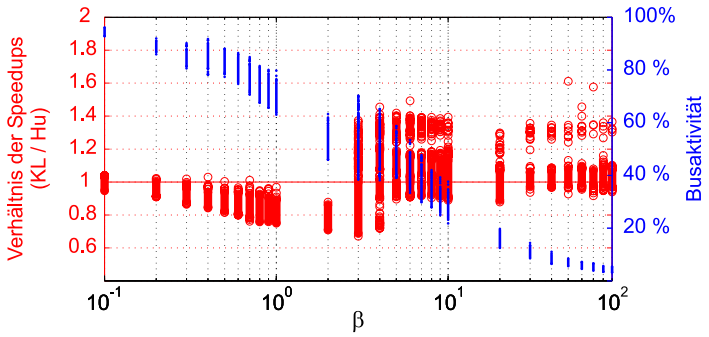


Bild 5.7 Triviale Startkonfiguration für KL, $B = 1$, Ablaufplan mit Pipelining

stark, dass die Vergleichswerte des Speedups für viele Realisierungen ebenfalls hoch liegen. Daher erzielt der KL-Algorithmus bei einem schnellen Bus in den meisten Fällen maximal 10% Gewinn und nur sehr selten Gewinne zwischen 30% und 40%.

Abschließend wird überprüft, ob die Leistungsfähigkeit des KL-Algorithmus merklich von der Startkonfiguration abhängt. Die Bilder 5.7 und 5.8 zeigen das Speedup-Verhältnis zwischen KL- und Hu-Algorithmus für die triviale und eine bezüglich der Partitionslaufzeiten in etwa ausgewogene Startkonfiguration [80]. Bei der trivialen Konfiguration sind alle Knoten des Graphen bis auf den Zielknoten einer einzigen Partition zugeordnet. Der KL-Algorithmus zielt darauf ab, die Ausgewogenheit der Partitionen bei gleichzeitiger Minimierung der externen Kosten durch

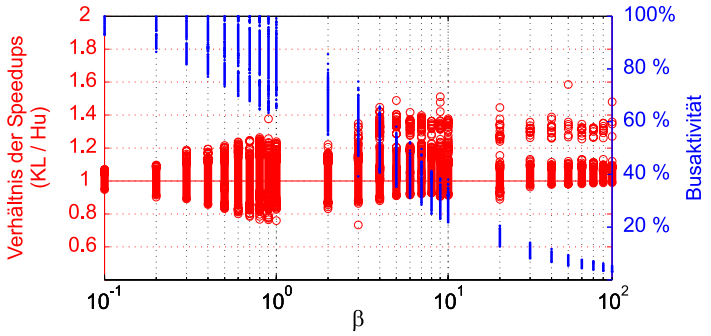


Bild 5.8 Ausgewogene Startkonfiguration für KL, $B = 1$, Ablaufplan mit Pipelining

Einzelknotentransfer und Tauschen von Knotenpaaren zwischen den Partitionen herzustellen. Das Bild 5.7 zeigt, dass das erst für $\beta > 2$ gelingt und dass die Fälle, in denen der KL-Algorithmus **keinen** Gewinn erzielt, deutlich zahlreicher sind als in Bild 5.6. Eine ausgewogene Startkonfiguration zu wählen scheint zumindest für $\beta \leq 2$ erfolgversprechender zu sein. Dennoch treten im Bild 5.8 Gewinn und Verlust gegenüber der Hu-Lösung in etwa gleich häufig auf. Für $\beta > 2$ ähneln sich die Ergebnisse aus Bild 5.8 und Bild 5.6, wobei der Anteil mit niedrigem Speedup in Bild 5.8 jedoch noch etwas größer ist. Unter diesen drei Fällen erzeugt die Hu-Startkonfiguration insgesamt mit hoher Wahrscheinlichkeit den größten Speedup.

Aus diesem ersten Vergleich zwischen Hu- und KL-Algorithmus lassen sich folgende Schlüsse ziehen [80]:

- Die Priorisierung von Bustransfers ist notwendig, um die Wahrscheinlichkeit für hohen Speedup zu vergrößern.
- Obwohl der Graph in Bild 5.1 über zahlreiche Parallelzweige verfügt und damit für einen Parallelisierungsansatz hohen Speedup verspricht, ist der Hu-Algorithmus durch eine endliche Busgeschwindigkeit und rein serielle Bustransfers in seiner Leistungsfähigkeit grundsätzlich beschränkt.
- Mindestens die Anwendung des Pipelining-Prinzips ist für den Erfolg des KL-Algorithmus notwendig.
- Für $B = 1$ Bus erzielt der KL-Algorithmus seine wesentlichen Gewinne bei moderaten Busgeschwindigkeiten im Bereich $2 < \beta \leq 10$.

Aufgrund der Beobachtungen bei den Bildern 5.6 bis 5.8 wird im Folgenden immer die Hu-Startkonfiguration für den KL-Algorithmus gewählt. Auch die Priorisierung von Bustransfers wird für alle folgenden Ablaufpläne fest eingestellt, solange diese ohne Pipelining auskommen müssen. Obwohl der direkte Vergleich zwischen Hu- und KL-Algorithmus bereits Aufschluss über die günstige Wahl von Systemparametern gibt, ist noch immer unklar, welche Methode mit großer Wahrscheinlichkeit an die theoretische Obergrenze für ideale Parallelisierung nach Gleichung (4.35) heranreicht. Auch steht der Vergleich mit spektraler Partitionierung noch aus. Die entsprechenden Simulationsergebnisse werden in den folgenden Abschnitten gezeigt.

Als Software-Modell kommt dabei eine leicht veränderte Darstellung von Bild 5.1 zum Einsatz: Das Bild 5.9 zeigt im Wesentlichen die gleiche logische Struktur, jedoch ohne die Bustransfer-Knoten, die als Module im Bild 5.1 mit 1 bis 9 nummeriert und mit den Kantengewichten ∞ fest an bestimmte Nachbarknoten gekoppelt sind. Die ursprüngliche Idee bei der Einführung dieser ausgezeichneten Bustransfers bestand darin, die Eingangsdaten so früh wie möglich von der I/O-Schnittstelle zu lesen (Bild 5.1, Knoten Nr. 1) und im Shared Memory des betrachteten Systems abzulegen, so dass die Daten nachfolgender Funkrahmen von der vorangegangenen OSI-Schicht in den gleichen Bereich der I/O-Schnittstelle geschrieben werden kön-

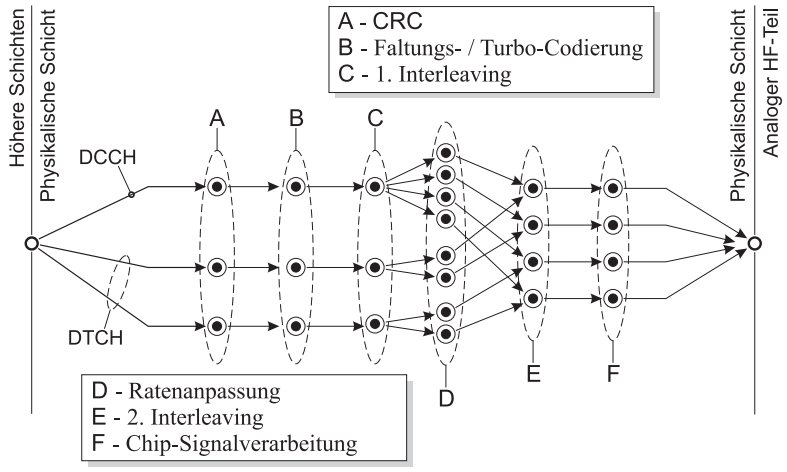


Bild 5.9 Gerichteter Graph, UTRA FDD (Sender), 64 kbps DTCH Uplink [23]

nen, während die Signalverarbeitung des aktuellen Funkrahmens unabhängig davon **im System** abläuft. Genauso sollten die anfallenden Ausgangsdaten zunächst komplett innerhalb des Systems zusammengestellt werden (Bild 5.1, Knoten Nr. $5-8$), bevor sie in einem Zug (Bild 5.1, Knoten Nr. 9) über die I/O-Schnittstelle an die nachfolgende OSI-Schicht weitergegeben werden.

Dieses Modell geht mit der Vorstellung einher, dass die Signalverarbeitung letztlich nur mit den Daten aus einem Speicherbereich arbeitet, der sich als Teil des Shared Memory **im Inneren** des Systems befindet, während der Bereich der I/O-Schnittstelle nur für die Daten des aktuellen Funkrahmens vorübergehend aktiv ist und ansonsten als Schattenspeicher (*engl.* shadow memory) zum Schreiben bzw. Lesen von Daten nachfolgender bzw. vorhergehender Rahmen genutzt wird.

Rein serielle Knoten wie die Knoten Nr. 1 und 9 führen aber zu dem Nachteil, das sie bei einer reinen Parallelisierungsstrategie und einer Ablaufplanung ohne Pipelining sequentiell abgearbeitet werden müssen und folglich das Entstehen von Leerlaufzeiten auf einem der Prozessoren unausweichlich ist. Zweitens ist es zeitraubend, erst alle Eingangsdaten im Shared Memory zu sammeln und sie dann an die Prozessoren zu verteilen bzw. die Ausgangsdaten von den verteilten Speichern wieder zu sammeln, bevor sie an die I/O-Schnittstelle übergeben werden. Ein Vorteil in der Speicherbelegung ergibt sich dadurch ebenfalls nicht, weil es für den Gesamtumfang des Speichers unerheblich ist, ob der aktive Teil des Speichers in Form des Shared Memory **innerhalb** des Systems oder an der I/O-Schnittstelle liegt.

Geht man zur Darstellung in Bild 5.9 über, so gibt es keine ausschließlich seriellen Bustransfers mehr, sondern als Knoten des Graphen treten nur noch nachrichtentechnische Funktionen auf. Die Bustransfers von Ein- und Ausgangsdaten werden nicht mehr gesondert behandelt, sondern in den regulären Partitionierungsprozess einbezogen. Zur Vorbereitung auf die Partitionierung werden die Kanten, die dadurch ausgezeichnet sind, dass sie vom Startknoten ausgehen oder in den Zielknoten münden, mit einem Kantengewicht Null versehen. Die **einfache** Knotenlaufzeit für einen I/O-Bustransfer wird zeitweise dem angrenzenden regulären Knoten zugeschlagen. Dieses Vorgehen ist unabhängig von dem verwendeten Partitionierungsansatz sinnvoll, weil die Knotenlaufzeiten für das Lesen und Schreiben der Ein- und Ausgangsdaten ja auf jeden Fall entstehen, unabhängig davon, ob der bei der Partitionierung entstehende Schnitt die ausgezeichneten Kanten betrifft oder nicht. Wenn die Ausgeglichenheit der Partitionen nicht dagegenspricht, ist die Wahrscheinlichkeit, dass ein Schnitt auch ausgezeichnete Kanten betrifft,

groß, weil deren Kantengewichte den Beitrag Null zur Summe der Schnittkosten liefern. Der genaue Verlauf des Schnittes ergibt sich aber nach wie vor aus der Partitionierungsphase.

Vor der Ablaufplanung werden die zeitweise erhöhten Laufzeiten der regulären Knoten wieder zurückgesetzt und der ursprüngliche Graph um Bustransfer-Knoten erweitert. Wenn Kanten zwischen regulären Knoten vom Schnitt betroffen sind, so wird die Knotenmenge $\mathbb{M}$ wie gewohnt durch ein Paar von Bustransfer-Knoten ergänzt. Im Gegensatz dazu werden alle ausgezeichneten Kanten bedingungslos durch **einzelne** Bustransfer-Knoten mit den entsprechenden Laufzeiten ersetzt. Die Ablaufplanung kann so ohne weitere Änderungen mit dem derart ergänzten Graphen arbeiten.

Dieses verbesserte Modell geht mit der Vorstellung einher, dass nun aktiver Speicher und Schattenspeicher (sowohl für Ein- als auch für Ausgangsdaten) in getrennten Bereichen der I/O-Schnittstelle liegen und ihre Rolle während der gesamten Echtzeitperiode aufrechterhalten wird. Lesezugriffe auf Eingangsdaten und Schreibzugriffe auf den aktiven Ausgangsspeicher können innerhalb von ΔT zu beliebigen Zeitpunkten auftreten, und der Datenaustausch erfolgt ohne Umweg über das Shared Memory direkt zwischen I/O-Schnittstelle und den verteilten Speichern der Prozessoren. An den Rahmengrenzen gilt es, lediglich die Rollen von aktivem Speicher und Schattenspeicher durch Veränderung der Zeiger zu tauschen, so dass die Signalverarbeitung ohne weitere zeitliche Synchronisation zwischen dem betrachteten System und der Außenwelt auf neuen Daten weiterlaufen kann [81].

Damit lässt sich eine weitere grundlegende Entwurfsrichtlinie für Mod-SDR-Systeme formulieren: Der gesamte Datenspeicher für Inter-System-Interaktionen sollte möglichst an der I/O-Schnittstelle, also an den Außengrenzen des Systems, bereitgestellt werden. Während der gesamten Signalverarbeitung des aktuellen Rahmens i in der betrachteten OSI-Schicht bearbeitet die vorangegangene OSI-Schicht bereits die Daten des nachfolgenden Rahmens $(i + 1)$. Entsprechendes gilt für die nachfolgende OSI-Schicht und den vorangegangenen Rahmen $(i - 1)$. Für die physikalische Schicht eines Mod-SDR-Senders bedeutet das beispielsweise, dass der analoge Hochfrequenzteil bereits den Funkrahmen $(i - 1)$ sendet, während die Basisbandverarbeitung des aktuellen Rahmens i noch läuft. Dieses Software-Modell findet bei allen nachfolgenden Untersuchungen Anwendung.

5.1.2 Vergleich der Partitionierungsansätze ohne Pipelining

Als gemeinsame Referenz für die Bewertung aller Partitionierungsverfahren aus Kapitel 4 wird der Speedup bei idealer Parallelisierung nach Gleichung (4.35) herangezogen. Die Bilder 5.10, 5.11 und 5.12 zeigen den relativen Speedup $y = s/\overline{s}$ für implizite Partitionierung nach Hu, für den KL-Algorithmus und für die spektrale Partitionierung in Abhängigkeit von der relativen Busgeschwindigkeit β. Als Software-Modell wird der Graph aus Bild 5.9 zugrunde gelegt. Wie gewohnt geben die Punkte die Messwerte von 2000 Realisierungen pro Parameterwert β wieder. Die Punkte liegen oft so dicht, dass ihre Überlagerung eine senkrechte Linie über dem entsprechenden Parameterwert ergibt. Zusätzlich zu den Messwerten sind die Schätzungen für die Höhenlinien des 5%-, der 50%- und des 95%-Quantils eingetragen. In Analogie zu Kapitel 3 beschreiben die Höhenlinien hier den maximal erreichbaren relativen Speedup für das entsprechende Quantil von Realisierungen in Abhängigkeit vom Parameter β. Der Stichprobenumfang von 2000 wurde gewählt, damit auch die Höhenlinien des 5%- und des 95%-Quantils zuverlässig geschätzt werden können. Wo die Höhenlinien eine starke Krümmung aufweisen, sind die Stützstellen entlang der β-Achse dichter gelegt worden.

Alle Messergebnisse zeigen, dass es offensichtlich am schwierigsten ist, hohe relative Speedups zu erzeugen, wenn $\beta \approx 1$, also wenn die Busgeschwindigkeit und die Verarbeitungsgeschwindigkeit der Prozessoren in etwa gleich sind. Im Vergleich der drei Bilder läßt sich feststellen, dass die Höhenlinien für kleine Busgeschwindigkeiten ($\beta < 1$) beim Hu- und KL-Algorithmus in etwa gleich verlaufen. Bemerkenswert ist hingegen der Verlauf der Medianlinie bei spektraler Partitionierung im Bild 5.12. Für $\beta < 1$ werden hier wesentlich höhere Speedups erzielt. An dieser Stelle muss allerdings betont werden, dass mit der Bedingung $\beta < 1$ ein Kommunkationssytem beschrieben wird, dessen wesentlicher Engpass in den langsamen Bustransfers besteht, nicht in der Signalverarbeitung. Zwar zeigt der relative Speedup die Tendenz, mit sinkendem β wieder gegen $y = 1$ anzusteigen, aber dieses Verhalten wird lediglich durch das Absinken den Bezugswertes $\overline{s}$ nach Gleichung (4.35) erzeugt, da in diesem Bereich der relativen Busgeschwindigkeit die Bedingung $T_P < T_{IO}$ gilt. Ein solcher Betriebspunkt ist keinesfalls wünschenswert für ein Mod-SDR.

Jenseits von $\beta = 1$ steigen die Höhenlinien zunächst alle an. Die Streuung beim KL-Algorithmus bleibt aber sehr groß, so dass das Verfahren in der Praxis so nicht einsetzbar ist, selbst wenn der Bus sehr schnell betrieben wird. Das impli-

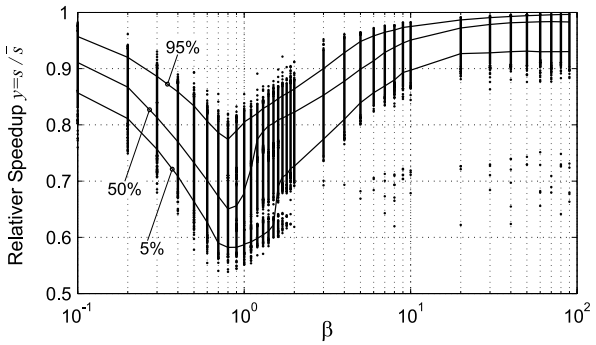


Bild 5.10 Implizite Partitionierung nach Hu, $B = 1$, Ablaufplan ohne Pipelining

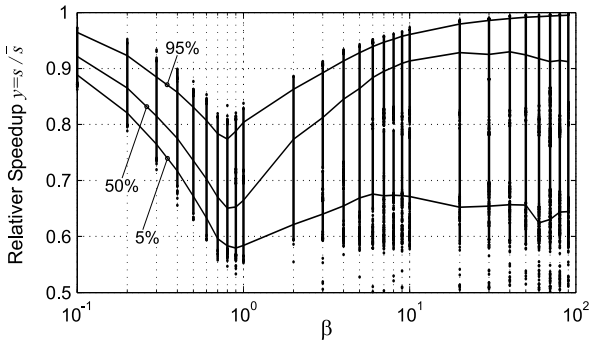


Bild 5.11 KL-Algorithmus, $B = 1$, Ablaufplan ohne Pipelining

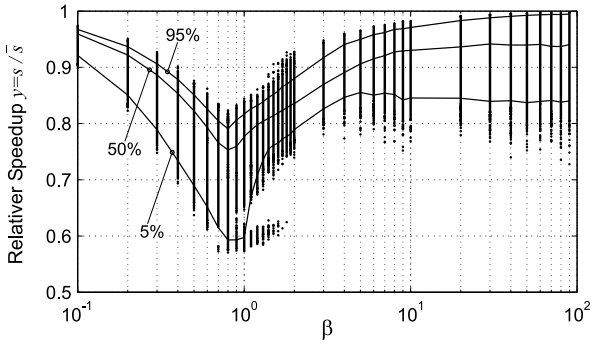


Bild 5.12 Spektrale Partitionierung, $B = 1$, Ablaufplan ohne Pipelining

zite Verfahren nach Hu und die spektrale Partitionierung weisen eine viel geringere Streuung des Speedups bei hohen Busgeschwindigkeiten auf. Der Hu-Algorithmus liefert die besten Ergebnisse mit der geringsten Streuung für $\beta > 5$ und einer Medianlinie, die der theoretischen Obergrenze von allen Verfahren am nächsten kommt. Leider weist das Ergebnis der impliziten Partitionierung auch Ausreißer um $s = 0{,}7\,\overline{s}$ auf. Ferner ist es äußerst unbefriedigend, dass keine Garantien für den Speedup in Abhängigkeit von β angegeben werden können.

Das Verhalten der spektralen Partitionierung ist unerwartet: Trotz der Fähigkeit, bei langsamen Bussen für die Mehrheit aller Realisierungen die besten Speedup-Ergebnisse zu erzeugen, wird das Verfahren bei schnellen Bussen von einem so einfachen und kaum theoretisch motivierten Ansatz wie dem Hu-Algorithmus übertroffen. Man könnte als nächstes klären, ob Buskonflikte für die Leerlaufzeiten verantwortlich sind, die den Speedup bei der spektralen Partitionierung reduzieren. Die Wahrscheinlichkeit für Buskonflikte ist bei $\beta \gg 1$ zwar gering, aber um das Speedupverhalten völlig ohne Buskonflike zu beobachten, werden die Simulationen mit $B = 2$ Bussen wiederholt.

5.1.3 Vergleich der Partitionierungsansätze mit zwei Bussen

Die Bilder 5.13, 5.14 und 5.15 zeigen für alle drei Partitionierungsansätze wiederum den relativen Speedup und jeweils die drei ausgewählten Höhenlinien. Es ist sofort zu erkennen, dass die Höhenlinien in allen drei Fällen im Bereich $\beta < 1$ die theoretische Grenze übersteigen. Das ist unmittelbar einsichtig, wenn man bedenkt, dass diese Grenze für ideale Parallelisierung und **einen** Bus hergeleitet wurde. Für $\beta < 1$ wird das System aber klar von Bustransfers dominiert, und gerade dann zahlt sich die Verfügbarkeit eines zweiten Busses aus, weil prinzipiell auch Bustransfer-Knoten von beiden Prozessoren gleichzeitig ausgeführt werden können.

