Latenzverringerung in Basisbandprozessoren

am Beispiel eines hochratigen OFDM-Kommunikationssystems

Von der Fakultät für Mathematik, Naturwissenschaften und Informatik
der Brandenburgischen Technischen Universität Cottbus

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
(Dr.-Ing.)

genehmigte Dissertation

vorgelegt von

Diplom-Ingenieur

Markus Petri

geboren am 31. Dezember 1979 in Berlin

Gutachter:	Prof. Dr. Eckhard Grass
Gutachter:	Prof. Dr. Hermann Rohling
Gutachter:	Prof. Dr. Heinrich Theodor Vierhaus

Tag der mündlichen Prüfung: 8. Juni 2012

Kurzfassung

In modernen digitalen Übertragungssystemen ist der gesamte Datendurchsatz nicht mehr allein durch die Datenrate – z. B. aufgrund der begrenzten Bandbreite – eingeschränkt. Stattdessen haben die bei der Verarbeitung entstehenden Latenzen einen immer größeren Einfluss auf die erzielbare Systemperformance. In der Fachliteratur beschriebene Untersuchungen zur Latenzverringerung in digitalen Basisbandprozessoren beschränken sich meist auf die Optimierung eigenständiger Module.

Im Rahmen dieser Arbeit werden, basierend auf einer Untersuchung der Basisbandmodule auf ihren Latenzbeitrag und einer Darstellung der Latenzauswirkungen, bestehende Verfahren der Latenzverringerung evaluiert und neue Verfahren zur Latenzreduktion entworfen. Die praktischen Auswirkungen werden dabei anhand der Latenzmodellierung eines bestehenden 60-GHz-OFDM-Kommunikationssystems aufgezeigt. Die zur Anwendung der neu entworfenen Verfahren notwendigen strukturellen Änderungen in der Basisbandverarbeitung werden detailliert dargestellt. Weiterhin wird eine allgemeine Methodik des latenzarmen Systementwurfs vorgestellt, die die Anwendung der neu entworfenen Verfahren beinhaltet.

Bei der Darstellung von Verfahren zur Latenzverringerung liegt der Schwerpunkt auf der zur Durchsatzsteigerung üblicherweise verwendeten, aber noch nicht im Zusammenhang mit der Latenzverringerung beschriebenen Anwendung von parallelen Datenverarbeitungsstrukturen sowie auf der neu vorgeschlagenen Anwendung von Spekulation. Mithilfe der spekulativen Demodulation und der spekulativen Dekodierung lassen sich Latenzen aufgrund von Abhängigkeiten in der Datenverarbeitung einfach vermeiden. Die sich aus den spekulativen Verfahren ergebenden Auswirkungen auf die MAC-Ebene werden zum Schluss kurz diskutiert.

English Abstract

The total data throughput of modern communication systems suffers not only from the limited data rate of the physical interface, e.g. due to bandwidth restrictions. Instead, the latencies caused by the data and signal processing in the baseband processor have a higher and higher influence on the possible overall system performance. There are some investigations on reducing the latency of digital baseband processors described in the literature, but they mostly focus on optimizing individual modules without paying attention to the complete system.

In this thesis, existing methods for latency reduction are evaluated and new methods are proposed, based on the analysis of baseband processing modules regarding to their latency contribution and a description of latency impacts. The reachable latency reduction with the proposed methods will be shown with the help of the latency modelling of an existing 60 GHz OFDM communication system. The necessary structural changes inside the baseband processing for the usage of the new methods are described in detail. Furthermore, a general methodology for a low-latency system design including the proposed new methods is presented.

The proposed new methods focus on the use of parallelism and speculation. While parallel data processing structures are usually used to increase the throughput of a system, but not described in the context of latency reduction, the proposed speculative methods were not mentioned before in literature. With the help of speculative demodulation and speculative decoding, it is possible to avoid latencies caused by dependencies in the data processing. The resulting influence of the new methods on the MAC layer is briefly discussed at the end of this thesis.

Gliederung

1. Einleitung und Motivation

Die digitale Datenkommunikation ist aus dem heutigen Alltag nicht mehr wegzudenken. Vor Jahren trat sie ihren Siegeszug an und ist nun integraler Bestandteil des gesellschaftlichen Lebens. Während Mobiltelefone ständige Erreichbarkeit und „Überalltelefonie“ ermöglichen, bietet das Internet Zugriff auf eine Vielzahl teilweise neuartiger Dienste und Serviceleistungen sowie einen fast unendlichen Nachrichtenvorrat. Mit dem Aufkommen der Smartphones ist die Nutzung dieser Dienste auch nicht mehr ortsgebunden, sondern jederzeit und überall möglich.

Auch lange Zeit der analogen Kommunikation vorbehaltene Bereiche wie Rundfunk und Fernsehen schließen sich aufgrund der entstehenden Vorteile dem digitalen Siegeszug an. Während der erste Anlauf des DAB[1] genannten digitalen Radioempfangs um die Jahrtausendwende als gescheitert angesehen werden konnte [1], ist mit der Umstellung auf DVB-T[2] das terrestrische Fernsehen in Deutschland vollständig digitalisiert. Die als *„digitale Dividende“* [2] bezeichneten frei werdenden Frequenzbereiche der analogen Ausstrahlung wurden sofort für neue digitale Kommunikationsdienste vergeben.

Neben diesen für die Allgemeinheit offensichtlichen Bereichen werden auch viele andere Gebiete der Kommunikationstechnik digitalisiert. Komplexe Industriesteuerungsanlagen kommunizieren untereinander nicht mehr über eine Vielzahl dedizierter analoger Steuerungsleitungen, sondern über digitale Bussysteme. Auch die vielerorts noch übliche analoge Bildverarbeitung wird mehr und mehr durch digitale Verfahren ersetzt.

Mit dieser Entwicklung geht der ständig steigende Bedarf nach höheren Datenraten einher. Während vor einigen Jahren im Internet noch der Download von Binärdateien, z. B. Programmen oder Dokumenten sowie der Mailversand den hauptsächlichen Datenverkehr erzeugten, ist es nun das Videostreaming [3]. Das Internet löst mit Diensten wie *Video-on-Demand* immer mehr das klassische Fernsehen ab.

Neben der reinen Steigerung der Datenrate, z. B. durch eine höhere Bandbreite, erfordert ein höherer Datendurchsatz jedoch auch eine Optimierung der Verarbeitungslatenzen, also der durch die digitale Verarbeitung entstehenden Verzögerungszeiten. Die sehr komplexen digitalen Basisbandprozessoren moderner Systeme haben auf diese Latenzen einen hohen Einfluss und begrenzen den möglichen Datendurchsatz. Für die Datenübertragung zwischen zwei Stationen ist es nutzlos, wenn einerseits durch die theoretisch mögliche Datenrate ein Datenpaket im eigentlichen Kanal nur 3 µs lang ist, die Verarbeitung eines Pakets im Empfänger jedoch 30 µs dauert. Bei Systemen mit unmittelbarer Paketempfangsbestätigung ist die entstehende

[1] DAB: Digital Audio Broadcasting
[2] DVB-T: Digital Video Broadcasting - Terrestrial

Totzeit im günstigsten Fall für andere Teilnehmer nutzbar, in einem reinen Zweistationsbetrieb bleibt sie ungenutzt.

In der Fachliteratur finden sich viele Untersuchungen zur Verringerung der MAC[3]-Latenzen wie in [4], [5] und [6], aber nur wenige zur Latenzverringerung in Basisbandprozessoren. Letztere beschränken sich hauptsächlich auf die eigenständige Untersuchung und Optimierung einzelner Module, wie in [7] auf die Sende- und Empfangsfilterbänke in DMT[4]-Systemen. Einige weitere Verfahren zur Latenzoptimierung eigenständiger Basisbandmodule werden beispielhaft im Kapitel 5 aufgeführt.

Latenzen entstehen jedoch nicht nur innerhalb einzelner Module eines Übertragungssystems, sondern auch durch das Zusammenspiel dieser. Für eine optimale Reduktion sind modulübergreifende Modellierungen und Verfahren notwendig. Mit der im Rahmen dieser Arbeit neu vorgeschlagenen Anwendung von spekulativen Methoden lassen sich Latenzen verringern, die sonst durch die Optimierung der einzelnen Module nur geringfügig beeinflusst werden könnten.

Die vorliegende Arbeit ist wie folgt strukturiert: Ausgehend von der schematischen Darstellung eines allgemeinen digitalen Datenübertragungssystems werden die einzelnen Blöcke des Basisbandprozessors auf ihren Beitrag zur Latenz untersucht. Dabei beschränkt sich die Untersuchung auf die digitale Basisbandverarbeitung und die daraus resultierenden Latenzen. Andere Ursachen, wie die Signallaufzeiten im Kanal oder Signalverzögerungen im analogen Sende- bzw. Empfangsteil, werden aufgrund des vergleichsweise vernachlässigbaren Einflusses nicht berücksichtigt. Neben einer Kategorisierung der gefundenen Latenzbeiträge werden die Auswirkungen von Latenzzeiten auf die Systemperformance dargestellt. Im Anschluss erfolgt eine beispielhafte Latenzmodellierung für ein 60-GHz-OFDM[5]-System.

Nach einem allgemeinen Überblick über Möglichkeiten zur Verringerung von Latenzen werden zwei Verfahren detailliert erörtert. Zum einen die Latenzverringerung durch eine Ausnutzung der im System vorhandenen oder einfach hinzuzufügenden Parallelität, zum anderen die Verwendung der bereits angesprochenen spekulativen Verfahren. Während die Nutzung von Parallelität üblich ist, jedoch noch nicht detailliert im Zusammenhang mit einer Latenzverkürzung für die Basisbandverarbeitung beschrieben wurde, handelt es sich bei der vorgestellten Nutzung von Spekulation um neue, im Rahmen dieser Arbeit entwickelte Methoden zur Latenzverringerung in der Basisbandverarbeitung. Letztere werden weiterhin auf ihre Eignung zur Verwendung mit aktuellen MAC-Prozessoren untersucht, um darauf aufbauend notwendige Änderungen und Anpassungen der MAC-Prozessoren zu diskutieren. Zugleich erfolgt ein

[3] MAC: Medium Access Control, Systemebene zur Steuerung der Datenübertragung, siehe Abschnitt 2.1
[4] DMT: Discrete Multi-Tone
[5] OFDM: Orthogonal Frequency Division Multiplex

kurzer Exkurs in die Möglichkeiten der Energieverbrauchsreduktion durch die Verwendung spekulativer Verarbeitungsverfahren.

Im Anschluss an die detaillierte Erörterung der Latenzverringerung durch Parallelität und durch Spekulation wird als Zusammenfassung eine allgemeine Methodik für den latenzarmen Systementwurf einer digitalen Basisbandverarbeitung vorgestellt. Diese Methodik erweitert die allgemeine Vorgehensweise des Systementwurfs und der Systemimplementierung um Teilprozesse für die Anwendung der latenzverringernden Verfahren.

2. Ursachen von Latenzen

2.1 Aufbau eines Kommunikationssystems

Ein Kommunikationssystem dient der Übermittlung von Nachrichten von einer Quelle zu einer Senke. Dabei werden die Nachrichten durch Signale repräsentiert, also durch die Darstellung der Nachricht durch physikalische Größen. Diese Arbeit beschäftigt sich mit digitalen Kommunikationssystemen, d. h. Systemen, bei denen die zu sendende Nachricht in Form von Digitalsignalen übermittelt wird. Die Nachricht besteht aus einer Folge von endlich vielen unterschiedlichen Symbolen. Auf diese Weise kann die Nachricht durch Digitalrechnersysteme verarbeitet werden.

Der schematische Aufbau eines allgemeinen digitalen Kommunikationssystems ist in Abb. 1 dargestellt. Dabei sind die einzelnen Blöcke mit Ausnahme von Quelle, Kanal und Senke nicht notwendigerweise in jedem praktischen System zu finden. Die Quellenkodierung entfernt Redundanz und verringert somit den Umfang der zu übertragenden Nachricht. Im Gegensatz dazu führt die Kanalkodierung der Nachricht zusätzliche Redundanz hinzu, um anhand dieser eventuelle, bei der Übertragung durch den Nachrichtenkanal aufgetretene Übertragungsfehler zu erkennen und / oder zu korrigieren. Die Leitungskodierung dient der Anpassung an den physikalischen Kanal, also das Übertragungsmedium. Sie stellt bestimmte Eigenschaften des Übertragungssignals sicher, z. B. die Gleichstromfreiheit bei einer kabelgebundenen Übertragung.

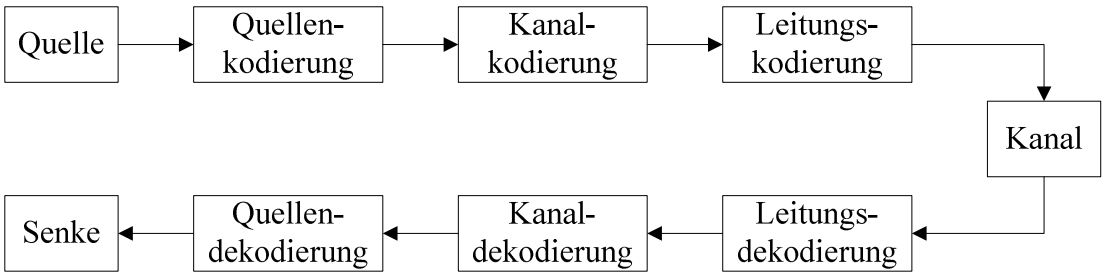

Abb. 1: Schematischer Aufbau eines Kommunikationssystems

In realen Systemen sind die einzelnen Funktionen Quellenkodierung, Kanalkodierung und Leitungskodierung auf unterschiedliche Module verteilt, deren Funktionen auch teilweise ineinander übergreifen. Den detaillierten schematischen Aufbau eines drahtlosen Kommunikationssystems zeigt Abb. 2. Dabei wurde die Quellenkodierung nicht dargestellt, da sie nicht notwendigerweise ein Systembestandteil ist. Die beiden Blöcke Mapper und Demapper stellen die eigentliche digitale Modulation dar, also die Zuordnung der eingehenden Datenbits zu Sendesymbolen. Die Signalformung soll die gewünschten spektralen Eigenschaften des Sendesignals herstellen. Sie kann genau wie die Aufbereitung des Empfangssignals aus einem Filter, aber auch aus ein oder mehreren verarbeitungsintensiven Blöcken, wie beispielsweise

einer FFT[6] bei OFDM-Systemen, bestehen. Eine analoge Modulation zur Übertragung in anderen Frequenzbereichen ist in der Darstellung nach Abb. 2 ein Bestandteil des analogen Kanals. Die einzelnen Module der digitalen Verarbeitung werden in den folgenden Abschnitten näher erläutert, gleichzeitig wird ihr Beitrag zur Latenz des Gesamtsystems analysiert.

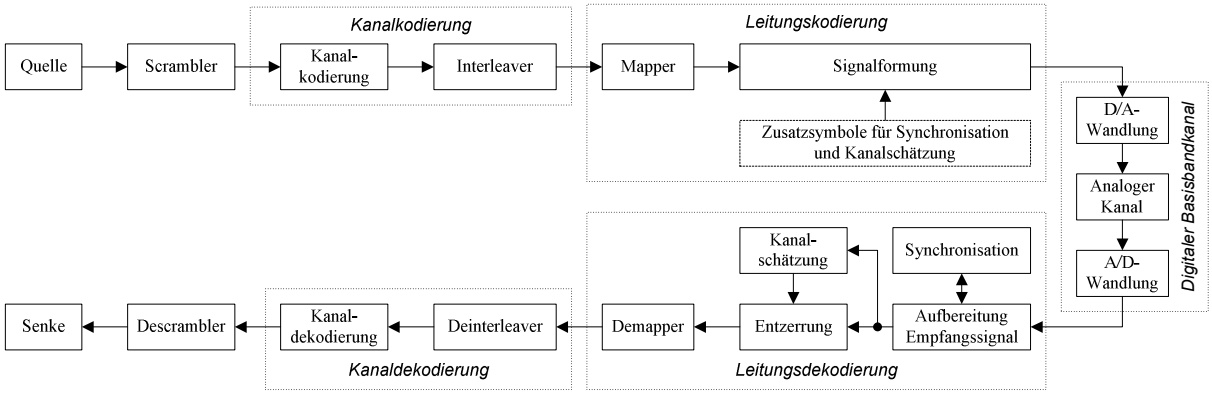

Abb. 2: Detaillierter schematischer Aufbau eines Kommunikationssystems

Während die beiden soeben eingeführten Strukturmodelle die reine Nachrichtenübertragung zwischen einer digitalen Quelle und einer digitalen Senke behandeln, werden die höheren Ebenen eines Kommunikationssystems in diesen Darstellungen nicht aufgegriffen. Sie sind im vorliegenden Modell Bestandteil der digitalen Quelle bzw. Senke. Auf den höheren Ebenen umfasst die Kommunikation unter anderem die Adressierung und die Fehlersicherung. In Mehrnutzerumgebungen bzw. bei mehreren unabhängigen Punkt-zu-Punkt-Verbindungen über einen gemeinsamen Kanal ist es notwendig, den beabsichtigten Empfänger der Nachricht zu spezifizieren. Fehlersicherung meint in diesem Zusammenhang nicht die bereits dargestellte Kanalkodierung zur Fehlerkorrektur, sondern das Sicherstellen einer einwandfreien Übertragung durch das Erkennen unkorrigierter Fehler. Gleichzeitig ermöglichen die höheren Ebenen unterschiedlichen Diensten den Zugriff auf den gleichen Kommunikationskanal.

Diese Ebenen können über das von der ISO[7] standardisierte OSI[8]-Schichtenmodell, auch als OSI-Referenzmodell bezeichnet, dargestellt werden. In Abb. 3 sind die 7 Schichten – *Layer* – aufgeführt, in die die Aufgaben der Kommunikation nach diesem Modell eingeteilt werden. Die Aufgabendefinition ist dabei allgemein gehalten und gibt keine konkrete Implementierung vor. So fassen manche Kommunikationssysteme die Aufgaben übereinander liegender Schichten des OSI-Referenzmodells auch in einer einzigen Schicht zusammen, beispielsweise im TCP/IP[9]-Referenzmodell [8].

[6] FFT: Fast-Fourier-Transform, schnelle Fouriertransformation
[7] ISO: International Organization for Standardization
[8] OSI: Open Systems Interconnetion
[9] TCP/IP: Transmission Control Protocol / Internet Protocol

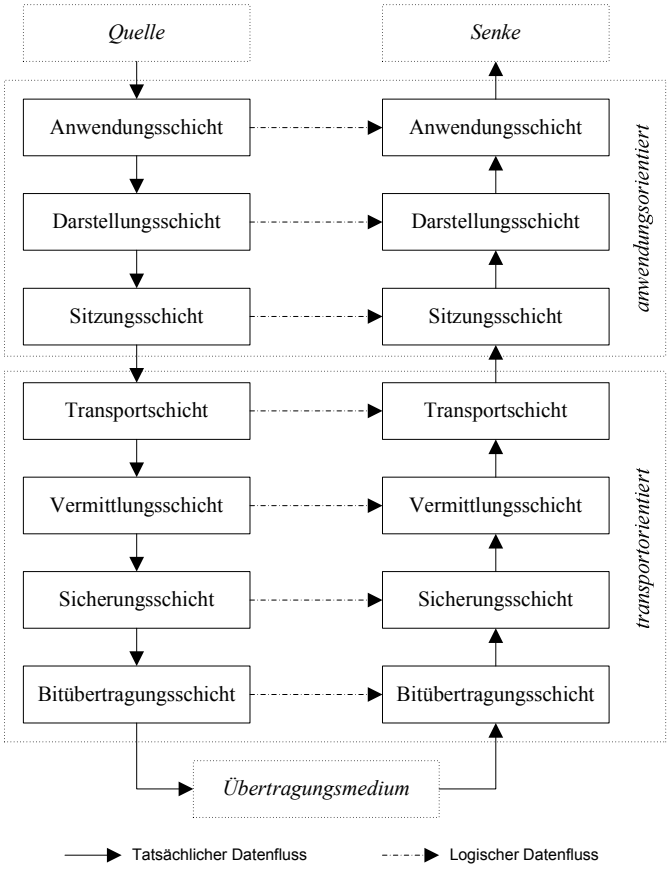

Abb. 3: OSI-Schichtenmodell

Die Bitübertragungsschicht – der *Physical Layer* oder kurz *PHY-Layer* – ist die unterste Ebene. Sie ermöglicht die Übertragung von Bits in Form von physikalischen Signalen. Der eigentliche Übertragungskanal, also das physikalische Medium, ist nicht Bestandteil dieser Schicht, sondern liegt darunter, da er nach der verwendeten Definition kein Bestandteil des Kommunikationssystems ist.

Die Sicherungsschicht – der *Data Link Layer* – soll eine fehlerfreie Datenübertragung zwischen Sender und Empfänger sicherstellen und den Medienzugriff regeln. Die Kanalkodierung, die Paketisierung des Datenstroms und die physikalische Adressierung gehören zu den Aufgaben dieser Schicht.

Die Vermittlungsschicht – der *Network Layer* – steuert den Datenfluss im gesamten Kommunikationsnetz durch die Bereitstellung von eindeutigen Netzwerkadressen und das Führen von Routingtabellen.

Die Transportschicht – der *Transport Layer* – bietet den darüber liegenden Schichten einen einheitlichen Zugriff auf die darunter liegenden.

Die vier zuvor genannten Schichten lassen sich zur Gruppe der transportorientierten Schichten zusammenfassen, während die drei höheren Schichten anwendungsorientiert sind.

Die fünfte Schicht – die Sitzungsschicht bzw. der *Session Layer* – stellt Dienste für einen synchronisierten Datenaustausch zur Verfügung und ermöglicht die Prozesskommunikation zwischen zwei Systemen.

Die Darstellungsschicht – der *Presentation Layer* – setzt eine systemabhängige Datendarstellung in eine unabhängige Form um. Damit wird der Datenaustausch zwischen Systemen mit unterschiedlichen Datendarstellungen ermöglicht. Weiterhin gehören zu dieser Schicht die Datenkompression und die Verschlüsselung.

Die Anwendungsschicht – der *Application Layer* – schließlich ermöglicht den Anwendungen und Diensten den Zugriff auf das Kommunikationsnetz.

Im OSI-Schichtenmodell sind die jeweils tiefer liegenden Schichten für die höheren transparent, d. h. der logische Datenfluss findet horizontal zwischen den gleichen Schichten statt, während der tatsächliche Datenfluss vertikal zwischen den einzelnen Schichten erfolgt.

Wie bereits aufgeführt, werden in der Praxis auch andere Modelle verwendet, die Aufgaben verschiedener Schichten zusammenfassen oder auf unterschiedliche Schichten verteilen. Im bereits erwähnten TCP/IP-Referenzmodell sind so die Bitübertragungs- und die Sicherungsschicht zur Netzzugangsschicht zusammengefasst, die oberen drei Schichten des OSI-Modells bilden im TCP/IP-Modell die vierte und höchste Schicht, die Anwendungsschicht.

Für die Analyse der Latenzzeiten in einem Kommunikationssystem können alle aufgeführten Modelle herangezogen werden. Der Fokus dieser Arbeit liegt auf den Latenzen der Basisbandverarbeitung, also den in der unteren Schicht des OSI-Referenzmodells beschriebenen Aufgaben. Deshalb wird das erste Modell nach Abb. 2 verwendet, welches die einzelnen Module detaillierter darstellt.

Die Basisbandverarbeitung, im Folgenden auch als PHY referenziert, umfasst dabei zusätzlich einen Teil der Aufgaben der Sicherungsschicht wie die Kanalkodierung, nicht jedoch die Paketisierung des Datenstroms, die Medienzugriffssteuerung und die physikalische Adressierung. Dies sind die Aufgaben der MAC-Schicht, einer Unterebene der Sicherungsschicht.