Nach dieser Interpretation ist es um so bemerkenswerter, dass die Einführung eines zweiten Busses bei $\beta > 1$ für keines der drei Partitionierungsverfahren eine wesentliche Verbesserung des Speedups erbringt. Die implizite Partitionierung nach Hu leidet nach wie vor an einzelnen Ausreißern, der breit gestreute Speedup des KL-Algorithmus ist für den praktischen Betrieb in einem Mod-SDR nicht akzeptabel und der Speedup bei der spektralen Partitionierung ist hier ebenfalls größer als bei der Hu-Lösung. Im Besonderen ist zu bemerken, dass sich alle Höhenlinien im Bild 5.15 für $\beta \gg 1$ genau gleich verhalten wie in Bild 5.12, also ohne den zweiten Bus. Da in den drei aktuellen Bildern die Grundlage für den Prozessor-

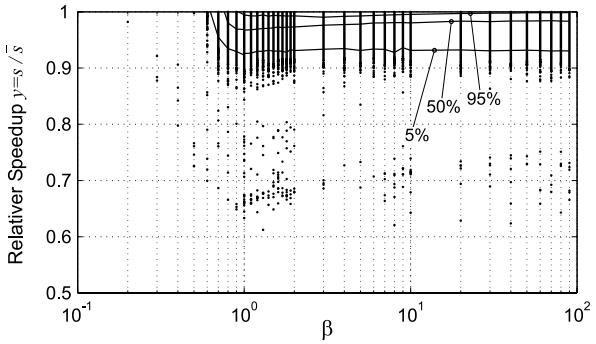


Bild 5.13 Implizite Partitionierung nach Hu, $B = 2$, Ablaufplan ohne Pipelining

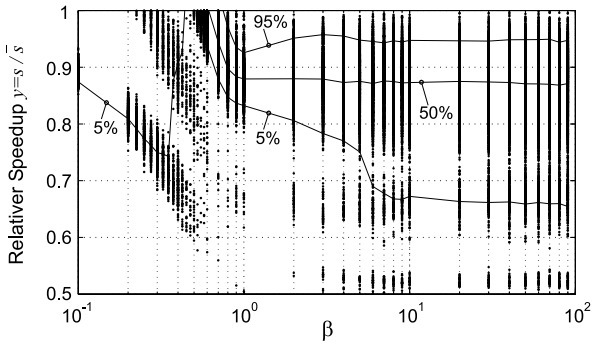


Bild 5.14 KL-Algorithmus, $B = 2$, Ablaufplan ohne Pipelining

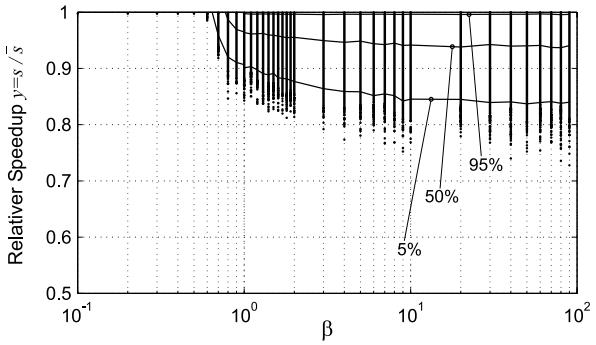


Bild 5.15 Spektrale Partitionierung, $B = 2$, Ablaufplan ohne Pipelining

zustand PHYSICAL_WAIT_IDLE durch den zweiten Bus entfällt, ist das beobachtete Speedupverhalten bislang als Folge ungünstiger logischer Abhängigkeiten zwischen Modulen zu interpretieren. Um diese Interpretation zu überprüfen, werden die Simulationen erneut durchgeführt, dieses Mal aber mit $B = 1$ Bus und unter Anwendung des Pipelining-Prinzips.

5.1.4 Vergleich der Partitionierungsansätze mit Pipelining

Die Bilder 5.16, 5.17 und 5.18 zeigen die Speedup-Ergebnisse für alle drei Partitionierungsansätze in der gleichen Form wie zuvor. Dabei fallen sofort die dicht an der theoretischen Obergrenze liegenden Ergebnisse auf, die mit dem KL-Algorithmus erzielt wurden. Offensichtlich verhilft in erster Linie das Vermeiden der Leerlaufzeiten zufolge des Zustands LOGICAL_WAIT_IDLE dem Ansatz von Kernighan und Lin zur erwarteten Überlegenheit gegenüber dem Hu-Algorithmus, der hier durchweg niedrigeren Speedup erzielt. Buskonflikte sind immer noch möglich, werden aber mit zunehmendem β immer unwahrscheinlicher, und so steigt die Wahrscheinlichkeit für hohen Speedup (siehe Bild 5.17 bei $\beta \gg 1$).

Im Vergleich dazu erzielt die spektrale Partitionierung keine befriedigenden Ergebnisse. Das Bild 5.18 ist fast identisch mit dem Bild 5.12, obwohl mit Pipelining und einem schnellen Bus fast ideale Anwendungsbedingungen herrschen. Gerade die vom KL-Algorithmus hervorgebrachten Ergebnisse zeigen ja, dass unter den genannten Randbedingungen fast idealer Speedup auch praktisch zu erreichen ist. Diese Beobachtungen lassen einzig und allein einen Schluss zu: Der Grund für die vergleichsweise schlechte Leistungsfähigkeit der spektralen Partitionierung ist nicht in den Randbedingungen der Ablaufplanung zu suchen, sondern bereits in der Phase der Partitionierung. Das Bild 5.19 zeigt das Verhältnis der Gesamtlaufzeiten beider Partitionen, wobei stets die Laufzeit der kleineren Partition auf die Laufzeit der größeren bezogen wird. Dieses Qualitätsmaß bewertet den Erfolg der Partitionierung, nicht den der Ablaufplanung. Es ist festzustellen, dass mit wachsender Busgeschwindigkeit auch ein wachsendes Lastungleichgewicht zwischen den Partitionen auftritt. Dieses Verhalten ist ein starker Hinweis darauf, dass die spektrale Partitionierung in der gegebenen Form nicht optimal arbeitet und im Hinblick auf die Ausgewogenheit der Lastverteilung verbessert werden sollte.

Nach den bisherigen Speedup-Ergebnissen geht der KL-Algorithmus eindeutig als leistungsfähigstes Partitionierungsverfahren hervor, wenn auch das Pipelining-Prinzip zur Anwendung kommt. Es liegt also zunächst nahe, den KL-Algorithmus als

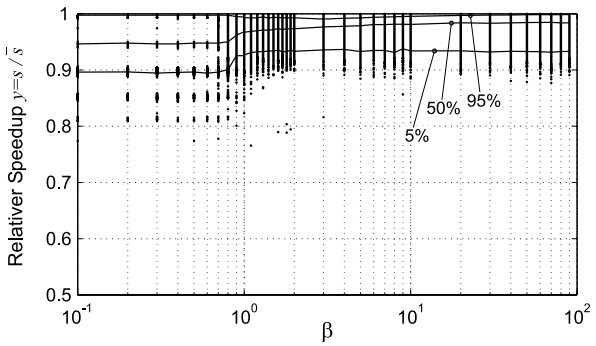


Bild 5.16 Implizite Partitionierung nach Hu, $B = 1$, Ablaufplan mit Pipelining

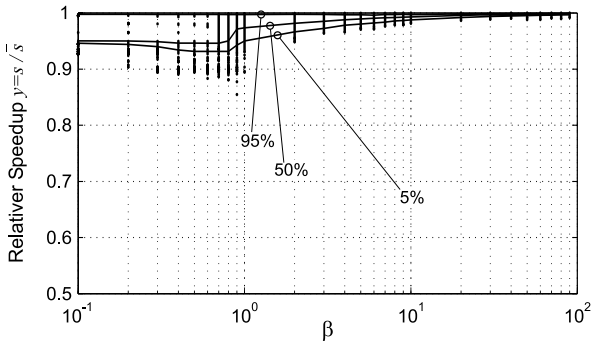


Bild 5.17 KL-Algorithmus, $B = 1$, Ablaufplan mit Pipelining

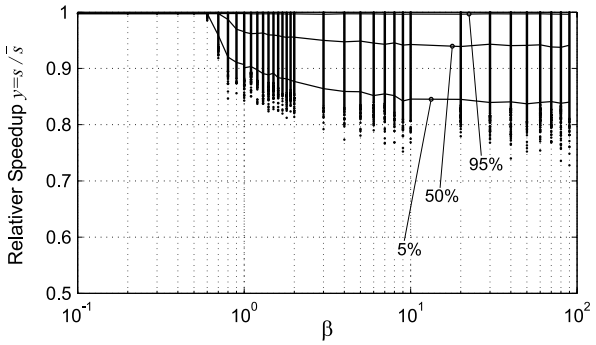


Bild 5.18 Spektrale Partitionierung, $B = 1$, Ablaufplan mit Pipelining

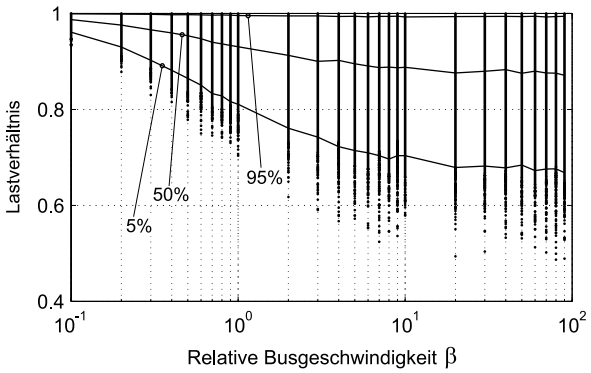


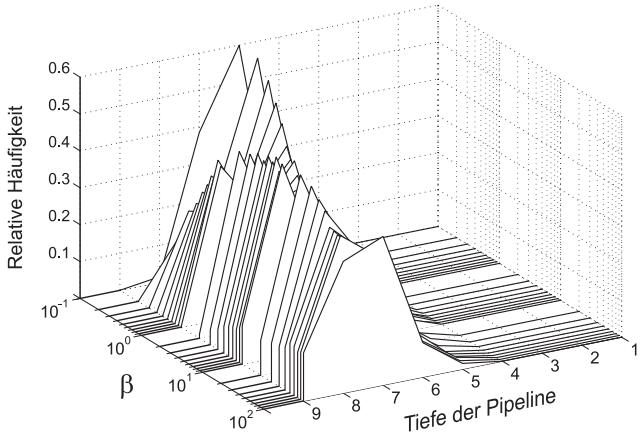
Bild 5.19 Lastverhältnis der kleineren zur größeren Partition

besten Kandidaten für die Aufgabe der Partitionierung im Kern eines Mod-SDR-Betriebssystems zu sehen und auf jeden Fall Pipelining als Verarbeitungsprinzip für Mod-SDR-Software festzuschreiben.

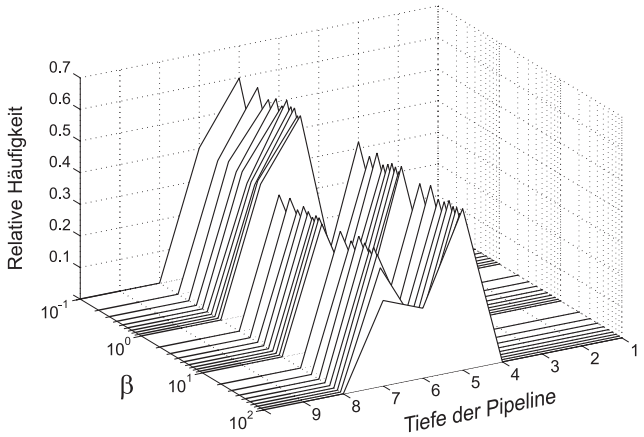
Bereits mehrfach wurde als grundlegender Nachteil von Pipelining die Verzögerung erwähnt, die allein durch die Art der Signalverarbeitung entsteht. An dieser Stelle wird die zu erwartende Verzögerung am Beispiel des Standard-Graphen für den UTRA FDD Uplink aus Bild 5.9 quantifiziert. Das Bild 5.20 zeigt die Histogramme der Pipeline-Tiefen in Abhängigkeit von der relativen Busgeschwindigkeit β für den KL-Algorithmus und für die spektrale Partitionierung. Der KL-Algorithmus benötigt zwischen 5 und 9 Rahmen in der Pipeline, um den Speedup von Bild 5.17 zu erzielen, die spektrale Partitionierung kommt mit einer Tiefe von 5 bis 7 aus und erreicht lediglich die Ergebnisse von Bild 5.18. Der Speicheraufwand, der mit solchen Pipeline-Tiefen zwangsläufig verknüpft ist, ist gerade für mobile Endgeräte enorm. Unter diesen Bedingungen ist es fraglich, ob die besagte Kombination aus KL-Algorithmus und Ablaufplanung mit Pipelining favorisiert werden sollte. Wenn Pipelining schon notwendig ist, um hohen Speedup zu erreichen, so gibt es doch sicherlich einfachere Betriebsarten für ein Mod-SDR.

5.1.5 Ergebnisse mit Graphenverdoppelung

In [77] wird ein Verfahren zur Organisation der Signalverarbeitung vorgeschlagen, das auf der künstlichen Erzeugung von Parallelzweigen durch Graphenverdoppe-



(a) Partitionierung mit dem KL-Algorithmus



(b) Spektrale Partitionierung

Bild 5.20 Histogramme der Pipeline-Tiefen

lung beruht. Es wird zunächst dafür gesorgt, dass **ein Paar** von Datenrahmen im Prinzip gleichzeitig verarbeitet werden kann. Allein Konflikte auf dem Bus sorgen für Leerlaufzeiten. Die Ablaufplanung zielt nur auf Parallelverarbeitung ab, und das bedeutet, dass die zeitlichen Grenzen der Signalverarbeitung für das Rahmenpaar übereinstimmen. Die Folge ist, dass nur noch jeder zweite Zeitschlitz eines

TDMA-basierten Mobilfunkstandards bedient werden kann. Um die mittlere Nutzdatenrate dennoch aufrechtzuerhalten, muss jeder Sender ein Bandspreizverfahren mit zwei orthogonalen Codes, also eine Multicode-Übertragung [55], anwenden. Dafür muss neben der TDMA- auch eine CDMA-Komponente in der Luftschnittstelle des Standards vorgesehen sein. Zusätzlich muss das Netzmanagement zwei in dieser Weise gleichartig operierenden mobilen Nutzern jeweils eine Sendeberechtigung für die geraden oder die ungeraden Zeitschlitze zuweisen, um die Kapazität einer Mobilfunkzelle aufrechtzuerhalten.

Dieses Verfahren ist zwar theoretisch möglich, für den praktischen Einsatz in einem Mod-SDR aber als untauglich einzustufen, und dafür gibt es zwei Gründe: Erstens ist das Einsatzgebiet auf hybride TDMA/CDMA-Standards beschränkt, und die Entwurfsrichtlinien für Mod-SDR sollen ja gerade nicht auf bestimmte Mobilfunkstandards zugeschnitten, sondern allgemein gültig sein. Zweitens entspringt das Verfahren allein der Unfähigkeit des mobilen Endgerätes, jeden Zeitschlitz zu bedienen, und das Verfahren verschiebt das Problem nicht nur vom Mobilgerät zur Basisstation, sondern würde auch zusätzlichen Signalisierungsaufwand nach sich ziehen, beispielsweise durch die Zuteilung der Sendeberechtigung für gerade und ungerade Zeitschlitze. Aus diesen Gründen ist der Vorschlag wieder verworfen worden.

Im Hinblick auf den Speedup ist weiterhin festzustellen, dass das in [77] beschriebene Verfahren es nicht vermag, die obere Schranke aus Gleichung (4.35) zu erreichen. Der wesentliche Punkt, der das Erreichen des maximal möglichen Speedups verhindert, besteht in dem starren Festhalten an der Idee der Parallelisierung im Zusammenhang mit der Graphenverdoppelung. Wendet man stattdessen das Pipelining-Prinzip an, so erhält man Graph Duplication Pipelining, das bereits im Abschnitt 4.3.2 ausführlich beschrieben und als optimal hinsichtlich des Speedups erkannt wurde.

Das Bild 5.21 zeigt den Ablaufplan für GDP, $L = 2$ und äquidistante Grenzen der Funkrahmen nochmals im Überblick. Im eingeschwungenen Zustand ist die Echtzeitperiode dicht mit Knotenlaufzeiten ausgefüllt, und das Füllen und Leeren der Pipeline ist gegenüber der Verbindungsdauer unerheblich. Während einer Echtzeitperiode ΔT werden immer Daten von genau zwei verschiedenen Rahmen verarbeitet, so dass GDP die Pipeline-Tiefe von 2 (unabhängig von den logischen Strukturen des Graphen) zugeschrieben werden kann. Damit ist die Tiefe kleiner als bei der Kombination von KL-Algorithmus und Pipelining aus Abschnitt 5.1.4. Gleichzeitig ist die Tiefe 2 die kleinste mögliche Pipeline-Tiefe, und folglich ist

GDP nicht nur bezüglich des Speedups, sondern auch bezüglich des Verzögerungsverhaltens optimal für die Mod-SDR-Signalverarbeitung bei leitungsvermittelten Diensten.

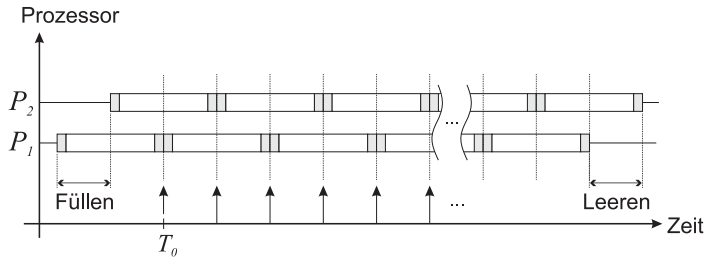


Bild 5.21 GDP, Ablaufplan mit äquidistanten Grenzen der Funkrahmen

5.1.6 Leistungseffizienter Betrieb eines Coprozessors

Pipelining hat sich in der Anwendung auf die Signalverarbeitung in Software bisher als ein sehr leistungsfähiges Prinzip erwiesen. In diesem Abschnitt wird daher untersucht, ob sich Pipelining auch auf den Betrieb von Hardware ausdehnen lässt und welche Vor- und Nachteile damit verknüpft sind.

Im Zusammenhang mit Mehrprozessorsystemen taucht immer wieder die Idee auf, einen Coprozessor als Beschleuniger für rechenintensive Teilaufgaben einzusetzen und so den Speedup insgesamt zu erhöhen. Als Hardware-Modell wird hier das System aus Bild 2.5 mit zwei Bussen zugrunde gelegt. Dieses System, in dem alle Hardware-Komponenten mit endlicher Geschwindigkeit betrieben werden, soll im Folgenden als **Gesamtsystem** bezeichnet werden.

Es stellt sich zunächst die Frage, welche Knoten eines gegebenen Graphen am besten auf den Coprozessor auszulagern sind. Als Entscheidungskriterium wird folgendes Ziel formuliert: Unter allen Knoten $m : 1 \leq m \leq M$ wird derjenige Knoten $\widehat{m}$ zur Auslagerung vorgesehen, bei dem eine Erhöhung der Ausführungsgeschwindigkeit um den Faktor $\kappa \in \mathbb{R} : \kappa > 1$ (durch den Coprozessor) den Gewinn $g(m)$ gegenüber der ursprünglichen Laufzeitsumme aller Knoten (ohne Beschleunigung durch den Coprozessor) maximiert:

$$\widehat{m} = \arg\max_{m} \frac{\partial g(m)}{\partial \kappa} \qquad \text{mit} \quad g(m) = \frac{\sum_{i=1}^{M} p_i}{\sum_{i \neq m} p_i + \frac{p_m}{\kappa}} \tag{5.1}$$

Ohne Einschränkung der Allgemeinheit werden die Knoten des Graphen jetzt so umindiziert, dass sie damit auch in aufsteigender Reihenfolge ihrer Laufzeiten geordnet sind:

$$p_1 \leq p_2 \leq \ldots \leq p_m \leq \ldots \leq p_{M-1} \leq p_M \tag{5.2}$$

Die partielle Differentiation ergibt:

$$\frac{\partial g(m)}{\partial \kappa} = \frac{1}{\kappa^2} \cdot \sum_{i=1}^{M} p_i \cdot \frac{p_m}{\left[\sum_{i \neq m} p_i + \frac{p_m}{\kappa}\right]^2} \tag{5.3}$$

$$= \frac{1}{\kappa^2} \cdot \sum_{i=1}^{M} p_i \cdot \frac{p_m}{\left[\sum_{i=1}^{M} p_i - p_m(1 - \frac{1}{\kappa})\right]^2} \tag{5.4}$$

Die Summe aller Laufzeiten sowie alle Terme in κ sind bezüglich der Wahl von m konstant, und so ergibt sich das Maximum von Gleichung (5.4), wenn der Zähler des Bruchterms maximal und der Nenner gleichzeitig minimal wird. Das ist der Fall, wenn $p_m = p_M$, also wenn der Knoten mit der größten individuellen Laufzeit als erstes auf den Coprozessor ausgelagert wird. Dies entspricht auch der intuitiven Lösung: Aus der Ordnung der Module in Gleichung (5.2) wird sofort klar, dass p_M den größten relativen Anteil an der Gesamtsumme der Laufzeiten hat. Bei linearer Beschleunigung mit dem Faktor κ bewirkt die Auswahl von p_M also auch den größten relativen Gewinn $g(M)$. Mit dem erweiterten Ressourcen-Laufzeit-Modell aus Abschnitt 2.2.4 ergibt sich für das Beispiel des Standard-Graphen, dass die vier Knoten der Chip-Signalverarbeitung (siehe Bild 5.9, Detail "F") auf den Coprozessor ausgelagert werden, weil sie unter allen Knoten die gleiche maximale Laufzeit haben.

Als nächstes ist die Frage zu klären, wie der Coprozessor im Zusammenspiel mit allen anderen Hardware-Komponenten am besten zu betreiben ist, und das bedeutet möglichst hoher Speedup bei möglichst geringer elektrischer Verlustleistung. Die Existenz des dritten Prozessors im Gesamtsystem stellt nun eine völlig neue Situation für den Mod-SDR-Entwurf dar, denn alle Algorithmen sind bisher nur auf Basis eines homogenen Zwei-Prozessor-Systems behandelt worden. Allerdings lässt sich für den Spezialfall **eines** Coprozessors zumindest eine Abschätzung für den Speedup des Gesamtsystems herleiten, und zwar durch Betrachtung zweier ausgezeichneter Zwei-Prozessor-Systeme: Einerseits das unbeschleunigte System,

das nach wie vor über zwei identische Prozessoren und zwei Busse verfügt. Andererseits das hypothetische System, das mit einem unendlich schnellen Coprozessor arbeitet. Dieses hypothetische System kann herkömmlich mit zwei Prozessoren modelliert werden, die auf einem **modifizierten Graphen** arbeiten, wobei die auf den Coprozessor ausgelagerten Knoten die Laufzeit Null zugewiesen bekommen. Die Ein- und Ausgangsdaten dieser Knoten müssen aber nach wie vor mit endlicher Geschwindigkeit über die beiden verfügbaren Busse transferiert werden.

Zum einen ist der Coprozessor nur sinnvoll, wenn er einen positiven Speedup-Gewinn gegenüber dem unbeschleunigten System erzielt. Zum anderen kann das betrachtete System niemals höheren Speedup erzeugen als das hypothetische System. Folglich liegen sinnvolle Speedup-Ergebnisse des Gesamtsystems mit drei Prozessoren irgendwo zwischen den Kurven für diese beiden ausgezeichneten Systeme.

Speedup wird in diesem Abschnitt relativ zum maximal erreichbaren Speedup angegeben. Dieses Maximum ist mit der minimalen Echtzeitperiode verknüpft, die sich ergibt, wenn alle Bustransfer-Knoten vernachlässigt werden und sich die Summe aller Knotenlaufzeiten des modifizierten Graphen völlig ausgewogen auf zwei Prozessoren verteilen lassen. Diese minimale Laufzeit ist zwar abhängig von der Realisierung, lässt sich während der Simulation aber einfach ermitteln. Das Verhältnis dieses Minimums zur tatsächlich realisierten Echtzeitperiode des Systems ist als Bruchteil des theoretisch maximal erreichbaren Speedups bei Coprozessornutzung zu interpretieren.

Für das Beispiel des Graphen aus Bild 5.9 und die beiden ausgezeichneten Systeme zeigen die Bilder 5.22 und 5.23 die entsprechenden Speedup-Ergebnisse (Punkte, linke Ordinate) und die Busaktivitäten in Prozent (Kreise/Kreuze für Bus 1/2, rechte Ordinate) in Abhängigkeit von der relativen Busgeschwindigkeit β. Der Stichprobenumfang in den Bildern beträgt 400 Realisierungen pro Parameterwert β. Zur Partitionierung wird der KL-Algorithmus eingesetzt und die Ablaufplanung erfolgt entsprechend dem Abschnitt 4.3.1 für $B = 2$ Busse mit Priorisierung der Bustransfers und sogar mit Software-Pipelining, obwohl damit nach den Ergebnissen aus Abschnitt 5.1.1 erhöhter Speicherbedarf und Verzögerung verbunden sind. Dennoch sind ohne Coprozessor nicht einmal 40% des maximal möglichen Speedups zu erreichen, selbst wenn die beiden Busse sehr schnell betrieben werden. Im Gegensatz dazu nähert sich das hypothetische System durchaus der theoretischen Obergrenze. Trotz des unendlich schnellen Coprozessors (und unendlich hoher Verlustleistung) gehen die umfangreichen Ausgangsdaten der Chip-Signalverarbeitung über das Bussystem, und so erfordert ein hoher Speedup auch

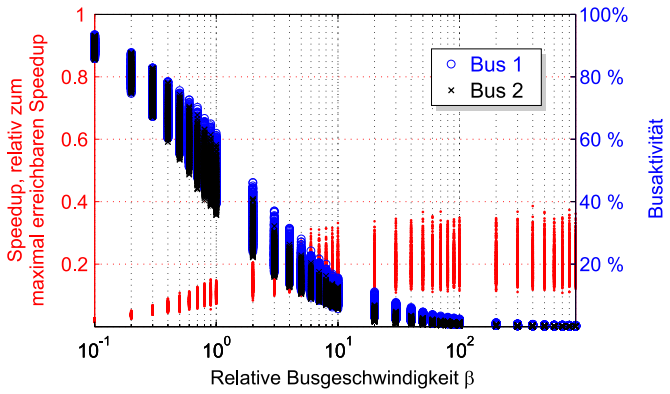


Bild 5.22 Unbeschleunigtes System, $B = 2$ Busse

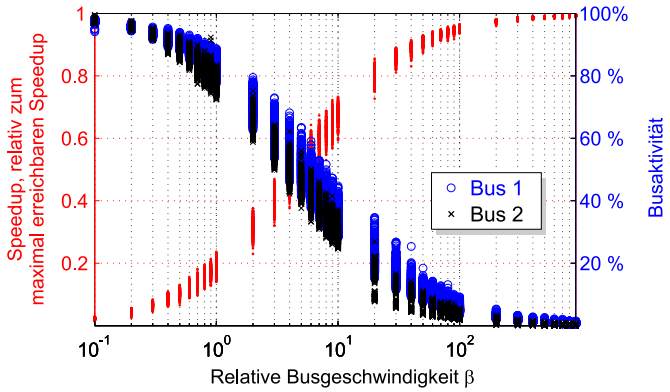


Bild 5.23 Hypothetisches System, $B = 2$ Busse

hohe Busgeschwindigkeiten. Das ist bezüglich der dynamischen Verlustleistung in CMOS-Schaltungen ein großer Nachteil.

Als Alternative zum Gesamtsystem wird im Folgenden ein Ansatz betrachtet, der in erster Linie die Hardware-Architektur betrifft, denn das betrachtete Gesamtsystem wird dabei in zwei getrennte Subsysteme unterteilt, die nur über das Shared Memory gekoppelt sind: Das Coprozessor-Subsystem nach Bild 2.6 einerseits und das Restsystem nach Bild 2.4 andererseits. Pipelining zwischen diesen Subsystemen

wird im Prinzip genau so organisiert wie das Pipelining zwischen dem Gesamtsystem und der Außenwelt, oder wie das Software-Pipelining zwischen zwei Halbrahmen: Während der Coprozessor noch die Chip-Signalverarbeitung des Funkrahmens i erledigt, beginnen die beiden Prozessoren des Restsystems bereits, die Daten des nachfolgenden Rahmens $(i + 1)$ zu verarbeiten. Als systeminterne Schnittstelle mit aktivem Speicher und Schattenspeicher dient nun das Shared Memory. Das Konzept eines solchen Subsystem-Pipelining erhöht die Tiefe einer bereits bestehenden Software-Pipeline um 1, wenn das Coprozessor-Subsystem das Pipelining-Prinzip nicht anwendet. Andernfalls addieren sich die Pipeline-Tiefen der beiden Subsysteme.

Der wesentliche Vorteil dieses architekturbasierten Ansatzes ist die Tatsache, dass das Restsystem nicht die großen Datenmengen der Chipsignalverarbeitung, sondern nur noch die Ausgangsdaten des zweiten Interleavers (siehe Bild 5.9, Detail "E") über den Bus austauschen muss. Ein denkbarer Nachteil besteht darin, dass beide Subsysteme jetzt nur noch jeweils einen Bus zum Datentransfer zur Verfügung haben, aber bereits die Ergebnisse in Abschnitt 5.1.4 haben gezeigt, dass der Erfolg des KL-Algorithmus im Hinblick auf hohen Speedup durch einen einzigen Bus nicht wesentlich gemindert wird, solange nur Software-Pipelining zum Einsatz kommt.

Für das Restsystem mit einem Bus zeigt das Bild 5.24 den Bruchteil des maximal erreichbaren Speedups (Punkte, linke Ordinate) und die Busaktivität in Prozent

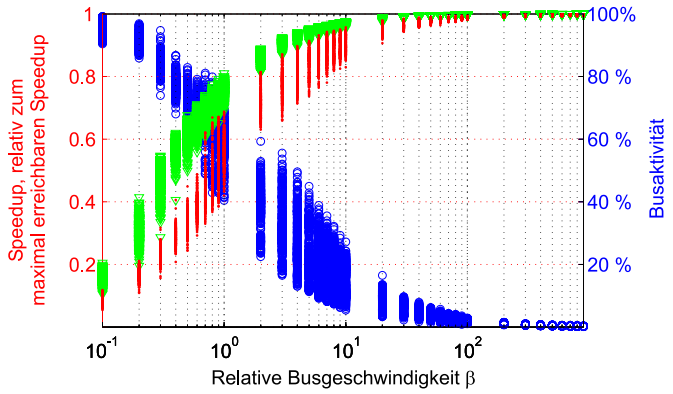


Bild 5.24 Restsystem, $B = 1$ Bus

(Kreise, rechte Ordinate). Zusätzlich ist eine realistische Grenze für den Speedup angegeben (graue Dreiecke), die im Gegensatz zur theoretischen Obergrenze die strikt serielle Ausführung aller Bustransfer-Knoten des modifizierten Graphen berücksichtigt. Diese praktische Grenze ist von der Realisierung abhängig und streut daher in ähnlicher Weise wie die Speedup-Messungen. Nichts desto trotz ist zu beobachten, dass der tatsächlich erreichbare Speedup des Restsystems sehr viel näher an der praktischen Grenze liegt als der Speedup des hypothetischen Systems. Ferner wird die theoretische Obergrenze von 1 vom Restsystem bereits bei viel geringeren Busgeschwindigkeiten ($\beta \approx 10$) gut angenähert. Eine notwendige Bedingung für die Gültigkeit der Speedup-Ergebnisse in Bild 5.24 ist, dass das in Pipeline arbeitende Coprozessor-Subsystem mindestens dieselbe Echtzeitperiode einhält wie das Restsystem. Deshalb ist zu prüfen, ob der Vorteil, das Restsystem bei niedrigen Busgeschwindigkeiten betreiben zu können, nicht von sehr großen Taktraten im Coprozessor-Subsystem wieder zunichte gemacht wird.

Im Laufe der Argumentation gilt es, die Verlustleistungen von Hardware-Komponenten zu vergleichen, und zwar für das Gesamtsystem ohne Pipelining einerseits und für die beiden Subsysteme mit Pipelining andererseits. Baugleiche Einzelkomponenten wie Prozessoren und Busse direkt miteinander zu vergleichen ist relativ einfach, wenn man ihre Aktivität bezüglich der Echtzeitperiode und die Taktraten, mit denen die Komponenten betrieben werden, vergleicht. Jedoch ist die Aufgabe, ein **zusammengesetztes System** wie das Coprozessor-Subsystem, das aus einem Prozessor **und** einem Bus besteht, bezüglich seiner dynamischen Verlustleistung zu beurteilen, ungleich schwieriger, denn Prozessor und Bus können erstens unterschiedlich getaktet sein und zweitens selbst bei gleicher Taktung verschieden große Verluste erzeugen. Daher ist für die Beurteilung dieses zusammengesetzten Systems zunächst eine passende Lösung zu finden. Als Modell für den Coprozessor dient dabei ein Prozessor, der baugleich mit den Prozessoren des Restsystems ist und der die Beschleunigung allein durch eine Erhöhung seines Prozessortaktes erzielt.

Unabhängig davon, ob man Prozessoren oder Bussysteme betrachtet, besteht der wesentliche Nachteil hoher Taktraten in der hohen dynamischen Verlustleistung

$$P_d \sim \overline{f} \cdot C \cdot V^2 \tag{5.5}$$

wobei $\overline{f}$ die mittlere Schaltfrequenz ist, C die effektive Kapazität der betrachteten CMOS-Struktur und V die Betriebsspannung. Für alle Leerlaufzeiten wird perfekte Taktabschaltung (*engl.* clock gating) der Prozessoren und Busse angenommen, d.h.

dynamische Verlustleistung wird nur dann erzeugt, wenn die CMOS-Komponenten tatsächlich aktiv sind. Im Rahmen der Rechnung wird weiterhin angenommen, dass der Coprozessor und der entsprechende Bus über verschiedene effektive Kapazitäten (Coprozessor: C_0, Bus: C_1) verfügen und unterschiedlich getaktet werden können (Coprozessor unbeschleunigt: f_0, Coprozessor beschleunigt: $f_0' = s_0 \cdot f_0$, Bus unbeschleunigt: f_1, Bus beschleunigt: $f_1' = s_1 \cdot f_1$, $s_0, s_1 \in \mathbb{R}$, $s_0 > 1$, $s_1 > 1$). Eine vom Restsystem vorgegebene Echtzeitperiode ΔT kann von zwei Taktraten f_0' und f_1' erreicht werden, die im Prinzip in einem beliebigen reellen Verhältnis zueinander stehen. Für den leistungseffizienten Betrieb eines Mod-SDR mit Subsystem-Pipelining wäre es aber wünschenswert, das **optimale** Verhältnis f_0'/f_1' zu kennen, sofern ein solches Optimum existiert.

Dazu soll jetzt angenommen werden, dass die Laufzeitsumme $(T_0 + T_1)$ aller auf dem (unbeschleunigten) Coprozessor ausgeführten regulären Knoten (T_0) und aller (unbeschleunigten) Bustransfer-Knoten (T_1) um den Betrag dT vermindert werden muss, um die Vorgabe von ΔT zu erreichen. Das Bild 5.25 zeigt diese Situation.

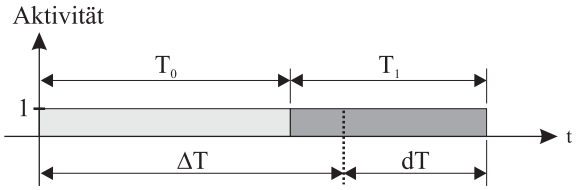


Bild 5.25 Laufzeiten aller ausgelagerten regulären Knoten und Bustransfer-Knoten

Weiterhin wird angenommen, dass der konstante Bruchteil $r \in \mathbb{R} : 0 \leq r \leq 1$ der notwendigen Laufzeitminderung dT durch eine Erhöhung der Busgeschwindigkeit im Subsystem erzielt wird, und der Rest, also $(1-r)dT$, durch die Beschleunigung des Coprozessors. Der Faktor s_1 der Beschleunigung gegenüber einem Bus mit der üblichen relativen Geschwindigkeit β kann folgendermaßen ausgedrückt werden:

$$s_1 = \frac{T_1}{T_1 - r \cdot dT} = \left[1 - r \cdot \frac{\chi}{1-\varphi}\right]^{-1} \tag{5.6}$$

wobei χ und φ abkürzend stehen für:

$$\chi := \frac{dT}{T_0 + T_1} \qquad \text{und} \qquad \varphi := \frac{T_0}{T_0 + T_1} \tag{5.7}$$

In gleicher Weise kann der Faktor s_0 der Beschleunigung des Coprozessors gegenüber einem unbeschleunigten Prozessor ausgedrückt werden:

$$s_0 = \frac{T_1}{T_1 - (1-r)\cdot dT} = \left[1 - (1-r)\cdot\frac{\chi}{1-\varphi}\right]^{-1} \tag{5.8}$$

Die mittleren Schaltfrequenzen ohne Beschleunigung lauten $\overline{f}_0 = \varphi \cdot f_0$ für den Coprozessor und $\overline{f}_1 = (1-\varphi)\cdot f_1$ für den Bus. Mit der Beschleunigung von s_0 bzw. s_1 ergeben sich neue mittlere Schaltfrequenzen (Coprozessor: $\overline{f}'_0$, Bus: $\overline{f}'_1$):

$$\overline{f}'_0 \;=\; s_0 \cdot f_0 \cdot \frac{T_0 - (1-r)\cdot dT}{T_0 + T_1 - dT} \tag{5.9}$$

$$\overline{f}'_1 \;=\; s_1 \cdot f_1 \cdot \frac{T_0 - r\cdot dT}{T_0 + T_1 - dT} \tag{5.10}$$

Der Leistungsfaktor r_p, um den die dynamische Verlustleistung des aus Bus und Coprozessor zusammengesetzten Subsystems unter den Bedingungen der Beschleunigung ansteigt, lautet:

$$r_p = \frac{\overline{f}'_0 C_0 + \overline{f}'_1 C_1}{\overline{f}_0 C_0 + \overline{f}_1 C_1} \tag{5.11}$$

Drückt man nun die zeitlichen Beziehungen in den Gleichungen (5.9) und (5.10) mit den Definitionen aus (5.7) aus und substituiert die Konstanten s_0 und s_1 durch die Ausdrücke in den Gleichungen (5.6) und (5.8), erhält man als Leistungsfaktor:

$$r_p = (1-\chi)^{-1} \tag{5.12}$$

Folglich hängt die Erhöhung der dynamischen Verlustleistung durch Beschleunigung des zusammengesetzten Coprozessor-Subsystems weder von dem Anteil φ regulärer Knoten am Gesamtzeitaufwand ab, noch von dem angestrebten Bruchteil r oder den effektiven Kapazitäten C_0 oder C_1, sondern einzig und allein von χ. Insbesondere existiert kein optimales Verhältnis f'_0/f'_1, sondern alle Kombinationen von f'_0 und f'_1, die zur Einhaltung von ΔT führen, sind bezüglich der erzeugten Verlustleistung gleich zu bewerten. Das reelle Verhältnis χ hängt von den ausgelagerten Knoten auf dem Coprozessor und von der Mod-SDR-Realisierung des Restsystems ab, und zwar über die Echtzeitperiode ΔT, die von einer bestimmten

Kombination aus Partitionierung und Ablaufplanung hervorgebracht wird. Für das Beispiel des UTRA FDD Standard-Graphen zeigt das Bild 5.26 den Leistungsfaktor r_p in Abhängigkeit von der relativen Busgeschwindigkeit β des Restsystems.

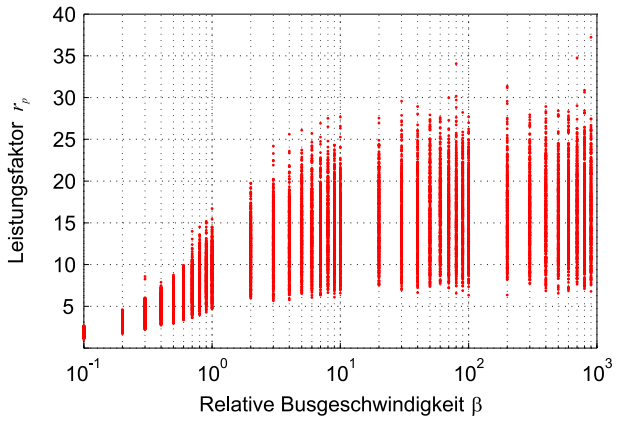


Bild 5.26 Leistungsfaktor r_p für das in Pipeline arbeitende Coprozessor-Subsystem

Schließlich ist die Verlustleistung des Gesamtsystems ohne Pipelining mit der Verlustleistung der beiden Subsysteme mit Pipelining zu vergleichen. Als gemeinsames Entwurfsziel sei das Erreichen von 90% des maximalen Speedups vorgegeben. Auf der einen Seite ist aus Bild 5.23 abzulesen, dass die relative Busgeschwindigkeit dafür bei mindestens $\beta = 50$ liegen muss. Beide Busse sind etwa 10% der Zeit aktiv. Auf der anderen Seite erreicht das Restsystem das Entwurfsziel bereits bei $\beta \approx 10$ (siehe Bild 5.24), wobei die Aktivität des einzigen Busses bei 20% liegt. Unter der Annahme, dass zwei separate Busse **jeweils** die gleiche effektive Kapazität aufweisen wie der einzelne Bus, ergibt sich daraus eine um den Faktor 5 geringere dynamische Verlustleistung für den Bus des Restsystems bei Subsystem-Pipelining.

Jetzt müssen noch zwei weitere Vergleiche durchgeführt werden: Erstens, die Prozessoren des in Pipeline arbeitenden Restsystems mit den zwei identischen Prozessoren des Gesamtsystems ohne Pipelining. Zweitens, das vollständige **zusammengesetzte Coprozessor-Subsystem** mit dem einzelnen Coprozessor des Gesamtsystems.

Die beiden identischen Prozessoren sind in beiden Betriebsarten nahezu ohne Leer-

lauf aktiv und tragen damit auf beiden Seiten des Leistungsbudgets gleichermaßen zu den Verlusten bei. Das Coprozessor-Subsystem erzeugt zwischen 5 und 25 Mal mehr dynamische Verlustleistung als das gleiche System ohne Beschleunigung. Der wesentliche Punkt ist aber: Unabhängig von dem tatsächlichen Wert ist diese Leistung sicherlich endlich, während die Verlustleistung des hypothetischen Systems unendlich ist. Man könnte also die ganze (endliche) Leistung des zusammengesetzen Coprozessor-Subsystems auch allein in den Coprozessor des Gesamtsystems investieren, und der Speedup würde trotzdem niemals besser werden als der Speedup in Bild 5.23.

Die Reduzierung der relativen Busgeschwindigkeit von $\beta = 50$ auf $\beta \approx 10$ bei Erreichen des gleichen Entwurfsziels bringt folglich die entscheidende Reduzierung der dynamischen Verlustleistung mit sich. Der Faktor 5 stellt eine konservative Schätzung dar, weil er mit den Ergebnissen des hypothetischen Systems ermittelt wurde. Der tatsächliche Gewinn bei Subsystem-Pipelining muss in der Tat größer als 5 sein. Die konkreten Werte hängen immer von der Zusammensetzung des ausgelagerten Teilgraphen aus regulären Knoten und Bustransfer-Knoten ab. Je größer der Anteil ausgelagerter Bustransfers, desto größer der zu erwartende Gewinn durch Subsystem-Pipelining. Ein weiterer Vorteil von Subsystem-Pipelining kann in dem geringeren Hardware-Aufwand gesehen werden: Alle Prozessoren benötigen nur noch Verbindung zu **einem** Bus und nicht jeweils zu beiden Bussen wie beim Betrieb des Gesamtsystems ohne Pipelining.

Insgesamt hat sich das Pipelining-Prinzip für leitungsvermittelte Dienste bisher als sehr leistungsfähig erwiesen, nicht nur im Hinblick auf den erreichbaren Speedup in der Software-Signalverarbeitung, sondern auch auf die Verlustleistung von Hardware. Es geht damit aus den vorliegenden Untersuchungen als **das** zentrale Entwurfsprinzip für Mod-SDR hervor. Ein Argument, das streng genommen immer noch an der Verallgemeinerung dieser Erkenntnis hindert, besteht allerdings darin, dass zwar mit dem erweiterten Ressourcen-Laufzeit-Modell bereits wichtige Aspekte der SDR-Umgebung für die Knoten eines gegebenen Graphen erfasst werden, die gerichteten Kanten in den Simulationen aber stets unverändert geblieben sind. Ein erster Schritt, den Raum der logischen Strukturen sinnvoll zu erweitern, besteht darin, alle Möglichkeiten der (geringfügig verschiedenen) UMTS-Luftschnittstellen (UTRA FDD Uplink, UTRA FDD Downlink, UTRA TDD) auszuschöpfen und die dort definierten Signalverarbeitungsgraphen für Untersuchungen bereitzustellen.