2.2 Kanalkodierung / -dekodierung

In realen Kanälen unterliegen die übertragenen Signale diversen Einflüssen, die Übertragungsfehler erzeugen können, z. B. Rauschen oder Fading[10]. Bereits 1948 zeigte Shannon in [9], dass es bei der Übertragung von Nachrichten über einen verrauschten Kanal durch das gezielte Hinzufügen von Redundanz möglich ist, die Fehlerrate bei der Übertragung beliebig

[10] Fading: Schwankungen der Empfangsfeldstärke durch Mehrwegeausbreitung oder Abschattung

zu reduzieren, sofern die Informationsrate unterhalb der Kanalkapazität liegt. Die Kanalkodierung als Verfahren zur Redundanzhinzufügung ist somit ein essentieller Bestandteil eines Kommunikationssystems. Es ist zwischen zwei unterschiedlichen Arten zu unterscheiden, den fehlererkennenden und den fehlerkorrigierenden Verfahren. Letztere beinhalten teilweise auch die Möglichkeit, zusätzliche nicht korrigierbare Fehler zu erkennen. Fehlererkennende Verfahren werden auch als Rückwärtsfehlerkorrektur (Backward Error Correction, BEC) und fehlerkorrigierende Verfahren als Vorwärtsfehlerkorrektur (Forward Error Correction, FEC oder auch Error Detection and Correction, EDAC) bezeichnet.

Bei der Rückwärtsfehlerkorrektur wird anhand der Redundanz erkannt, ob bei der Übertragung ein Fehler aufgetreten ist. In diesem Fall werden die Daten über eine entsprechende Signalisierung, z. B. über ein „Automatic Repeat Request“ (ARQ), erneut vom Empfänger beim Sender angefordert. Für diese erneute Anforderung ist offensichtlich ein Rückkanal notwendig. Ist dieser bei einer unidirektionalen Übertragung nicht vorhanden, muss entweder ein Vorwärtsfehlerkorrekturverfahren eingesetzt oder ganz auf eine Fehlerkorrektur verzichtet werden.

Ein einfaches Beispiel für eine Rückwärtsfehlerkorrektur ist das optionale Paritätsbit der RS-232-Schnittstelle [10]. Dieses Bit gibt an, ob in dem vorhergehenden 8-Bit-Datenwort eine gerade oder ungerade Anzahl von Einsen auftritt. Sofern der Empfänger aufgrund des Paritätsbits eine gerade Anzahl von Einsen im empfangenden Datenwort erwartet, dieses jedoch nur eine ungerade Anzahl von Einsen enthält, muss also ein Übertragungsfehler aufgetreten und mindestens ein Bit verfälscht worden sein. In diesem Fall muss der Empfänger über den Rückkanal das Datenwort erneut anfordern. Die Schwachstelle dieses einfachen Verfahrens ist sehr leicht erkennbar: Sollten zwei Bits verfälscht sein, stimmt die Parität der Einsen im empfangenen Datenwort wieder und der Empfänger kann somit keinen Übertragungsfehler erkennen. Allgemein gilt: „Bei binären Paritätscodes sind alle *n*-fachen Fehler mit *n* gerade nicht erkennbar.“ [11].

Abhilfe schafft die Erweiterung der Redundanz, z. B. durch einen zweidimensionalen Paritätscode wie der Kreuzparität. Hierfür werden die Datensymbole in einen Block zusammengefasst, der sich als Matrix darstellen lässt. Anschließend wird für jede Zeile und für jede Spalte ein Paritätsbit berechnet. Anhand dieser Quer- und Längsparität sind nun auch Doppelfehler innerhalb eines Datenwortes erkennbar, da ein Doppelfehler in Querrichtung aus zwei unabhängigen Einzelfehlern in Längsrichtung besteht. Gleichzeitig ermöglicht diese Erweiterung zusätzlich die Korrektur eines Einzelbitfehlers, da dieser über die Spalten- und Zeilenprüfung eindeutig lokalisiert werden kann.

Eine weitere Möglichkeit zur reinen Fehlererkennung ist die zyklische Redundanzprüfung (Cyclic Redundancy Check, CRC), basierend auf der Übertragung einer Prüfsumme. Details finden sich z. B. in [12]. Die Berechnung erfolgt dabei im Gegensatz zur Bezeichnung *Prüf-*

summe nicht durch eine Addition, sondern beruht auf einer Polynomdivision. Auch die bereits erwähnte Längsparitätsprüfung nutzt eine Prüfsumme in Form eines zusätzlichen Paritätswortes, das nach einer definierten Anzahl von Datenwörtern übertragen wird. Allerdings gilt für die Längsparitätsprüfung die gleiche Einschränkung wie für alle binären Paritätscodes, eine gerade Anzahl an Fehlern ist nicht erkennbar. Mit einer zyklischen Redundanzprüfung sind dagegen bei einer geeigneten Wahl des Prüfsummenalgorithmus alle Einbitfehler, jede ungerade Anzahl von verfälschten Bits, Bündelfehler bis zur Länge des Generatorpolynoms und eine Mehrzahl an möglichen Zweibitfehlern erkennbar [12], [13].

Die Vorwärtsfehlerkorrektur ermöglicht dem Empfänger, evtl. aufgetretene Übertragungsfehler anhand der Redundanz zu erkennen und zu korrigieren. Die Verfahren lassen sich in zwei Gruppen unterteilen. Die erste Gruppe arbeitet mit unabhängigen Datenblöcken fester Länge, die keinen Einfluss aufeinander haben. Die zweite Gruppe von Fehlerkorrekturverfahren unterteilt die Daten nicht in einzelne Blöcke, sondern kodiert einen fortlaufenden Datenstrom. Ein einfaches Beispiel für einen Blockcode ist die bereits aufgeführte Kreuzparität. Weitere Blockcodes mit besseren Korrektureigenschaften sind Hamming-Codes, Bose-Chaudhuri-Hocquenghem-Codes (BCH-Codes), Reed-Solomon-Codes (RS-Codes) und Low-Density-Parity-Check-Codes (LDPC-Codes). Faltungscodes, wie sie häufig in Mobilfunksystemen zum Einsatz kommen, sind ein Beispiel für auf kontinuierlichen Datenströmen arbeitende Verfahren. Details zu Hamming-Codes, BCH-Codes, RS-Codes, LDPC-Codes und Faltungscodes finden sich z. B. in [14] und [15].

Im Gegensatz zur Rückwärtsfehlerkorrektur werden bei der alleinigen Verwendung der Vorwärtskorrektur keine Übertragungswiederholungen durchgeführt. Gleichzeitig verringert sich jedoch durch die Redundanz die Übertragungsrate der Nutzdaten oder aber die für die Übertragung erforderliche Bandbreite steigt. Zum Beispiel erhöht sich bei dem einfachen Vorwärtsfehlerkorrekturverfahren mit Verdreifachung des zu sendenden Bits und einer Mehrheitsentscheidung am Empfänger die notwendige Bandbreite um den Faktor drei. Bei gleichbleibender Bandbreite des übertragenen Signals verringert sich die nutzbare Datenrate auf ein Drittel der Datenrate ohne Fehlerkorrektur.

Aufgrund der Arbeitsweise lassen sich die Vorwärts- und Rückwärtsfehlerkorrekturverfahren auf unterschiedlichen Schichten im OSI-Schichtenmodell einordnen. Die Vorwärtskorrektur erfordert kein spezielles Protokoll zur Steuerung der Fehlererkennung und -korrektur. Sie kann in die unterste Schicht, den *PHY-Layer* bzw. die PHY-Schicht, einsortiert werden. Bei der Rückwärtskorrektur ist dagegen der Austausch von weiteren Nachrichten über einen Rückkanal notwendig, für den bei einer bidirektionalen Übertragung der gleiche Übertragungskanal verwendet werden kann. Die für diese Anforderung einer Übertragungswiederholung notwendigen Protokollschritte sind Aufgabe der MAC-Schicht. Die reine Fehlererkennung kann unabhängig davon auch Bestandteil der PHY-Schicht sein. In diesem Fall würde

an der PHY-MAC-Schnittstelle über eine spezielle Signalisierung das Auftreten eines Übertragungsfehlers gemeldet.

Speziell im drahtgebundenen Umfeld werden für die digitale Übertragung aufgrund der geringen Bitfehlerwahrscheinlichkeiten im Bereich von 10^{-5} bis 10^{-12} [16] Systeme ohne Vorwärtsfehlerkorrektur verwendet, z. B. bei USB[11] und Firewire [17]. Durch die nur selten notwendigen wiederholten Übertragungen wird der gesamte Datendurchsatz geringer verschlechtert, als dies durch ein Vorwärtskorrekturverfahren mit fester Verkleinerung der Datenrate (bei gleichbleibender Bandbreite) möglich wäre. Zugleich erhöhen Vorwärtskorrekturverfahren den Ressourcen- und / oder Implementierungsaufwand des Systems gegenüber einer einfacheren Prüfsummenberechnung und gleichzeitig die Verarbeitungszeit des eingehenden Datenpakets.

In typischen drahtlosen Systemen werden dagegen aufgrund der höheren Bitfehlerwahrscheinlichkeiten die Verfahren der Vorwärts- und Rückwärtsfehlerkorrektur miteinander kombiniert. Erstere stellen dabei eine möglichst gute Datenübertragung mit einer geringen Restfehlerwahrscheinlichkeit sicher. Eine zusätzliche Prüfsumme verringert die nutzbare Kapazität nur geringfügig, sodass anhand dieser die Fehlerfreiheit der vorwärtsfehlerkorrigierten Daten getestet werden kann. Sofern die Prüfung ein negatives Ergebnis hat, werden die entsprechenden Daten mit dem Rückwärtskorrekturverfahren erneut angefordert. Eine weitere gebräuchliche Erweiterung ist die zusätzliche Verwendung einer positiven Empfangsbestätigung. Es wird nicht nur beim Empfang eines fehlerhaften Paktes ein ARQ ausgelöst, sondern stattdessen wird jedes fehlerfreie Paket explizit mit einem ACK[12] durch den Empfänger beim Sender bestätigt. Damit wird gewährleistet, dass auch verloren gegangene Pakete, z. B. durch ein Fehlschlagen der Framesynchronisation, erneut übertragen werden können.

Je nach den Anforderungen an das System wird bei der Vorwärtsfehlerkorrektur nicht nur ein einzelnes Verfahren, sondern eine Kombination mehrerer verwendet. So findet sich in Funkübertragungen, insbesondere bei der Satellitenkommunikation, häufig eine Kombination aus innerem Faltungscode und äußerem RS-Code. Bei der Viterbi-Dekodierung des Faltungscodes entstehen bei einer fehlerhaften Dekodierung Bündelfehler. Die Korrektur solcher Bündelfehler ist die Stärke des RS-Codes, sodass sich aus der Kombination beider Verfahren eine bessere Fehlerkorrekturleistung im Vergleich zur Leistung der beiden Verfahren an sich ergibt. [15]

Sowohl Faltungs- als auch Blockcodes erzeugen bei der Verarbeitung einen signifikanten Beitrag zur Latenzzeit der gesamten Basisbandverarbeitung. Den größeren Anteil hat dabei die Dekodierung, während die Kodierung meistens so gut wie keinen Einfluss hat. So muss beim Faltungskodierer der eingehende Datenstrom eine Kette von wenigen Flipflops durchlaufen.

[11] USB: Universal Serial Bus
[12] ACK: Acknowledge, positive Empfangsbestätigung

Auch bei Blockkodierern können die einzelnen Bits bzw. Bytes des Blocks je nach Implementierung den Kodierer als Datenstrom durchlaufen. Bei einem Paritätscheckcode werden die Paritätsbits an das Nachrichtenwort angehängt und beim RS-Code die berechneten Prüfsymbole. Sofern für die Kodierung der letzten Stellen eines Datenblocks Informationen über die ersten, bereits ausgegebenen Stellen des Datenblocks benötigt werden, können diese im Kodierer intern gespeichert werden.

Bei der Blockdekodierung ist diese Vorgehensweise nicht möglich. Da nicht nur Fehler erkannt, sondern auch korrigiert werden sollen, muss erst der gesamten Codeblock vorliegen, bevor die Fehlerbestimmung erfolgen kann. Arbeitet die digitale Verarbeitung auf einem N_p-parallelen Datenstrom mit N_p als Anzahl der parallel übertragenen Bits und besteht ein Codeblock aus N_{block} Datenbits mit $N_{block} > N_p$, sind

$$N_{takt} = \left\lceil \frac{N_{block}}{N_p} \right\rceil \tag{1}$$

Takte notwendig, bevor die Dekodierung starten kann. Es entsteht somit für das Einschreiben des Datenblocks mit der Taktrate f_{clk} bereits eine Verzögerungszeit von:

$$T_{wd} = \frac{N_{takt}}{f_{clk}} \tag{2}$$

Unter der Voraussetzung gleicher Schnittstellen am Ein- und Ausgang des Dekodierers entsteht die gleiche Verzögerungszeit beim Auslesen des Blocks, gekürzt um die Anzahl der nicht ausgegebenen redundanten Prüfbits N_{chkbit}:

$$T_{rd} = \frac{N_{block} - N_{chkbit}}{N_p \cdot f_{clk}} \tag{3}$$

Der zweite Beitrag zur Gesamtverzögerung bei der Blockverarbeitung entsteht durch die Dekodierung an sich. Mit einfachen Verfahren wie der Kreuzparität lässt sich dieser Aufwand stark reduzieren. Die Paritätsberechnung in Längs- und Querrichtung kann bereits parallel mit dem Einschreiben erfolgen, sodass zeitgleich mit dem blockabschließenden Längsparitätswort über einen Vergleich die fehlerhaften Bits erkannt werden können. Algorithmen mit besseren Korrektureigenschaften, wie z. B. RS-Codes, erzeugen dagegen einen höheren Berechnungsaufwand für die Dekodierung. So erfordert die Dekodierung von RS-kodierten Datenblöcken das Lösen umfangreicher Gleichungssysteme. Infolgedessen stellt die Dekodierungszeit einen großen Anteil an der Gesamtverzögerung. Da es sich bei RS-Codes um systematische Codes handelt, bietet sich aufgrund des hohen Anteils der reinen Verarbeitungszeit an der Gesamtverzögerung das neu entwickelte Verfahren der *spekulativen Dekodierung* für die Latenzverringerung an. Dieses Verfahren wird im Abschnitt 7.3 ausführlich erläutert.

Wie bereits ausgeführt, arbeitet die Faltungskodierung nicht auf Blöcken fester Länge, sondern auf einem kontinuierlichen Datenstrom. Für die Dekodierung wird üblicherweise die Maximum-Likelihood-Dekodierung mit einem Viterbi-Dekodierer verwendet. Auch hier muss eine Sequenz von Eingangswerten im Dekodierer vorliegen, bevor dieser über das erste dekodierte Bit entscheiden kann. Es entstehen also auch hier neben der Verarbeitungszeit Verzögerungen durch Wartezeiten.

2.3 Interleaver

Typische Störungen in realen Kanälen erzeugen meist Bündelfehler, d. h. sie verfälschen mehrere zeitlich aufeinanderfolgende Symbole bzw. Datenbits. Im Falle einer Einzelträgerübertragung könnte dies ein zeitlich begrenzter Störimpuls durch Fading sein, bei einer OFDM-Übertragung dagegen die Störung mehrerer benachbarter Unterträger durch einen Störsender im betreffenden Frequenzbereich. Während bei einer statischen bzw. deterministischen Störquelle die Störung durch Auslassen des entsprechenden Zeitschlitzes bzw. der Unterträger vermieden werden kann, ist dies bei zufälligen Störeinwirkungen nicht möglich. Auch bei einer deterministischen Störung kann das Auslassen des gestörten Zeitschlitzes bzw. Unterträgers unmöglich sein, wenn wie z. B. bei der digitalen Rundfunkausstrahlung kein Rückkanal für eine entsprechende Signalisierung zur Verfügung steht.

Im vorhergehenden Abschnitt wurde bereits ausgeführt, dass RS-Codes gut geeignet sind, um Bündelfehler zu korrigieren. Andere Codes, wie z. B. Faltungscodes, sind dagegen anfällig gegenüber einer Fehlerfolge [15], sie eignen sich hauptsächlich für die Korrektur von Einzelfehlern. Daher werden beim Einsatz solcher Kanalkodierungen zusätzlich Interleaver verwendet. Diese ändern die Reihenfolge der kodierten Datenbits bzw. Symbole entsprechend eines festgelegten Schemas. Vor der Kanaldekodierung wird das Vertauschen mit einem Deinterleaver zurückgenommen. Sofern auf dem Kanal ein Bündelfehler auftrat, ist dieser nun in Abhängigkeit von dem verwendeten Schema auf mehrere einzelne Stellen im Datenstrom verteilt. Für die Dekodierung stellen sich die verfälschten Stellen als verteilte Einzelbitfehler dar. Allgemein lässt sich der Interleaver durch die Funktion $y(n) = x(a(n))$ beschreiben. $a(n)$ ordnet jedem Index n_i der Eingangsfolge $x(n)$ eineindeutig einen neuen Index n_o der Ausgangsfolge $y(n)$ zu. Der Deinterleaver stellt durch die Funktion $z(a(n)) = y(n)$ die Ursprungsreihenfolge wieder her. Wie zum Ende dieses Abschnitts beim Faltungsinterleaver aufgezeigt wird, kann die Ausgangsfolge $y(n)$ des Interleavers länger als dessen Eingangsfolge sein. Die zusätzlichen Werte werden im Deinterleaver bei der Wiederherstellung der ursprünglichen Folge eliminiert.

Eine einfache und sehr gebräuchliche Art von Interleavern ist der Blockinterleaver, dessen schematischer Aufbau in Abb. 4 dargestellt ist. Er besteht aus einem Speicher der Breite N_c und der Tiefe N_r. Ein Block mit der Blockgröße $N_{block} = N_c \cdot N_r$ wird reihenweise in den Spei-

cher eingeschrieben und anschließend spaltenweise ausgelesen. Mit $N_c = N_r = 3$ ergibt sich bei der Eingangsfolge x(n) = 1 2 3 4 5 6 7 8 9 eine Ausgangsfolge y(n) = 1 4 7 2 5 8 3 6 9. Die Interleavingtiefe, also der minimale Abstand benachbarter Eingangswerte, beträgt 3. Diese wird durch die Anzahl an Reihen festgelegt, während die Anzahl der Spalten den minimalen Abstand zweier benachbarter Ausgangssymbole angibt.

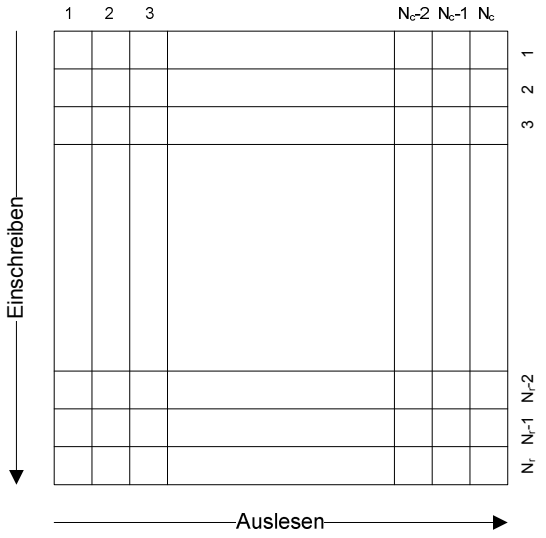

Abb. 4: Prinzip eines Blockinterleavers

Damit Bündelfehler vollständig in Einzelfehler umgewandelt werden können, muss die Reihenanzahl größer oder gleich der Länge des Bündelfehlers sein. Der Abstand der resultierenden Einzelfehler wird durch die Spaltenanzahl festgelegt und sollte groß genug sein, damit der Dekodierer die Einzelfehler noch korrigieren kann. Durch Permutationen von Spalten und Zeilen lassen sich bei gleichem Aufbau andere Interleavingsequenzen erreichen. Wird ein Speicher mit 2 Spalten und 6 Reihen verwendet, ist die Interleavingtiefe 6, die Ausgangsreihenfolge lautet y(n) = 1 3 5 7 9 11 2 4 6 8 10 12. Der minimale Abstand der aus einem Bündelfehler erzeugten Einzelfehler beträgt 2. Werden die 6 Reihen in der Reihenfolge 1 4 2 5 3 6 beschrieben, ergibt sich die Ausgangsfolge y(n) = 1 5 9 3 7 11 2 6 8 10 12. Der minimale Abstand beträgt nun 4.

Während bei der Verwendung eines üblichen synchronen Speicherblocks in einem ASIC[13] oder FPGA[14] die Daten bitparallel entsprechend der Breite eines Speicherwortes eingeschrieben werden können, ist aufgrund der Reihenadressierung ein spaltenweises Auslesen innerhalb eines Taktes nicht möglich. Um gegenüber einer bitseriellen Ausgabe den Durchsatz zu vergrößern, wird der Interleaver N_p-fach parallel aufgebaut. Die Eingangsdaten werden stets parallel in die gleiche Reihe der N_p Speicherblöcke eingeschrieben. Beim Auslesen werden dagegen unterschiedliche Adressen verwendet. Aus den N_p Ausgabebits eines jeden Speicherblocks wird dann jeweils genau ein Bit ausgewählt, um ein N_p-bit-breites Ausgabedatenwort

[13] ASIC: Application-Specific Integrated Circuit
[14] FPGA: Field-Programmable Gate Array

zu formen. Prinzipiell ist mit dieser Variante durch eine geeignete Adresssteuerung jedes beliebige auf einem Block basierende Vertauschungsschema realisierbar. Gleichzeitig vervielfacht diese Lösung jedoch den Speicherbedarf des Interleavers um den Faktor N_p und erhöht den Verwaltungsaufwand sowie die Ressourcenanforderungen, unter anderem durch die Notwendigkeit von großen Multiplexern zum Zusammensetzen des Ausgangsdatenwortes. Das im Kapitel 4 vorgestellte EASY-A Beispielsystem verwendet eine solche Variante mit einem konfigurierbaren 32-bit-parallelen Interleaver zur Sicherstellung des Durchsatzes von rund 4 Gbit/s.

Eine Unterart des Blockinterleavers ist der pseudozufällige Blockinterleaver, bei dem die Daten sequentiell in einen eindimensionalen Speicher eingeschrieben und entsprechend einer pseudozufälligen Reihenfolge wieder ausgelesen werden.

Es ist offensichtlich, dass durch die Verwendung von Interleavern zusätzliche Verzögerungszeit in das System eingebracht wird. Bei dem vorgestellten Blockinterleaver muss erst ein gesamter Datenblock der Größe N_{block} eingeschrieben werden, bevor die Ausgabe beginnen kann. Die Verzögerung beträgt $T_v = N_{block} \cdot T_b$ mit T_b als der Taktperiode für das sequentielle Einschreiben und Auslesen. Da bei einem sequentiellen Einschreiben das Auslesen bereits beginnen kann, wenn die erste Zelle der letzten Reihe beschrieben wurde, beträgt die Verzögerungszeit genau genommen nur $T_v = (N_c \cdot (N_r - 1) + 1) \cdot T_b$. Bei einem parallelen Einschreiben und Auslesen ist dies nicht mehr möglich. Mit dem Parallelitätsfaktor N_p verkürzt sich die notwendige Zeit auf:

$$T_v = \frac{N_{block}}{N_p} \cdot T_b \tag{4}$$

Faltungsinterleaver, deren Aufbau beispielhaft in Abb. 5 dargestellt ist, sind eine effizientere Art von Interleavern. Analog zu Faltungskodierern arbeiten sie nicht auf Blöcken fester Länge, sondern auf einem kontinuierlichen Eingangsstrom. Der eingehende Datenstrom wird reihum auf eine definierte Anzahl unterschiedlich langer Schieberegister verteilt, die somit alle eine unterschiedliche Verzögerungszeit haben. Der erste Pfad enthält dabei keine Verzögerung, für jeden folgenden nimmt die Anzahl der Verzögerungsschritte um einen konstanten Wert zu. Aus der Eingangsfolge x(n) = 1, 2, 3, ..., 13 ergibt sich für den dargestellten Faltungsinterleaver die Ausgangsfolge y(n) = 1 X X X 5 2 X X 9 6 3 X 13 10 7 4. X steht hierbei für den Inhalt der Schieberegister vor dem Einschreiben der Datenfolge. Die Struktur der Ausgangsdaten weist nicht mehr die einfach erkennbare Regularität wie beim Blockinterleaver auf. Der im Empfänger notwendige Faltungsdeinterleaver lässt sich durch eine horizontale Spiegelung des Faltungsinterleavers aufbauen.