5.1.7 Familien von UMTS-Graphen

Das Software-Modell aus Abschnitt 2.2.5 ermöglicht es im Prinzip, Graphen für beliebige Kombinationen von leitungsvermittelten Transportkanälen konform zum UMTS-Standard zu erzeugen. Die Kombinationsmöglichkeiten sind dabei so vielfältig, dass es allein aus diesem Grund schon schwerfällt, sinnvolle Testfälle zu konstruieren. Der Standard bietet deshalb mehrere Beispiele für den Ablauf der Signalverarbeitung von Kanälen mit verschiedenen Datenraten an: Ein DTCH mit $12{,}2$ kbps, 64 kbps, 144 kbps und 384 kbps, jeweils begleitet von einem DCCH mit $2{,}5$ kbps [23]. Jedes dieser Beispiele kann auf abstrakte Weise als Graph repräsentiert werden. Die Datenraten sind dabei nicht ausgewählt worden, weil sie für die Übertragung über den Funkkanal optimal wären, sondern weil sie kompatibel sind zu den Standardraten und Komponenten existierender Kommunikationssysteme: GSM EFR Sprachencoder ($12{,}2$ kbps, [27, 28]), Nutzdatenraten eines ISDN B-Kanals (64 kbps) bzw. des ISDN Basisanschlusses (144 kbps, [34]), und eine Standard-Rate für Videodatenstöme (384 kbps, [86]). Im Zusammenhang mit der Abbildung von Signalverarbeitungssoftware stellt sich nun die Frage, ob sich gerade die Graphen zu diesen Datenraten günstig auf die Mod-SDR-Hardware abbilden lassen und automatisch hohen Speedup erzeugen. Insbesondere könnten Schwankungen in der Speedup-Streuung auf bessere oder schlechtere Vorhersagbarkeit des Systemverhaltens schließen lassen. Ebenso könnte es Datenraten geben, die für eine leistungseffiziente Signalverarbeitung im mobilen Endgerät besonders gut geeignet sind.

Für die Untersuchung sind Datenraten im Intervall von 64 kbps bis 144 kbps ausgewählt worden. Die Signalverarbeitung für die beiden Grenzen ist im Standard wohldefiniert: Beide setzen für den DTCH eine verkettete Kanalcodierung aus einem CRC-16 und einer Turbo-Codierung mit der Coderate $R_c = 1/3$ ein, und auch die restlichen Parameter, insbesondere das Verhältnis der TTIs von DTCH und DCCH (20ms und 40ms) bleibt hier konstant. Die einzige wesentliche Änderung beim Durchlaufen der Datenraten besteht in der Änderung des Slotformats [25]. Die Grenze für die Punktierung der codierten Nutzdaten (*engl.* puncturing limit) wurde auf 20% festgelegt. Damit ergibt sich das Umschalten im Downlink (Slotformat $13 \rightarrow 14$) bei einer Datenrate von $82{,}6$ kbps, im Uplink (Slotformat $4 \rightarrow 5$) bei einer Datenrate von $94{,}6$ kbps.

Aus Sicht der Nachrichtentechnik ist die Punktierung nur bis zu einem gewissen Grad sinnvoll. Zwar erlaubt ein steigender Punktierungsgrad immer größere Nutz-

datenraten bei gleichem Slotformat, aber dieser Vorteil wird mit einer wachsenden BER erkauft, und eine vereinbarte Übertragungsqualität ist damit ab einer bestimmten Grenze für die Punktierung nicht mehr einzuhalten. Ein Wechsel des Slotformats bringt zunächst immer den Wechsel von der Punktierung zur Bitwiederholung mit sich. Die Nutzdatenrate kann mit dem neuen Slotformat weiter erhöht werden, aber die Bitwiederholung zieht letztlich eine Verschwendung von Funkressourcen des Mobilfunknetzes nach sich, weil die wiederholten Bits noch nicht einmal zur Kanaldecodierung genutzt werden, sondern lediglich dem Auffüllen von Blöcken mit einer bestimmten Bitanzahl dienen, so dass die Blocklänge gerade in einen Funkrahmen passt.

Aus Sicht der Signalverarbeitung sind noch keine Bewertungsgrundlagen für bestimmte Datenraten bekannt. Beim Durchlaufen des Intervalls von 64 kbps bis 144 kbps ändern sich die Größen der Ausgangsspeicher r_m aller Knoten m : $1 \leq m \leq M$ und damit die Kantengewichte eines gegebenen Graphen, nicht aber seine grundlegende Struktur. Zusätzlich kommt das erweiterte Ressourcen-Laufzeit-Modell zum Einsatz und mit den Änderungen der r_m geht nach Gleichung (2.4) auch eine Änderung der Knotenlaufzeiten p_m einher. Solange das Slotformat gleich bleibt, gehen die Änderungen quasi kontinuierlich vonstatten. Der Wechsel des Slotformats ändert die Verhältnisse zwischen den Ausgangsspeichergrößen der Knoten jedoch abrupt. Die Auswirkungen dieser Einflüsse auf das Speedup-Verhalten, besonders im Zusammenspiel mit den Verfahren der Partitionierung und Ablaufplanung, werden durch Simulation im Rechner untersucht.

Es ist bereits festgestellt worden, dass bei Anwendung des Pipelining-Prinzips GDP bezüglich Speedup und Verzögerung optimal ist. Ferner ist GDP von den logischen Strukturen eines gegebenen Graphen (und damit auch von den Verhältnissen der Laufzeiten und den Kantengewichten) völlig unabhängig und erreicht das Maximum des theoretisch erreichbaren Speedups nach Gleichung (4.35) unabhängig von β. Demnach ist es nur sinnvoll, das Speedup-Verhalten ohne Pipelining weiter zu untersuchen.

Das Hardware-Modell verfügt über $L = 2$ Prozessoren und nur einen Bus, und die Ablaufplanung erfolgt nach Abschnitt 4.3.1. Die Bilder 5.27 und 5.28 zeigen typische Ergebnisse für das Beispiel von UTRA FDD Uplink und Downlink, die mit spektraler Partitionierung erzielt wurden: Der erreichte Anteil am theoretisch maximal möglichen Speedup ist in Abhängigkeit von der Nutzdatenrate des DTCH angegeben. Zusätzlich sind wieder die Schätzungen der Höhenlinien für die 5%-, 50%- und 95%-Quantile eingezeichnet. Die relative Busgeschwindigkeit ist für al-

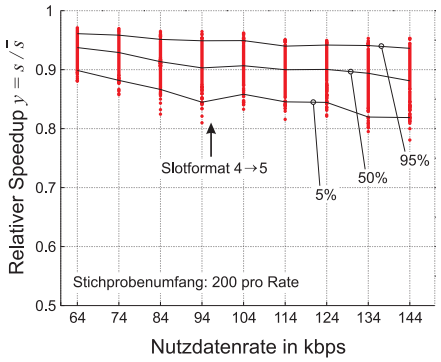


Bild 5.27 Uplink, $\beta = 100$

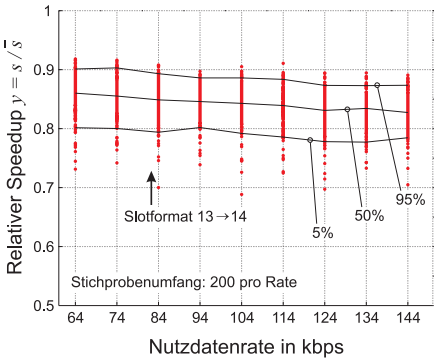


Bild 5.28 Downlink, $\beta = 100$

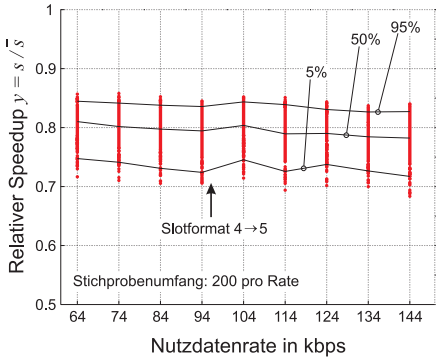


Bild 5.29 Uplink, $\beta = 4$

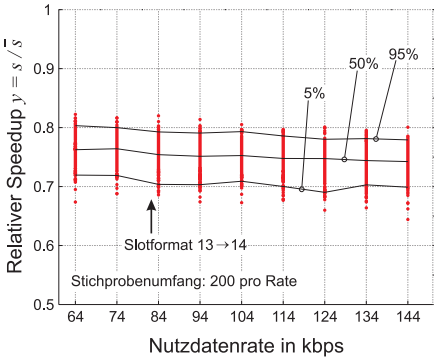


Bild 5.30 Downlink, $\beta = 4$

le Datenraten konstant und mit $\beta = 100$ recht hoch. Die Bilder 5.29 und 5.30 zeigen ein vergleichbares Verhalten, wobei die Ergebnisse bei einer relativen Busgeschwindigkeit von $\beta = 4$ erwartungsgemäß insgesamt niedriger liegen.

Neben der spektralen Partitionierung sind im Zusammenhang mit dem Einfluss von Datenraten auch der KL- und der Hu-Algorithmus untersucht worden. Die Ergebnisse dieser Versuchsreihen, die hier nicht im Einzelnen aufgeführt werden, weisen auf die gleichen Nachteile hin wie der Abschnitt 5.1.2: Einzelene Ausreißer sind ebenso möglich wie eine große Speedup-Streuung, die nah an $s = 0.5\,\bar{s}$ heranreichen kann. Bei welchen Datenraten der KL- und der Hu-Algorithmus so reagieren, ist nicht vorherzusagen.

Das beste Verhalten mit dem durchgängig höchsten Speedup wird von der spektralen Partitionierung erzielt. Die Bilder 5.27 bis 5.30 zeigen, dass die Höhenlinien mit zunehmender Datenrate leicht abfallen, und zwar sowohl im Uplink als auch im Downlink. Dieses Verhalten beruht **nicht** darauf, dass bei wachsender Nutzdatenrate mehr und mehr Eingangsdaten in der gleichen Zeit ΔT verarbeitet werden müssen und es deshalb ”schwieriger” ist, die gleichen Echtzeitbedingungen einzuhalten. Speedup ist ein **relatives** Qualitätsmaß für den Betrieb eines Mehrprozessor-Systems, das ja gerade wegen der Unabhängigkeit von konkreten Echtzeitanforderungen ausgewählt wurde. Vielmehr nimmt unter dem erweiterten Ressourcen-Laufzeit-Modell bei wachsender Nutzdatenrate und sonst gleichen Bedingungen die mittlere Laufzeit aller Knoten zu, während die Anzahl der Knoten M konstant bleibt. Insgesamt enthalten die Graphen des Downlinks im Vergleich zu denen des Uplinks weniger Knoten. In beiden Fällen bedeutet das eine Abnahme der Granularität G nach Gleichung (3.6). Die Beobachtung, dass geringere Granularität auch einen geringeren Speedup nach sich zieht, bestätigt erneut das Ergebnis aus Kapitel 3. Der Wechsel des Slotformats erzeugt bei spektraler Partitionierung kaum eine Änderung des Speedup-Verhaltens.

Damit ergeben sich aus den Simulationen mit einer Familie von UMTS-Graphen keine Hinweise auf bestimmte Graphen-Konstellationen, die bevorzugt für hohen Speedup, gute Vorhersagbarkeit des Systemverhaltens oder einen besonders leistungseffizienten Betrieb des Mod-SDRs sorgen. Abgesehen von dem Einfluss der Granularität kommen alle untersuchten Datenraten für die Signalverarbeitung gleichermaßen in Frage.

5.1.8 Verschiedenartig strukturierte Graphen

Schließlich sind noch Graphen zu untersuchen, bei denen nicht nur die Laufzeiten und die Gewichte der Kanten variiert werden, sondern auch die Kantenmenge selbst, d.h. die Gesamtheit der logischen Strukturen. Das Bild 5.31 zeigt einen einfachen Testgraphen, der über zwei zusammenhängende Abschnitte verfügt: Eine strikt serielle Kette und zwei parallele Ketten von Knoten. Indem bei konstanter Gesamtzahl M der Anteil der Knoten im seriellen Abschnitt variiert wird (während die beiden Parallelketten so ausgewogen wie möglich mit Knoten ausgestattet bleiben), ist es einfach möglich, die Struktur des Testgraphen auf eingängige Weise zu kontrollieren.

Eine wesentliche Eigenschaft, die im Rahmen der Untersuchungen quantifiziert

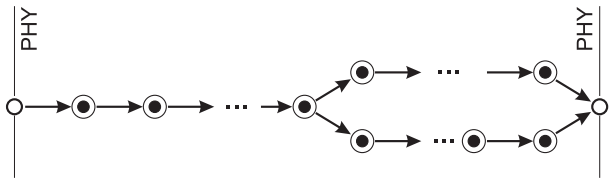


Bild 5.31 Testgraph mit seriellen und parallelen Teilgraphen, M konstant

werden muss, ist der Grad der inhärenten Parallelität eines Graphen. In [97] wird dafür folgender Ausdruck vorgeschlagen:

$$d = 1 - \frac{L_c - 1}{M - 1} \tag{5.13}$$

wobei L_c die Länge des kritischen Pfades und M die Gesamtzahl von Knoten ist. Mit diesem Ausdruck wird ein gegebener Graph als vollständig parallel bewertet, wenn $d = 1$, und als strikt sequentiell, wenn $d = 0$ ist. Da jedoch die Gleichung (5.13) nur die Anzahl M beinhaltet, und nicht die Laufzeiten aller Knoten, wird in dieser Arbeit ein entsprechend modifizierter Ausdruck vorgeschlagen:

$$\tilde{d} = 1 - \frac{T(L_c)/\min_m\{p_m\} - 1}{\sum_m p_m/\min_m\{p_m\} - 1} \tag{5.14}$$

wobei $T(L_c)$ die Gesamtlaufzeit des kritischen Pfades ist und p_m die gewohnten Knotenlaufzeiten. Beide Ausdrücke quantifizieren die Parallelität in Bezug auf eine unbegrenzte Anzahl von Prozessoren, nicht relativ zum vorliegenden Fall $L = 2$. Deshalb geht der Grad der Parallelität auch nicht gegen 1, sondern $\tilde{d} \to 0{,}5$, wenn die Zahl der Knoten im parallelen Abschnitt des Testgraphen ansteigt.

Als Hardware-Modell dient das System aus Bild 2.4 mit $L = 2$ Prozessoren und $B = 1$ Bus. Wie im vorangegangenen Abschnitt ist es nicht sinnvoll, eine andere Methode als GDP für leitungsvermittelte Dienste anzuwenden, da GDP bereits unabhängig von β als optimal bezüglich Speedup und Verzögerung erkannt wurde. Der einzige prominente Nachteil von GDP besteht in der Graphenverdoppelung. HFP biete hier eine Alternative, da es nur auf einem Exemplar des Graphen operiert. Als Partitionierungsverfahren für HFP kommt zunächst die modifizierte Variante der spektralen Partitionierung [75] zum Einsatz, die kurz in Abschnitt 4.2.3 beschrieben wurde. Im Vergleich dazu werden auch Ergebnisse für die implizite

Partitionierung nach Hu gezeigt, weil der Hu-Algorithmus ganz im Gegensatz zu HFP die reine Parallelisierung als Partitionierungsziel anstrebt. Folglich lässt sich in diesem Vergleich gut zeigen, wie wichtig die bewusste Unterscheidung der Begriffe Parallelisierung und Pipelining ist.

Die folgenden Bilder zeigen Simulationsergebnisse in Form des relativen Speedups in Abhängigkeit vom Parallelitätsgrad $\bar{d}$. Der Stichprobenumfang beträgt stets $2 \cdot 10^4$ Realisierungen, und die konstante Busgeschwindigkeit von $\beta = 10$ ist als ein sinnvoller Kompromiss zwischen der dynamischen Verlustleistung des Bussystems und dem erreichbaren Speedup gewählt worden. Da die Speedup-Messungen jetzt nicht mehr über festen Parameterwerten der Abszisse stehen, werden die Höhenlinien der 5%-, 50%- und 95%-Quantile aus einem gleitenden Fenster von jeweils 2000 benachbarten Messpunkten geschätzt und der Fenstermitte zugeordnet.

Das Bild 5.32 zeigt die Ergebnisse, die der Hu-Algorithmus bei Anwendung auf den Testgraphen hervorbringt. Es kommt kaum zu einer Speedup-Streuung, und die Höhenlinien liegen hier so dicht beieinander, das es nicht sinnvoll ist sie einzuzeichnen. Die starke Abhängigkeit der Parallelisierungsstrategie von der inhärenten Parallelität des Graphen ist deutlich zu erkennen. Es ist auch unmittelbar plausibel, dass ein reiner Parallelisierungsansatz um so weniger erfolgreich ist, je weniger parallele Teilstrukturen überhaupt anzutreffen sind.

Im Gegensatz dazu zeigt HFP im Bild 5.33 deutlich weniger Abhängigkeit vom

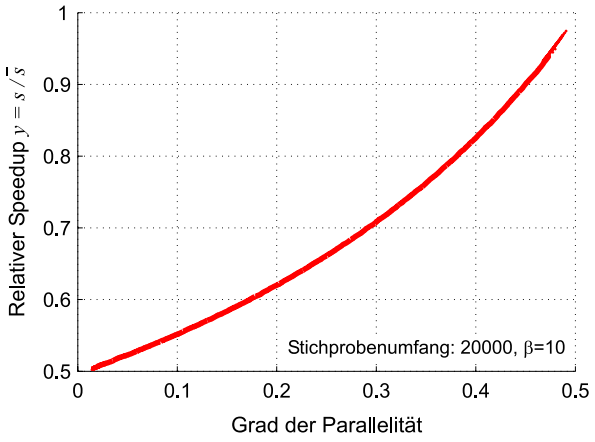


Bild 5.32 Testgraph, implizite Partitionierung nach Hu, kein Pipelining

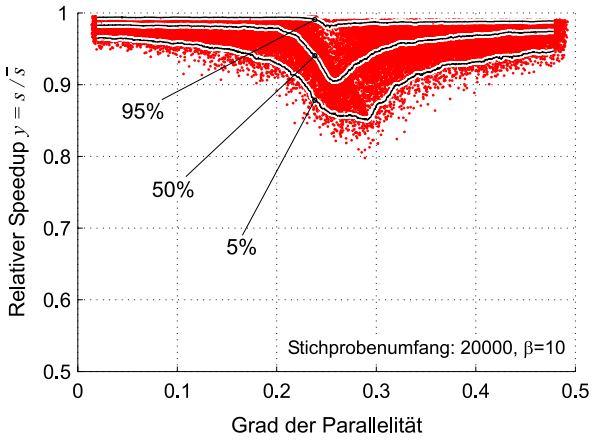


Bild 5.33 Testgraph, modifizierte spektrale Partitionierung, HFP

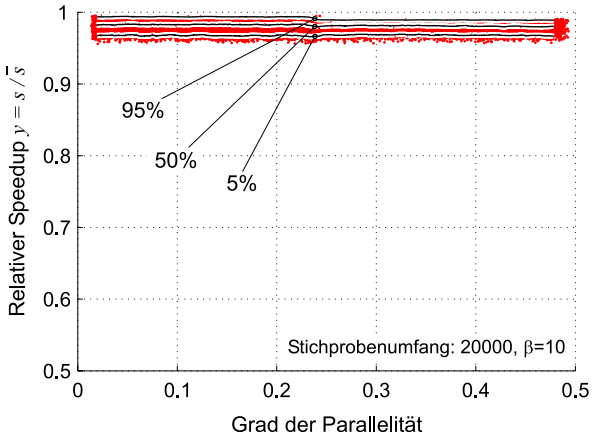


Bild 5.34 Testgraph, mod. spektrale Partitionierung mit Nachverarbeitung, HFP

Grad der Parallelität. Trotzdem bricht der Speedup für $0{,}2 \leq \tilde{d} \leq 0{,}3$ ein: Für die Hälfte aller Realisierungen geht der Speedup auf weniger als 90% des maximal erreichbaren Speedups herunter. Im Hinblick auf die spektrale Partitionierung ist bereits festgestellt worden, dass dort mit Blick auf die Ausgewogenheit der Lastverteilung noch Verbesserungspotential vorhanden ist (siehe Abschnitt 5.1.4).

Daher ist zunächst die Nachverarbeitung nach Abschnitt 4.3.3 eingeführt worden: Nach erfolgter spektraler Partitionierung werden so lange Knoten aus der größeren Partition in die kleinere überführt, bis die Partitionslaufzeiten ausgewogen sind. Den Erfolg der Nachverarbeitung bei Anwendung auf den Testgraphen zeigt das Bild 5.34. Unabhängig von $\tilde{d}$ wird in allen Realisierungsfällen fast maximaler Speedup erreicht. Demnach scheint die modifizierte spektrale Partitionierung [75] in Kombination mit HFP ein vielversprechender Ansatz zu sein, um unabhängig vom Parallelitätsgrad der logischen Strukturen hohen Speedup zu erreichen.

Trotz seiner variablen Struktur ist der Testgraph aus Bild 5.31 sicherlich nur ein Spezialfall unter den vielen verschiedenen in der Kommunikationstechnik auftretenden Graphen. Daher kommt im Folgenden das Software-Modell mit Zufallsgraphen aus Abschnitt 2.2.6 zum Einsatz. Im Gegensatz zum Testgraphen erzeugt das Zufallsmodell auch Graphen mit $\tilde{d} > 0{,}5$.

Wiederum wird zunächst der grundlegende Unterschied zwischen Parallelisierung und Pipelining offengelegt. Im Bild 5.35 sind die vom Hu-Algorithmus erzielten Ergebnisse in Abhängigkeit vom Parallelitätsgrad $\tilde{d}$ zu sehen. Unterhalb von $\tilde{d} = 0{,}5$ zeigt das Bild das typische Verhalten eines reinen Parallelisierungsansatzes, das bereits vom Testgraphen her bekannt ist. Oberhalb von $\tilde{d} = 0{,}5$ erreicht die Mehrzahl der Realisierungen noch nicht einmal $s = 0{,}9\,\overline{s}$. Zwar gibt es mit wachsendem $\tilde{d}$ immer weniger Realisierungen mit niedrigem Speedup, aber im Grunde

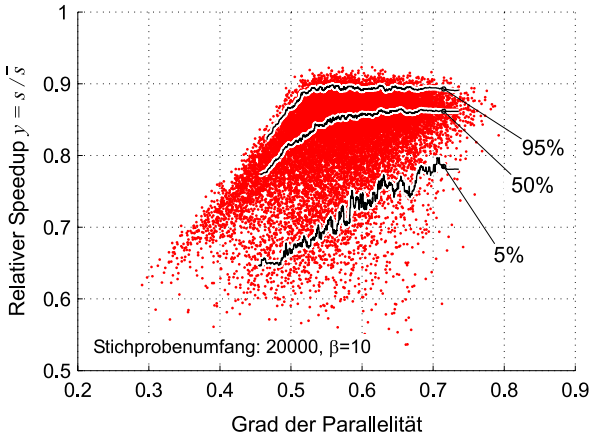


Bild 5.35 Zufallsgraphen, implizite Partitionierung nach Hu, kein Pipelining

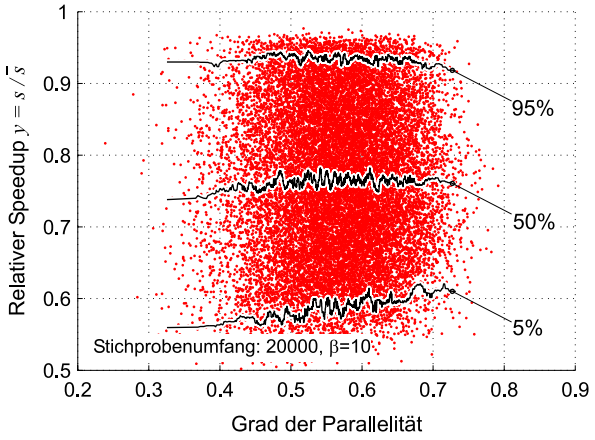


Bild 5.36 Zufallsgraphen, modifizierte spektrale Partitionierung, HFP

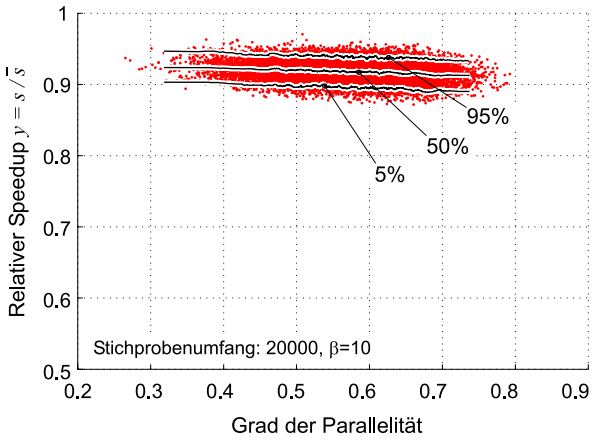


Bild 5.37 Zufallsgraphen, mod. spektrale Partitionierung mit Nachverarbeitung, HFP

ist mit solchen Ergebnissen der Einsatz dieses Verfahrens in einem Mod-SDR nicht sinnvoll.

Das Bild 5.36 zeigt eine extrem breite Streuung für die modifizierte spektrale Partitionierung, nachdem der erste Nachverarbeitungsschritt einen senkrechten Schnitt

sichergestellt hat. Obwohl hier die Höhenlinie des 95%-Quantils durchweg höher liegt als beim Hu-Algorithmus, ist das Gesamtverhalten völlig unakzeptabel für den praktischen Einsatz. Das Bild 5.37 hingegen zeigt praktisch sehr brauchbare Ergebnisse. Fast alle Realisierungen liegen hier zwischen 90% und 95% des theoretisch maximal erreichbaren Speedups, unabhängig vom Grad der Parallelität des Graphen. Diese Ergebnisse sind nach dem zweiten Nachverarbeitungsschritt zu beobachten.

Eine erneute Untersuchung von HFP hat gezeigt [83], dass die modifizierte spektrale Partitionierung lediglich eine Startkonfiguration erzeugt, der in Bild 5.37 beobachtete Speedup aber im Wesentlichen auf die Nachverarbeitung zurückzuführen ist. Als aufwandsgünstige Alternative bietet sich sofort die triviale Partitionierung in Kombination mit dem KL-Algorithmus an, der zusätzlich den Vorteil aufweist, einen Kompromiss zwischen Verbindungskosten und Lastverteilung zu schließen. Das Bild 5.38 zeigt die Speedup-Ergebnisse in Abhängigkeit vom Parallelitätsgrad der Zufallsgraphen. In der Tat sind mit dem Verfahren von Kernighan und Lin noch bessere Ergebnisse für HFP zu erzielen, und das auch noch schneller als mit allen anderen untersuchten Varianten. Das Bild 5.39 zeigt das Laufzeitverhalten für die entsprechenden Partitionierungsansätze in Abhängigkeit von der relativen Busgeschwindigkeit. Die Messungen basieren auf jeweils 2000 Realisierungen pro Parameterwert β und sind in der Form $\hat{\mu} \pm \hat{\sigma}$ angegeben [83], wobei $\hat{\mu}$ und $\hat{\sigma}$ jeweils

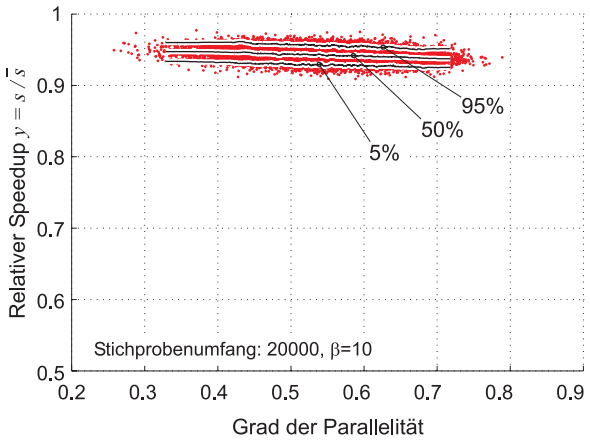


Bild 5.38 Zufallsgraphen, Partitionierung mit dem KL-Algorithmus, HFP

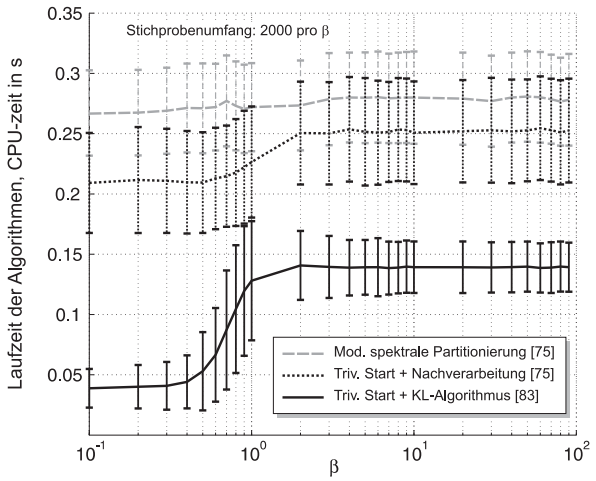


Bild 5.39 Laufzeitverhalten verschiedener Partitionierungsansätze für HFP

Schätzungen für die mittlere Laufzeit bzw. die Standardabweichung der Ansätze ist. Die beobachtete Standardabweichung beruht **nicht** auf Fehlern bei der Laufzeitmessung, sondern auf dem Antwortverhalten der Algorithmen auf das Zufallsgraphenmodell. Dieser Sachverhalt wird um so deutlicher, wenn man die erhöhten $\hat{\sigma}$-Werte des KL-Algorithmus im Bereich $\beta \approx 1$ betrachtet, wo die Laufzeiten von Bustransfer-Knoten und regulären Knoten in die gleiche Größenordnung fallen.

Für die Untersuchungen zur Abhängigkeit des Speedup-Verhaltens vom Parallelitätsgrad $\tilde{d}$ wurde in diesem Abschnitt bisher eine konstante Busgeschwindigkeit von $\beta = 10$ angenommen. Allerdings ist es für den praktischen Einsatz von großer Wichtigkeit, auch die Abhängigkeit der zu erwartenden Speedups von der relativen Busgeschwindigkeit zu kennen. HFP verfügt im Vergleich zu GDP über den Vorteil des geringeren Speicherbedarfs, jedoch prinzipbedingt zum Preis eines niedrigeren Speedups. Es ist zu erwarten, dass dieser Verlust gering gehalten werden kann, wenn β groß ist. Andererseits sind hohe Busgeschwindigkeiten bezüglich der dynamischen Verlustleistung von Nachteil. HFP wird also nur als Kompromisslösung zwischen hohem Speedup und akzeptabler Verlustleistung existieren können, und zwar auch nur dann, wenn der Gesamtspeicher im mobilen Endgerät für den Einsatz von GDP zu klein ist.

Das Bild 5.40 zeigt wie gewohnt Speedupmessungen und Höhenlinien für die Kom-

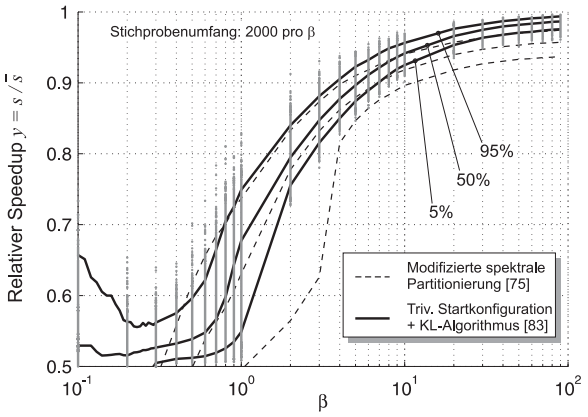


Bild 5.40 Vergleich von mod. spektraler Partitionierung und KL-Algorithmus für HFP

bination von KL-Algorithmus und HFP in Abhängigkeit von der relativen Busgeschwindigkeit β. Der Stichprobenumfang beträgt 2000 Realisierungen von Zufallsgraphen pro Parameterwert β. Zum direkten Vergleich mit der modifizierten spektralen Partitionierung sind die entsprechenden Höhenlinien gestrichelt eingezeichnet. Es ist festzustellen, dass HFP in Kombination mit dem KL-Algorithmus bei großen Busgeschwindigkeiten höhere Speedup-Werte erreicht und weniger streut. Bei Verringerung der Busgeschwindigkeit degradiert der Speedup nicht so abrupt wie bei der Variante mit der modifizierten spektralen Partitionierung. Um für die Mehrheit der zufälligen Realisierungen etwa 95% des maximalen Speedups zu erreichen, bedarf es mit $\beta = 10$ eines relativ schnellen und verlustbehafteten Bussystems. Diese Interpretation bestätigt die bereits getroffene Vorhersage: HFP kann als Alternative zu GDP nur bestehen, wenn der verfügbare Speicher den Einsatz von GDP absolut unmöglich macht.

Insgesamt zeigen sich bei der Signalverarbeitung deutliche Vorteile der Pipelining-Methoden GDP und HFP. Die bisher beobachteten Ergebnisse gelten stets für den eingeschwungenen Zustand des Ablaufplanes, der für leitungsvermittelte Dienste maßgeblich ist. Wenn aber die Signalverarbeitung für paketorientierte Dienste bewertet werden soll, so ist neben dem eingeschwungenen Zustand auch das Füllen und Leeren der Pipeline in die Speedup-Berechnung einzubeziehen. Im Folgenden werden daher die beiden erfolgversprechendsten Methoden GDP und HFP auf ihre Leistungsfähigkeit für paketorientierte Dienste hin untersucht.

5.2 Paketorientierte Dienste

Im Rahmen eines möglichst einfachen Betriebssystems seien zunächst die gleichen Annahmen getroffen wie in Abschnitt 4.3.2: Für eine erste Analyse bestehen die Paketsendungen nur aus Paketen der gleichen Länge N_F, und sobald ein Paket zur Verarbeitung anliegt, seien beide Prozessoren komplett für die Signalverarbeitung auf der physikalischen Schicht des Mod-SDR reserviert. Damit ist es beispielsweise nicht möglich, dass ein Prozessor noch Berechnungen zu Aufgaben von anderen Schichten ausführt, während der zweite Prozessor bereits mit der hier betrachteten Paket-Signalverarbeitung beginnt.

Mit den Bestimmungsgleichungen für den Speedup von paketorientiertem GDP und HFP aus den beiden Abschnitten 4.3.2 und 4.3.3 können die Dichten des Speedup (und damit die Höhenlinien) im Prinzip berechnet werden, wenn das erweiterte Ressourcen-Laufzeit-Modell zugrunde gelegt wird und alle Grapheneigenschaften bekannt sind. Da es sich bei den Graphen selbst um zufällige Realisierungen handelt und zusätzlich noch viele Fallunterscheidungen in den Bestimmungsgleichungen auftreten, ist es jedoch viel einfacher, die Höhenlinien aus der Simulation zu schätzen.

5.2.1 Ergebnisse mit Half-Frame Pipelining

Das Bild 5.41 zeigt die Speedup-Ergebnisse für HFP in Abhängigkeit von der relativen Busgeschwindigkeit. Aus Gründen der Übersichtlichkeit sind hier keine Messpunkte mehr angegeben, sondern nur noch die Höhenlinientripel für einige Funkrahmen pro Paket, $2 \leq N_F \leq 7$. Der Stichprobenumfang beläuft sich auf 2000 Zufallsgraphen pro Parameterpaar (β, N_F). Zusätzlich sind die kleinsten oberen Schranken [83] für den relativen Speedup gestrichelt angegeben:

$$y_{HFP} < \frac{N_F}{N_F + 1} \tag{5.15}$$

Die Tatsache, dass der Speedup bei sinkender Busgeschwindigkeit relativ schnell einbricht, hängt mit den stets erforderlichen Bustransfer-Knoten entlang des senkrechten Schnittes zusammen. Das Bild 5.42 zeigt dazu nochmals den Ablaufplan von HFP im Überblick und die Ausführungsreihenfolge der verschiedenen Knotentypen in der Vergrößerung. Das Füllen und Leeren der Pipeline umfasst einen Zeitraum, der in etwa der Bearbeitungszeit für einen kompletten Funkrahmen entspricht. Dieser Zeitraum enthält nicht nur Bustransfer-Knoten (deren Laufzeit mit

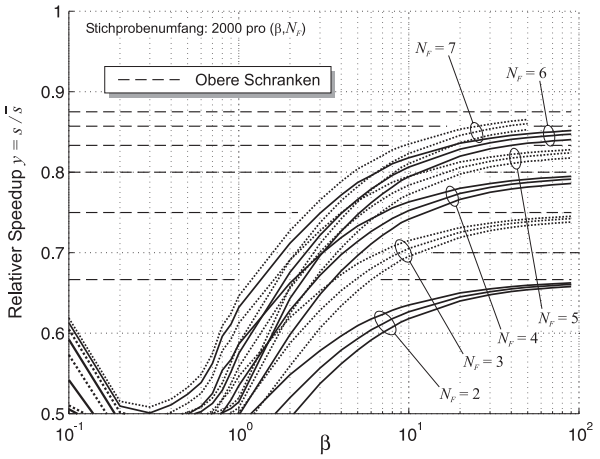


Bild 5.41 Paket-Signalverarbeitung mit Half-Frame Pipelining

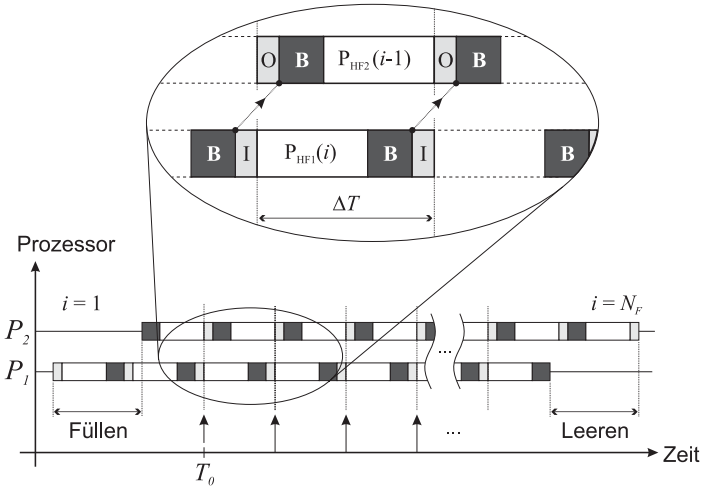


Bild 5.42 Ablaufplan für die Paket-Signalverarbeitung mit Half-Frame Pipelining

$\beta \to \infty$ sehr klein wird), sondern auch reguläre Knoten. Folglich ist es bei HFP prinzipiell unmöglich, den Einflusses des Füllens und Leerens der Pipeline durch Erhöhung von β zu vermindern.

Eine vorteilhafte Eigenschaft, die bisher noch nicht diskutiert wurde, besteht in der Regelmäßigkeit, mit der die Ausgangsdaten zu äquidistanten Zeiten an die I/O-Schnittstelle übergeben werden. HFP verfügt über diese Eigenschaft und unterstützt damit automatisch das kontinuierliches Senden und Empfangen von Paketen ohne weitere Zwischenspeicherung. Der Beginn des Schreibens von Ausgangsdaten wird hier für jeden Funkrahmen als Referenzzeitpunkt verstanden. Alle diese ausgezeichneten Zeitpunkte sind in Bild 5.42 als Impulszug auf der Zeitachse dargestellt. Der Beginn der Datenausgabe für den ersten Rahmen eines Pakets wird mit T_0 gekenzeichnet.

Wie bei den leitungsvermittelten Diensten zeigt HFP hier Vor- und Nachteile. Im Vergleich dazu wird nun die Leistungsfähigkeit von GDP für paketorientierte Signalverarbeitung bewertet.

5.2.2 Ergebnisse mit Graph Duplication Pipelining

Das Bild 5.43 zeigt einen Überblick über den Ablaufplan, der die schnellste mögliche Berechnung von N_F Funkrahmen pro Paket gestattet. Die für das Füllen und Leeren der Pipeline benötigte Zeit hängt hier unmittelbar von der relativen Busgeschwindigkeit ab, und so ist mit wachsendem β auch ein hoher Speedup zu erwarten.

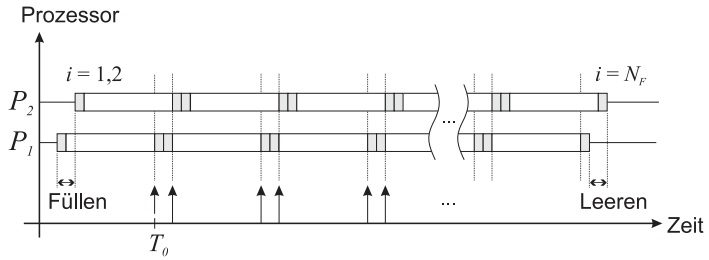


Bild 5.43 GDP, nicht-äquidistante Referenzzeitpunkte

Das Bild 5.44 zeigt die üblichen Höhenlinientripel für einige Funkrahmen pro Paket. Zusätzlich sind die größten unteren Schranken [9] für den relativen Speedup gestrichelt eingezeichnet:

$$y_{GDP} \geq \frac{N_F}{N_F + 1} \tag{5.16}$$

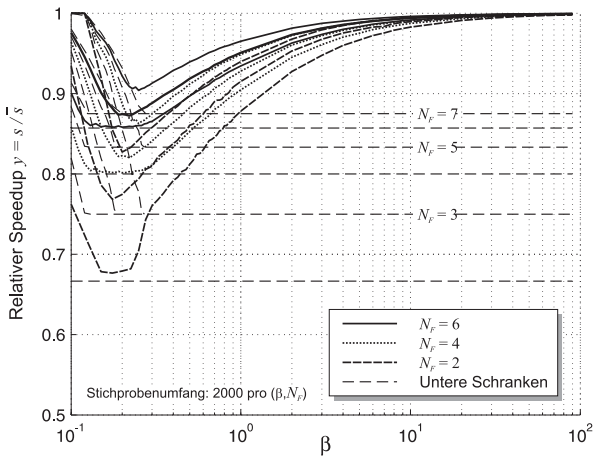


Bild 5.44 Paket-Signalverarbeitung mit Graph Duplication Pipelining

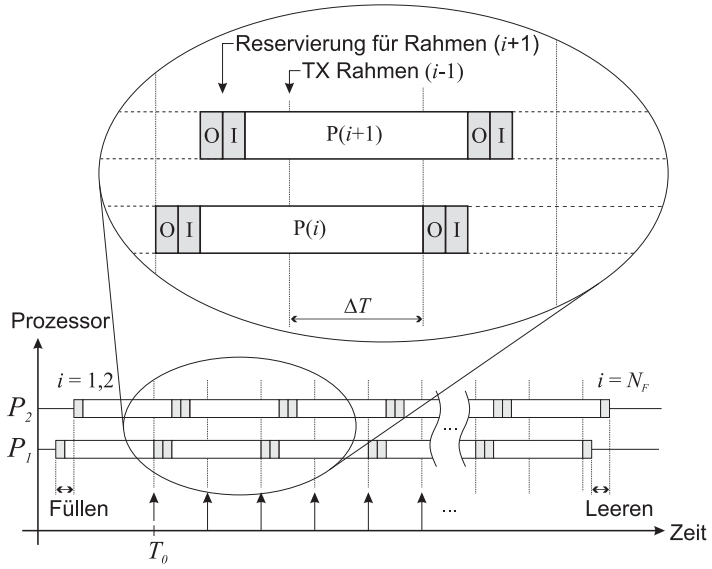


Bild 5.45 GDP, äquidistante Referenzzeitpunkte, erhöhter Speicherbedarf

Aus der Gleichung (4.38) geht bereits hervor, dass der Speedup für ungerade N_F und hinreichend große Busgeschwindigkeiten ($T_P \geq T_{IO}$) den Wert der Schranke erreicht. Für ungerade N_F ergibt sich ein Speedup-Verhalten, das mit wachsendem β schnell gegen die theoretische Obergrenzes strebt. Selbst bei moderaten Busgeschwindigkeiten und relativ kleinen N_F lassen sich bereits hohe Speedups erzielen.