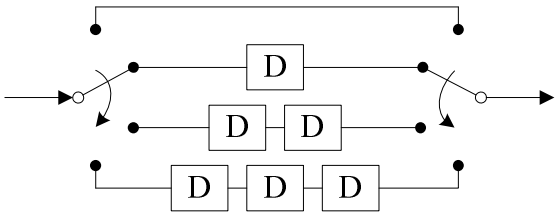

Abb. 5: Faltungsinterleaver mit 4 Pfaden

Die Gesamtverzögerung aus Faltungsinterleaver und -deinterleaver mit N_p parallelen Pfaden und einer maximalen Speichertiefe von (N_p-1) ergibt sich zu $N_p \cdot (N_p-1) \cdot T_b$. Der notwendige Speicherbedarf für den Interleaver beträgt $N_p \cdot (N_p-1)/2$, die Interleavertiefe N_p+1. Ein Blockinterleaver mit der gleichen Interleavertiefe würde in etwa den doppelten Speicherbedarf und die doppelte Verzögerungszeit erfordern.

Eine triviale Möglichkeit zur Verringerung der durch einen Interleaver erzeugten Latenz ist der Verzicht auf dieses Modul. Abhängig von den Eigenschaften des Übertragungskanals und der gewählten Kanalkodierung hat dies jedoch deutliche Auswirkungen auf die Performance der Datenübertragung im Hinblick auf die Fehlerrate. Trotzdem kann eine solche Vorgehensweise sinnvoll sein, wenn für eine schnellere Ankunft der Datenpakete eine höhere Paketfehlerrate in Kauf genommen werden kann. Ein Beispiel hierfür ist die Fastpath-Option bei ADSL[15]-Anschlüssen [18]. ADSL verwendet ein OFDM-basiertes Übertragungssystem, um im nicht genutzten Frequenzbereich von kupfernen Telefonleitungen Daten mit hoher Geschwindigkeit zu übertragen. Um eine möglichst fehlerfreie Übertragung zu gewährleisten, wird ein Interleaver verwendet. Dieser erhöht die Gesamtzeit für die Durchführung eines ICMP[16]-Echo-Requests[17] [19] von 15-25 ms auf ca. 60 ms [20],[21]. Für das Herunterladen von Daten aus dem Internet ist diese Verzögerung irrelevant, da keine Echtzeitanforderungen bestehen und der gesamte Datendurchsatz durch die Verringerung der Paketfehlerrate steigt. Bei Onlinespielen und der Internettelefonie dagegen werden keine großen Datenpakete übertragen, stattdessen bestehen hier hohe Echtzeitanforderungen. Es ist unerheblich, ob ein Paket fehlerhaft oder zwar fehlerfrei, aber zu spät ankommt. Das Hauptaugenmerk liegt auf der schnellen Zustellung der Pakete. Mit der Fastpath-Option wird die Verwendung des Interleavers abgeschaltet. Die Paketlaufzeiten werden somit auf Kosten der Paketfehlerrate verringert.

Doch auch wenn es nicht möglich ist, die Laufzeit auf Kosten der Fehlerrate zu verringern, sondern bei annähernd gleichbleibender Paketfehlerrate die Latenzzeit verringert werden soll, kann dies durch einen Verzicht auf den Interleaver erfolgen. Durch das Ausnutzen von Parallelität lässt sich ein implizites Interleaving durchführen. Implizit bedeutet, dass kein dedizierter Interleaver verwendet wird. Stattdessen nutzt ein implizites Interleaving die Parallelität der Kanalkodierung aus, um Bündelfehler in Einzelfehler umzuwandeln. Ohne einen zusätzlichen

[15] ADSL: Asymmetric Digital Subscriber Line
[16] ICMP: Internet Control Message Protocol
[17] umgangssprachlich: „Ping“

Beitrag zur Latenzzeit durch einen Interleaver wird also die Paketfehlerrate verbessert bzw. bei gleichbleibender Paketfehlerrate die Latenzzeit reduziert. Das Verfahren des impliziten Interleavings wird im Abschnitt 6.4 detailliert vorgestellt und analysiert.

2.4 Framesynchronisation

Zum Aufbau der Kommunikationsverbindung zwischen Quelle und Senke ist es notwendig, dass der Senke bekannt ist, wann die Quelle Nachrichten sendet, d. h. Quelle und Senke müssen sich synchronisieren. Insbesondere bei der paketorientierten Übertragung, wie z. B. im Ethernet, ist dies offensichtlich. Die einzelnen Datenpakete haben kein festes Zeitraster, sondern werden je nach Bedarf abgesendet. Eine Senke muss also erkennen, ob der Kanal frei ist oder ob ein neues Paket empfangen werden soll.

Zur Synchronisation kann ein zweiter dedizierter Kanal zwischen Empfänger und Sender verwendet werden, üblich ist jedoch die Verwendung einer Synchronisationssequenz, anhand derer die Senke den Beginn des neuen Datenpakets erkennen kann. Im einfachsten Fall handelt es sich um eine Pegeländerung wie bei RS-232 [10]. Hier hat das Startbit den inversen Pegel des Stopbits bzw. des Ruhezustandes. Eine andere Möglichkeit ist eine bewusste Codeverletzung, wie z. B. zur Rahmensynchronisation bei ISDN[18] [22]. Im Bereich der drahtlosen Kommunikation werden hauptsächlich lange Bit- bzw. Symbolsequenzen mit guten Korrelationseigenschaften verwendet, wie sie z. B. komplementäre Golay-Sequenzen [23] bieten. Gute Korrelationseigenschaften bedeutet, dass die Korrelationswerte außerhalb des Maximums möglichst klein sind. Im Empfänger kann dann durch die Kenntnis der gesendeten Sequenz über einen Korrelationsfilter der Frameanfang bestimmt werden. Dieser Korrelationsfilter kann entweder auf einer Kreuzkorrelation mit der bekannten Sequenz oder aber auf einer Autokorrelation des Empfangssignals selbst basieren.

Auch im Fall einer kontinuierlichen Datenstromverbindung bzw. bei einer Verbindung mit einem festen Zeitraster ist eine Synchronisation sowohl für den initialen Verbindungsaufbau als auch während der bestehenden Verbindung notwendig. Über die fortlaufende Synchronisation wird sichergestellt, dass sich die Senke in einem Fehlerfall mit Verlust der Synchronisation wieder selbstständig auf den Datenstrom synchronisieren kann. Essentiell wichtig ist dies insbesondere in unidirektionalen Übertragungssystemen, z. B. beim Betrieb eines Monitors. Basierend auf dem ursprünglichen VGA[19]-Standard [24] zur analogen Signalübertragung zwischen Grafikkarte und Bildschirm werden auch bei den digitalen Schnittstellen wie HDMI[20] [25] zusätzlich zur Bildinformation zwei Synchronisationsimpulse – H-Sync und V-Sync – zur Zeilen- und Bildsynchronisation übertragen. Anhand dieser Informationen kann der Moni-

[18] ISDN: Integrated Services Digital Network
[19] VGA: Video Graphics Array
[20] HDMI: High-Definition Multimedia Interface

tor den Beginn eines Bildes erkennen und sich bei einer Störung der Datenübertragung wieder auf den Bildbeginn synchronisieren.

Synchronisation umfasst allerdings nicht nur das (Wieder-) Erkennen des Kommunikationsstarts, sondern auch die Taktwiederherstellung bzw. die Bitsynchronisation. Bei vielen drahtgebundenen Datenübertragungssystemen wird der Sendetakt über eine dedizierte Leitung mitgeliefert. Dieser Sendetakt wird dann im Empfangsteil der Senke zur Abtastung der Daten verwendet, wenn nicht gar das gesamte System mit diesem Takt operiert. Bei einer drahtlosen Übertragung ist eine solche parallele leitungsgebundene Übertragung des Taktsignals sehr unüblich. Quelle und Senke verfügen über verschiedene Taktquellen für die Abtastung der Daten. Diese Taktquellen können – und sollten – zwar nominell den gleichen Takt aufweisen, werden in einem realen System aber trotzdem einen bestimmten, teilweise zeitvarianten, Unterschied aufweisen. Ursache hierfür sind zum einen die fertigungsbedingten Toleranzen (bei Quarzoszillatoren typischerweise einige ppm[21]), zum anderen eine Temperaturdrift bei unterschiedlicher Erwärmung und der Jitter, auch als Phasenrauschen bezeichnet. Somit muss die Senke ihre Abtastrate auf den ursprünglichen Sendetakt einstellen, um den optimalen Abtastzeitpunkt zu treffen. Wird bei einer drahtgebundenen Übertragung der Sendetakt nicht separat mitgeliefert, ist auch bei dieser eine Taktsynchronisation notwendig.

Zur Takt- bzw. Bitsynchronisation gibt es verschiedene Verfahren mit unterschiedlich hohem Aufwand, z. B. eine alternierende Bitfolge als Synchronisationsmuster wie bei Ethernet [26] oder die direkte Taktwiederherstellung aus dem Datensignal mittels einer PLL[22] ohne Verwendung einer dedizierten Synchronisationssequenz [27]. Zur direkten Taktwiederherstellung aus dem Datensignal muss sichergestellt sein, dass im laufenden Datenstrom genug Flanken-, also Bitwechsel auftreten. Dies kann unter anderem über eine spezielle Art der Kodierung oder einen Scrambler (siehe Abschnitt 2.7) erfolgen.

Die soeben angeführte Taktwiederherstellung bzw. -synchronisation für die Abtastzeitpunkte ist nur bei einer Basisbandübertragung ausreichend. Sofern das digitale Basisbandsignal noch für die (drahtlose) Übertragung moduliert, d. h. in einen höheren Frequenzbereich transferiert wird, ergibt sich eine neue Anforderung für die Senke, die Trägerwiederherstellung bzw. -synchronisation. Auch die zur Erzeugung des Trägersignals verwendeten Referenzsignale unterliegen den genannten Abweichungen, sodass die zur Demodulation verwendete Frequenz von der ursprünglichen Modulationsfrequenz abweichen kann. Bei OFDM-Systemen ist es üblich, die Frequenzabweichung im Empfänger zu schätzen und diese im digitalen Basisband zu korrigieren.

Bei drahtgebundenen Systemen liegt zwischen Quelle und Senke ein Medium mit definierter Länge und Eigenschaften, z. B. ein Kupferdraht oder eine Glasfaser. Die Länge des Mediums

[21] ppm: parts per million
[22] PLL: Phase-locked Loop

ist fest und somit ist die Dämpfung des Signals, die es auf der Leitung erfährt, bekannt bzw. kann bei der Inbetriebnahme ausgemessen werden. Bei drahtlosen Systemen ist dies nicht gegeben. Hier sind die einzelnen Kommunikationsteilnehmer nicht über eine feste Leitung miteinander verbunden und können demzufolge ihre Entfernung und Position zueinander verändern. In einem drahtlosen System mit mobilen Teilnehmern ist die Dämpfung des Signals während der Betriebszeit also nicht mehr konstant, sondern abhängig von der Entfernung der Teilnehmer und evtl. auftretender Fadingeffekte. Eine für die Signalverarbeitung optimale Aussteuerung des Empfängers ist selbst mit dem während der Systeminbetriebnahme erworbenen Wissen über die initiale Dämpfung nicht mehr gegeben.

Zur Abhilfe kann eine automatische Verstärkungsregelung (automatic gain control, AGC) verwendet werden. Bei dieser handelt es sich um eine Regelschleife, die über eine variable Verstärkung am Schleifenausgang einen konstanten mittleren Ausgangspegel unabhängig vom Eingangspegel sicherstellt. [28]

Alle aufgeführten Punkte, wie die Framesynchronisation, die Takt- und Bitsynchronisation inkl. der Trägerfrequenzabweichungsbestimmung und / oder -kompensation sowie die Aussteuerung einer AGC, können über eine Präambel realisiert werden, die dem eigentlichen zu sendenden Signal vorangestellt wird. Dabei kann es sich um eine relativ kurze Sequenz wie der angesprochenen alternierenden Bitfolge bei Ethernet oder aber über eine im Vergleich dazu sehr lange Sequenz in einem drahtlosen System handeln.

Durch Hinzufügen einer geeigneten Sequenz lässt sich im Rahmen der Framesynchronisation auch eine referenzdatengestützte Kanalschätzung vornehmen. Insbesondere bei einer drahtlosen Verbindung ist es notwendig, die Kanalschätzung nicht nur einmalig bei der Initialisierung, sondern in regelmäßigen Abständen bzw. für jeden Frame durchzuführen. Der drahtlose Kanal kann meist nicht als zeitinvariant betrachtet werden. Weiterhin sind, wie schon bei der AGC erläutert, durch eine andere Entfernung und Position der Kommunikationsteilnehmer zueinander deutliche Änderungen in den Kanaleigenschaften möglich. Auf die Kanalschätzung wird zusammen mit der Kanalkorrektur im Abschnitt 2.5 eingegangen.

Es ist offensichtlich, dass die Verwendung einer Präambel Einfluss auf den maximal möglichen Datendurchsatz sowie die entstehenden Latenzzeiten hat. Insbesondere die relativ langen Präambelsequenzen in vielen drahtlosen Systemen haben einen nicht zu vernachlässigenden Einfluss durch den entstehenden Overhead. Die Präambelsequenzen verringern zum einen die Zeit, die für die reine Datenübertragung genutzt werden kann. Zum anderen entsteht bei der zusätzlichen Verwendung einer Präambel eine höhere Verzögerungszeit zwischen der Anforderung der MAC-Schicht der Quelle, Daten zu senden und dem Empfang dieser Daten durch die Senke. Auch wenn die notwendige Verarbeitungszeit sowie die Laufzeiten vernachlässigt werden können, ergibt sich eine minimale Verzögerung um die Präambellänge.

Typischerweise werden für die Frameerkennung sehr niedrige Schwellwerte verwendet. Ein nicht erkannter Frame führt in jedem Fall zu einem Datenverlust und somit zu einem Performanceeinbruch, während ein Falschalarm, also das fehlerhafte Erkennen eines Frames, ohne dass ein solcher gesendet wurde, bei der ersten Datenintegritätsprüfung im Empfänger auffällt. Die Erkennungsrate wird somit auf Kosten einer steigenden Falschalarmrate gesteigert. In der praktischen Umsetzung hat eine zu hohe Falschalarmrate durch die Latenz in der Verarbeitung jedoch wieder einen negativen Einfluss auf die Erkennungsrate. Nach dem Erkennen eines Frames wird die Frameerkennung gestoppt und erst nach der fehlgeschlagenen Datenintegritätsprüfung oder der vollständigen Frameverarbeitung wieder gestartet. Ein Verzicht auf das Stoppen der Frameerkennung ist im Allgemeinen nicht möglich, da für die Präambel- und die Datenverarbeitung unterschiedliche Pfade genutzt werden. Sofern die Framerkennung nicht gestoppt wird, könnte bei einem korrekt erkannten Frame ein Falschalarm während des Datenempfangs ausgelöst werden, aufgrund dessen die Daten dann in den falschen Verarbeitungspfad überführt werden. Erfolgt bei dieser Vorgehensweise vor dem richtigen Frame ein Falschalarm, wird der eigentliche Frame nicht mehr erkannt.

2.5 Kanalschätzung / Kanalentzerrung

In realen Kanälen unterliegen die Sendesignale zahlreichen Störeinflüssen. Die Kanäle sind meist frequenzselektiv, und aus den resultierenden linearen Verzerrungen entstehen Symbolinterferenzen (ISI[23]). Damit diese im Empfänger durch einen Entzerrer korrigiert werden können, muss der Empfänger die Kanalimpulsantwort bzw. die Übertragungsfunktion des Kanals kennen. Anhand dieser können die Parameter des Entzerrers optimal eingestellt werden. Auf die Vielfalt an Methoden zur Kanalschätzung und Kanalentzerrung wird hier nicht weiter eingegangen, stattdessen sei auf die einschlägige Fachliteratur wie [29] verwiesen. Je nach verwendetem Algorithmus ergibt sich ein signifikanter Anteil an der Gesamtlatenz durch die notwendigen Verarbeitungsschritte.

Ist der Übertragungskanal zwar frequenzselektiv, aber zeitinvariant, reicht eine einmalige Ermittlung der Kanalübertragungsfunktion während der Initialisierungsphase aus. Bei zeitvarianten Kanälen ist diese dagegen periodisch zu wiederholen. Im vorhergehenden Abschnitt zur Framesynchronisation wurde bereits erwähnt, dass durch das Hinzufügen einer geeigneten Sequenz zur Präambel diese für eine referenzdatenbasierte Schätzung der Kanalübertragungsfunktion verwendet werden kann. Ist die maximale Framedauer kürzer als die Kohärenzzeit des Kanals, reicht die einmalige Kanalschätzung für jeden Empfangsframe aus. Bei einer kürzeren Kohärenzzeit muss zwischenzeitlich eine erneute Schätzung erfolgen. Für eine referenzdatengestützte Kanalschätzung werden in diesem Fall oftmals sogenannte Midambeln als Zwischensequenzen im Datenstrom eingefügt. Wie schon bei der Framesynchronisation entsteht unabhängig von der Verarbeitungszeit ein zusätzlicher Latenzbeitrag mit der akkumu-

[23] ISI: Intersymbolinterferenz

lierten Gesamtlänge der Midambeln für die Datenverarbeitung der auf die Midambeln folgenden Datenbereiche des Frames.

Im Allgemeinen kann für die Unterträger eines OFDM-Systems ein flacher Frequenzgang angenommen werden, d. h. das Empfangssignal eines Unterträgers ergibt sich aus der Multiplikation des Sendesignals des entsprechenden Unterträgers mit einer komplexen Konstante [30]. Im Gegensatz zur Kanalschätzung kann der Beitrag der Kanalkorrektur somit bei der Latenzanalyse eines OFDM-Systems vernachlässigt werden.

2.6 Fast Fourier Transform

Der FFT als Algorithmus zur schnellen Berechnung einer diskreten Fouriertransformation kommt insbesondere in OFDM-Systemen eine besondere Bedeutung zu. Mithilfe einer inversen FFT werden die im Frequenzbereich generierten Sendesymbole in den Zeitbereich transformiert. Am Empfänger erfolgt mit einer FFT die umgekehrte Transformation.

Zur Durchführung einer FFT existieren verschiedene Algorithmen, wie Radix-2, Radix-4 und der Winograd-Algorithmus, die durch einen unterschiedlichen Aufwand in Bezug auf die durchzuführenden Multiplikationen bzw. durch unterschiedliche Länge des Eingangsvektors gekennzeichnet sind. Allen gemein ist jedoch die grundlegende Eigenschaft, dass zur Berechnung des Ausgangsvektors der gesamte Eingangsvektor bekannt sein muss. Es handelt sich folglich um einen blockverarbeitenden Algorithmus. Analog zur Blockdekodierung ergeben sich bei der Anwendung der FFT nicht vermeidbare Latenzzeiten durch das Warten auf das vollständige Einschreiben eines Blocks, auch wenn durch eine Fließbandarchitektur ein kontinuierliches Einschreiben und Auslesen von Blöcken möglich ist. Die Gesamtverzögerung setzt sich aus der Einschreibeverzögerung und der Verarbeitungszeit zusammen.

Eine weitere Verzögerung aufgrund von Wartezeiten kann sich in einem OFDM-System durch die Reihenfolge der Unterträger ergeben. Diese werden meist in der Reihenfolge $-\frac{N_{fft}}{2},...,-1,0,1,...,\left(\frac{N_{fft}}{2}-1\right)$ verarbeitet, mit N_{fft} als Anzahl der FFT-Stützstellen eines OFDM-Symbols. Praktische FFT-Realisierungen erwarten dagegen die einzelnen Stützstellen in natürlicher Reihenfolge 0, 1, ..., (N_{fft}-1). Die negativen Unterträger der ersten Notation entsprechen bei dieser Vorgabe der rechten Hälfte des Spektums, also den Unterträgern $\frac{N_{fft}}{2}$, ..., (N_{fft}-1). Durch diese notwendige Umordnung der Reihenfolge über einen Zwischenspeicher entsteht eine Verzögerung um die Einschreibezeit für ein halbes OFDM-Symbol.

2.7 Weitere Module

In Abb. 2 (Seite 11) sind weitere Module wie Mapper und Demapper sowie Scrambler und Descrambler aufgeführt. Diese leisten im Vergleich zu den anderen, bereits angesprochenen Modulen, unter der Voraussetzung einer günstigen Implementierung, keinen signifikanten Beitrag zur Gesamtlatenz. Aus Gründen der Vollständigkeit wird ihre Funktion im Folgenden kurz vorgestellt. Nicht aufgeführt sind in der Abbildung die Piloteneinfügung, die digitale Filterung sowie die Prüfsummen, die auch kurz in diesem Abschnitt erwähnt werden.

Mapper / Demapper: Mapper und Demapper sind Module der Leitungskodierung bzw. -dekodierung. Ein oder mehrere zu sendende Datenbits werden im Mapper einem bestimmten Datensymbol zugeordnet, welches dann noch in weiteren Leitungskodiermodulen verarbeitet oder aber direkt an den Digital-Analog-Wandler (D/A-Wandler) gegeben wird. Bei einer QPSK[24]-Modulation, die im resultierenden Signal einer 4-QAM[25]-Modulation entspricht, werden im Mapper anhand von jeweils zwei eingehenden Bits die Vorzeichen im I- und Q-Ausgangssignal bestimmt. Der Demapper extrahiert aus der eingehenden Konstellation der Symbole wieder die gesendeten Datenbits anhand von Entscheidungsschwellen. Bei QPSK wären diese die beiden Koordinatenachsen des Konstellationsdiagramms. Dabei können den Ausgangsdatenbits für eine *Soft-Decision*-Dekodierung Wahrscheinlichkeiten anhand des Abstandes des empfangenen Symbols zu der jeweiligen Entscheidungsschwelle zugeordnet werden.

Scrambler / Descrambler: Ein Scrambler – die deutsche Bezeichnung „Verwürfler“ ist eher unüblich – tauscht eine Bitfolge umkehrbar gegen eine andere Bitfolge aus. Die verwendeten Algorithmen sind meist sehr einfach und basieren üblicherweise auf einem linear rückgekoppelten Schieberegister. Das Prinzip des Schieberegisters kann, insbesondere für eine Softwareimplementierung, durch eine Tabelle mit festen Einträgen abgebildet werden. Das Ausgangssignal des Scramblers ist stets eine pseudozufällige Bitfolge, wenn eine geeignete Rückkopplung und ein geeigneter Startwert verwendet werden. Bei Kenntnis der Rückkopplungsstruktur und des Startwertes kann die ursprüngliche Bitfolge fehlerfrei wiederhergestellt werden.

Scrambler werden für verschiedene Einsatzzwecke verwendet. Als Bestandteil der Leitungskodierung stellen sie auch bei statischem Eingangssignal Pegelwechsel am Ausgang sicher. Sie können so zusammen mit einem geeigneten Mapping die Gleichanteilsfreiheit herstellen. Durch die vorhandenen Pegelwechsel wird weiterhin eine Taktsynchronisation ermöglicht. Das Spektrum des Ausgangssignals ist gegenüber dem Spektrum des Eingangssignals im Allgemeinen gleichmäßiger über den gesamten Spektralbereich verteilt.

[24] QPSK: Quadrature Phase-Shift Keying
[25] QAM: Quadraturamplitudenmodulation

Piloteneinfügung: Piloten bzw. Pilottöne sind Zusatzinformationen, die von der Quelle dem eigentlichen Nutzdatensignal hinzugefügt werden. Sie dienen meist Steuerungs- oder Referenzaufgaben. So können in OFDM-Systemen die Piloten zur Kanalschätzung oder zur Phasenkorrektur verwendet werden.