Ein Nachteil, der im Zusammenhang mit GDP noch nicht genannt wurde, wird mit Blick auf die Referenzzeitpunkte in Bild 5.43 deutlich: Die Ausgabe von Ausgangsdaten erfolgt bei GDP ursprünglich nicht äquidistant, wenn die schnellstmögliche Ablaufplanung zur Anwendung kommt. Eine Alternative, die den hohen Speedup aus Bild 5.44 aufrechterhält, besteht darin, äquidistante Referenzzeitpunkte zu erzwingen. Damit kommt es aber zwangsläufig zu einer weiteren Erhöhung des Speicherbedarfs. Das Bild 5.45 zeigt die Übersicht über den zugehörigen Ablaufplan und in der Vergrößerung, wie es zu dem erhöhten Speicherbedarf kommt: Noch bevor die Ausgangsdaten des Rahmens $(i-1)$ über den Funkkanal übertragen werden, muss bereits neuer Ausgangsdatenspeicher für den Rahmen $(i+1)$ reserviert werden, und das zusätzlich zu dem Speicher für den Rahmen i, der auf dem zweiten Prozessor berechnet wird. Damit ergibt sich mindestens eine Speicherbedarfszunahme um 50% gegenüber dem Fall von GDP mit nicht-äquidistanten Referenzzeitpunkten.

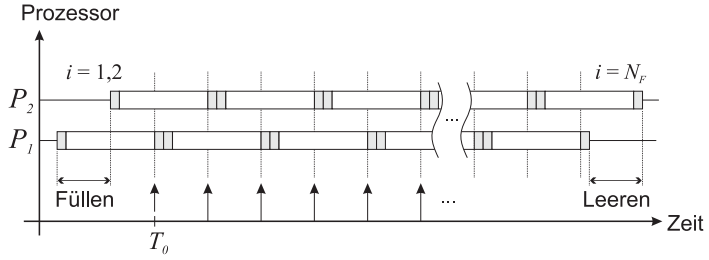


Bild 5.46 GDP, äquidistante Referenzzeitpunkte, verminderter Speedup

Den Nachteil des erhöhten Speicherbedarfs kann man auch durch eine zeitliche Verschiebung im Ablaufplan vermeiden, allerdings nur um den Preis eines verminderten Speedups. Das Bild 5.46 zeigt den resultierenden Ablaufplan mit äquidistanten Referenzzeitpunkten. Aus dieser Darstellung kann der konstante relative Speedup $y = N_F/(N_F+1)$ für hinreichend große Busgeschwindigkeiten hergeleitet werden. Also wird mit dieser Variante des GDP-Ablaufplans kein größerer Speedup erzeugt als von den gestrichelten Linien in Bild 5.44 angegeben. Diese Linien stel-

len für GDP zwar untere Grenzen dar, für HFP aber **obere** Grenzen. Folglich sind alle in diesem Abschnitt vorgestellten GDP-Varianten dem HFP im Hinblick auf den Speedup prinzipiell überlegen.

5.2.3 Relevanz der Ergebnisse für etablierte WLAN-Standards

Bisher ist für paketorientierte Dienste stets eine konstante Anzahl N_F von Funkrahmen pro Paket zur Berechnung des Speedups vorausgesetzt worden. In tatsächlich einsetzbaren WLANs herrscht jedoch vorwiegend IP-Verkehr, d.h. die Paket-Sendungen bestehen aus einer Mischung der unterschiedlichsten Längen, von einigen Bytes für die Signalisierung von Protokollereignissen bis hin zur maximalen IP-Paketlänge von 1500 Bytes für den Transport großer Nutzdatenmengen. Eine naive Annahme könnte daher sein, dass Pakete auf der Funkstrecke deshalb bei etablierten WLAN-Standards zu einem Großteil ebenfalls aus vielen Funkrahmen ($N_F \gg 7$) bestehen, weil kurze Pakete naturgemäß einen relativ großen Overhead aus Kontrolldaten erzeugen und den Durchsatz des Kommunikationssystems so niedrig halten. Damit würden sich die Bilder in den beiden vorangegangenene Abschnitten, die lediglich Ergebnisse für $2 \leq N_F \leq 7$ zeigen, als irrelevant erweisen.

Der zu erwartende mittlere Speedup des Mod-SDR-Systems bei Paket-Signalverarbeitung kann nur ermittelt werden, wenn eine Paketlängen-Statistik bekannt ist. Um

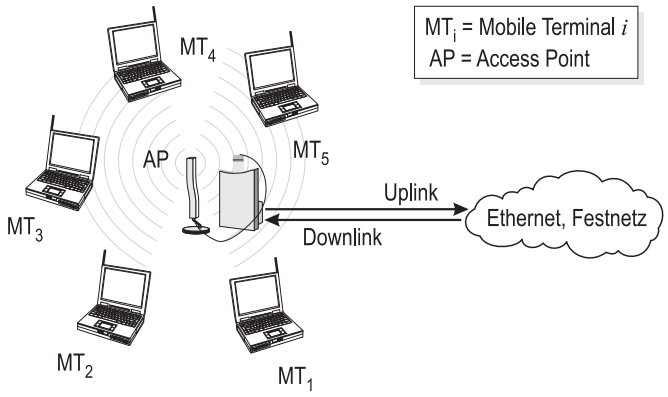


Bild 5.47 Szenario für die Simulation einer WLAN-Zelle

zu zeigen, dass kurze Pakete auf der Funkstrecke zwischen mobilem Endgerät und Basisstation in der Tat relevanten Einfluss haben, ist ein Szenario simuliert worden, das auf dem IEEE802.11a-Standard basiert. Der abstrakte Begriff des Funkrahmens entspricht in diesem konkreten Fall dem Begriff des OFDM-Symbols. Die Länge von Paketen wird für höhere Schichten meist in Bytes angegeben. Da über den Funkkanal aber nur Symbole übertragen werden, wird die Paketlänge $N_F \in \mathbb{N}$ für die physikalische Schicht sinnvollerweise in Vielfachen von OFDM-Symbolen (**ohne** Präambel von 4 OFDM-Symbolen und **ohne** SIGNAL-Feld der Länge eines OFDM-Symbols [1]) angegeben.

Das Bild 5.47 zeigt das Szenario [82, 83] mit fünf mobilen Endgeräten (*engl.* MT, mobile terminal), die sich im Abdeckungsbereich der Basisstation (*engl.* AP, access point) befinden. Die nach Dienstklassen getrennten Verkehrsmodelle stammen aus [14], das integrale Modell für IP-Verkehr aus [110]. Für den Datenverkehr im Uplink und Downlink wird ein Verhältnis von $1:4$ angenommen. Die physikalische Schicht arbeitet im Folgenden mit der Modulationsart 64-QAM bei einer Coderate von $R_c = 3/4$, wenn die MAC-Schicht *Datenpakete* zur Verarbeitung liefert.

Für *Kontrollpakete* der MAC-Schicht (RTS, CTS, ACK, usw.) erlaubt IEEE802.11a die Verwendung verschiedener Modulationsarten: BPSK, QPSK oder 16-QAM, jeweils in Kombination mit der Coderate $R_c = 1/2$. Das Bild 5.48 zeigt Histogramme der Paketlängen für IP-Verkehr und alle drei Betriebsarten. Als Kanalmodell kommt das Indoor-Modell aus der Spezifikation einer UMTS-Fallstudie des ETSI

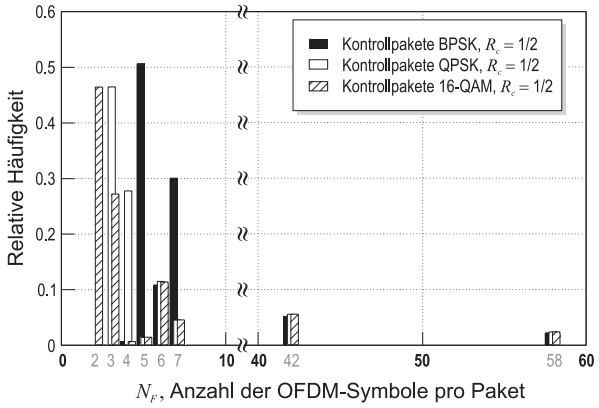


Bild 5.48 Histogramme der Paketlängen für IP-Verkehr, SNR = 30 dB

[22] zum Einsatz. Die mittlere Geschwindigkeit der Terminals sei $v = 5$ km/h, und das mittlere SNR ist mit 30 dB bewusst hoch gewählt: Sollte die Anzahl kurzer Pakete für den Betrieb eines Mod-SDRs relevant sein, so ist ein hohes SNR sicherlich noch ein günstiger Testfall. Jedes niedrigere SNR würde den Anteil kurzer Pakete nur noch vergrößern, weil es dann vermehrt zu Paketfehlern käme und folglich zu einem wachsenden Anteil kurzer Kontrollpakete [82].

Im Bild 5.48 ist festzustellen, dass mehr als 90% aller Pakete Längen von weniger als 10 OFDM-Symbolen aufweisen. Offensichtlich sind also in dem betrachteten WLAN-Standard mit IP-Verkehr kurze Pakete tatsächlich dominant, selbst wenn BPSK mit $R_c = 1/2$ verwendet wird. Folglich ist die Dominanz kurzer Pakete

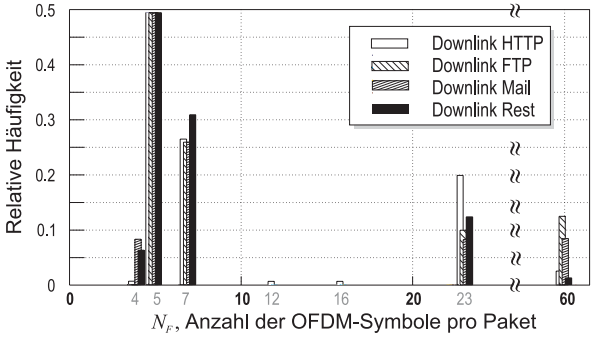


Bild 5.49 Histogramme der Paketlängen, nach Dienstklasse (Downlink), SNR = 30 dB

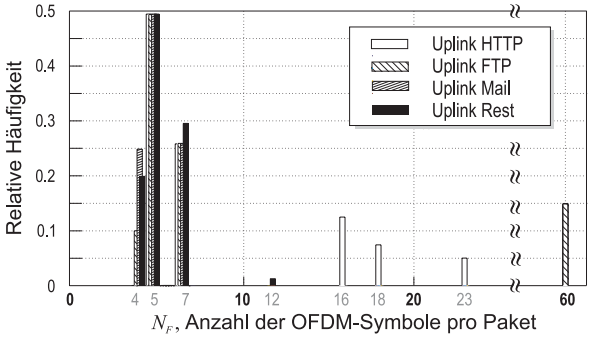


Bild 5.50 Histogramme der Paketlängen, nach Dienstklasse (Uplink), SNR = 30 dB

unabhängig davon, wie die Kontrollpakete moduliert werden. Daher wird im Folgenden üblicherweise BPSK für die Kontrollpakete verwendet, weil diese Modulationsart aus Sicht der Nachrichtentechnik am wenigsten fehleranfällig ist.

Die Bilder 5.49 und 5.50 zeigen die Histogramme der Paketlängen nach den verschiedenen Dienstklassen geordnet, jeweils für Uplink und Downlink. Das Kanalmodell und das mittlere SNR bleiben unverändert. Wiederum kann bei allen Dienstklassen die Dominanz kurzer Pakete beobachtet werden. Lediglich Mail-Verkehr im Downlink und FTP-Verkehr im Up- und Downlink zeigen eine Tendenz zur bevorzugten Erzeugung längerer Pakete. Die Idee, mehrere von der Dienstklasse abhängige Betriebsarten für Mod-SDR einzuführen, erscheint mit diesen Ergebnissen trotzdem nicht sinnvoll. Vielmehr ist darauf zu achten, dass die Signalverarbeitung generell mit vielen kurzen Paketen auch hohen Speedup erzeugt.

Zuletzt wird die Abhängigkeit der Dominanz kurzer Pakete vom SNR untersucht, und zwar für zwei verschiedene Kanalmodelle: Das bereits bekannte Indoor-Modell [22] und ein Kanalmodell für die Fahrzeug-Fahrzeug-Kommunikation [56, 57], die beide so eingestellt werden, dass die Kohärenzzeit jeweils $T_C \approx f_{D,max}^{-1} = 6$ ms beträgt. Das Bild 5.51 zeigt die Simulationsergebnisse für das Indoor-Kanalmodell

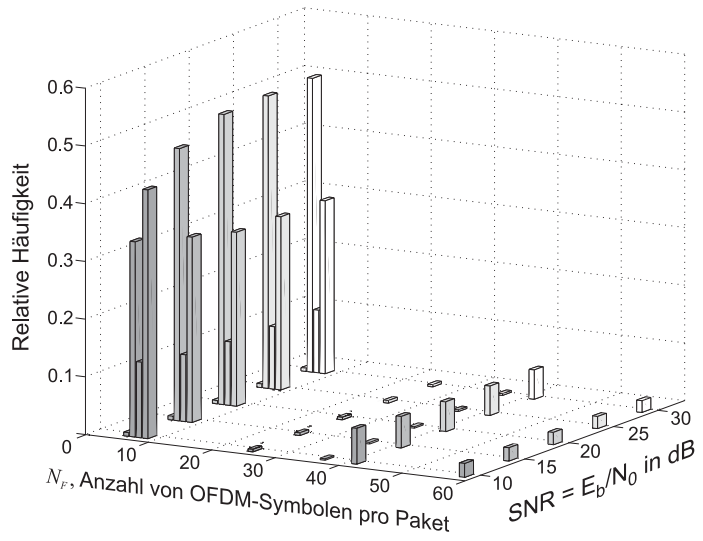


Bild 5.51 Histogramme der Paketlängen, Indoor-Kanalmodell

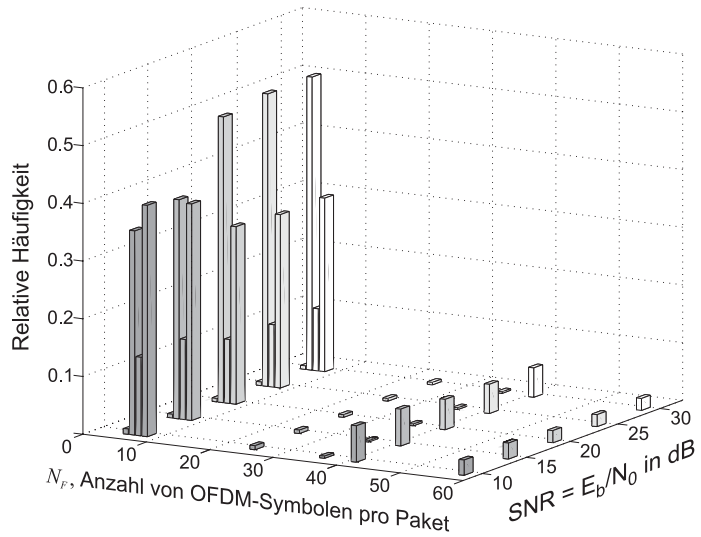


Bild 5.52 Histogramme der Paketlängen, Fahrzeug-Fahrzeug-Kanalmodell

und ein mittleres SNR zwischen 10 dB und 30 dB. Unabhängig vom SNR ist hier wiederum die Dominanz von Paketen mit Längen unter 10 OFDM-Symbolen zu beobachten. Mit wachsendem SNR steigt die relative Häufigkeit der Paketgröße $N_F = 5$, und die der Größe $N_F = 7$ sinkt, während Pakete mit $N_F = 6$ und lange Pakete mit nahezu unveränderter Häufigkeit auftreten. Die Ergebnisse für das Kanalmodell zur Fahrzeug-Fahrzeug-Kommunikation in Bild 5.52 zeigen ein ähnliches Verhalten.

Es ist bereits argumentiert worden, dass ein hohes SNR sicherlich einen günstigen Testfall darstellt, während niedrigere SNRs eher einen noch größeren Anteil kurzer Pakete zur Folge haben werden. Diese Vorhersage wird mit den Ergebnissen aus den Bildern 5.51 und 5.52 in der Tat bestätigt. Die Simulationen lassen aber auch eine wesentlich detailliertere Interpretation zu: $N_F = 5$ entspricht CTS- und ACK-Paketen der Länge 14 Byte, wohingegen RTS-Pakete der Länge 20 Byte auf der physikalischen Schicht in $N_F = 7$ OFDM-Symbole umgesetzt werden. Natürlich sind die relativen Häufigkeiten von Paketlängen Folge eines Gemisches von MAC-Paketen mit Kontroll- und Nutzdaten, deren individuelle Beiträge zu den Histogrammen in den Bildern 5.51 und 5.52 streng genommen nicht zu trennen sind. Es wird jedoch aus der Argumentation später klar werden, dass die Beiträge der

Datenpakete nahezu unabhängig vom mittleren SNR sind. Unter dieser Annahme kann dann von den Histogrammwerten eine (pro Paketlänge N_F) konstante Grundlast für die Nutzdaten abgezogen werden, so dass der Rest nur noch von den Kontrollpaketen der MAC-Schicht stammen kann.

Bei hohem SNR ist das Verhältnis der MAC-Paketlängen $(N_F = 5) : (N_F = 7)$ in etwa $2 : 1$. Diese Beobachtung entspricht der Anschauung, dass "für gute Kanäle" die Datenübertragung bei beliebigen Paketlängen erfolgreich ist und jedes Datenpaket mit einem vorausgehenden RTS/CTS Handshake und einem abschliessenden ACK verknüpft ist. Mit sinkendem SNR werden alle Pakete immer fehleranfälliger. Wenn insbesondere der RTS fehlschlägt, wird der Handshake physikalisch vom Kanal unterbrochen, und der Sender versucht die Übertragung des RTS-Paketes nach einer Wartezeit erneut. Auf diese Weise steigt bei sinkendem SNR die Häufigkeit von RTS relativ zu allen anderen Pakettypen, aber besonders im Vergleich zu CTS und ACK (siehe beide Bilder 5.51 und 5.52).

Schließlich muss noch geklärt werden, warum die Datenpakete nahezu unabhängig vom mittleren SNR sind: Ihre Übertragung erfolgt **nur dann, wenn** der RTS/CTS Handshake erfolgreich ist. Selbst wenn das mittlere SNR des Kanals niedrig ist, kann das momentane SNR hoch sein. Sollte der einleitende RTS/CTS Handshake erfolgreich sein, dann zufolge eines hohen momentanen SNR, das mit hoher Wahrscheinlichkeit auch über die gesamte verbleibende Übertragungszeit aufrechterhalten wird. Selbst die längsten IP-Pakete (1500 Byte) werden bei 64-QAM lediglich in $N_F = 56$ OFDM-Symbole (224μs) umgeformt. Unter Einbeziehung der Präambel und des SIGNAL-Feldes kommt man damit auf eine Übertragungsdauer von 244μs. Eine komplette Protokollsequenz aus RTS/CTS/data/ACK beläuft sich auf etwa 480μs [1]. Die gesamte Übertragungsdauer ist also immer noch kleiner als die Kohärenzzeit T_C des Kanals. Eine Datenübertragung findet daher nur dann statt, wenn über einen erfolgreichen RTS/CTS Handshake der Zustand des Mobilfunkkanals mit hoher Wahrscheinlichkeit als gut erkannt wurde.

5.3 Zusammenfassung

In diesem Kapitel ist die Leistungsfähigkeit von Mod-SDR für die Signalverarbeitung bei leitungs- und paketvermittelten Diensten bewertet worden. Die wesentlichen Aspekte eines Mehrprozessor-Hardware-Systems wie Speedup, Verzögerung, Speicherbedarf und dynamische Verlustleistung sind in verschiedenen Abschnitten

betrachtet worden. Die Reihenfolge der in den Untersuchungen eingesetzten Software-Modelle spiegelt die chronologische Verbesserung der Simulationsbedingungen wider: Ausgehend von einem Graphen mit festen logischen Strukturen und zufälligen Laufzeiten ist das Modell von SDR-Software um variable Strukturen und schließlich um zufällige Strukturen erweitert worden. Trotz der enthaltenen Zufallsprozesse für Knoten und Kanten werden nur Graphen erzeugt, die wesentliche Eigenschaften von modularer Kommunikationssoftware aufweisen. Mit diesem allgemeinsten Modell können allgemein gültige Aussagen über die Leistungsfähigkeit der Aufgaben Partitionierung und Ablaufplanung getroffen werden. Im Hinblick auf Hardware und Software geht Pipelining als die universell erfolgreich einsetzbare Richtlinie für den Entwurf und Betrieb von Mod-SDR-Systemen hervor.

A Laplace-Operatoren in der digitalen Bildverarbeitung

Im Abschnitt 4.2.3 wird aus der symmetrischen Gewichtsmatrix $\mathbf{W}$ aller Kantengewichte $w_{l,m}$ eines Graphen die Laplace-Matrix $\mathbf{L}$ gebildet:

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \tag{A.1}$$

$$\text{mit} \quad \mathbf{D} = \text{diag}(d_l), \qquad d_l = \sum_{m=1}^{M} w_{l,m} \tag{A.2}$$

Die Berechnung von $\vec{q}^{\,T}\mathbf{L}\vec{e}_m$ entspricht der Anwendung des diskreten Laplace-Operators auf den Knoten m. In der digitalen Bildverarbeitung gibt es verschiedene Laplace-Operatoren, die zur Hervorhebung der Konturen eines Bildes benutzt werden. Das Bild A.1 zeigt drei Matrizen, die als zweidimensionale Faltungskerne zu verstehen sind. Die Anwendung dieser Operatoren auf ein gegebenes Bild erfolgt durch Faltung des Bildes mit der jeweiligen Matrix.

0	-1	0
-1	4	-1
0	-1	0

(a) Operator Δ_1

-1	-1	-1
-1	8	-1
-1	-1	-1

(b) Operator Δ_2

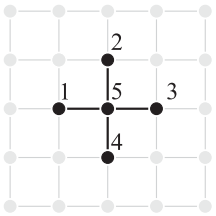
1	-2	1
-2	4	-2
1	-2	1

(c) Operator Δ_3

Bild A.1 Laplace-Operatoren in der digitalen Bildverarbeitung

Der Zusammenhang zwischen der Laplace-Matrix $\mathbf{L}$ und den zweidimensionalen Operatoren ergibt sich, wenn man die Pixel des Bildes als Knoten eines Graphen auffasst und die lokalen Nachbarschaftsbeziehungen der Pixel durch Kanten mit dem Einheitsgewicht anzeigt. Das Bild A.2 zeigt die entsprechende Skizze für den Operator Δ_1, daneben ist die zugehörige Laplace-Matrix $\mathbf{L}_1$ angegeben.

Die Zuordnung des Wertes $\vec{q}^{\,T}\mathbf{L}_1\vec{e}_5$ zum zentralen Pixel des Faltungskerns entspricht der Anwendung des Operators Δ_1 auf ein Bild, das in dem Vektor $\vec{q}$ die Farbinformation aller Pixel speichert, also beispielsweise die Graustufenwerte. Im



(a) Graph

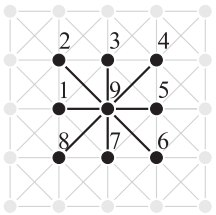
$$\mathbf{L}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 \\ -1 & -1 & -1 & -1 & 4 \end{bmatrix}$$

(b) Laplace-Matrix des Graphen

Bild A.2 Darstellung lokaler Nachbarschaftsbeziehungen von Pixeln

Falle der Partitionierung enthalten die Komponenten von $\vec{q}$ jeweils ein Maß für die Partitionszugehörigkeit des entsprechenden Knotens.

Entsprechend der Korrespondenz zwischen dem Operator Δ_1 und dem Graphen im Bild A.2 existiert auch eine Darstellung der lokalen Nachbarschaftsbeziehungen für den Operator Δ_2 (siehe Bild A.3). Der Operator Δ_3 ist lediglich eine gewichtete Summe der beiden ersten Matrizen: $\Delta_3 = 3\Delta_1 - \Delta_2$.



(a) Graph

$$\mathbf{L}_2 = \left[\begin{array}{c|c} \mathbf{I}_{8,8} & -\vec{1} \\ \hline -\vec{1}^T & 8 \end{array}\right]$$

(b) Laplace-Matrix des Graphen

Bild A.3 Darstellung lokaler Nachbarschaftsbeziehungen von Pixeln

Anschaulich liefert der Laplace-Operator ein richtungsunabhängiges Maß für die lokale Bildkrümmung im Farbraum. Alle Konturen eines Bildes verfügen über einen großen Betrag dieser Krümmung, und aus diesem Grund eignen sich die Laplace-Operatoren in der Bildverarbeitung als Konturdetektoren. Eine anschauliche Interpretation im Zusammenhang mit der Aufgabe der Partitionierung zu formulieren ist für den allgemeinen Fall $\vec{q} \in \mathbb{R}^M$ kaum möglich. Zumindest kann man sagen, dass der Operator für die Indikatorvektoren $\vec{a}, \vec{b} : a_m, b_m \in \{0; 1\}$ keine Beiträge innerhalb einer Partition liefert, sondern lediglich an den Partitionsgrenzen. Die dortige Krümmung im Raum der Partitionszugehörigkeiten ergibt gerade die externen Schnittkosten der Partitionierung.

Abkürzungen, Terminologie und Formelzeichen

Abkürzungen

ACK	ACKnowledgement
A/D	Analog-to-Digital
ADSL	Asynchronous Digital Subscriber Line
ASIC	Application Specific Integrated Circuit
ASSP	Application Specific Standard Product
BER	Bit Error Rate
BOS	Behörden und Organisationen mit Sicherheitsaufgaben
BPSK	Binary Phase Shift Keying
CCTrCH	Coded Composite Transport CHannel
CDMA	Code Division Multiple Access
CMOS	Complementary Metal Oxide Semiconductor
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CTS	Clear To Send
D/A	Digital-to-Analog
DAG	Directed Acyclic Graph
DCCH	Dedicated Control CHannel
DCH	Dedicated CHannel
DECT	Digital Enhanced Cordless Telecommunications
DPCCH	Dedicated Physical Control CHannel
DPDCH	Dedicated Physical Data CHannel
DS-CDMA	Direct Sequence CDMA
DSP	Digital Signal Processor
DTCH	Dedicated Traffic CHannel
EDGE	Enhanced Data rates for Global Evolution
EFR	Enhanced Full Rate
FFT	Fast Fourier Transform
FLOPS	FLoating point Operations Per Second
FPGA	Field Programmable Gate Array

FTP	File Transfer Protocol
GDP	Graph Duplication Pipelining
GMSK	Gaussian Minimum Shift Keying
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HF	HochFrequenz
HFP	Half-Frame Pipelining
HSCSD	High Speed Circuit Switched Data
HTTP	Hyper Text Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
ILP	Integer Linear Programming
IMT-2000	International Mobile Telecommunications-2000
I/O	Input/Output
IP	Internet Protocol, Intellectual Property
IPv6	IP version 6
IS	Interim Standard
ISDN	Integrated Services Digital Network
ISO	International Standardization Organization
ITU	International Telecommunication Union
JPO	Joint Program Office
JTRS	Joint Tactical Radio System
KL	Kernighan-Lin
LAN	Local Area Network
MAC	Medium Access Control
MC-CDMA	Multi Carrier CDMA
MMR	Multi Mode Radio
Mod-SDR	Modular Software Defined Radio
MUD	Multi User Detection
NP	Non-deterministic Polynomial-time
OFDM	Orthogonal Frequency Division Multiplexing
OSI	Open Systems Interconnect
OVSF	Orthogonal Variable Spreading Factor
PaC-SDR	Parameter Controlled Software Defined Radio
PDU	Protocol Data Unit
PHS	Personal Handyphone System
PHY	PHYsical layer
PSK	Phase Shift Keying

QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
RF	Radio Frequency
RTS	Request To Send
SAP	Service Access Point
SCA	Software Communications Architecture
SDR	Software Defined Radio
SDRF	Software Defined Radio Forum
SMTP	Simple Mail Transfer Protocol
SNR	Signal-to-Noise Ratio
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
TETRA	TErrestrial Trunked Radio Access
TETRAPOL	TETRA for POLice
TrCH	Transport CHannel
TTI	Transmission Time Interval
TX	Transmit
UMTS	Universal Mobile Telecommunications System
UTRA	UMTS Terrestrial Radio Access
UTRA-FDD	UTRA Frequency Division Duplex
UTRA-TDD	UTRA Time Division Duplex
VoIP	Voice over IP
WLAN	Wireless Local Area Network
WSSUS	Wide Sense Stationary Uncorrelated Scattering

Terminologie

Ablaufplanung (*engl.* scheduling)	Der Vorgang der Festlegung zeitlicher Abläufe in einem System. In dieser Arbeit: Die Bestimmung von Startzeitpunkten für alle Module eines Mod-SDRs. Der Ablaufplan (*engl.* schedule) ist statisch, d.h. während des Betriebs bleiben alle Startzeitpunkte unverändert.

Betriebssystem (*engl.* operating system)	Eine spezielle Software, die zusammen mit einem bestimmten Hardware-System vom Gerätehersteller ausgeliefert wird (*engl.* firmware). Das Betriebssystem ermöglicht dem Nutzer des Gerätes ebenso wie dem Programmierer von Software den bestimmungsgemäßen Betrieb des Hardware-Systems. In dieser Arbeit: Das Betriebssystem eines Mod-SDR nimmt Anfragen nach der Realisierung eines bestimmten Mobilfunkstandards entgegen und entscheidet über die Machbarkeit solcher Realisierungen. Es verwaltet alle Ressourcen und steuert die Ausführung von Software. Im Hinblick auf Mehrprozessor-Systeme gehören die Partitionierung und das Scheduling zu den Hauptaufgaben des Betriebssystems.
Block	Ein geordneter Verbund von Daten.
Funkrahmen (*engl.* radio frame)	Ein allgemein gültiger Begriff für die Darstellung eines Rahmens in der Form, die unmittelbar zur Datenübertragung über einen Funkkanal genutzt werden kann.
Funktion	Eine klassische nachrichtentechnische Funktion (z.B. Kanalcodierung, Entzerrung), die in einem Blockschaltbild zur Darstellung der Funktionsweise eines Funkgerätes dienen kann.
Graph	Das Tupel $(\mathbb{M}, \mathbb{K})$ einer Knotenmenge $\mathbb{M}$ und einer Kantenmenge $\mathbb{K}$. In dieser Arbeit: Ein gerichteter azyklischer Graph, so dass $\mathbb{K} \subset (\mathbb{M} \times \mathbb{M})$.
Luftschnittstelle (*engl.* air interface)	Die Beschreibung der Aufbereitung von Binärdaten mit dem Ziel der Darstellung in Form analoger Wellenzüge, die elektromagnetisch über einen physikalischen Funkkanal übertragen werden können. Ein wesentlicher Aspekt dieser Beschreibung ist die Festlegung von Echtzeitbedingungen für jeden Funkrahmen.
Modul	Ein Softwaremodul. Der kleinste zusammenhängende Verbund von ausführbaren Maschinenbefehlen. Die Ausführung eines Moduls kann vom Betriebssystem nicht unterbrochen werden.

Partitionierung (*engl.* partitioning)	Die Zerlegung der Knotenmenge $\mathbb{M}$ eines Graphen in disjunkte Teilmengen $\mathbb{A}$ und $\mathbb{B}$. In dieser Arbeit: Die Zuordnung von Softwaremodulen zu Prozessoren.
Rahmen (*engl.* frame)	Ein allgemein gültiger Begriff für einen Block von Binärdaten, der zur Übertragung über eine Luftschnittstelle vorgesehen ist.
Ressourcen	Ein allgemein gültiger Begriff für die funktionell unterscheidbaren Hardware-Bestandteile eines Systems.
Software	Die Gesamtheit der Module zur Erfüllung aller Signalverarbeitungsaufgaben auf der physikalischen Schicht und der Datensicherungsschicht eines Mod-SDRs.
Standard	Ein Mobilfunkstandard. Eine Realisierung von vielen möglichen zur drahtlosen Anbindung eines mobilen Endgerätes an das Festnetz. Ein Standard umfasst mindestens die Luftschnittstelle.
System	Die Gesamtheit aller Bestandteile und physikalischen Strukturen der Hardware. In dieser Arbeit: Die Realisierung einer Hardware für Mod-SDR.

Formelzeichen

α	Absolutes Maß für die Prozessorgeschwindigkeit, $\alpha \in \mathbb{R}^+$
$\mathbb{A}$	Partition, Teilmenge von $\mathbb{M}$
$\mathbf{A}$	Matrix, reell symmetrisch
β	Relative Busgeschwindigkeit, $\beta \in \mathbb{R}^+$
B	Gesamtanzahl der Busse eines Systems, $B \in \mathbb{N}$
$\mathbb{B}$	Partition, Teilmenge von $\mathbb{M}$
$\mathbf{B}$	Matrix, reell symmetrisch
c	Realisierung der Zufallsvariablen C , $c \in \mathbb{R}$
$\mathbf{C}$	Matrix, reell symmetrisch
d	Grad der inhärenten Parallelität eines Graphen
D	Charakteristischer Wert eines Knotens, $\rightarrow$ KL-Algorithmus
$\mathbf{\Delta}$	Diagonalmatrix

Δp	Differenz zwischen den Gesamtlaufzeiten zweier Partitionen
Δt	Zeitauflösung, Zeitintervall $\Delta t \in \mathbb{R}^+$
ΔT	Echtzeitperiode, Zeitintervall $\Delta T \in \mathbb{R}^+$
E	Externe Kosten eines Knotens, $E \in \mathbb{R}_0^+$
$\mathcal{E}$	Ereignis
ϵ	Fehlermaß, $\epsilon \in \mathbb{R}^+$
f	Funktion
F	Verteilungsfunktion
g	Gewinn, $g \in \mathbb{R}$
G	Granularität, $G \in \mathbb{R}^+$
$\mathcal{G}$	Graph $\mathcal{G} = (\mathbb{M}, \mathbb{K})$ mit den gerichteten Kanten $\langle l,m \rangle$
γ	Parameter einer Funktion, $\gamma \in \mathbb{R}$
i	Index, $i \in \mathbb{N}$
I	Interne Kosten eines Knotens, $I \in \mathbb{R}_0^+$
$\mathbf{I}$	Einheitsmatrix
j	Index, $j \in \mathbb{N}$
k	Index für Hardware-Ressourcen R_k, $k \in \mathbb{N} : 1 \leq k \leq K$
K	Gesamtanzahl der Hardware-Ressourcen eines Systems, $K \in \mathbb{N}$
$\mathbb{K}$	Menge aller Kanten eines gerichteten Graphen $\mathcal{G}$
l	Index eines Softwaremoduls, $l \in \mathbb{N}$
L	Gesamtanzahl der Prozessoren eines Systems, $L \in \mathbb{N}$
λ	Parameter der Exponentialverteilung, $\lambda \in \mathbb{R}$
m	Index eines Softwaremoduls, $m \in \mathbb{N}$
M	Gesamtanzahl der betrachteten Module
$\mathbb{M}$	Menge aller Knoten eines Graphen $\mathcal{G}$
μ	Parameter der Gaußschen Wahrscheinlichkeitsdichte
μ_{eff}	Erwartungswert einer beliebig verteilten Zufallsvariablen
n	Zeitindex, $n \in \mathbb{Z}$

N	Gesamtanzahl von Zeitintervallen Δt in ΔT, $N \in \mathbb{N}$
N_F	Gesamtanzahl von Funkrahmen pro Paket, $N_F \in \mathbb{N}$
$\mathbb{N}$	Menge der natürlichen Zahlen
ν	Testzeitindex, $\nu \in \mathbb{N} : 1 \leq \nu \leq N$
p	Laufzeit einer Signalverarbeitungsaufgabe
P	Wahrscheinlichkeit, $P \in \mathbb{R} : 0 \leq P \leq 1$
$\mathbf{P}$	Stochastische Matrix
$\mathbf{Q}$	Matrix von Eigenvektoren, im Allgemeinen nicht orthogonal
r	Konstante, $r \in \mathbb{R}^+$
R_c	Coderate, $R_c \in \mathbb{R}^+$
$\mathbb{R}$	Menge der reellen Zahlen
$\mathbb{R}^+$	Menge der positiven reellen Zahlen
$\mathbb{R}_0^+$	Menge der nichtnegativen reellen Zahlen, $\mathbb{R}_0^+ = \mathbb{R}^+ \cup \{0\}$
ρ	Reelle Konstante, Eigenwert, $\rho \in \mathbb{R}$
s	Speedup, $s \in \mathbb{R}^+$
σ	Parameter der Gaußschen Wahrscheinlichkeitsdichte
σ_{eff}	Standardabweichung einer beliebig verteilten Zufallsvariablen
σ_p	Effektive relative Laufzeit-Streuung $\sigma_p = \sigma_{\mathit{eff}} / \mu_{\mathit{eff}} \in \mathbb{R}^+$
t	Zeit, $t \in \mathbb{R}$
T_0	Startzeitpunkt, $T_0 \in \mathbb{R}$
$\mathbf{U}$	Orthogonale Matrix von Eigenvektoren
W	Gewicht einer Partition, $W \in \mathbb{R}_0^+$
$\mathbf{W}$	Matrix der Kantengewichte $w_{l,m} \in \mathbb{R}_0^+$ eines Graphen $\mathcal{G}$
$x_{m,n}$	Triggervariable des Moduls m im Zeitintervall n, $x_{m,n} \in \{0; 1\}$
$\mathbb{Z}$	Menge der ganzen Zahlen

Literaturverzeichnis

[1] *ANSI/IEEE Std 802.11, Wireless LAN MAC and PHY Specifications*. 1999 Edition, and IEEE 802.11a-1999, High-speed Physical Layer in the 5GHz Band.

[2] W. Abu-Al-Saud und G. Stuber: *Efficient Sample Rate Conversion for Software Radio Systems*. In: *IEEE Global Telecommunications Conference (GLOBECOM'02)*, Bd. 1, S. 559–563, Taipeh (Taiwan, ROC), Nov 2002. IEEE.

[3] W. Abu-Al-Saud und G. Stuber: *Modified CIC Filter for Sample Rate Conversion in Software Radio Systems*. IEEE Signal Processing Lett., Bd. 10, Nr. 5, S. 152–154, May 2003.

[4] G. Ahlquist, M. Rice und B. Nelson: *Error Control Coding in Software Radios: An FPGA Approach*. IEEE Personal Commun. Mag., Bd. 6, Nr. 4, S. 35–39, Aug 1999.

[5] H. Ali, Hesham und H. El-Rewini: *An Optimal Algorithm for Scheduling Interval Ordered Tasks with Communication on N Processors*. Journal of Computer and System Sciences, Bd. 51, Nr. 2, S. 301–306, Oct 1995.

[6] G. Amdahl: *Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities*. In: *Proc. AFIPS 1967 Spring Joint Comp. Conf.*, S. 483–485, Atlantic City, NJ (USA), April 1967.

[7] M. Beach, J. MacLeod und P. Warr: *Radio Frequency Translation for Software Defined Radios*. In: W. Tuttlebee (Hrsg.): *Software Defined Radio: Enabling Technologies*, S. 25–78, London (UK), 2002. John Wiley & Sons, Inc.

[8] M. Berkelaar, K. Eikland und P. Notebaert: *lp_solve V5.0*, May 2004. An Open Source (Mixed-Integer) Linear Programming System. License: GNU LGPL (Lesser General Public Licence). [Online]. Available: http://groups.yahoo.com/group/lp_solve/files/.

[9] U. Berthold, A.-R. Rhiemeier und F. K. Jondral: *A Pipelining Approach to Operating Modular Software Defined Radio*. In: *IEEE Sarnoff Symp. (SARNOFF'04)*, Princeton, NJ (USA), April 2004.

[10] U. Berthold, A.-R. Rhiemeier und F. K. Jondral: *Spectral Partitioning for Modular Software Defined Radio*. In: *IEEE Vehicular Tech. Conf. (VTC'04) Spring*, Milano (Italy), May 2004.

[11] J. Brakensiek, R. Wittmann und M. Darianian: *Software Defined Radio Technology for Multistandard Terminals*. In: *2nd Karlsruhe Workshop on Software Defined Radios*, S. 87–92, Karlsruhe (Germany), March 2002. Universität Karlsruhe (TH), Institut für Nachrichtentechnik. ISSN 1616-6019.

[12] I. Bronstein und K. Semendjajew: *Taschenbuch der Mathematik*. B.G. Teubner Verlagsgesellschaft, 25. Aufl. 1991.

[13] E. Buracchini: *The Software Radio Concept*. IEEE Commun. Mag., Bd. 38, Nr. 9, S. 138–143, Sept 2000.

[14] M.-D. Cano, J. Malgosa-Sanahuja, F. Cerdan und J. Garcia-Haro: *Internet Measurements and Data Study*. In: *IEEE Pacific Rim Conf. Communications, Computers and Signal Processing (PACRIM'01)*, S. 393–396. Victoria, BC (Canada), Aug 2001.

[15] C. Cavalcante et al.: *Scheduling Projects with Labor Constraints*. J. Discrete Applied Mathematics, Bd. 112, Nr. 1, S. 27–52, 2001.

[16] *The Vanu Perspective*. Software Defined Radio Forum, June 2002. Document Nr. SDRF-02-I-0039-V0.00 [Online]. Available: http://www.sdrforum.org.

[17] M. Cummings und S. Haruyama: *FPGA in the Software Radio*. IEEE Commun. Mag., Bd. 37, Nr. 2, S. 108–112, Feb 1999.

[18] C. Dick: *Reinventing the Signal Processor*. Xcell Journal, Bd. 45, S. 72–75, Spring 2003. Xilinx, Inc., San Jose, CA (USA).

[19] M. Dillinger, K. Madani und N. Alonistioti (Hrsg.): *Software Defined Radio: Architectures, Systems and Functions*, Bd. 5 d. Reihe *Wiley Series in Software Radio*. John Wiley & Sons Ltd., 2003.

[20] C. H. Q. Ding et al.: *A Min-max Cut Algorithm for Graph Partitioning and Data Clustering*. In: *IEEE Int'l Conf. Data Mining (ICDM 2001)*, S. 107–114, San Jose, CA (USA), 2001.

[21] J. Dodley, R. Erving und C. Rice: *In-Building Software Radio Architecture, Design and Analysis*. In: *11th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2000)*, Bd. 1, S. 479–483, London (UK), Sept 2000. IEEE.

[22] *ETSI TR 101 146; UMTS 30.06; UMTS Terrestrial Radio Access (UTRA); Concept evaluation*. ETSI, 1997. Version 3.0.0. (1997-12). [Online]. Available: http://www.etsi.org.

[23] *ETSI TS 125 101; UMTS; UE Radio Transmission and Reception (FDD)*. ETSI, 2002. Version 5.5.0 (2002-12), pp. 51-52. [Online]. Available: http://www.etsi.org.

[24] *ETSI TS 125 200 Series; UMTS Physical Layer (TS 125 21X: FDD, TS 125 22X: TDD)*. ETSI, 2003. Version 6.0.0 (2003-12). [Online]. Available: http://www.etsi.org.

[25] *ETSI TS 125 211; UMTS; Physical Channels and Mapping of Transport Channels onto Physical Channels (FDD)*. ETSI, 2003. Version 6.0.0 (2003-12), p. 11 + p. 21. [Online]. Available: http://www.etsi.org.

[26] *ETSI TS 125 212 (FDD) / TS 125 222 (TDD); UMTS; Multiplexing and Channel Coding*. ETSI, 2003. Version 6.0.0 (2003-12). [Online]. Available: http://www.etsi.org.

[27] *ETSI TS 125 090; UMTS; AMR speech Codec; Transcoding Functions*. ETSI, 2002. Version 5.0.0 (2002-06). [Online]. Available: http://www.etsi.org.

[28] *ETSI TS 125 103; Digital cellular telecommunications system (Phase 2+); UMTS; Speech codec list for GSM and UMTS*. ETSI, 2002. Version 5.4.0 (2002-12). [Online]. Available: http://www.etsi.org.