Das Einfügen von Pilottönen in den Datensymbolstrom erzeugt im Allgemeinen keine zusätzliche Latenz, stattdessen wird bei einer paketbasierten Übertragung die gesamte Symbolsequenz nur länger. Bei bestimmten Implementierungen lässt sich jedoch ein signifikanter Beitrag zur Gesamtlatenz nicht vermeiden. Im später eingeführten Beispielsystem (siehe Kapitel 4) erfolgt die Piloteneinfügung speicherbasiert zusammen mit der Anordnung der einzelnen Datenunterträger im gesamten OFDM-Symbol. Ein OFDM-Symbol bietet hier insgesamt 1024 verschiedene Unterträgerpositionen, von denen 768 für die Daten und 60 für die Piloten verwendet werden. Die Struktur des entsprechenden Moduls ist in Abb. 6 aufgezeigt. Mit jedem Takt werden acht eingehende Datensymbole parallel auf die zugehörigen Stellen im Speicher geschrieben, der aus acht einzelnen Dualportspeichern besteht. Die Adressen sind in einem ROM[26] gespeichert. Die Piloten selbst sind bereits im Speicher vorbelegt und werden zur Laufzeit des Systems nicht überschrieben. Die Reihenfolge der Datenunterträgersymbole am Moduleingang entspricht bereits der Reihenfolge in der endgültigen Zuordnung, es sind nur zwischendurch die Piloten einzufügen.

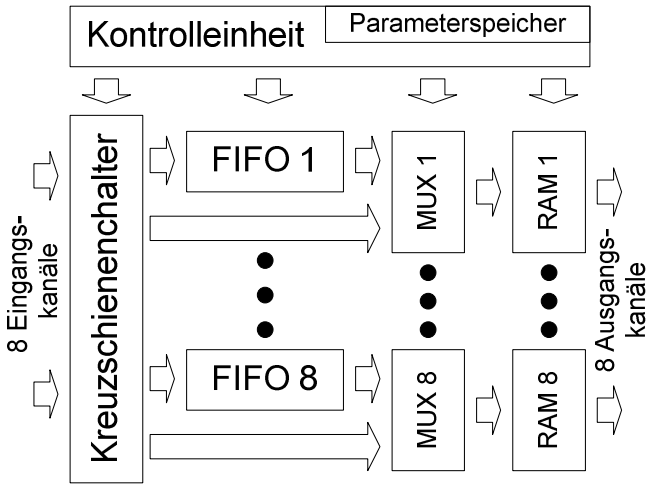

Abb. 6: Piloteinfügungsmodul des EASY-A-Systems

Würde der Abstand der Piloten der Anzahl der parallelen Eingangsdatensymbole oder einem Vielfachen davon entsprechen, könnten mit jedem Takt alle acht Datensymbole an die entsprechenden Speicherpositionen geschrieben werden, da in jedem einzelnen elementaren Speicherblock nur eine Schreiboperation erfolgen muss. Gleichzeitig könnten mit jedem Takt bereits acht Datenwörter ausgelesen werden. Es entstünde eine sehr geringe Verzögerung von wenigen Takten für das Einschreiben und Auslesen aus dem Speicher in die Register der Verarbeitungspipeline. Damit mit jedem Takt alle acht Datensymbole in den Speicher geschrie-

[26] ROM: Read-Only Memory

ben werden können, muss weiterhin vorausgesetzt werden, dass keine zusätzlichen Nullträger oder nur Nullträger im genannten Abstand eingefügt werden. Nullträger sind nicht verwendete, also zu Null gesetzte Unterträger bei der OFDM-Übertragung. Am Rand des genutzten Spektrums erleichtern sie die Filterung durch die Möglichkeit flacherer Filterübergänge zwischen Durchlass- und Sperrbereich, im Inneren des benutzten Spektrums können mit Nullträgern Teilbereiche ausgeschlossen werden.

Die genannte Voraussetzung des Pilotenabstandes ist in der Spezifikation des Beispielsystems nicht gegeben. Stattdessen liegen zwischen den einzelnen Pilotunterträgern 12 Datenunterträger. Es können somit nicht immer alle acht Datenunterträgersymbole gleichzeitig in einem Takt in den Speicher geschrieben werden, da zwei Symbole in denselben Speicherblock geschrieben werden müssen. Die aktuell nicht schreibbaren Symbole werden zusammen mit der Adresse in einer dem jeweiligen elementaren Speicherblock zugeordneten FIFO[27] gespeichert. Nach dem Einschreiben eines gesamten OFDM-Symbols wird die Dateneingabe gestoppt und die FIFOs werden geleert. Dies ist zulässig, da durch die Hinzufügung eines zyklischen Präfixes im FFT-Modul eine entsprechende Pause im Datenstrom auftreten darf.

Da das gesamte Symbol erst nach dem Leeren der FIFOs ausgegeben werden kann, entsteht eine zusätzliche Verzögerung in der Verarbeitung. Diese entspricht der Summe aus der Zeit für das Schreiben aller Datenunterträger in den Speicher sowie der Zeit für das Leeren der FIFOs. Da zeitgleich mit dem Einfügen der Piloten das Speichermodul auch die notwendige FFT-Verschiebung (siehe Abschnitt 2.6) durchführt, reduziert sich der eigentliche Latenzbeitrag durch die Piloten auf die Summe aus der halben Zeit für das Einschreiben in den Speicher und der Zeit für das Leeren der FIFOs.

Digitale Filterung: Zur Begrenzung der benötigten Übertragungsbandbreite durchlaufen die Sendesignale üblicherweise ein Pulsformungsfilter. Auf der Empfängerseite werden Eingangsfilterstrukturen verwendet, um unerwünschte Nebensignale zu unterdrücken und die Auswirkungen eines breitbandigen Rauschens zu minimieren. Die Filter können sowohl analog als auch digital realisiert werden. Digitale Filterstrukturen erzeugen zum einen durch ihren grundsätzlichen Aufbau, zum anderen durch die Art der Implementierung eine Verzögerung des Ausgangssignals. Da im später vorgestellten Beispielsystem (siehe Kapitel 4) nur eine analoge Tiefpassfilterung erfolgt und gleichzeitig in der Fachliteratur viele Verfahren zur Reduktion der Latenzen digitaler Filter beschrieben sind, wird auf eine Analyse der Auswirkungen digitaler Filter im Rahmen dieser Arbeit verzichtet. Als Beispiel für die Reduktion der Latenzen in FIR[28]-Filterstrukturen sei auf [31] und [32] sowie die dort aufgeführte Literatur verwiesen.

[27] FIFO: First-In First-Out, spezielle Speicherstruktur zum Puffern von Daten
[28] FIR: Finite Impulse Response

Prüfsummen: Die Notwendigkeit des Hinzufügens von Prüfsummen zum Test auf eine fehlerfreie Datenübertragung sowie der Ermöglichung einer Rückwärtsfehlerkorrektur wurde bereits in Abschnitt 2.2 aufgeführt. Eine CRC-Erzeugung und ein CRC-Test beruhen stets auf Bitoperationen. In Hardware lässt sich eine solche Struktur einfach aufbauen und parallelisieren, in Software ist die effiziente Realisierung dagegen aufwändiger.

Im Allgemeinen kann der Beitrag von Prüfsummen zur Gesamtlatenz, speziell bei einer hardwarebasierten Lösung, vernachlässigt werden.

2.8 Signalfeld

Je nach Systemdesign kann es notwendig sein, dass die Quelle der Senke auf PHY-Ebene Informationen übermittelt, wie die gesendeten Datensymbole zu verarbeiten sind. So muss bei einer paketbasierten Übertragung der Senke mitgeteilt werden, wie lang das übermittelte Datenpaket ist, sofern nicht von vornherein eine feste Paketlänge vorgegeben wird. Letzteres schränkt die Flexibilität des Systems jedoch stark ein. Werden bestimmte Datensymbole für die Kommunikationssteuerung reserviert, dürfen diese also nicht in den zu übertragenden Daten auftreten, kann ein solches Symbol als Paketendeerkennung herangezogen werden.

In allen anderen Fällen muss die Quelle die Paketlängeninformation auf eine geeignete Weise bereitstellen. Weitere mögliche Informationen sind die verwendete Modulationsart, die Initialisierungswerte für Scrambler und die Art der verwendeten Kanalkodierung. Auf PHY-Ebene ist eine Übermittlung der verwendeten Modulationsart und Kanalkodierung nur notwendig, wenn auf den höheren Ebenen kein Protokoll zur synchronen Anpassung durchgeführt wird. Gleiches gilt für den Scrambler, dessen Initialisierungsvektor nur übertragen werden muss, wenn er sich zwischen den einzelnen Frames ändert.

Analog zur MAC-Schicht, die Informationen zur Paketadressierung in einem speziellen MAC-Header den Daten voranstellt, kann auf PHY-Ebene ein PHY-Header bzw. Signalfeld verwendet werden, um die genannten Informationen zu übertragen. Die Position des Signalfeldes im Datenpaket, die Länge sowie die Modulation und Kodierung desselben sind dabei durch die Systemspezifikation vorgegeben und der Quelle und Senke bekannt. Somit ist der Informationsaustausch zur Kommunikationssteuerung auf PHY-Ebene gewährleistet.

Durch das Signalfeld und seine Verarbeitung entstehen jedoch zusätzliche Verzögerungszeiten. Allein durch das Vorhandensein einer zusätzlichen Sequenz vor den Daten verlängert sich der gesamte Frame und die Daten können nicht direkt nach der Präambel gesendet werden. Weiterhin ist die Demodulation und Dekodierung der Datensymbole im Empfänger erst möglich, wenn das Signalfeld ausgewertet und die Modulations- und Kodierungsart bestimmt wurden. Der Beginn der Datensymbolverarbeitung verzögert sich um die gesamte Verarbeitungszeit des Signalfeldes.

Doch nicht nur beim Empfänger, auch im Sender entstehen weitere Verzögerungen. Das Signalfeld muss aus den gegebenen Parametern berechnet werden. Hierfür kann die Zeit während der Präambelausgabe genutzt werden. Ist es jedoch erforderlich, erst eine Parameterüberprüfung auszuführen, um die Gültigkeit dieser festzustellen und im Fehlerfall die Übertragung nicht zu starten, kann die Präambel nicht gleich gesendet werden. Der Parametertest stellt aber üblicherweise keinen hohen Rechenaufwand dar und erzeugt demzufolge keinen signifikanten Beitrag zur Latenz des Gesamtsystems.

2.9 Implementierungsspezifische Aspekte

Die bisherige Analyse der Module beschränkte sich größtenteils auf implementierungsunabhängig auftretende Eigenschaften der jeweils zugrunde liegenden Algorithmen und Verfahren. Doch auch die Art der Implementierung hat einen Einfluss auf die entstehenden Latenzen und kann diese sowohl positiv als auch negativ beeinflussen. Teilweise wurden implementierungsspezifische Aspekte bereits angesprochen, z. B. für die Prüfsummen oder die Piloteneinfügung im Abschnitt 2.7.

Der größte Unterschied ergibt sich beim Vergleich von hardware- gegenüber softwarebasierten Lösungen. Software-Defined Radio (SDR) ist ein Schlagwort, welches seit geraumer Zeit viel Aufmerksamkeit auf sich zieht und Forschungsschwerpunkte stellt. Die Idee lässt sich sehr einfach beschreiben: Klassische Funksysteme sind für einen speziellen Anwendungsfall zugeschnitten, d. h. der Frequenzbereich, die Modulationsarten und die Zugriffsschemata sind fest im System verankert und im Allgemeinen nicht variabel. Dies ist z. B. bei einer rein hardwarebasierten ASIC-Lösung eines Übertragungssystems mit einem Großteil an analogen Komponenten der Fall.

Der SDR-Ansatz versucht, die mangelnde Flexibilität der Übertragungssysteme zu beseitigen, indem die gesamte Signalverarbeitung in den digitalen Bereich verschoben wird. Die einzigen analogen Elemente sind eine Antenne und ein Verstärker mit einem möglichst sehr breiten Operationsfrequenzbereich. Anschließend folgen sofort Wandler mit hohen Abtastraten. Die Einhaltung des Abtasttheorems wird durch ein vorgeschaltetes Antialiasingfilter sichergestellt, welches sich auch indirekt aus den Eigenschaften von Antenne und Verstärker ergeben kann. Im Rahmen der digitalen Signalverarbeitung können nun durch Filterung die gewünschten Übertragungsfrequenzen selektiert werden. Sollte ein Frequenzbereich bereits durch ein anderes System belegt sein, kann das SDR-System einfach auf einen anderen Bereich umschalten. Dabei spielt ein System mit der direkten Analog-Digital-Wandlung hinter der Antenne – bzw. des Antennenverstärkers – seine größtmögliche Flexibilität genau dann aus, wenn die anschließende Signalverarbeitung programmgesteuert auf einem DSP[29] oder einem GPP[30] erfolgt. Durch eine Änderung im Programmablauf, z. B. den Aufruf einer anderen Un-

[29] DSP: Digital Signal Processor
[30] GPP: General Purpose Processor

terfunktion, können so sämtliche Eigenschaften und Algorithmen des Übertragungssystems geändert werden.

Aber auch mit einer auf FPGAs basierenden teilweisen Hardwarelösung lässt sich dieser Ansatz erreichen. So kann dieser zum einen die Filterung mittels einer konfigurierbaren Struktur vornehmen, während die Parameter von einem externen Kontrollprozessor vorgegeben werden. Zum Zweiten wäre eine (partielle) Rekonfiguration des FPGAs zum (teilweisen) Austausch von Verarbeitungsalgorithmen möglich. In einer dritten Variante wären alle Module parallel implementiert und über ein Steuerungsprogramm würden Multiplexer zur Datenflusskontrolle geschaltet.

Erfolgt die Datenverarbeitung zumindest teilweise auf einem DSP oder GPP, ergeben sich neue Quellen von Latenzen, die bei einer rein hardwarebasierten Lösung nicht auftreten. In [33] wird beschrieben, dass in dem untersuchten SDR-System die Busanbindung des Radio-Frontends an den Prozessor die Hauptquelle der Latenzen ist. Es werden einige Lösungen aufgezeigt, unter anderem die Verlagerung von rechenintensiven Prozessen zurück in die Hardware, um die zu transferierende Datenmenge zu verringern.

Auch die unterschiedliche Abarbeitung der Algorithmen in hardware- gegenüber softwarebasierten Lösungen kann einen Einfluss auf die Latenzen ausüben. Hardware ermöglicht die Verwendung von nahezu beliebiger Parallelität, die aber bereits während des Systemdesigns vorgesehen werden muss. Softwarebasierte Verfahren bieten dagegen annähernd beliebig viel Speicher, zumindest auf den heute verwendeten Rechnerplattformen. In einer Hardwarelösung lässt sich gut ein kontinuierlicher Datenstrom durch eine Fließbandverarbeitung realisieren. Die Prozessorarchitektur von DSPs und GPPs ist zwar auch fließbandbasiert, die Softwareprozesse arbeiten allerdings auf Speicherblöcken. Soll ein im Speicher liegendes Datenpaket für die Übertragung aufbereitet werden, erfolgt zuerst die Kodierung eines gesamten Blocks und anschließend das Mapping desselben. Eine mehr datenstromorientierte Abarbeitung mit mehreren parallelen Prozessen lässt sich in Software schwerer als in Hardware realisieren. Dadurch können zusätzliche Latenzen entstehen, z. B. durch ein Kopieren von Speicherblöcken. Andererseits stehen die möglichen Taktraten von DSPs und GPPs meist an der Spitze des technisch Möglichen, sodass keine pauschale Wertung über die Art des Einflusses der unterschiedlichen Abarbeitung in Hard- und Software möglich ist.

Der Fokus dieser Arbeit liegt in der Verringerung von Latenzen, wie sie bei einer rein hardwarebasierten Implementierung des Basisbandprozessors auftreten. Besonders im Bereich der ultrahochratigen Funksysteme, also in Systemen mit Datenraten im Bereich von mehreren Gbit/s, sind reine DSP/GPP-basierende Systeme in Bezug auf die praktische Relevanz noch utopisch. Die Datenrate hinter dem Analog-Digital-Wandler (A/D-Wandler) beträgt bei sol-

chen Systemen leicht bis zu 72 Gbit/s[31], während die maximale theoretische Speicherbandbreite eines aktuellen High-End-Prozessor-Systems[32] bei 256 Gbit/s liegt [34]. Mehr als ein Viertel der theoretisch verfügbaren Speicherbandbreite wird allein für das Einschreiben der Daten benötigt. Eine prozessorbasierte Lösung erfordert während der Verarbeitung einen mehrfachen Speicherzugriff. Abhängig von der gesamten Systemarchitektur und dem ausgeführten Programm ist damit die verbleibende Speicherbandbreite unter Umständen zu gering.

2.10 Kategorisierung der Latenzursachen

Die in den vorherigen Abschnitten beschriebenen Latenzursachen innerhalb der Basisbandverarbeitung lassen sich in die in Abb. 7 dargestellten sechs Kategorien begrenzte Verarbeitungsgeschwindigkeit, begrenzte Übertragungsgeschwindigkeit, Datenabhängigkeit, Parameterabhängigkeit, Verarbeitungsreihenfolge und Blockverarbeitung unterteilen. Die einzelnen Kategorien stehen in Bezug zueinander und bedingen sich teilweise. So folgt eine Wartezeit aufgrund der Parameterabhängigkeit zwischen Signalfeld und Datensymbol aus der begrenzten Übertragungs- und Verarbeitungsgeschwindigkeit für das Signalfeld.

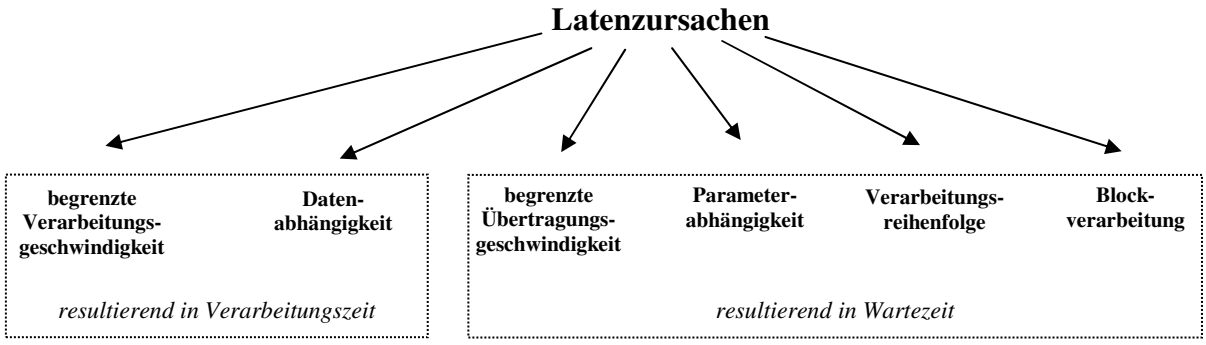

Abb. 7: Kategorisierung von Latenzen

Die Gesamtverzögerung eines Moduls zwischen der Datenein- und -ausgabe ergibt sich aus der eigentlichen Verarbeitungszeit innerhalb des Moduls sowie aus Wartezeiten, z. B. der Zeit für das vollständige Einschreiben eines Datenblocks. Demzufolge lassen sich die sechs Kategorien von Latenzursachen in zwei Gruppen zusammenfassen: Die erste, bestehend aus begrenzter Verarbeitungsgeschwindigkeit und der Datenabhängigkeit, resultiert in Verarbeitungszeiten, während sich die zweite Gruppe von Latenzursachen innerhalb eines Moduls durch Wartezeiten äußert.

Die begrenzte Verarbeitungsgeschwindigkeit ist eine offensichtliche Latenzursache. Sie erzeugt bei einer kontinuierlichen Eingabe von Daten eine initiale Verzögerung vor der Ausgabe des Datenstroms. Da sie aus der Implementierung des Moduls resultiert, lässt sie sich durch Änderungen am Modul selbst optimieren, d. h. unabhängig von den anderen Modulen.

[31] 12 Bit A/D-Wandlung mit 3 GHz Abtastrate auf zwei Kanälen
[32] Beispiel: Intel Xeon X5690 mit 3,46 GHz Taktrate

Die Schnittstellen eines Moduls und des Systems haben eine begrenzte Übertragungsgeschwindigkeit. Am Eingang des Empfängers entspricht diese der Übertragungsgeschwindigkeit auf dem Kanal, die durch die Abtastrate und die Modulationsart vorgegeben ist. Für die Untersuchung von Methoden zur Latenzverringerung wird die Übertragungsrate auf dem Kanal als fest und als nicht beeinflussbar angenommen. Weiterhin gehört zu dieser Kategorie die Verzögerung aufgrund einer Präambelhinzufügung zum Sendeframe. Wie bereits erläutert, verzögert sich dabei der Empfang des ersten Datensymbols mindestens um die Länge der Präambel.

Aus der begrenzten Übertragungsgeschwindigkeit folgt selbst bei einer unendlichen Verarbeitungsgeschwindigkeit die Unmöglichkeit einer latenzfreien Übertragung. Durch die Begrenzung können die Daten nicht mehr unmittelbar, sondern nur in einer endlichen Zeit > 0 übertragen werden. Erst nach dieser Zeit sind die Daten vollständig im Empfänger vorhanden.

Durch eine Datenabhängigkeit ist die Verarbeitungszeit eines Moduls ggf. abhängig von den Eingangsdaten und somit variabel. Ein Beispiel ist die iterative LDPC-Dekodierung, bei der durch eine Abbruchbedingung nicht notwendigerweise die maximale Anzahl an Iterationen zur Dekodierung durchgeführt werden muss. Unter „guten“ Übertragungsbedingungen konvergiert der LDPC-Algorithmus schneller, also mit einer geringeren Anzahl an Iterationen, gegen den Endwert.

Während sich die Datenabhängigkeit aus dem Datenfluss ergibt, folgt die Parameterabhängigkeit aus dem Kontrollfluss. Bei der Verwendung eines Signalfeldes zur Übermittlung von Modulationsart und Datenwortanzahl können die Datensymbole erst nach der vollständigen Auswertung des Signalfeldes verarbeitet werden. Es entsteht eine Wartezeit aufgrund der Parameterabhängigkeit. Auch wenn sich die ursächliche Latenz der Signalfeldverarbeitung, resultierend aus begrenzter Übertragungs- und Verarbeitungsgeschwindigkeit, nicht weiter verringern lassen sollte, kann die durch die Parameterabhängigkeit entstehende Latenz mithilfe der *spekulativen Demodulation* (siehe Abschnitt 7.2) verringert werden.

Bei einer Parallelverarbeitung mehrerer Datensymbole bzw. -blöcke muss am Modulausgang die ursprüngliche Reihenfolge wiederhergestellt werden. Es ergeben sich mitunter Wartezeiten an der Ausgangsschnittstelle aufgrund dieser Wiederherstellung der Verarbeitungsreihenfolge.

Die Blockverarbeitung als letzte Kategorie beschreibt die Latenz innerhalb eines blockverarbeitenden Moduls, die sich bei einer begrenzten Übertragungsgeschwindigkeit aufgrund des Wartens auf einen vollständigen Block ergibt. So kann eine RS-Blockdekodierung erst dann beginnen, wenn der gesamte Block in den Dekodierer eingeschrieben wurde.

3. Auswirkungen von Latenzen

3.1 Durchsatzverringerung

Für die praktische Anwendung eines Übertragungssystems ist nicht die theoretisch mögliche maximale Datenrate entscheidend, sondern der mögliche Datendurchsatz, also die Anzahl an fehlerfreien Paketübertragungen pro Zeiteinheit. Diese bestimmt sich einerseits aus der Datenrate, jedoch andererseits auch aus dem verwendeten Fehlerkorrekturverfahren, der Paketflusssteuerung und der Latenz.

Bei einer rein unidirektionalen Kommunikation ergibt sich der maximal mögliche Durchsatz direkt aus der Datenrate. Gleiches gilt für den Fall einer Rückwärtsfehlerkorrektur bei einer fehlerfreien Übertragung. Müssen dagegen Pakete aufgrund von Übertragungsfehlern wiederholt werden, verringert sich der Durchsatz. Während hier die Latenz abhängig vom Protokolldesign noch nicht notwendigerweise einen Einfluss ausübt, ist dieser Einfluss bei einer unmittelbaren Paketempfangsbestätigung offensichtlich. Ein neues Paket kann erst dann gesendet werden, wenn der Empfang des vorherigen bestätigt wurde.

Kann ein neues Paket erst nach dem Empfang des vorherigen gesendet werden, ergibt sich der maximale Paketdurchsatz bei einer Paketgröße N_{ps} und einer Datenrate R als Kehrwert der Paketübertragungszeit T_p, die sich aus der eigentlichen Übertragungszeit und der Latenz T_L zusammensetzt:

$$T_p = \frac{N_{ps}}{R} + T_L \tag{5}$$

Bei einer unmittelbaren Empfangsbestätigung kann die Quelle erst dann ein neues Paket senden, wenn der Empfang des vorherigen Pakets durch die Senke bestätigt wurde. Wird die Größe des Bestätigungspakets vernachlässigt bzw. in die Paketgröße N_{ps} mit aufgenommen, ergibt sich die gesamte Paketübertragungszeit unter Berücksichtigung der Empfangsbestätigung zu:

$$T_p = \frac{N_{ps}}{R} + 2 \cdot T_L \tag{6}$$

Bei einer Optimierung der Paketübertragungszeit durch eine Vergrößerung der Datenrate um den Faktor n darf die Latenz also maximal um einen bestimmten Faktor m steigen, damit die resultierende Paketübertragungszeit verringert wird. Mit $R_{neu} = n \cdot R$ und $T_{Lneu} = m \cdot T_L$ ergibt sich aus der Bedingung $T_{pneu} < T_p$:

$$\frac{N_{ps}}{R_{neu}} + 2 \cdot T_{Lneu} < \frac{N_{ps}}{R} + 2 \cdot T_L \tag{7}$$

$$\frac{N_{ps}}{n \cdot R} + 2 \cdot m \cdot T_L < \frac{N_{ps}}{R} + 2 \cdot T_L \tag{8}$$

$$m < \frac{N_{ps} \cdot (n-1)}{2 \cdot n \cdot R \cdot T_L} + 1 \tag{9}$$

Bei einer Paketgröße von 2000 Byte, einer Übertragungsrate von 500 Megabyte / Sekunde, einer Latenz von 4 µs und einer beabsichtigten Erhöhung der Datenrate um den Faktor 2 darf also die neue Latenz maximal 25 % größer als die ursprüngliche sein, damit der gesamte Durchsatz gegenüber vorher gesteigert wird. Ist die Bedingung (9) dagegen nicht erfüllt, verringert sich der Paketdurchsatz trotz der Erhöhung der Datenrate. Steigt im gegebenen Beispiel die Latenz um den Faktor 1,5, verringert sich der Paketdurchsatz trotz der Verdoppelung der Übertragungsrate um rund 15 %.

3.2 Echtzeitfähigkeit

Während bei einer Durchsatzoptimierung die Anzahl an fehlerfrei übertragenen Paketen pro Zeiteinheit auch auf Kosten der Laufzeit eines einzelnen Paketes gesteigert werden soll, wird für die Echtzeitfähigkeit eines Übertragungssystems die Laufzeit eines einzelnen Paketes betrachtet. Bei der Darstellung der Latenzursachen im Abschnitt 2.3 wurde anhand des Beispiels ADSL bereits die Beeinflussung der Echtzeitfähigkeit durch die Interleaververzögerung aufgezeigt.

Echtzeit bedeutet, dass die Paketübertragung innerhalb eines bestimmten Zeitintervalls abgeschlossen wurde. Es gibt viele Systeme, vor allem im Bereich der Industrieautomation und der Audio- / Videoübertragung, bei denen neben der erforderlichen Datenrate eine möglichst geringe Verzögerungszeit das Hauptkriterium ist. Die Verzögerungszeit muss die Echtzeitbedingung einhalten. Bei vielen dieser Systeme wird auf eine Rückwärtsfehlerkorrektur und eine erneute Übertragung fehlerhafter bzw. fehlender Pakete verzichtet, da diese Informationen dann bereits „veraltet“ sind. Auch bei einer rein unidirektionalen Übertragung kann z. B. durch eine sehr komplexe Vorwärtsfehlerkorrektur eine Latenz entstehen, die die vorgegebenen Grenzwerte für die Echtzeitbedingungen verletzt. Beim Systementwurf muss das Hauptaugenmerk auf der Reduzierung der Verarbeitungszeit liegen, unter Beachtung der minimal erforderlichen Datenrate sowie der maximal zulässigen Fehlerrate.

Als weitere Latenzauswirkung gehört die im Abschnitt 2.4 vorgestellte Beeinflussung der Frameerkennungsrate durch Fehlalarme in die Kategorie Echtzeitfähigkeit. Idealerweise würde ein System eine lückenlose Folge von N Präambeln mit folgendem Signalfeld als N einzelne Frames erkennen, ohne dass einzelne verloren gehen. Durch die Latenz für die Signalfeldauswertung ist dies jedoch nicht immer gegeben.

Die auch im Abschnitt 2.4 eingeführten Midambeln erzeugen nicht nur eine zusätzliche Latenz, sondern unterliegen Bedingungen an die Echtzeitfähigkeit. Der maximale Abstand zwischen den Midambeln ist durch die Kohärenzzeit des Kanals gegeben. Gleichzeitig entsteht durch die Berechnung der neuen Kanalkoeffizienten eine Verzögerungszeit, während der die neu einlaufenden Symbole entweder nicht verarbeitet werden können oder noch mit den alten Koeffizienten korrigiert werden.

3.3 Kanalkollisionen

In drahtlosen Systemen sowie in drahtgebundenen Busstrukturen greifen mehrere Stationen auf das gleiche Übertragungsmedium zu. Es sind also Verfahren notwendig, durch die ein gleichzeitiger Zugriff verschiedener Stationen auf das Medium erkannt und möglichst verhindert werden kann. Von vielen denkbaren Möglichkeiten hat sich im drahtgebundenen Ethernet CSMA/CD[33] [35] durchgesetzt, während in Funknetzwerken überwiegend CSMA/CA[34] [36] zum Einsatz kommt. Bei beiden handelt es sich um Mehrfachzugriffverfahren mit Trägerüberprüfung. Während bei CSMA/CD Kollisionen erkannt und signalisiert werden, wird bei CSMA/CA versucht, Kollisionen weitestgehend zu vermeiden.

Der typische Ablauf bei CSMA/CD ist wie folgt: Möchte eine Station Daten senden, überprüft sie als erstes, ob der Kanal frei ist. Ist dies für eine definierte Zeit gegeben, werden die Daten gesendet, während gleichzeitig der Kanal weiter überwacht wird. Sollte durch die Überwachung eine Kanalkollision erkannt werden, wird die Übertragung abgebrochen und ein Störsignal gesendet, damit alle anderen Stationen die Kollision ebenfalls erkennen können. Anschließend wird eine zufällige Zeit gewartet, bis ein erneuter Sendeversuch mit vorheriger Überprüfung des Kanals unternommen wird. Ist eine festgelegte Maximalanzahl an Übertragungsversuchen erreicht, ohne dass die Daten kollisionsfrei übertragen werden konnten, wird die Übertragung endgültig abgebrochen und ein Fehler an die Datenquelle gemeldet.

Drahtlose Systeme sind nicht unbedingt vollduplexfähig, z. B. wird zum Senden und Empfangen meist die gleiche Antenne verwendet. Damit ist eine Überwachung und somit eine Kollisionserkennung während des Sendens nicht möglich. Das CSMA/CA-Verfahren basiert auf CSMA/CD, versucht jedoch die Anzahl an Kollisionen weitestgehend zu minimieren. Dazu wird vor einer Übertragung zuerst wie bei CSMA/CD überprüft, ob der Kanal für eine definierte Zeit frei ist. Anschließend wird jedoch nicht gleich gesendet, sondern erst eine weitere, zufällige Zeit gewartet, bevor das Senden gestartet wird. Wird während dieser Zeit eine Belegung erkannt, wird der Ablauf der zufälligen Wartezeit bis zum Ende der laufenden Übertragung gestoppt. Über eine ausbleibende Bestätigung des Datenempfangs durch die Empfangsstation kann eine entstandene Kanalkollision erkannt werden.

[33] CSMA / CD: Carrier Sense Multiple Access / Collision Detection
[34] CSMA / CA: Carrier Sense Multiple Access / Collision Avoidance

Offensichtlich muss die Kanalbelegungserkennung möglichst latenzfrei sein, durch die ansonsten entstehende Blindzeit erhöht sich die Gefahr von Kollisionen. Hohe Blindzeiten entstehen u. a., wenn umfangreiche Berechnungen in einer Softwareimplementierung vorgenommen werden müssen oder die Belegungserkennung auf einer Präambeldetektion beruht.

3.4 Energieverbrauch

Ein anderer Aspekt hoher Latenzzeiten wird nur sehr selten beachtet: Durch eine Verringerung der Latenzzeit ist es möglich, die Leistungsaufnahme des Systems zu verringern. In der Fachliteratur, z. B. in [37], gibt es eine breite Vielfalt von Untersuchungen zur Verringerung des Energieverbrauchs eines digitalen Systems, sodass auf diese im Rahmen dieser Arbeit nicht weiter eingegangen wird. Zur beispielhaften Erläuterung der Aussage, dass durch eine Verringerung der Latenzzeit eines Systems der Energieverbrauch reduziert werden kann, seien die dort beschriebenen Verfahren Clock-Gating und Power-Gating aufgeführt.

Beim Clock-Gating werden durch gezieltes Abschalten des Taktes für bestimmte Schaltungsteile diese in einen operationslosen Zustand versetzt, in dem nur noch der statische Ruhestrom fließt. Über Power-Gating ist es möglich, den betreffenden Schaltungsteil komplett von der Versorgungsspannung zu trennen und in einen vollständig stromlosen Zustand zu versetzen.

Wenn es möglich ist, die Verarbeitungszeit eines Systems zu reduzieren, kann das System während der eingesparten Zeit über Clock- oder Power-Gating in einen stromsparenden Ruhezustand geschaltet werden. Zur Verringerung der Verarbeitungszeit dürfen dabei keine Methoden wie die Anhebung der Taktfrequenz verwendet werden, die in einer erhöhten Stromaufnahme resultieren, sonst ist die Reduktion des Energieverbrauchs nicht gegeben.

Eine Möglichkeit zu einer solchen Reduktion der Verarbeitungszeit ist die im Abschnitt 7.3 eingeführte *spekulative Dekodierung*. Ohne Änderungen am Aufbau eines Blockdekodierers und ohne Änderungen der Taktfrequenz ist es möglich, die für einen Block benötigte Verarbeitungszeit durch eine Spekulation auf Fehlerfreiheit zu verringern. Bei einem Spekulationserfolg lässt sich in Verbindung mit den oben genannten Methoden eine Energieeinsparung erreichen. Im Falle eines Spekulationsmisserfolges dagegen erhöht sich der Energieverbrauch nur geringfügig, abhängig von der gewählten Umsetzung der Spekulationshypothesenüberprüfung. Eine detaillierte Analyse der durch spekulative Verfahren ermöglichten Energieeinsparung findet sich im Abschnitt 7.6.

4. Das EASY-A 60 GHz VHR-E-System

4.1 Systembeschreibung

Im Rahmen des EASY-A Projektes [38] wurden umfassende Untersuchungen zur Nutzung des 60 GHz-Bandes für die drahtlose Kommunikation vorgenommen. Ein essentieller Aufgabenteil des Projektes war die Ausarbeitung von Systemkonzepten sowie deren Realisierung. Dazu wurde das Projekt in zwei Teilbereiche unterteilt: Der erste, das UHR-C[35]-System, beschäftigte sich mit der Spezifikation und Realisierung eines Systems zur drahtlosen Datenübertragung mit Datenraten bis zu 10 Gbit/s über Entfernungen bis zu 1 m. Der zweite Aufgabenbereich, das VHR-E[36]-System, fokussierte sich auf ein drahtloses System, welches Datenraten von 1 Gbit/s über Entfernungen bis zu 10 m ermöglicht. Das Projekt basierte in Teilen auf dem vorhergegangenen WIGWAM-Projekt [39], welches ebenfalls die Demonstration einer hochratigen Datenübertragung im 60-GHz-Band zum Ziel hatte.

Für das VHR-E-System wurden verschiedene Anwendungsszenarien definiert, unter anderem die Installation eines hochratigen Datenübertragungssystems mit mehreren Teilnehmern in einer Flugzeugkabine und ein typisches Büroszenario zur drahtlosen Kommunikation zwischen zwei Teilnehmern. Während für Ersteres aufgrund der Definition des Anwendungsszenarios die Zieldatenrate von 1 Gbit/s nicht unbedingt notwendig ist, soll Letzteres eine maximale Datenrate von 4 Gbit/s erreichen. Die Einschränkung auf zwei Teilnehmer basiert auf der Definition des Anwendungsszenarios, das die Demonstration eines „*drahtlosen Netzwerkkabels*“ vorsieht. Dies bedeutet, dass das implementierte System transparent anstelle eines Netzwerkkabels in eine Ethernet-Kommunikationsumgebung eingebracht werden kann. Die Basisbandverarbeitung dagegen ist unabhängig von dieser Spezifikation und kann auch in Mehrnutzerumgebungen zum Einsatz kommen.

Die im Rahmen des EASY-A-Projektes entstandene VHR-E-Systemkonzeption sowie der entworfene und realisierte Basisbandprozessor für das letztgenannte Anwendungsszenario werden im Rahmen dieser Arbeit zur beispielhaften Analyse der Latenzbeiträge der Basisbandmodule herangezogen. Gleichzeitig wird anhand dieser Implementierung eine Darstellung der resultierenden Verzögerungszeiten vorgenommen.

Zunächst wird im Abschnitt 4.2 die Spezifikation der physikalischen Ebene, bestehend aus den analogen Komponenten zur Übertragung im 60-GHz-Band sowie der digitalen Basisbandverarbeitung, erläutert. Anschließend erfolgt im Abschnitt 4.3 die Darstellung der Architektur des entworfenen digitalen Basisbandprozessors sowie dessen Implementierung. Darauf aufbauend wird im Abschnitt 4.4 ein komplettes Latenzmodell dieses Basisbandprozessors

[35] UHR-C: Ultra-High-Rate - Cordless
[36] VHR-E: Very-High-Rate - Extended Range

abgeleitet. Mit diesem werden die Beiträge der einzelnen Module zur Gesamtverzögerungszeit ermittelt.

4.2 PHY-Spezifikation

Bei einer Inhausübertragung im 60-GHz-Bereich über einen beabsichtigten Entfernungsbereich von bis zu 10 m muss von einer Multipfadumgebung ausgegangen werden [40]. Daher basiert das VHR-E-System auf einer OFDM-Übertragung. Dieses Übertragungsprinzip ist gut für frequenzselektive Kanäle geeignet.

Zur Modulation des Basisbandsignals auf die Übertragungsfrequenz von 60 GHz wird die in Abb. 8 dargestellte Superheterodynarchitektur mit einem analogen Quadraturmodulator für die Zwischenfrequenz von 12 GHz verwendet. Der Aufbau des Empfängers ist komplementär zu dieser Darstellung. Die gesamte analoge Verarbeitung ist dabei in einem einzigen Chip realisiert. Näheres zu diesem analogen Frontend (AFE) findet sich in [41] und [42].

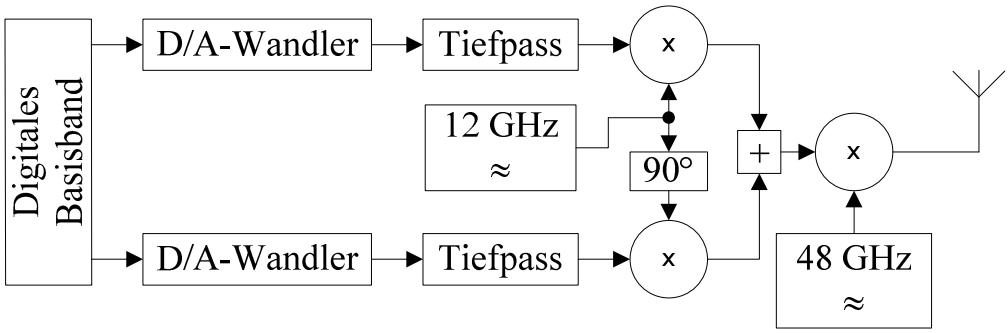

Abb. 8: VHR-E-Systemarchitektur Sender

Die Spezifikation der Basisbandverarbeitung [43] des EASY-A VHR-E-Systems basiert auf der des WIGWAM-Systems. Letztere ist in [44] detailliert beschrieben. Tabelle 1 zeigt eine Zusammenfassung der wichtigsten PHY-Parameter des VHR-E-Systems.

Kanalbandbreite	2160 MHz
Genutzte Signalbandbreite	≈ 1760 MHz
FFT-Größe	1024 Punkte
Unterträgerabstand	2,11 MHz
Schutzintervall	119 ns
OFDM-Symbollänge	593 ns
Anzahl Datenunterträger	768
Anzahl Pilotunterträger / DC-Nullträger[37]	60 / 5
Unterträger-Modulationsschema	BPSK[38], QPSK, 16-QAM

Tabelle 1: PHY-Parameter VHR-E-System

[37] DC-Nullträger: Nullträger im Basisband-Spektralbereich um 0 Hz (DC: Direct Current, Gleichstrom)
[38] BPSK: Binary Phase-Shift Keying

Zur Kanalkodierung sind zwei verschiedene Schemata definiert. Zum einen der (133,171)-Faltungscode aus dem IEEE[39] 802.11a Standard [45] , zusammen mit einem optionalen äußeren (255,239)-RS-Code. Letzterer ist im IEEE 802.15.3c [46] Standard definiert. Als Punktierungsschemata des Faltungscodes werden die Raten ½, ⅔ und ¾ unterstützt, letztere nur für die 16-QAM-Modulation. Im Folgenden wird zur einfachen Referenz auf eine Kombination aus Modulationsart und Punktierungsschema das gewählte Punktierungsschema als Suffix an die Modulationsart angefügt. Die Notation QPSK-$\frac{2}{3}$ verweist so auf eine QPSK-Modulation mit Coderate ⅔.

Das zweite Kodierungsschema besteht aus einem (768,384)-LDPC-Code mit Coderate ½ aus dem IEEE 802.16e Standard [47]. Aufgrund der FPGA-basierten Implementierung des digitalen Basisbandes ist das LDPC-Kodierungsschema nicht Bestandteil der im nächsten Abschnitt 4.3 beschriebenen Basisbandumsetzung. Für die zusätzliche Aufnahme der LDPC-Dekodierung standen nicht genug Logik-Ressourcen innerhalb des FPGAs zur Verfügung.

Aus den drei Modulationsarten mit den drei unterschiedlichen Punktierungsschemata und einer optionalen RS-Kodierung ergibt sich die Unterstützung von 14 verschiedenen nominellen Datenraten, die in Tabelle 2 zusammengefasst sind. Gleichzeitig werden in dieser Tabelle die entstehenden Datenraten des kodierten Datenstroms nach der Punktierung gezeigt.

Modulationsart / Punktierung	Nettodatenrate ohne RS-Kodierung / Mbit/s	Nettodatenrate mit RS-Kodierung / Mbit/s	Bruttodatenrate nach Kodierung / Mbit/s
BPSK-½	650	605	1300
BPSK-⅔	860	806	1290
QPSK-½	1300	1210	2600
QPSK-⅔	1720	1610	2580
16-QAM-½	2600	2430	5200
16-QAM-⅔	3450	3230	5180
16-QAM-¾	3890	3645	5190

Tabelle 2: Datenraten VHR-E-System

4.3 Basisbandimplementierung

Die digitale Basisbandverarbeitung ist entsprechend Abb. 9 in drei Hauptmodule unterteilt. Während die sendeseitige Verarbeitung im Transmitter in einer Gruppe zusammengefasst wird, wurde in der empfängerseitigen Verarbeitung eine Unterteilung in zwei Gruppen vorgenommen. Der Empfänger ist deutlich komplexer als der Transmitter, kann jedoch in die beiden Bereiche digitales Frontstage und digitaler Datenpfad aufgeteilt werden.

[39] IEEE: Institute of Electrical and Electronics Engineers

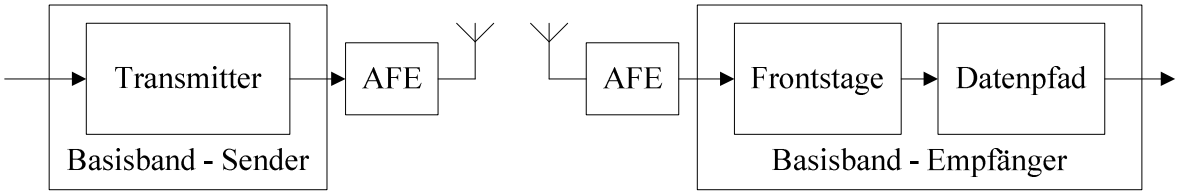

Abb. 9: Struktur digitale Basisbandverarbeitung

Das digitale Frontstage umfasst dabei die gesamte auf Datensymbolen basierende Verarbeitung bis zum Demapper, also die Framesynchronisation, die Frequenz- und Kanalkorrektur sowie die FFT und die Extraktion der Datenunterträger aus dem OFDM-Symbol. Der Demapper ist der erste Teil des Datenpfades, welcher weiterhin aus der datenbitbasierten Verarbeitung, wie dem Deinterleaving und der Kanaldekodierung, besteht. Die Architektur sowie die Implementierung des Transmitters und des Empfänger-Datenpfades wurden bereits in [48] durch den Autor dieser Arbeit vorgestellt. Im Folgenden werden sie noch einmal erläutert, da sie die Basis für die beispielhafte Latenzmodellierung im Abschnitt 4.4 darstellen. Das digitale Frontstage steht nicht im Fokus dieser Arbeit, sondern stellt eine erweiterte Version der im Rahmen des WIGWAM-Projektes entwickelten Frontstage-Verarbeitung dar. Die zugrunde liegende Konzeption sowie Details der Implementierung finden sich in [49]. Am Ende dieses Abschnitts wird aus Gründen der Vollständigkeit das Blockschaltbild des digitalen Frontstages des EASY-A-Systems kurz vorgestellt.

Zuerst ist eine Erläuterung der Rahmenbedingungen für die Basisbandimplementierung zweckmäßig. Wie schon erwähnt, handelt es sich bei der digitalen Basisbandverarbeitung um eine FPGA-basierte Implementierung auf kommerziell verfügbarer Hardware. Die Abtastrate der Analog-Digital- und Digital-Analog-Wandler von 2,16 GHz ist nicht als Taktrate für die digitale Logik in einem FPGA geeignet, die spezifizierte maximale Operationsfrequenz der verwendeten Bausteine liegt bei 550 MHz. Folglich müssen mehrere Abtastwerte parallel verarbeitet werden. Für die vorliegende Implementierung wurde ein Parallelisierungsfaktor von acht gewählt, d. h. jeweils acht Abtastwerte für die I- und die Q-Komponente des Signals werden parallel verarbeitet. Die Wahl des Faktors basiert zum einen auf einer Abschätzung der tatsächlich erreichbaren Taktfrequenz für das komplexe Design. Für diese wurde aufgrund der vorhandenen Erfahrungen mit den verwendeten FPGAs eine obere Grenze von 275 MHz angenommen. Zum anderen wurde die Anbindung der Wandler an die FPGAs berücksichtigt. Diese Schnittstelle besteht aus jeweils vier parallelen Datenbussen, die im DDR[40]-Modus mit einer Taktrate von 270 MHz arbeiten. Für die FPGA-intern typische SDR-Verarbeitung ergibt sich somit die Verarbeitung von acht parallelen Abtastwerten mit einer Taktrate von 270 MHz.

Für die Parallelisierung der datenbitbasierten Verarbeitung wurde die erforderliche Datenrate zugrunde gelegt. Vor der Kodierung ist ein maximaler Durchsatz von 4 Gbit/s ausreichend,

[40] DDR: Double Data Rate

dies entspricht der parallelen Verarbeitung von 32 Bits bei einer Taktfrequenz von 125 MHz und gleichzeitig dem spezifizierten Interface zur MAC-Schicht. Nach der Kanalkodierung erhöht sich die Datenrate auf maximal 5,2 Gbit/s, sodass hier eine fortgeführte 32-Bit-parallele Verarbeitung mit einer Taktfrequenz von 200 MHz vollständig ausreicht. Theoretisch wäre mit dieser Taktfrequenz eine kodierte Datenrate von 6,4 Gbit/s möglich, es stehen somit genug Sicherheitsreserven zur Verfügung.