[29] M. Fiedler: *Algebraic Connectivity of Graphs*. Czechoslovak Mathematical Journal, Bd. 23, S. 298–305, 1973.

[30] M. Fiedler: *A Property of Eigenvectors of Non-Negative Symmetric Matrices and its Application to Graph Theory*. Czechoslovak Mathematical Journal, Bd. 25, S. 619–633, 1975.

[31] P. Galicki: *FPGAs Have the Multiprocessing I/O Infrastructure to Meet 3G Base Station Design Goals*. Xcell Journal, Bd. 45, S. 80–84, Spring 2003. Xilinx, Inc., San Jose, CA (USA).

[32] *SDR Market Survey*. Gartner Consulting Group, commissioned by the Software Defined Radio Forum, engagement no. 220079600, 2002. [Online]. Available to SDR Forum Members: http://www.sdrforum.org.

[33] N. Georganopoulos et al.: *Terminal-Centric View of Software Reconfigurable System Architecture and Enabling Components and Technologies*. IEEE Commun. Mag., Bd. 42, Nr. 5, S. 100–110, May 2004.

[34] W.-D. Haaß: *Handbuch der Kommunikationsnetze*. Springer Verlag, 1997. ISBN 3-540-61837-6.

[35] H. Harada, Y. Kamio und M. Fujise: *Multimode Software Radio System by Parameter Controlled and Telecommunication Block Embedded DSP Hardware*. IEICE Trans. Commun., Bd. E83-B, Nr. 6, S. 1217–1228, June 2000. Special Issue on Software Defined Radio Technology and its Applications.

[36] S. Haykin: *Adaptive Filter Theory*. Prentice Hall Information and System Sciences Series. Prentice Hall, 4 Aufl., 2002. ISBN 0-13-090126-1.

[37] T. Hentschel und G. Fettweis: *Sample Rate Conversion for Software Radio*. IEEE Commun. Mag., Bd. 38, Nr. 8, S. 142–150, Aug 2000.

[38] T. Hu: *Parallel Sequencing and Assembly Line Problems*. Operations Research, Bd. 9, S. 841–848, 1961.

[39] X. Huang, K.-L. Du, A. Lai und K. Cheng: *A Unified Software Radio Architecture*. In: *IEEE 3rd Workshop on Signal Processing Advances in Wireless Communications (SPAWC'01)*, Bd. 2, S. 330–333, Taoyuan (Taiwan, ROC), March 2001. IEEE.

[40] *RFC791: Internet Protocol*. Internet Engineering Task Force, 1981. [Online]. Available: http://www.ietf.org/.

[41] International Organization for Standardization, 1994. ISO/IEC 7498-1:1994, ICS 35.100 series.

[42] *Recommendation ITU-R M.1457*. International Telecommunication Union, May 2000. [Online]. Available: http://www.itu.int.

[43] F. Jondral: *Parametrization - a Technique for SDR Implementation*. In: W. Tuttlebee (Hrsg.): *Software Defined Radio: Enabling Technologies*, S. 232–256, London (UK), 2002. John Wiley & Sons, Inc.

[44] M. Jordan: *Turbo Code Codec Implementation and Performance in a Software Radio*. In: *Military Communications Conference (MILCOM'99)*, Bd. 1, S. 525–529. IEEE, Nov 1999.

[45] *IEEE J. Select. Areas Commun.*, vol. 17, no. 4, April 1999. Special Issue on Software Radio.

[46] P. B. Kennington und L. Astier: *Power Consumption of A/D Converters for Software Radio Applications*. IEEE Trans. Veh. Technol., Bd. 49, Nr. 2, S. 643–650, March 2000.

[47] B. W. Kernighan und S. Lin: *An Efficient Heuristic Procedure for Partitioning Graphs*. Bell System Technical Journal, Bd. 49, S. 291–307, 1970.

[48] B. Kraemer: *Data Conversion Considerations for Software Radios*. In: *IEEE 5th International Symposium on Spread Spectrum Techniques and Applications (ISSSTA'98)*, Bd. 2, S. 546–550, South Africa, Sept 1998. IEEE.

[49] K. Kroschel: *Statistische Nachrichtentheorie*. Springer-Verlag, 1996.

[50] K. Kuipers: *bnb V2.0: A Mixed Integer Nonlinear Optimization Problem Solver*, 2002. Matlab R11 Code. [Online]. Available: http://www.mathworks.com/matlabcentral/.

[51] Y.-K. Kwok und I. Ahmad: *Benchmarking and Comparison of the Task Graph Scheduling Algorithms*. Journal of Parallel and Distributed Computing, Bd. 59, Nr. 3, S. 381–422, 1999.

[52] P. Leaves et al.: *Dynamic Spectrum Allocation in Composite Reconfigurable Wireless Networks*. IEEE Spectr., Bd. 41, Nr. 5, S. 72–81, May 2004.

[53] E. Lee und D. Messerschmitt: *Static Scheduling of Synchronous Data Flow Programs for DSP*. IEEE Trans. Computers, Bd. 36, Nr. 1, S. 24–35, Jan 1987.

[54] J. Llosa, A. González, E. Ayguadé und M. Valero: *Swing Modulo Scheduling*. In: *Int'l Conf. Parallel Architectures and Compilation Techniques (PACT'96)*, S. 80–86, Boston, MA (USA), Oct 1996.

[55] R. Machauer: *Multicode-Detektion im UMTS*. Dissertation, Forschungsberichte aus dem Institut für Nachrichtentechnik, Universität Karlsruhe (TH), Oct 2002. ISSN 1433-3821.

[56] J. Maurer, T. Fügen, T. Schäfer und W. Wiesbeck: *A New Inter-Vehicle Communications (IVC) Channel Model*. In: *IEEE Vehicular Tech. Conf. (VTC'04) Fall)*, Los Angeles, CA (USA), Sept 2004. IEEE.

[57] J. Maurer, T. Schäfer und W. Wiesbeck: *Wave Propagation Modelling for Inter-Vehicle Communications*. In: *URSI 27th General Assembly, Paper-No. 1231, Maastricht (The Netherlands)*. International Union of Radio Science, Aug 2002.

[58] J. Ming, H. Y. Weng und S. Bai: *An Efficient IF Architecture for Dual-Mode GSM/W-CDMA Receiver of a Software Radio*. In: *IEEE International Workshop on Mobile Multimedia Communications (MoMuC'99)*, S. 21–24, San Diego, CA (USA), Nov 1999. IEEE.

[59] J. Mitola: *Software Radios*. In: *National Telesystems Conf.*, S. 13/15–13/23. IEEE, May 1992.

[60] J. Mitola: *Software Radios: Survey, Critical Evaluation and Future Directions*. IEEE Aerosp. Electron. Syst. Mag, Bd. 8, Nr. 4, S. 25–36, April 1993.

[61] J. Mitola: *The Software Radio Architecture*. IEEE Commun. Mag., Bd. 33, Nr. 5, S. 26–38, May 1995.

[62] J. Mitola: *Software Radio Architecture: A Mathematical Perspective*. IEEE J. Select. Areas Commun., Bd. 17, Nr. 4, S. 514–538, 1999.

[63] J. Mitola: *Technical Challenges in the Globalization of Software Radio*. IEEE Commun. Mag., Bd. 37, Nr. 2, S. 84–89, Feb 1999.

[64] J. Mitola: *Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio*. Dissertation, Computer Communication System Laboratory, Dept of Teleinformatics, Royal Institute of Technology (KTH), Stockholm, Sweden, May 2000. ISSN 1403-5286.

[65] J. Mitola und G. Maguire: *Cognitive Radio: Making Software Radios More Personal*. IEEE Personal Commun. Mag., Bd. 6, Nr. 4, S. 13–18, 1999.

[66] B. Oelkrug, D. Bücker, A. Dröge, J. Brakensiek und M. Darianian: *Programmable Hardware Accelerator for Universal Telecommunication Applications*. In: *2nd Karlsruhe Workshop on Software Defined Radios*, S. 93–98, Karlsruhe (Germany), March 2002. Universität Karlsruhe (TH), Institut für Nachrichtentechnik. ISSN 1616-6019.

[67] M. Öner und F. K. Jondral: *Air Interface Recognition For A Software Radio System Exploiting Cyclostationarity*. In: *15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2004)*, Barcelona (Spain), Sept 2004. IEEE.

[68] M. Öner und F. K. Jondral: *Extracting the Channel Allocation Information in a Spectrum Pooling System Exploiting Cyclostationarity*. In: *15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2004)*, Barcelona (Spain), Sept 2004. IEEE.

[69] C. H. Papadimitriou und M. Yannakakis: *Scheduling Interval-Ordered Tasks*. SIAM Journal of Computing, Bd. 8, Nr. 3, S. 405–409, Aug 1979.

[70] A. Papoulis: *Probability, Random Variables, and Stochastic Processes*. McGraw Hill, 3rd Aufl., 1991.

[71] A. Perez-Neira, X. Mestre und J. Fonollosa: *Smart Antennas in Software Radio Base Stations*. IEEE Commun. Mag., Bd. 39, Nr. 2, S. 166–173, Feb 2001.

[72] J. Razavilar, F. Rashid-Farrokhi und K. Liu: *Software Radio Architecture with Smart Antennas*. IEEE J. Select. Areas Commun., Bd. 17, Nr. 4, S. 662–676, April 1999.

[73] J. H. Reed: *Software Radio: A Modern Approach to Radio Engineering*. Prentice Hall Communciations Engineering and Emerging Technologies Series. Prentice Hall, 2002.

[74] A.-R. Rhiemeier: *Benfits and Limits of Parameterized Channel Coding for Software Radio*. In: *2nd Karlsruhe Workshop on Software Radios (WSR'02)*, S. 107–112, Karlsruhe (Germany), April 2002.

[75] A.-R. Rhiemeier: *A Comparison of Scheduling Approaches in Modular Software Defined Radio*. In: *3rd Karlsruhe Workshop on Software Radios (WSR'04)*, S. 33–38, Karlsruhe (Germany), March 2004. ISSN 1616-6019. — Als Reprint erschienen in: *Frequenz*, Zeitschrift für Telekommunikation, Bd 58, Nr. 5/6, Mai/Juni 2004, S. 115-120.

[76] A.-R. Rhiemeier und F. Jondral: *A Software Radio View of Modulation and Spreading for UTRA FDD and UTRA TDD*. In: *3rd Int'l Conf. Mobile Commun. Tech. (3G2002)*, S. 464–468, London (UK), May 2002. , IEE.

[77] A.-R. Rhiemeier und F. K. Jondral: *Enhanced Resource Utilization in Software Defined Radio Terminals*. In: *Int'les Wissenschaftliches Kolloquium (IWK'03)*, Ilmenau (Germany), Sept 2003. Technische Universität Ilmenau (Germany).

[78] A.-R. Rhiemeier und F. K. Jondral: *Mathematical Modeling of the Software Radio Design Problem*. IEICE Trans. Commun., Bd. E86-B, Nr. 12, S. 3456–3467, Dec 2003. Special Issue on Software Defined Radio Technology and its Applications.

[79] A.-R. Rhiemeier und F. K. Jondral: *On the Design of Modular Software Defined Radio Systems*. In: *IEE Colloquium on DSP Enabled Radio*, Alba Campus, Livingston, Scotland (UK), Sept 2003. The Institute for System Level Integration (ISLI).

[80] A.-R. Rhiemeier und F. K. Jondral: *A Software Partitioning Algorithm for Modular Software Defined Radio*. In: *6th Int'l Symp. Wireless Personal Multimedia Communications (WPMC'03)*, S. 42–46, Yokosuka (Japan), Oct 2003.

[81] A.-R. Rhiemeier und F. K. Jondral: *Software Partitioning and Hardware Architecture for Modular SDR Systems*. In: *SDR Forum Technical Conf. and Product Exhibition (SDR'03)*, Bd. 2, S. 9–15, Orlando, FL (USA), Nov 2003. SDR Forum.

[82] A.-R. Rhiemeier, T. Weiss und F. K. Jondral: *Half-Frame Pipelining for Modular Software Defined Radio*. In: *15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2004)*, Barcelona (Spain), Sept 2004. IEEE.

[83] A.-R. Rhiemeier, T. Weiss und F. K. Jondral: *A Simple and Efficient Solution to Half-Frame Pipelining for Modular Software Defined Radio*. In: *SDR Forum Technical Conf. and Product Exhibition (SDR'04)*, Phoenix, AZ (USA), Nov 2004. SDR Forum. to appear.

[84] R. J. Richards und H. J. De Los Santos: *MEMS for RF/Microwave Appliations: The Next Wave*. Microwave Journal, Bd. 44, Nr. 3, S. 20–41, March 2001.

[85] R. J. Richards und H. J. De Los Santos: *MEMS for RF/Microwave Appliations: The Next Wave – Part II*. Microwave Journal, Bd. 44, Nr. 7, S. 142–152, July 2001.

[86] I. E. G. Richardson: *H.264 and MPEG-4 Video Compression – Video Coding for Next-Generation Multimedia*. John Wiley & Sons, 2003. ISBN 0-470-84837-5.

[87] J. R. Sacha und M. J. Irwin: *Number Representations for Reducing Data Bus Power Dissipation*. In: *32nd Asilomar Conf. on Signals, Systems and Computers*, Bd. 1, S. 213–217. IEEE, Nov 1998.

[88] A. Salkintzis, H. Nie und P. Mathiopoulos: *ADC and DSP Challenges in the Development of Software Radio Base Stations*. IEEE Personal Commun. Mag., Bd. 6, Nr. 4, S. 47–55, Aug 1999.

[89] *Software Communications Architecture Specification, JTRS-5000SCA V2.2.1*. Joint Tactical Radio System (JTRS) Joint Program Office, April 2004. [Online]. Available: http://jtrs.army.mil.

[90] W. Schacherbauer, A. Springer, T. Ostertag, C. Ruppel und R. Weigel: *A Flexible Multiband Frontend for Software Radios Using High IF and Active Interference Cancellation*. In: *IEEE MTT-S International Microwave Symposium Digest*, Bd. 2, S. 1085–1088. IEEE, May 2001.

[91] *2nd SDR Forum Technical Conference and Product Exhibition (SDR'03)*. Software Defined Radio Forum, Nov 2003. Orlando, FL (USA).

[92] *Architecture and Elements of SDR Systems as Related to Standards*. Software Defined Radio Forum, Nov 1999. Technical Report V2.1.

[93] E. Seneta: *Non-Negative Matrices*. George Allen & Unwin Ltd, London, 1973. ISBN 0-04-519011-9.

[94] I. Seskar und N. Mandayam: *A Software Radio Architecture for Linear Multiuser Detection*. IEEE J. Select. Areas Commun., Bd. 17, Nr. 5, S. 814–823, May 1999.

[95] J. Siedersleben: *Branch and Bound*. Tech. Report No. 218, Universität Karlsruhe (TH), Inst. für Wirtschaftstheorie und Operations-Research (WIOR), 1983.

[96] J. Singh: *High Speed Analog-to-Digital Converter for Software Radio Applications*. In: *11th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2000)*, Bd. 1, S. 39–42, London (UK), Sept 2000. IEEE.

[97] M. Solar und I. Mario: *A Parallel Compiler Scheduler*. In: *21st Int'l Conference of the Chilean Computer Science Society (SCCC'01)*, S. 256–263, Punta Arenas (Chile), Nov 2001. IEEE.

[98] S. Srikanteswara, R. C. Palat, J. H. Reed und P. Athanas: *An Overview of Configurable Computing Machines for Software Radio Handsets*. IEEE Commun. Mag., Bd. 41, Nr. 7, S. 134–141, July 2003.

[99] W. Stehle: *Digitale Netze: Grundlagen, Protokolle, Anwendungen*. J. Schlembach Fachverlag, Weil der Stadt, 2001. ISBN 3-935340-09-5.

[100] V. Thara und M. Siddiqi: *Power Efficiency of Software Radio Based Turbo Codec*. In: *IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering (TENCON'02)*, Bd. 2, S. 1060–1063, Beijing (China), Oct 2002. IEEE.

[101] N. Tredennick und B. Shimamoto: *Go Reconfigure: Programmable Logic in Handheld Devices*. IEEE Spectr., Bd. 40, Nr. 12, S. 36–40, Dec 2003.

[102] W. Tuttlebee (Hrsg.): *Software Defined Radio: Enabling Technologies*, Bd. 4 d. Reihe *Wiley Series in Software Radio*. John Wiley & Sons Ltd., 2002.

[103] W. Tuttlebee (Hrsg.): *Software Defined Radio: Origins, Drivers and International Perspectives*, Bd. 3 d. Reihe *Wiley Series in Software Radio*. John Wiley & Sons Ltd., 2002.

[104] W. Tuttlebee (Hrsg.): *Software Defined Radio: Baseband Technology for 3G Handsets and Basestations*, Bd. 6 d. Reihe *Wiley Series in Software Radio*. John Wiley & Sons Ltd., 2004.

[105] M. Valenti: *An Efficient Software Radio Implementation of the UMTS Turbo Codec*. In: *12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2001)*, Bd. 2, S. G108–G113, San Diego, CA (USA), Sept 2001. IEEE.

[106] D. Verkest: *Machine Chameleon*. IEEE Spectr., Bd. 40, Nr. 12, S. 41–46, Dec 2003.

[107] J. Verriet: *Scheduling Interval Orders with Release Dates and Deadlines*. In: *7th Int'l Symposium on Algorithms and Computation*, Osaka (Japan), Dec 1996. also appeared in LNCS 1178, Springer, Berlin.

[108] R. H. Walden: *Analog-to-Digital Converter Survey and Analysis*. IEEE J. Select. Areas Commun., Bd. 17, Nr. 4, S. 539–550, 1999.

[109] F. Weinberg: *Einführung in die Methode Branch and Bound*, Bd. 4 d. Reihe *Lecture Notes in Operations Research and Mathematical Systems*. Springer Verlag, 1968.

[110] J. Weinmiller, M. Schlager, A. Festag und A. Wolisz: *Performance Study of Access Control in Wireless LANs - IEEE 802.11 DFWMAC and ETSI RES 10 Hiperlan*. Journal on Mobile Networks and Applications, Bd. 2–1, June 1997.

[111] T. Weiss und F. Jondral: *Spectrum Pooling: An Innovative Strategy for the Enhancement of Spectrum Efficiency*. IEEE Commun. Mag., Bd. 42, Nr. 3, S. 8–14, March 2004.

[112] N. H. E. Weste: *Principles of CMOS VLSI Design*. Addison-Wesley VLSI Systems Series. Addison-Wesley, 1988. ISBN 0-201-08222-5.

[113] O. Wiener, M. Bonik und R. Hödicke: *Eine elementare Einführung in die Theorie der Turing-Maschinen*. Springer Verlag, 1998. ISBN 3-211-82769-2.

[114] A. Wiesler: *Parametergesteuertes Software Radio für Mobilfunksysteme*. Dissertation, Forschungsberichte aus dem Institut für Nachrichtentechnik, Universität Karlsruhe (TH), May 2001. ISSN 1433-3821.

[115] J. Wilkinson: *The Algebraic Eigenvalue Problem*. Oxford University Press, 1965. ISBN 0-19-853403-5.

[116] *2nd Karlsruhe Workshop on Software Radios (WSR'02)*. Institut für Nachrichtentechnik, Universität Karlsruhe (TH), March 2002. ISSN 1616-6019.

[117] *3rd Karlsruhe Workshop on Software Radios (WSR'04)*. Institut für Nachrichtentechnik, Universität Karlsruhe (TH), March 2004. ISSN 1616-6019.

[118] Y.-C. Wu und T.-S. Ng: *FPGA Implementation of Digital Timing Recovery in Software Radio Receiver*. In: *IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS 2000)*, S. 703–707, Tianjin (China), Dec 2000. IEEE.

Studien- und Diplomarbeiten

Wojciech Kuropatwiński:	*Simulation einer digitalen Datenübertragung nach dem PMR-Standard TETRAPOL* *15. Mai 2001*
Bertrand Souville:	*Simulation einer digitalen Datenübertragung nach dem PMR-Standard TETRA* *08. Juni 2001*
Serge-Patrick Tchuente:	*Einsatz von Markov-Prozessen in der Modellierung nachrichtentechnischer Vorgänge* *30. Juli 2001*
Piotr Rykaczewski:	*Integration von UTRA FDD und UTRA TDD in ein Multi Mode Radio (1): Turbo Coding und OSI Layer 2 Signalverarbeitung* *29. September 2001*
Mariusz Koc:	*Integration von UTRA FDD und UTRA TDD in ein Multi Mode Radio (2): CDMA Multicode Transmission und OSI Layer 1 Signalverarbeitung* *15. November 2001*
Mounir Hanana:	*Algorithmik in UMTS* *06. Februar 2002*
Ulrich Berthold:	*Algorithmen für Partitioning und Scheduling in Mod-SDR Systemen* *15. November 2003*
Diego Vicente Llorca Rubí:	*Ressourcennutzung in Modularen Software Defined Radio Systemen* *15. Juni 2004*

Index

Lebenslauf

	Persönliche Daten
Name	Arnd-Ragnar Rhiemeier
Geburtsdatum	24. August 1972
Geburtsort	Bochum
Staatsangehörigkeit	deutsch
	Schulausbildung
1979-1983	Grundschule Natorpstraße, Bochum
1983-1992	Gymnasium am Ostring, Bochum
	Grundwehrdienst
1992-1993	Luftwaffenausbildungsregiment 3, Roth Elektronikstaffel, JaboG 31 “Boelcke”, Nörvenich
	Studium und Berufsweg
1993-1995	Ruhr-Universität Bochum, Fakultät für Elektrotechnik, Grundstudium
1995-1996	Universität Karlsruhe (TH), Fakultät für Elektrotechnik und Informationstechnik, Kernfachstudium
1996-1997	Institut National des Sciences Appliquées de Lyon Département de Génie Electrique/Laboratoire Créatis Vertiefungsstudium/Studienarbeit
1998-1999	Universität Karlsruhe (TH), Fakultät für Elektrotechnik und Informationstechnik, Vertiefungsstudium
1999	New Jersey Institute of Technology, Newark, NJ, USA Center for Communications and Signal Processing Research, Diplomarbeit
seit 2000	Universität Karlsruhe (TH), Institut für Nachrichtentechnik, wissenschaftlicher Mitarbeiter