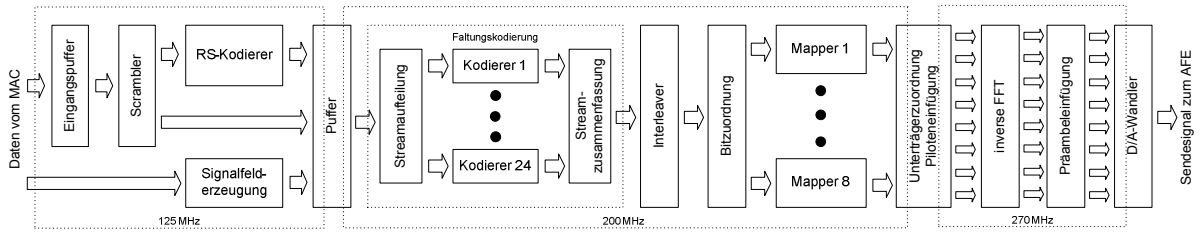

Abb. 10: Basisbandverarbeitungsstruktur Transmitter

Die in Abb. 10 und Abb. 11 aufgezeigten Strukturen des Transmitters sowie des Empfänger-Datenpfades basieren auf einem strikt modularen Designansatz. Die Schnittstelle zwischen den Modulen besteht aus einem allgemein gehaltenen Datenbus mit einem zusätzlichen Data-Valid-Signal. Jedes Modul beinhaltet am Ein- und Ausgang eine asynchrone FIFO sowie Synchronisationsstufen für evtl. notwendige globale Steuerungssignale. Asychron bedeutet in diesem Zusammenhang, dass die Schreib- und Lesetakte der FIFO unterschiedlich sein können. Den zugrunde liegenden generellen Aufbau eines solchen Moduls zeigt Abb. 12.

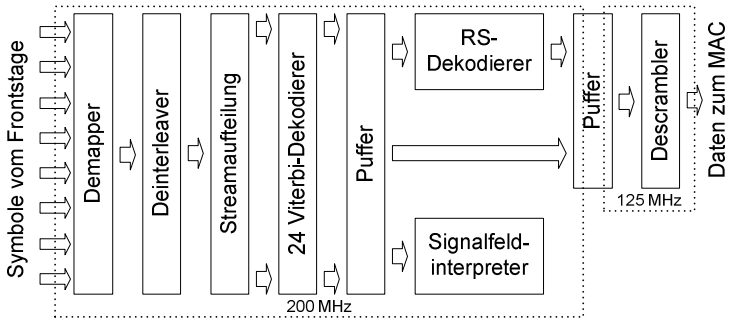

Abb. 11: Basisbandverarbeitungsstruktur Datenpfad Empfänger

Weiterhin verzichtet die Basisbandprozessorarchitektur für Transmitter und Empfänger-Datenpfad weitestgehend auf eine globale Steuerung. Jedes Modul beinhaltet eine eigene Steuerungslogik. Somit können die einzelnen Module sehr einfach ausgetauscht werden. Insbesondere für die vorgenommenen Untersuchungen zu den Latenzen unterschiedlicher Implementierungen ist diese Architektur sehr gut geeignet. Des Weiteren ist es möglich, auf diese Weise für jedes Modul die notwendige Taktrate zu verwenden, d. h. die bereits angesprochene Differenzierung auf verschiedene Taktdomänen vorzunehmen.

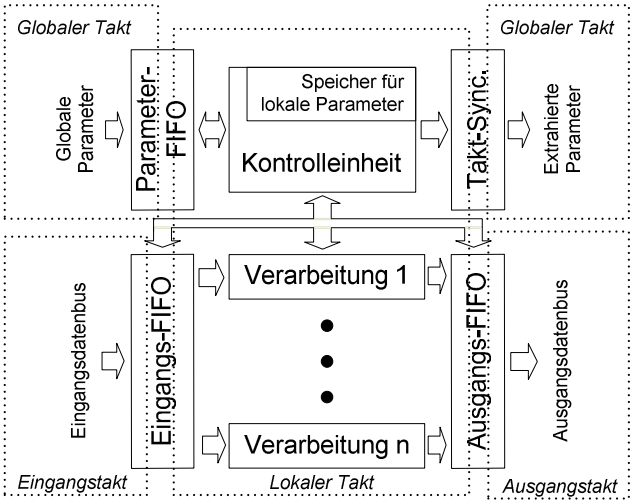

Abb. 12: Generelle Struktur eines einzelnen Basisband-Verarbeitungsmoduls

Zur Kanalkodierung und -dekodierung wurde analog zum WIGWAM-System eine Parallelisierung durch die Aufteilung des Datenstroms in bis zu 24 einzelne Streams vorgenommen. Für jeden Stream gibt es eine Instanz des Kodier- bzw. Dekodiermoduls. Die Minimalanzahl an benötigten Streams $N_{streammin}$ ergibt sich dabei aus der durch Modulationsart und Punktierungsschema vorgegebenen Datenrate und des maximalen Durchsatzes eines Dekodierungsmoduls. Der im Dekodierungsmodul enthaltene Viterbidekoder bietet bei der Taktrate von 200 MHz einen Durchsatz von 200 Mbit/s. Somit müssen für die Kanalkodierung eines BPSK-½-modulierten Datenpakets mindestens vier Streams verwendet werden. Die Anzahl an zu verwendenden Streams N_{stream} ist nach oben nur durch die Anzahl der parallel implementierten Kodier- / Dekodiermodule beschränkt und kann innerhalb dieser Schranken frei gewählt werden. In den Abschnitten 4.4 und 6.3 wird untersucht, ob sich mit einer Durchsatzsteigerung durch die Erhöhung der zu verwendenden Streams über die Mindestanzahl hinaus Vorteile für das System ergeben.

Die Aufteilung des eingehenden Datenstroms auf die Teilströme wird wie folgt vorgenommen: Die Datenbits, die im ersten OFDM-Symbol kodiert werden, werden durch das erste Kodier- bzw. Dekodiermodul verarbeitet, die des zweiten durch die zweite Instanz und so fort. Ist die vorher spezifizierte Anzahl N_{stream} mit $N_{stream} \geq N_{steammin}$ erreicht, werden die nächstfolgenden Datenbits wieder dem ersten Teilstrom zugeordnet und von dem entsprechenden Modul verarbeitet. Die Faltungskodierer werden dabei nicht nach jedem OFDM-Symbol auf den Initialzustand zurückgesetzt, sondern arbeiten kontinuierlich. Ein Abschluss erfolgt erst mit den jeweils letzten zu kodierenden Bits des gesamten Datenpakets durch das Anfügen von Tail-Bits.

Die Aufteilung auf Teildatenströme bedingt, dass am Ein- und Ausgang des Kodierungs- und Dekodierungsmoduls FIFOs verwendet werden, die jeweils genug Speicherkapazität für das Zwischenspeichern der Daten eines kompletten OFDM-Symbols bieten.

Wie in den Abbildungen 10 und 11 zur Struktur von Transmitter und Empfänger-Datenpfad ersichtlich, ist für die optionale RS-Kodierung ein Umgehungspfad vorgesehen. Da entsprechend der Spezifikation, abhängig von der gesamten Datenwortanzahl, auch bei der Verwendung der RS-Kodierung die letzten Datenwörter zur Overheadverringerung nicht RS-kodiert werden, ist nicht nur eine statische Umschaltung, sondern ein dynamisches Multiplexen zwischen den beiden Pfaden notwendig.

Zur Gewährleistung der vorgegebenen Durchsatzanforderungen wurde für die RS-Kodierung und -Dekodierung ein dem Streamverfahren ähnlicher Ansatz durch die parallele Verarbeitung aufeinanderfolgender Blöcke gewählt. Da diese Blöcke jedoch voneinander unabhängig sind, ist die Verarbeitung des gesamten Moduls transparent, d. h. ein äußerer Beobachter sieht einen einzelnen RS-Kodierer bzw. -Dekodierer mit dem vorgegebenen Durchsatz. Intern bestehen diese jedoch aus jeweils drei einzelnen Kodierern bzw. Dekodiern mit einer 1-Byte-breiten Eingangs- und Ausgangsschnittstelle. Jeder dieser Kodierer und Dekodierer bietet bei einer Taktrate von 200 MHz einen Durchsatz von 1,6 Gbit/s, sodass sich ein maximaler Gesamtdurchsatz von 4,8 Gbit/s ergibt.

Im Gegensatz zur Streamarchitektur bei der Kanalkodierung ergeben sich für die RS-Verarbeitung Herausforderungen aufgrund der Blockgröße von 239 bzw. 255 Byte. Die Eingangs- und die Ausgangsschnittstelle des RS-Kodierers bzw. -Dekodierers bestehen aus einem 32-bit-breiten Datenbus, pro Takt werden vier Datenbytes parallel übergeben. Das Einschreiben der Blöcke erfolgt dabei sequentiell, d. h. zuerst werden sämtliche Bytes des ersten Blocks eingeschrieben, dann die des zweiten usw. Die RS-Blockgröße ist somit kein Vielfaches der parallel an den Schnittstellen ein- und ausgehenden Daten. Demzufolge müssen innerhalb des Moduls Strukturen vorgesehen werden, die ein 4-Byte-breites Eingangswort, welches Bytes unterschiedlicher RS-Blöcke enthält, auf die jeweiligen internen Pfade verteilen. Aufgrund der Transparenzbedingung soll das Gesamtmodul nach außen wie ein einzelner RS-Kodierer bzw. -Dekodierer mit einer 32-bit-breiten Eingangs- und Ausgangsschnittstelle erscheinen. Folglich ist es nicht möglich, außerhalb des Moduls die Blockgrenzen an der Schnittstellenbreite auszurichten, d. h. im letzten parallelen Datenwort eines Blockes nur drei gültige Datenbytes zu übergeben. Die resultierende Struktur des RS-Dekodierungsmoduls ist in Abb. 13 dargestellt.

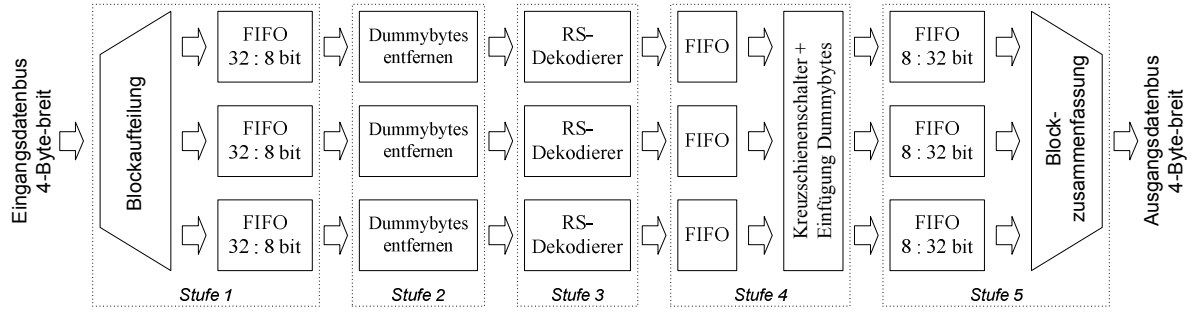

Abb. 13: Struktur Reed-Solomon-Dekodierer

In der ersten Stufe werden die eingehenden Daten entsprechend der Blockzugehörigkeit auf die drei einzelnen Dekodiererpfade aufgeteilt. Dabei wird der erste Block im ersten Pfad, der zweite im zweiten, der dritte im dritten, der vierte wieder im ersten Pfad usf. verarbeitet. Weiterhin erfolgt in dieser Stufe die Anpassung des 4-Byte-breiten Eingangsdatenbusses an den 1-Byte-breiten internen Datenbus des jeweiligen Pfades. Für diese Anpassung werden FIFO-Strukturen mit einer 32-bit-breiten Eingangs- und einer 8-bit-breiten Ausgangsschnittstelle verwendet. Jedes Eingangdatenwort wird in der FIFO auf vier sequentielle Ausgangsdatenworte aufgeteilt. Überschreitet nun ein 4-Byte-breites Datenwort der Eingangsschnittstelle die RS-Blockgrenze, wird es in der ersten Stufe in die beiden zu diesen Blöcken gehörenden Pfade eingeschrieben. Nach dem Durchlaufen der jeweiligen FIFO mit einer Serialisierung der Datenbytes können die überzähligen Bytes des nicht zu diesem Verarbeitungspfad gehörenden Blockes in Stufe 2 gelöscht werden. Gleiches gilt für möglicherweise überzählige Bytes, wenn das letzte Byte des letzten RS-Blocks nicht an den Datenwortgrenzen der Eingangsschnittstelle ausgerichtet ist.

Die Zusammenführung der einzelnen Pfade zur Wiederherstellung der Blockreihenfolge und zur Anpassung an die 4-Byte-breite Ausgangsschnittstelle in Stufe 5 erfolgt analog zur Vorgehensweise an der Eingangsschnittstelle. Für die Anpassung werden FIFO-Strukturen verwendet, die aus vier sequentiell eingeschrieben Datenbytes ein 4-Byte-breites Ausgangsdatenwort formen. Zum Auffüllen der Datenbyteanzahl pro Pfad auf ein Vielfaches der Busbreite der Ausgangsschnittstelle werden nach der RS-Dekodierung (Stufe 3) in Stufe 4 einzelne Bytes des jeweiligen Blockanfangs in einen anderen Pfad transferiert. Je nach Gesamtblockanzahl ist es zusätzlich notwendig, durch das Einfügen von ein oder mehreren Dummybytes das letzte Datenwort des letzten Blockes aufzufüllen, da die FIFOs nur vollständige Datenwörter ausgeben können. Diese zusätzlich eingefügten Bytes sind beim Zusammenführen mit den nicht dem RS-Dekodierer übergebenen Datenwörtern zu löschen.

Das Blockschaltbild des digitalen Frontstages zeigt Abb. 14. Während der übrige Basisbandprozessor wie erläutert zu einem hohen Grad modular aufgebaut ist und auf eine globale Steuerungslogik verzichtet, verfolgt die Frontstageimplementierung einen anderen Designansatz. Auch das Frontstage besteht aus einzelnen Modulen, beinhaltet jedoch einen umfangreichen globalen Steuerungsautomaten, der den gesamten Datenfluss kontrolliert. Die Datenverarbeitung erfolgt burstweise, d. h. auf einen Steuerungsimpuls des globalen Kontrollautomaten hin wird ein komplettes OFDM-Symbol verarbeitet. Im Gegensatz dazu erfolgt die Verarbeitung im Transmitter und dem Datenpfad des Empfängers datenstrombasiert, d. h. der Datenstrom ist kontinuierlich. Sowohl im Transmitter als auch im Datenpfad des Empfängers wird über ein Gültigkeitssignal festgelegt, ob das aktuelle Datenwort zu verarbeiten ist, wenn Lücken im Datenstrom auftreten.

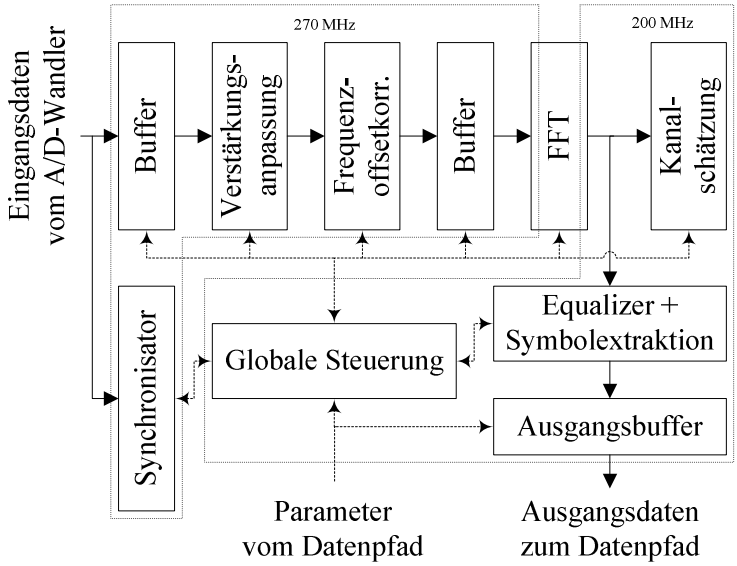

Abb. 14: Struktur des digitalen Frontstages

4.4 Latenzmodellierung

4.4.1 Einführung

Die Gesamtverzögerungszeit eines modularen Systems ergibt sich aus der Summe der Verzögerungszeiten der einzelnen hintereinander geschalteten Module sowie deren Schnittstellen. Dadurch könnte leicht vermutet werden, eine ausführliche Latenzmodellierung ist überflüssig bzw. schnell vorgenommen. Letzteres ist jedoch bei komplexeren Systemen nicht mehr unbedingt gegeben. Der in den vorhergehenden Abschnitten vorgestellte OFDM-Basisbandprozessor bietet verschiedene von außen wählbare Übertragungsparameter, die einen Einfluss auf die Verarbeitungszeit haben. Offensichtlich gehören die Modulationsart sowie die optionale RS-Kodierung dazu, aber auch die Anzahl der parallel zur Kodierung verwendeten Streams hat einen Einfluss.

Die Analyse der im Beispielsystem entstehenden Latenzbeiträge erfolgt getrennt für Sender und Empfänger in den folgenden Abschnitten 4.4.2 und 4.4.3. Zunächst werden noch einige wichtige Verzögerungszeiten definiert. Dazu ist es notwendig, den Interaktionsablauf zwischen MAC und PHY beim Senden und Empfangen von Daten darzustellen. Gleichzeitig ist anzumerken, dass nur eine Modellierung für die durch die digitale Verarbeitung entstehenden Latenzzeiten vorgenommen wird. Die Berechnung erfolgt dabei auf Grundlage der benötigten Taktperioden. Verzögerungen durch Leitungslaufzeiten und die analogen Sende- bzw. Empfangsmodule liegen im Bereich von wenigen Nanosekunden. Sie sind somit gegenüber den Latenzen der digitalen Verarbeitung vernachlässigbar klein und werden daher nicht berücksichtigt.

Schematisch ist der Interaktionsablauf zwischen MAC und PHY in den Diagrammen der Abb. 15 und Abb. 16 zusammengefasst. Sollen Daten gesendet werden, werden als erstes die Über-

tragungsparameter durch den MAC im PHY über eine dedizierte Kontrollschnittstelle eingestellt. Anschließend wird durch den MAC ein Startimpuls ausgelöst. Parallel dazu werden die zu sendenden Daten über den Datenbus an den PHY übergeben.

Im Anschluss an das Startsignal beginnt nach einer Verzögerungszeit T_{fs} die Ausgabe des Frames. Parallel dazu erfolgen die Berechnung der Framedauer T_{fd} und der Symbolanzahl sowie die Kodierung des Signalfeldes. Die berechneten Parameter werden nach der Zeit T_{pv} an den MAC übergeben. Die Zeit T_{pre} kennzeichnet die Dauer der den eigentlichen Datensymbolen vorangestellten Präambel zur Framesynchronisation.

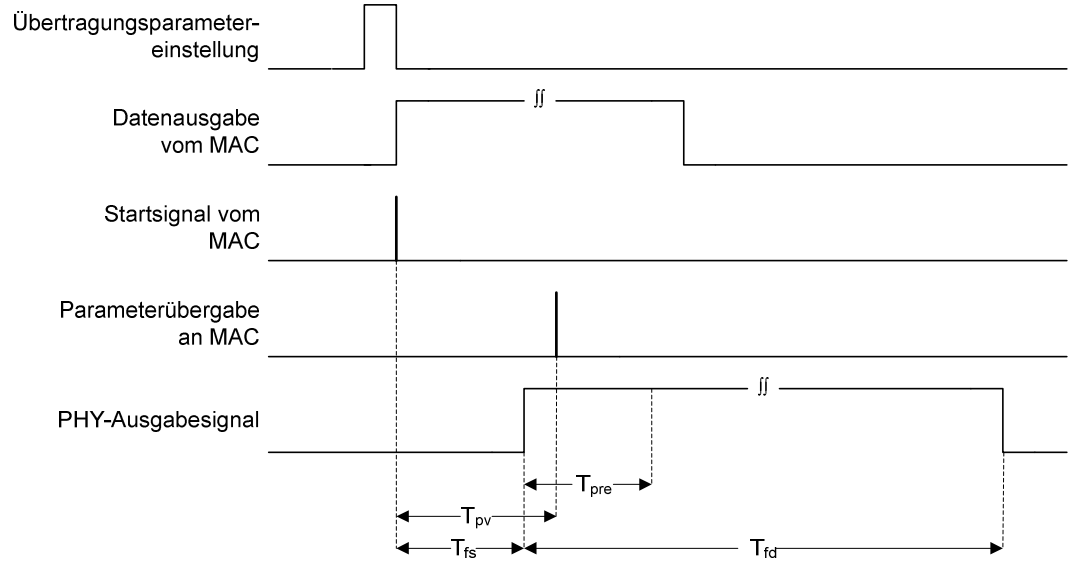

Abb. 15: Timingdiagramm Basisband Sender

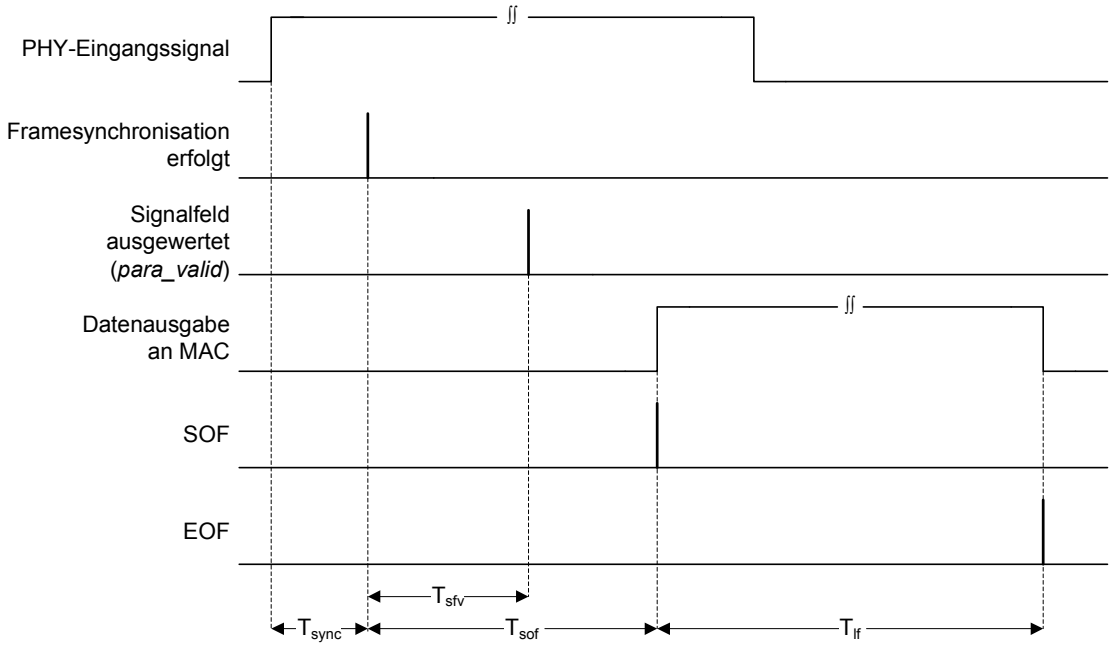

Abb. 16: Timingdiagramm Basisband Empfänger

Im Empfänger wird der ankommende Frame nach der Zeit T_{sync} erkannt. Anschließend wird innerhalb der Zeit T_{sfv} das Signalfeld verarbeitet. Nach der Zeit T_{sof} wird das erste Datenwort parallel mit einem SOF[41]-Impuls über die Ausgangsschnittstelle an den MAC übergeben, das letzte Datenwort folgt nach T_{lf} mit einem gleichzeitigen EOF[42]-Impuls.

Die bereits genannten und folgenden Zeitangaben T werden in zwei unterschiedlichen Notationen verwendet. Eine Angabe der Form T_{index} bezieht sich dabei auf die Zeitdauer in Sekunden, während ein hochgestelltes $*$ die Zählung der benötigten Taktzyklen kennzeichnet. Zur Umrechnung in die tatsächliche Zeitdauer ist diese Angabe mit der jeweiligen Taktperiodendauer T_{px} zu multiplizieren. Durch die drei unterschiedlichen Taktdomänen im Design ergeben sich die folgenden Faktoren:

$$T_{p125} = \frac{1}{125\,\mathrm{MHz}} = 8\,\mathrm{ns} \quad T_{p200} = \frac{1}{200\,\mathrm{MHz}} = 5\,\mathrm{ns} \quad T_{p270} = \frac{1}{270\,\mathrm{MHz}} \approx 3{,}7\,\mathrm{ns} \tag{10}$$

4.4.2 Sender

Für den Sender gibt es eine wichtige, nach außen sichtbare Verzögerungszeit: T_{fs}, die Zeit zwischen dem Setzen des PHY-Startsignals durch den MAC und den Beginn der Ausgabe des Frames auf dem Kanal. Diese Verzögerungszeit ergibt sich aus unterschiedlichen internen Verzögerungszeiten der Basisbandverarbeitung.

Das digitale Basisband des Senders muss an seiner Ausgangsschnittstelle zu den Digital-Analog-Wandlern einen kontinuierlichen Strom der zu sendenden Signalform sicherstellen. Die digitale Verarbeitung in den verschiedenen Taktdomänen ist jedoch burstorientiert. Die sich ergebenden internen momentanen Datenraten der einzelnen Module entsprechen nicht immer einem gleichmäßigen Datenstrom mit der Ausgaberate. Zur Angleichung an die Ausgaberate muss also ein Zwischenspeicher vorgesehen werden, der einen kontinuierlichen Ausgabedatenstrom erzeugt. Obwohl dies im vorgestellten System innerhalb der Datenverarbeitung durch das Piloteneinfügungsmodul vorgenommen wird, lässt sich das Prinzip durch einen Verarbeitungsblock mit folgendem Ausgangspuffer modellieren. Durch die den OFDM-Datensymbolen vorangestellte Präambel der Länge T_{pre} kann die Verarbeitung der Daten parallel zur Präambelausgabe erfolgen. Das Prinzip der parallelen Präambelgenerierung und der OFDM-Datensymbolerzeugung ist in Abb. 17 dargestellt.

[41] SOF: Start of frame
[42] EOF: End of frame

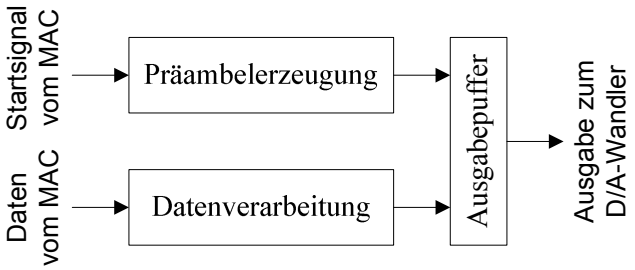

Abb. 17: Struktur der parallelen Präambelerzeugung

Die durch die Datenverarbeitung entstehende reine Verarbeitungslatenz zwischen der Eingabe des ersten Datenwortes und der Bereitstellung des ersten verarbeiteten OFDM-Datensymbols an der Schnittstelle zum Ausgabepuffer spiegelt sich in der Zeit T_{sd} wider. T_{sd} beinhaltet sämtliche Verarbeitungslatenzen der OFDM-Symbolerzeugung, also unter anderem die Verzögerungen aufgrund der bereits beschriebenen Piloteneinfügung sowie des Interleavers, und ist abhängig von der gewählten Kanalkodierung und Modulationsart:

$$T_{sd} = f(\text{Modulationsart, Punktierungsschema, RS-Kodierung}) \tag{11}$$

Der in Abb. 17 dargestellte Block „Präambelerzeugung“ bedeutet nicht, dass die zur Synchronisation verwendete Präambel anhand algorithmischer Vorschriften für jeden Frame neu berechnet wird. Stattdessen ist die vorgegebene feste Präambelsequenz in einem Speicher abgelegt, der nach dem Startsignal ausgelesen wird. Analog zu T_{sd} wird durch T_{sp} die Verzögerungszeit zwischen dem vom MAC generierten Startsignal und der Übertragung des ersten Präambelsymbols an den Ausgabepuffer charakterisiert. Im Gegensatz zu T_{sd} ist T_{sp} parameterfrei und somit konstant.

Zur Sicherstellung eines kontinuierlichen Ausgangsdatenstroms müssen unter Vernachlässigung der Verzögerungszeit des Ausgabepuffers zwei Fälle unterschieden werden. Gilt unter Berücksichtigung der Präambellänge T_{pre} die Beziehung $T_{pre} + T_{sp} \geq T_{sd}$, ist T_{sd} nach außen nicht mehr sichtbar. Die Zeit T_{fs} entspricht T_{sp}. Gilt dagegen $T_{pre} + T_{sp} < T_{sd}$, beeinflusst T_{sd} allein die nach außen sichtbare Verzögerungszeit T_{fs}.

Im Beispielsystem ist T_{fs} abhängig von der Modulationsart und dem Punktierungsschema. Für BPSK und QPSK ist $T_{sd} < T_{pre}$ und somit $T_{fs} = T_{sp}$. Für die 16-QAM-Modulation dagegen gilt $T_{sd} > T_{pre} + T_{sp}$. Die Präambelausgabe ist um die Zeit:

$$T_{pd} = T_{sd} - \left(T_{pre} + T_{sp}\right) \tag{12}$$

zu verzögern. Ohne diese Verzögerung um T_{pd} würde eine Lücke zwischen Präambel und erstem OFDM-Datensymbol im Ausgabedatenstrom zu den Digital-Analog-Wandlern entstehen. Die Verzögerung um T_{pd} erfolgt in der Basisbandimplementierung über einen Dekrementierungszähler, der mit dem entsprechenden Verzögerungswert T_{pd} geladen wird. Mit jedem

Takt in der 270-MHz-Domäne wird der Zählerstand um 1 verringert und beim Erreichen der Null der interne Startpuls für das Auslesen des Ausgabepuffers generiert.

Insgesamt ergibt sich folgende Formulierung für T_{fs}:

$$T_{fs} = \left(T_{sp}^{*} + T_{pd}^{*}\right) \cdot T_{p270} + T_{ss270} \tag{13}$$

mit

$$T_{pd}^{*} = \begin{cases} 0 & | \text{Modulationstyp} = \text{BPSK} \vee \text{QPSK} \\ 232 & | \text{Modulationstyp} = \text{16-QAM-}\frac{1}{2} \\ 487 & | \text{Modulationstyp} = \text{16-QAM-}\frac{2}{3} \\ 567 & | \text{Modulationstyp} = \text{16-QAM-}\frac{3}{4} \end{cases} \tag{14}$$

und

$$T_{sp}^{*} = 6 \tag{15}$$

Der Term T_{ss270} in (13) berücksichtigt, dass die Generierung des Startpulses und die Ausgabe der Präambel in verschiedenen Taktdomänen erfolgen. Er definiert die Verzögerungszeit zur Synchronisation des Startsignals zwischen den verschiedenen Taktdomänen, also die durch die Synchronisationsflipflops entstehende Verzögerung und ergibt sich entsprechend der Implementierung zu:

$$T_{ss270} = 1 \cdot T_{p125} + 4{,}5 \cdot T_{p270} + \tilde{T}_{cdc270} \tag{16}$$

$\tilde{T}_{cdc270}$ modelliert den ungewissen Zeitpunkt der Übernahme der Impulse durch das erste Flipflop in der Zieltaktdomäne. Dieser entsteht durch den unbekannten Phasenlagenbezug der beiden Takte und die physikalischen Laufzeiten und entspricht somit einer Taktperiode der Zieldomäne:

$$\tilde{T}_{cdc270} = \pm 0{,}5 \cdot T_{p270} \tag{17}$$

Auf die Synchronisation zwischen verschiedenen Taktdomänen wird in dieser Arbeit nicht weiter eingegangen, stattdessen sei auf [50] verwiesen. In diesem Zusammenhang ist erwähnenswert, dass in [51] spekulative Verfahren zur schnelleren Synchronisation zwischen verschiedenen Taktdomänen genutzt werden.

Anhand der genannten Angaben ergibt sich die minimale Verzögerung zwischen dem Setzen des Startsignals und dem Beginn der Frameausgabe zu:

$$\min T_{fs} = 45 \text{ ns} \tag{18}$$

und das Maximum bei Verwendung der 16-QAM-3/4-Modulation zu:

$$\max T_{fs} = 2146{,}6\ \text{ns} \tag{19}$$

Wie in der Einführung zur Latenzmodellierung beschrieben, werden die vorgegebenen Übertragungsparameter nach dem Setzen des Startsignals auf ihre Gültigkeit geprüft. Gleichzeitig werden weitere notwendige Parameter, wie die Anzahl der OFDM-Symbole, berechnet. Aus diesen Parametern wird dann das Signalfeld generiert. Die für die Generierung des Signalfeldes benötigte Zeit ist übertragungsparameterabhängig. In der Formulierung der Startverzögerung T_{fs} wurde dieser parameterabhängige Einfluss vernachlässigt. Die Signalfeldgenerierung ist im verwendeten abstrakten Modell aus paralleler Präambel- und Datensymbolerzeugung ein Bestandteil der Datensymbolverarbeitung und somit in T_{sd} enthalten. Da T_{sd} nur für die Modulationsart 16-QAM einen Einfluss auf T_{fs} hat, wurde die maximal entstehende Verzögerungszeit für den jeweiligen Punktierungstyp zur Bestimmung des notwendigen Verzögerungswertes T_{pd} zugrunde gelegt. Eine Erweiterung der Verzögerungswerte auf beliebige Parameterkombinationen würde den Steuerungs- und Implementierungsaufwand stark erhöhen, ohne dass die erzielbare Latenzverringerung diesen Aufwand rechtfertigen würde.

Die Ergebnisse der Gültigkeitsüberprüfung der vom MAC eingestellten Übertragungsparameter und die Berechnungsresultate der Signalfeldgenerierung, insbesondere die Symbolanzahl und die Framedauer, werden nach der Zeit T_{pv} an die MAC-Schicht zurückgegeben. Diese Zeit ist somit von außen sichtbar und wird aus Gründen der Vollständigkeit hier angegeben.

T_{pv} setzt sich entsprechend:

$$T_{pv} = T_{p125} \cdot \left(T^*_{pchk} + \begin{cases} 0 & \mid \text{Parameterkombination ungültig} \\ T^*_{sw} + T^*_{rsb} + T^*_{rso} + T^*_{sc} + T^*_{ns} & \mid \text{Parameterkombination ungültig} \end{cases} \right) \tag{20}$$

aus sechs einzelnen Summanden zusammen: T_{pchk} als Zeit für das Testen der von außen vorgegebenen Parameterkombination auf Gültigkeit, T_{sw} als Zeit für die Summation der Datenwortanzahlen der einzelnen Pakete des Sendeframes, T_{rsb} als Zeit für die Berechnung der gesamten RS-Blockanzahl, T_{rso} zur Bestimmung des Overheads durch die RS-Kodierung, T_{sc} als Zeit für die Bestimmung der notwendigen Streamanzahl und T_{ns} als Zeit für die Bestimmung der Symbolanzahl. Diese einzelnen Summanden hängen wiederum von bestimmten Parametern bzw. deren Kombination ab:

$$T^*_{pchk} = 1 \qquad T^*_{sw} = 3 \tag{21}$$

$$T^*_{rsb} = \begin{cases} 0 & \mid \text{RS - Kodierung aus} \\ 21 & \mid \text{RS - Kodierung an} \end{cases} \tag{22}$$

$$T_{rso}^{*} = \begin{cases} 0 & | (N_{res} < 48) \vee \text{RS - Kodierung aus} \\ 1 & | N_{res} \geq 48 \end{cases} \tag{23}$$

$$T_{sc}^{*} = \begin{cases} (2 \cdot (N_{stream} - 1) + 2) & | N_{res} < N_{sym} \\ (2 \cdot (N_{sym} - 1) + 3) & | N_{res} \geq N_{sym} \end{cases} \tag{24}$$

$$T_{ns}^{*} = \begin{cases} 21 & | N_{pad} = 0 \\ 24 & | N_{pad} \neq 0 \end{cases} \tag{25}$$

N_{stream} bezeichnet die Anzahl der verwendeten Streams, N_{sym} die Anzahl der benötigten OFDM-Datensymbole, N_{res} die Anzahl der Datenbytes im letzten RS-Block sowie N_{pad} die Anzahl an Bytes, die zum Auffüllen des letzten OFDM-Symbols benötigt werden. Diese vier Parameter ergeben sich aus der Berechnung der notwendigen Symbolanzahl während der Signalfelderzeugung. Der Algorithmus dafür ist in [43] beschrieben.

Unter der Voraussetzung einer gültigen Parameterkombination ergibt sich das Minimum von T_{pv} zu:

$$\min T_{pv} = 224 \text{ ns} \tag{26}$$

und das Maximum zu:

$$\max T_{pv} = 784 \text{ ns} \tag{27}$$

Neben T_{fs} und T_{pv} wurde im Abschnitt 4.4.1 noch die Framedauer T_{fd} als dritte von außen sichtbare Größe eingeführt. Sie bestimmt die Zeit, in der der Kanal belegt ist und nicht von anderen Teilnehmern genutzt werden kann. Demzufolge hat sie Einfluss auf das MAC-Protokoll. Weiterhin entstehen durch die im Frame enthaltenen Overheadanteile wie Präambel und Signalfeld zusätzliche Basisband-Verzögerungen beim Empfang. Diese werden im folgenden Abschnitt zur Latenzmodellierung des Empfängers aufgeführt. Mit den bereits eingeführten Größen T_{pre} und N_{sym} sowie N_{mid} als Anzahl der eingefügten Midambeln zur Wiederholung der Kanalschätzung, T_{mid} als deren Dauer, T_{ov} als Überlappungsbereich zwischen den einzelnen OFDM-Symbolen, der Präambel und den Midambeln sowie T_{sym} als Dauer eines OFDM-Symbols bestimmt sich T_{fd} zu:

$$T_{fd} = T_{pre} + (N_{sym} + 1) \cdot (T_{sym} - T_{ov}) + N_{mid} \cdot (T_{mid} - T_{ov}) \tag{28}$$

4.4.3 Empfänger

Die Empfängerverarbeitung ist, wie im Abschnitt 4.3 vorgestellt, in zwei Verarbeitungsgruppen unterteilt, das digitale Frontstage und den Datenpfad. Erstere übernimmt die OFDM-symbolbasierte Verarbeitung wie die Framesynchronisation, die Kanalschätzung und die Kanalkorrektur, während der Datenpfad u. a. die Kanaldekodierung beinhaltet. Der grundsätzli-

che Zeitablauf beim Empfang wurde bereits im Abschnitt 4.4.1 vorgestellt. Wie in Abb. 16 gezeigt, charakterisieren vier Zeiten die Verarbeitung eines Frames. T_{sync} bestimmt die Zeit, die für die Erkennung und Synchronisation eines Frames anhand der Präambel benötigt wird. Das zugrunde liegende Verfahren zur Framesynchronisation, welches neben dem Erkennen eines Frames auch die Frequenzsynchronisation und die Kanalschätzung beinhaltet, ist ausführlich in [49] dargestellt und wird hier nicht weiter erläutert. T_{sfv} beinhaltet die komplette Signalfeldverarbeitung bis zu dem Zeitpunkt, an dem die Auswertung des Signalfeldes abgeschlossen ist und die Übertragungsparameter des aktuellen Frames bereitstehen. T_{sof} beschreibt die eigentliche Latenz der Basisbandverarbeitung für die Daten, nämlich die Zeit bis zur Ausgabe des ersten Datenwortes an der Schnittstelle zum MAC. Nach einer zusätzlichen Zeit T_{lf} wird das letzte Datenwort des Frames an der MAC-Schnittstelle ausgegeben.

Die Zeiten T_{sfv} und T_{sof} basieren jeweils auf dem Zeitpunkt der vollständigen Framesynchronisation. Zur Bestimmung der Gesamtverzögerung zwischen dem Framebeginn am Empfängereingang und dem Start der Datenausgabe ist also die Zeit T_{sync} noch zu addieren. Während T_{sync} entsprechend der Kanaleigenschaften variabel ist, sind T_{sfv} und T_{sof} unabhängig von diesen und können als feste Zeiten angegeben werden. Einschränkend ist festzustellen, dass T_{sof} zwar unabhängig von den Kanaleigenschaften, jedoch nicht unabhängig von den Übertragungsparametern ist. Offensichtlich gibt es eine höhere Verarbeitungs- und damit Verzögerungszeit bis zur Ausgabe des ersten Datenwortes, wenn der optionale äußere RS-Code verwendet wird. Es gilt:

$$T_{sof} = f(\text{Modulationstyp, Punktierungsschema, RS-Kodierung}) \tag{29}$$

Die vierte in Abb. 16 dargestellte charakteristische Größe T_{lf} beschreibt die Zeit bis zur Ausgabe des letzten Datenwortes an der MAC-Schnittstelle. Als Nullpunkt ist der Zeitpunkt der Ausgabe des ersten Datenwortes gewählt, sodass T_{lf} im Prinzip die Länge einer Ausgabesequenz beschreibt. Im Gegensatz zum Sender, bei dem am Ausgang ein kontinuierlicher Datenstrom bereitgestellt werden muss, ist der Ausgangsdatenstrom des Empfängers nicht kontinuierlich. Die Datenrate an der Empfängerschnittstelle zum MAC muss mindestens der maximalen spezifizierten Übertragungsrate des Kommunikationssystems entsprechen. Werden niedrigere Datenraten, z. B. durch eine Änderung in der Modulationsart, verwendet, entstehen Lücken im Ausgangsdatenstrom. Somit ist T_{lf} genau wie T_{sof} abhängig von den gewählten Übertragungsparametern, zusätzlich jedoch auch von der Anzahl der gesendeten Symbole und der verwendeten Streams, also:

$$T_{lf} = f(\text{Modulationstyp, Punktierung, RS-Kodierung, Symbolanzahl, Streamanzahl}) \tag{30}$$

Die sich aus Symbolanzahl und Modulationstyp ergebende Anzahl an gesendeten Datenwörtern beeinflusst offensichtlich die Länge einer Ausgabesequenz. Wie später ausgeführt wird, kann die anfängliche Verzögerung der Signalfeldauswertung durch die höhere Verarbeitungsleistung des Empfängers in Abhängigkeit von der Symbolanzahl und den Dekodie-

rungsstreams verringert werden. Daraus ergibt sich eine mögliche Verkürzung von T_{lf}. Bevor diese Ableitung erfolgt, werden zunächst die Zeiten T_{sync}, T_{sfv} und T_{sof} genauer analysiert.

Wie bereits angemerkt, ist der vom Beginn eines Frames gemessene Zeitpunkt der Frame-synchronisation nicht fest, sondern abhängig von den Kanaleigenschaften. Der Zeitpunkt der Frameerkennung schwankt in Abhängigkeit des Rauschens. Trotz der unterschiedlichen Verarbeitungsschritte für die Präambel mit Frameerkennung sowie grober und feiner Zeitsynchronisation und der Kanalschätzung lässt sich die Latenz der Präambelverarbeitung durch die fixe Präambellänge und eine variable Verzögerungszeit $\tilde{T}_{pp}$ modellieren:

$$T_{sync} = T_{pre} + \tilde{T}_{pp} \tag{31}$$

$\tilde{T}_{pp}$ berücksichtigt nicht nur die festen Verarbeitungszeiten, sondern auch die Variabilität des Frameerkennungszeitpunktes durch das rauschbehaftete Eingangssignal sowie die notwendige Umsynchronisation zwischen den einzelnen Taktdomänen aufgrund des globalen Steuerungsmoduls. Unter idealen Bedingungen ergibt sich $\tilde{T}_{pp}$ zu 570 Takten in der 270 MHz Taktdomäne, der Anteil der Verarbeitungszeit an T_{sync} beträgt rund 35 %. Durch Rauschen ergibt sich eine simulativ ermittelte Schwankung des Frameerkennungszeitpunktes um bis zu 30 Takte.

Nachdem die Framesynchronisation erfolgreich vorgenommen wurde, folgt die Verarbeitung des Signalfeldes. Entsprechend der einzelnen Verarbeitungsschritte wie Kanalkorrektur, Demapping, Deinterleaving und Dekodierung lässt sich T_{sfv} wie folgt formulieren:

$$T_{sfv} = T_{sffs} + T_{sfdd} + T_{sfde} + T_{sfchk} \tag{32}$$

T_{sffs} umfasst die Zeit für die Frontstage-interne Verarbeitung, also die Zeitspanne zwischen dem Erkennen eines Frames und der Übergabe des ersten Unterträgers des Signalfeld-OFDM-Symbols an den Demapper. Die Verzögerungszeit durch Demapping und Deinterleaving wird in T_{sfdd} zusammengefasst. Der Demapper selbst erzeugt keine nennenswerte Verzögerung, den größten Beitrag liefern noch die enthaltenen Multiplizierer zur Gewichtung der Ergebnisse mit den Aussteuerungskoeffizienten. Dieser kurzen Zeitspanne von einigen Takten steht der Blockdeinterleaver gegenüber, bei dem erst der gesamte Datenblock eingeschrieben werden muss, bevor die Ausgabe beginnen kann. Die für die Kanaldekodierung benötigte Zeit wird mit T_{sfde} beschrieben und beinhaltet die vollständige Dekodierung des Signalfeldes, also die Zeit bis zur Ausgabe des letzten Signalfeld-Datenwortes. Im Anschluss an die Dekodierung müssen anhand einer CRC-Prüfsumme und einer Plausibilitätsprüfung die dekodierten Parameter auf Gültigkeit überprüft werden. Dies erzeugt den letzten Verzögerungsbeitrag T_{sfchk}.

Während die ersten drei Bestandteile T_{sffs}, T_{sfdd} und T_{sfde} parameterfrei und somit unabhängig von den gewählten Übertragungsparametern sind, ist T_{sfchk} von diesen abhängig:

$$T_{sfchk} = f(\text{CRC-Fehler, RS-Kodierung, Symbolanzahl, Streamanzahl}) \tag{33}$$

Der Algorithmus zum Überprüfen des Signalfeldes auf Plausibilität entspricht dabei weitestgehend dem Algorithmus zur Signalfeld-Parameterberechnung. Neben dem Plausibilitätstest werden auch weitere für die folgende Datenverarbeitung notwendige Parameter, wie die Anzahl der hinzugefügten Pad-Bytes, berechnet, sodass die Zeit T_{sfchk} auch bei einem Verzicht auf einen Plausibilitätstest nicht deutlich verringert werden kann. T_{sfchk} setzt sich entsprechend:

$$T_{sfchk} = T_{p200} \cdot \left(T^*_{crcchk} + \begin{cases} 0 & | \text{CRC - Fehler aufgetreten} \\ T^*_{sw} + T^*_{rsb} + T^*_{rso} + T^*_{by} + T^*_{ns2} + T^*_{plchk} & | \text{kein CRC - Fehler aufgetreten} \end{cases} \right) \tag{34}$$

aus den Zeiten T_{crcchk} für den Prüfsummentest, T_{plchk} für den Plausibilitätstest, T_{by} für die Berechnung der Anzahl von hinzugefügten Pad-Bytes, T_{ns2} zur Berechnung der Symbolanzahl sowie den bereits im vorherigen Abschnitt als Bestandteil von T_{pv} aufgeführten Größen T_{sw}, T_{rsb} und T_{rso} zusammen. Der einzige Unterschied für die drei letztgenannten ergibt sich durch den verwendeten Takt. Die Auswertung wird in der 200-MHz-Taktdomäne vorgenommen, sodass die Taktanzahl jeweils mit T_{p200} zu multiplizieren ist, um die entsprechende Verarbeitungszeit zu erhalten. T_{ns2} unterscheidet sich von T_{ns} dahingehend, dass die Anzahl an Takten unabhängig von der Anzahl an hinzugefügten Pad-Bytes ist.

Entsprechend der vorgenommenen Implementierung ergeben sich die folgenden Zeiten:

$$\begin{aligned} &T_{sffs} = 609 \cdot T_{p200} \qquad T_{sfdd} = 115 \cdot T_{p200} \qquad T_{sfde} = 561 \cdot T_{p200} \\ &T^*_{crcchk} = 1 \qquad T^*_{ns2} = 21 \qquad T^*_{plchk} = 3 \\ &T^*_{by} = \begin{cases} 2 & | \text{RS - Kodierung aus} \\ 5 & | \text{RS - Kodierung an} \end{cases} \end{aligned} \tag{35}$$

In Abhängigkeit von der Verwendung der RS-Kodierung und der CRC-Fehlerfreiheit lässt sich T_{sfv} zusammenfassen zu:

$$T_{sfv} = 1285 \cdot T_{p200} + \begin{cases} 1 \cdot T_{p200} & | \text{CRC - Fehler} \\ 30 \cdot T_{p200} & | \text{RS - Kodierung aus} \wedge \text{kein CRC - Fehler} \\ 54 \cdot T_{p200} & | \text{RS - Kodierung an} \wedge (N_{res} < 48) \wedge \text{kein CRC - Fehler} \\ 55 \cdot T_{p200} & | \text{RS - Kodierung an} \wedge (N_{res} \geq 48) \wedge \text{kein CRC - Fehler} \end{cases} \tag{36}$$

Bei einer fehlerfreien Übertragung mit RS-Kodierung entsteht durch die Signalfeldauswertung eine Verzögerungszeit von rund 6,7 µs, ohne RS-Kodierung verkürzt sie sich geringfügig

auf 6,6 µs. Auf Frontstage und Datenpfad entfällt dabei jeweils etwa die Hälfte der Gesamtzeit. Für die folgende Ableitung von T_{sof} sei vorweggenommen, dass die Zeit T_{dd} für das Demapping und Deinterleaving der Datensymbole unabhängig von der Modulationsart ist und der ermittelten Zeit T_{sfdd} entspricht. Da im Beispielsystem die Blockgröße des Deinterleavers der Datenanzahl innerhalb eines OFDM-Symbols entspricht, steigt die Blockgröße bei Verwendung einer höheren Modulationsart an. Es müssen mehr Daten eingeschrieben werden, bevor die Ausgabe beginnen kann. Somit steigt die native Latenz des Blockdeinterleavers. Gleichzeitig erhöht sich jedoch bei einer höheren Modulationsart mit einer konstanten Symbolrate am Eingang des Demappers dessen Ausgangsdatenrate um den gleichen Faktor. Damit verringert sich die für das Einschreiben in den Blockdeinterleaver benötigte Zeit. Die Gesamtverzögerung aus Demapper und Deinterleaver ist demzufolge unabhängig von der Modulationsart und damit konstant. Dies gilt für alle vergleichbaren Systemarchitekturen mit einer konstanten Symbolrate am Eingang des Demappers.

Für T_{sof} wurde als Bezug der Zeitpunkt der vollständigen Framesynchronisation gewählt, obwohl die Datensymbole erst nach der Signalfeldauswertung und der Signalisierung *para_valid* vom Frontstage ausgegeben und im Datenpfad verarbeitet werden können. Dies begründet sich in der Implementierung des digitalen Frontstages. Im Gegensatz zum Datenpfad, der stark modularisiert ist und größtenteils auf globale Kontroll- und Steuerungslogik verzichtet, basiert das Frontstage auf einem einzigen globalen Steuerungsmodul. Der Steuerungsprozess arbeitet in einem Raster von Zeitschlitzen, die sich aus den FFT-Zyklen und somit aus der Symboldauer ergeben. Der Startpunkt des ersten Zeitschlitzes wird dabei durch den Zeitpunkt der erfolgten Framesynchronisation bestimmt. Nur zu Beginn eines Zeitschlitzes sind Parameterübernahmen möglich, ansonsten verzögert sich die Übernahme bis zum nächsten Zeitschlitz. Unabhängig von den Einflussfaktoren CRC-Fehler und RS-Kodierung liegt die Signalisierung von *para_valid* in der vorliegenden Implementierung immer innerhalb ein und desselben Zeitschlitzes, sodass die Parameter immer zum gleichen Zeitpunkt übernommen werden. Damit erfolgt auch die Ausgabe des ersten Datensymbols aus dem Frontstage stets zu dem gleichen Zeitpunkt in Bezug auf den Zeitpunkt der Framesynchronisation. Würde T_{sof} stattdessen auf *para_valid* bezogen, enthielte es einen variablen Anteil entsprechend der verbleibenden Zyklusdauer zum Zeitpunkt der Signalisierung. Aus Gründen einer einfacheren Modellierung ist die Wahl der Framesynchronisation als Bezugspunkt vorteilhaft. Gleichzeitig werden vorerst die optionale RS-Kodierung und die Verwendung von Midambeln ausgeschlossen.

Somit lässt sich T_{sof} analog zur Signalfeldverarbeitung formulieren als:

$$T_{sof} = T_{dfs} + T_{dd} + T_{de1} + T_{do} \tag{37}$$

T_{dfs} beschreibt die Zeit zwischen der Framesynchronisation und dem Beginn der Ausgabe des ersten Datensymbols aus dem Frontstage, T_{dd} die Zeit für das Demapping und Deinterleaving,

T_{de1} die Zeit für die Dekodierung des ersten Datenwortes und T_{do} die Zeit für das Durchlaufen des Datenscramblers und der Ausgabe-FIFO, die eine Anpassung an die Taktdomäne der MAC-Schnittstelle vornimmt.

Die parameterunabhängigen Bestandteile von T_{sof} ergeben sich zu:

$$\begin{aligned} T_{dfs} &= 1405 \cdot T_{p200} & T_{de1} &= 229 \cdot T_{p200} \\ T_{dd} &= T_{sfdd} = 115 \cdot T_{p200} & T_{do} &= 8 \cdot T_{p125} \pm \frac{T_{p200}}{2} \end{aligned} \tag{38}$$

Die Bestimmung von T_{lf} erfolgt zunächst wieder unter Ausschluss der RS-Kodierung und der Midambelverwendung. T_{lf} gibt die Länge des Datenwortbereiches eines Frames bei der Ausgabe zum MAC an. Soll die gesamte Latenzzeit des Frameempfangs bestimmt werden, sind T_{sync} und T_{sof} zu T_{lf} zu addieren. Bei der Bestimmung von T_{lf} muss berücksichtigt werden, dass durch die geringere Datenrate eines einzelnen Dekodierungstreams die notwendige Zeit zur Dekodierung eines Symbols größer als T_{sym} ist. Erst durch die parallele Verarbeitung ergibt sich nach einer anfänglichen Zusatzverzögerung T_{deo} eine Zunahme von T_{lf} um T_{sym}.

T_{deo} spiegelt die aus der geringeren Datenrate resultierende zusätzliche Verzögerungszeit im Dekodierer wider und ist abhängig von der Anzahl der zu verarbeitenden Datenwörter pro Symbol, also von Modulationsart und Punktierungschema. Am Ausgang des Dekodierers muss die Reihenfolge der Symbole wieder mit der Eingangsreihenfolge übereinstimmen. Somit kann die Ausgabe des zweiten Symbols erst erfolgen, wenn das erste Symbol vollständig verarbeitet und ausgeben wurde.

Zusammenfassend stellt sich T_{lf} dar als:

$$T_{lf} = T_{deo} + \left(N_{sym} - 1\right) \cdot T_{sym} \tag{39}$$

mit

$$T_{deo} = \begin{cases} 293 \cdot T_{p200} & \mid \text{Modulationstyp} = \text{BPSK - } \frac{1}{2} \\ 419 \cdot T_{p200} & \mid \text{Modulationstyp} = \text{BPSK - } \frac{2}{3} \\ 677 \cdot T_{p200} & \mid \text{Modulationstyp} = \text{QPSK - } \frac{1}{2} \\ 929 \cdot T_{p200} & \mid \text{Modulationstyp} = \text{QPSK - } \frac{2}{3} \\ 1445 \cdot T_{p200} & \mid \text{Modulationstyp} = \text{16 - QAM - } \frac{1}{2} \\ 1955 \cdot T_{p200} & \mid \text{Modulationstyp} = \text{16 - QAM - } \frac{2}{3} \\ 2213 \cdot T_{p200} & \mid \text{Modulationstyp} = \text{16 - QAM - } \frac{3}{4} \end{cases} \tag{40}$$

Während das Signalfeld im Datenpfad dekodiert wird, werden die einlaufenden Datensymbole im Frontstage bereits verarbeitet und in einem Ausgabepuffer zwischengespeichert. Die Verarbeitung der einzelnen Symbole erfolgt dabei in einem festen Zeitraster, welches durch

die Dauer eines Symbols auf dem Kanal festgelegt ist. Da sowohl die Datenrate der Schnittstelle zwischen Frontstage und Datenpfad als auch der maximale Datendurchsatz des Datenpfades selbst größer sind als die mit den spezifizierten Übertragungsparametern erzielbare maximale Datenrate, können die im Puffer gespeicherten Symbole schneller ausgegeben und verarbeitet werden als neue Symbole einlaufen. Auf diese Weise kann die anfängliche Verzögerung durch die Signalfelddekodierung wieder verringert werden. Die Verringerung wirkt sich dabei nicht auf die Ausgabe des ersten Datensymbols eines Frames aus, sondern nur auf die Ausgabe der folgenden Datensymbole. Zur besseren Modellierung wird $T_{dw}(n_{sym})$ als Zeit zwischen dem Erkennen eines Frames sowie dem Ausgabestart des n_{sym}-ten OFDM-Datensymbols aus dem Frontstage eingeführt. Für einen Eingangsstrom von fortlaufend nummerierten Symbolen gilt in Abhängigkeit einer Schwelle N_{th}:

$$T_{dw}(n_{sym}+1) < T_{dw}\left(n_{sym}\right) + T_{sym} \qquad \forall\, n_{sym} < N_{th} \tag{41}$$

und

$$T_{dw}(n_{sym}+1) = T_{dw}\left(n_{sym}\right) + T_{sym} \qquad \forall\, n_{sym} \geq N_{th}\,. \tag{42}$$

Der Zeitpunkt des Ausgabebeginns des zweiten OFDM-Datensymbols $T_{dw}(2)$ entspricht also nicht der Summe von $T_{dw}(1)$ und der Dauer eines OFDM-Symbols auf dem Kanal, sondern ist aufgrund der Vorverarbeitung innerhalb des Frontstages kürzer. Da diese burstweise Verarbeitung auf Symbolen basiert und auch die Datenflusssteuerung auf die Symboldauer ausgerichtet ist, lässt sich die unterschiedliche Verzögerung durch eine variable Verarbeitungssymboldauer $T_{symvar}(n_{sym})$ beschreiben. In Abhängigkeit der Schwelle N_{th} ist T_{symvar} kleiner oder gleich T_{sym}.

Die variable Symboldauer T_{symvar} lässt sich anhand eines FIFO-Modells des Frontstage-Ausgabepuffers beschreiben. In diese FIFO werden mit der Rate:

$$R_{in} = \frac{1}{T_{sym}} \tag{43}$$

Symbole eingeschrieben. Da während der Signalfeldauswertung keine Symbole ausgegeben werden, sind zum Beginn der Datensymbolausgabe bereits:

$$N_{symfifo} = R_{in} \cdot (T_{sfv} - T_{dfs}) \tag{44}$$

Symbole in der FIFO gespeichert. T_{sfv} wird in (44) um T_{dfs} verringert, da auch die folgenden Datensymbole im Frontstage dieser Verzögerung unterliegen und somit nur die Verzögerungszeit der Signalfeldverarbeitung im Datenpfad für die Anzahl der Symbole in der Ausgabe-FIFO entscheidend ist. Solange in der FIFO Daten vorhanden sind, kann die Ausgabe mit der Rate $R_{out} > R_{in}$ erfolgen. Sind alle gespeicherten Symbole ausgegeben, verringert sich die

Ausgaberate auf $R_{out} = R_{in}$. Die anfängliche Ausgabedatenrate entspricht dabei unter Berücksichtigung etwaiger notwendiger Wartetakte der maximalen Datenrate auf der Schnittstelle zwischen Frontstage und Datenpfad. Mit T_{min} als der minimalen Zeit für die Übertragung eines Symbols auf der Schnittstelle ergibt sich die anfängliche Ausgabedatenrate zu:

$$R_{out} = \frac{1}{T_{min}} \tag{45}$$

Anhand der Überschussrate $R_{out} - R_{in}$ lässt sich die Zeit $T_{outfifo}$ bestimmen, die für die Ausgabe der $N_{symfifo}$ gespeicherten Symbole benötigt wird:

$$T_{outfifo} = \frac{N_{symfifo}}{R_{out} - R_{in}} \tag{46}$$

Damit ergibt sich für die Schwelle N_{th}:

$$N_{th} = \left\lfloor \frac{T_{outfifo}}{T_{sym}} \right\rfloor \tag{47}$$

und für die variable Symboldauer:

$$T_{symvar}(n_{sym}) = \begin{cases} T_{min} & | \, n_{sym} \le N_{th} \\ T_{Nth} & | \, n_{sym} = N_{th} + 1 \\ T_{sym} & | \, n_{sym} > N_{th} + 1 \end{cases} \tag{48}$$

T_{Nth} entspricht der um eine Wartezeit T_{wait} verlängerten Periode T_{min}, da in der FIFO zum Zeitpunkt $N_{th} \cdot T_{sym}$ noch der Teil eines Symbols gespeichert sein kann, sodass nach:

$$T_{wait} = \left(1 - (N_{symfifo} - \lfloor N_{symfifo} \rfloor)\right) \cdot R_{in} \tag{49}$$

die Ausgabe beginnen kann.

Mit der mittleren Symboldauer:

$$\overline{T}_{sym}(n_{sym}) = \frac{\sum_{m=1}^{n_{sym}} T_{symvar}(m)}{n_{sym}} \tag{50}$$

und $T_{dw}(1) = T_{dfs}$ als fester Verzögerungszeit für die Ausgabe des ersten Symbols ergibt sich T_{dw} für ein beliebiges Symbol mit $n_{sym} > 1$ zu:

$$T_{dw}(n_{sym}) = T_{dw}(1) + \overline{T}_{sym}(n_{sym}) \cdot n_{sym} \tag{51}$$

Durch die Variabilität der Symboldauer bei der Ausgabe aus dem Frontstage ändert sich auch die Gesamtzeit T_{lf} für die Dekodierung eines Frames zu:

$$T_{lf} = T_{deo} + (N_{sym} - 1) \cdot \overline{T}_{sym} \tag{52}$$

Anhand der eingeführten variablen Symboldauer T_{symvar} lässt sich die durch die Verwendung von Midambeln entstehende zusätzliche Latenz auf eine einfache Weise beschreiben. Die Durchführung der auf den Midambeln basierenden erneuten Kanalschätzung erfolgt parallel zur Verarbeitung der auf die Midambel folgenden Datensymbole. Somit werden diese Datensymbole jeweils nur um die akkumulierte Länge der vorhergehenden Midambeln zusätzlich verzögert. Für das direkt auf eine Midambel folgende Datensymbol muss die variable Symboldauer für dieses Symbol also nur um die Midambellänge vergrößert werden. Durch die sich daraus ergebende Vergrößerung der mittleren Symboldauer wird der Midambeleinfluss auf die weiteren Datensymbole erfasst.

Wird die optionale äußere RS-Kodierung verwendet, ergeben sich Änderungen für T_{sof} und T_{lf}. Zum einen ist für die Ausgabe die Verzögerungszeit des RS-Dekodierers T_{rs} zu berücksichtigen. Zum anderen kann die Blockdekodierung immer erst dann starten, wenn der gesamte Block in den Dekodierer eingeschrieben wurde. Zur Modellierung dieser zusätzlichen Verzögerungszeit für den Start einer Frameausgabe wird der Term T_{wrs} eingeführt. Damit ergibt sich T_{sof} zu:

$$T_{sof} = T_{dfs} + T_{dd} + T_{de1} + T_{wrs} + T_{rs} + T_{do} \tag{53}$$

Für die Ermittlung der Verzögerungszeit T_{rs} wird ein kontinuierlicher Eingangsstrom vorausgesetzt. T_{rs} bestimmt also die Zeit zwischen dem Einschreiben des ersten Datenwortes eines Blocks in den RS-Dekodierer und der Ausgabe des ersten dekodierten Datenwortes:

$$T_{rs} = 548 \cdot T_{p200} + 8 \cdot T_{p125} + \tilde{T}_{cdcfifo} \tag{54}$$

Der Term $\tilde{T}_{cdcfifo}$ ergibt sich durch die Taktumsynchronisation in der asynchronen FIFO-Struktur und kann abhängig von der jeweiligen Phasenlage beliebige Werte zwischen 0 und T_{p200} annehmen.

Zur Bestimmung von T_{wrs} muss zunächst die Modellierung für die übertragungsparameterabhängige Verzögerung des Kanaldekodierers im allgemeinen Fall vorgenommen werden. $T_{de}(n_{sym})$ mit $n_{sym} > 1$ beschreibt dabei die Zeit zwischen der Eingabe des ersten Datenwortes des ersten Symbols in die Kanaldekodierung und der Ausgabe des ersten dekodierten Datenwortes des n_{sym}-ten Symbols. Für diese Modellierung sind die Auswirkungen der Viterbide-

kodierung auf den gewählten Parallelisierungsansatz durch mehrere Streams zu beachten. So kann aufgrund der Reihenfolgenwiederherstellung ein Symbol erst nach vollständiger Ausgabe des vorherigen ausgegeben werden. Durch die fortlaufende Faltungskodierung sind jedoch die Symbole eines Streams nicht voneinander unabhängig, sondern stellen einen kontinuierlichen Datenstrom dar. Sofern dieser nicht terminiert wird, können die letzten Daten des n-ten Symbols eines Streams erst nach der Eingabe der ersten Daten des (n+1)-ten Symbols dieses Streams ausgegeben werden. Erst mit der Terminierung zum Ende des letzten Symbols eines Streams ist die direkte Ausgabe möglich. Damit ergibt sich eine Abhängigkeit von der Anzahl der verwendeten Streams. Abhängig von der Gesamtanzahl an Datensymbolen N_{sym} und der Streamanzahl N_{stream} lassen sich drei Fälle für $T_{de}(n_{sym})$ unterscheiden:

$$\begin{aligned} &1.\ N_{sym} = N_{stream} \\ &2.\ N_{sym} > N_{stream}, \quad n_{sym} \le N_{sym} - N_{stream} \\ &3.\ N_{sym} > N_{stream}, \quad n_{sym} > N_{sym} - N_{stream} \end{aligned} \tag{55}$$

Entspricht wie im ersten Fall die Anzahl der verwendeten Streams der Gesamtanzahl an Datensymbolen, stellt jedes Symbol die gesamten Daten eines Streams dar und ist somit terminiert. Die Ausgabe erfolgt entsprechend der Eingabe fortlaufend mit der Symbolperiode T_{sym} und der anfänglichen Verzögerung T_{dvs} als Verarbeitungszeit eines Symbols. $T_{de}(n_{sym})$ ergibt sich im ersten Fall zu:

$$T_{de}(n_{sym}) = T_{dvs} + (n_{sym} - 2) \cdot \overline{T}_{sym}(n_{sym}) \ \mid N_{stream} = N_{sym} \tag{56}$$

T_{dvs} beschreibt analog zu T_{deo} in (39) die Verzögerung für die Symboldekodierung, diesmal allerdings mit anderen Bezugszeitpunkten. T_{dvs} ist übertragungsparameterabhängig und ergibt sich zu:

$$T_{dvs} = \begin{cases} 589 \cdot T_{p200} & \mid \text{Modulationstyp} = \text{BPSK - } \frac{1}{2} \\ 712 \cdot T_{p200} & \mid \text{Modulationstyp} = \text{BPSK - } \frac{2}{3} \\ 994 \cdot T_{p200} & \mid \text{Modulationstyp} = \text{QPSK - } \frac{1}{2} \\ 1238 \cdot T_{p200} & \mid \text{Modulationstyp} = \text{QPSK - } \frac{2}{3} \\ 1721 \cdot T_{p200} & \mid \text{Modulationstyp} = \text{16 - QAM - } \frac{1}{2} \\ 2248 \cdot T_{p200} & \mid \text{Modulationstyp} = \text{16 - QAM - } \frac{2}{3} \\ 2508 \cdot T_{p200} & \mid \text{Modulationstyp} = \text{16 - QAM - } \frac{3}{4} \end{cases} \tag{57}$$

Werden weniger Streams als Datensymbole verwendet, muss unterschieden werden, ob das aktuelle Datensymbol terminiert wird oder nicht. Diese Unterscheidung entspricht dem zweiten und dritten Fall von (55). Durch die Verwendung einer Anzahl von Streams, die über die für das gewählte Modulations- und Kodierungsschema minimal erforderliche Streamanzahl hinausgeht, ergeben sich Wartezeiten innerhalb der Symbolverarbeitung. Der Zeitabstand zwischen dem Einschreiben von zwei aufeinanderfolgenden Symbolen eines Streams ist nun

größer als die Symbolverarbeitungszeit innerhalb des Streams. Dieser zusätzliche Beitrag wird durch $T_{streamov}$ erfasst und beträgt in Abhängigkeit von $N_{streammin}$ als der minimal für das jeweilige Modulations- und Kodierungsschema erforderlichen Streamanzahl:

$$T_{streamov} = \left(N_{stream} - N_{streammin}\right) \cdot T_{sym} \tag{58}$$

Für $T_{streamov}$ ist die Symbolperiodendauer der terminierten Symbole heranzuziehen. Bei hinreichend langen Frames entspricht diese der anfänglich genannten konstanten Symbolperiode T_{sym}. Damit ergibt sich $T_{de}(n_{sym})$ für den zweiten Fall von (55) zu:

$$T_{de}(n_{sym}) = T_{dvs} + (n_{sym} - 2) \cdot \overline{T}_{sym} + T_{streamov} \mid \left(N_{sym} > N_{stream}\right) \wedge \left(n_{sym} \leq N_{sym} - N_{stream} + 1\right) \tag{59}$$

Die sich aus den Datenraten für die spezifizierten Modulations- und Punktierungsschemata ergebenden minimalen Streamanzahlen sind in Tabelle 3 dargestellt.

Modulationstyp	Minimale Streamanzahl
BPSK-½	4
BPSK-⅔	5
QPSK-½	8
QPSK-⅔	10
16-QAM-½	15
16-QAM-⅔	20
16-QAM-¾	24

Tabelle 3: Minimale Streamanzahlen nach Modulationstyp und Punktierungsschema

Im dritten Fall von (55) können durch die Terminierung die letzten Datenwörter eines Symbols direkt ausgegeben werden, ohne auf ein weiteres Folgesymbol zu warten. Da die maximale Ausgaberate des Kanaldekodierers (6,4 Gbit/s) den erforderlichen maximalen Datendurchsatz des Systems (3,89 Gbit/s) übersteigt, kann die Ausgabe mit dieser höheren Datenrate erfolgen. Somit wird der zusätzliche Verzögerungsanteil durch $T_{streamov}$ wieder reduziert und bis zur Ausgabe des letzten Symbols vollständig abgebaut. $T_{de}(n_{sym})$ ergibt sich in diesem Fall zu:

$$T_{de}(n_{sym}) = \max\left\{\left(T_{de}(n_{sym} - 1) + T_{out}\right), \left(T_{deo} + T_{de1} + \left(n_{sym} - 2\right) \cdot \overline{T}_{sym}\right)\right\} \mid n_{sym} > \left(N_{sym} - N_{stream} + 1\right) \tag{60}$$

T_{out} gibt die für die Ausgabe eines terminierten Symbols über die Ausgangsschnittstelle des Kanaldekodierers benötigte Zeit an. Die Auswahl des Maximums aus den beiden gegebenen Termen in (60) erklärt sich dadurch, dass ein Symbol ausgegeben werden kann, wenn es bereits verarbeitet und gleichzeitig das vorherige Symbol bereits ausgegeben wurde. Die Zeit T_{out} kann aus der Anzahl an Datenbits pro Terminierungssymbol, der Datenrate der Schnittstelle und der für den Streamwechsel benötigten Zeit ermittelt werden:

$$T_{out} = 3 \cdot T_{p200} + \begin{cases} 11 \cdot T_{p200} & | \text{Modulationstyp} = \text{BPSK - } \frac{1}{2} \\ 15 \cdot T_{p200} & | \text{Modulationstyp} = \text{BPSK - } \frac{2}{3} \\ 23 \cdot T_{p200} & | \text{Modulationstyp} = \text{QPSK - } \frac{1}{2} \\ 31 \cdot T_{p200} & | \text{Modulationstyp} = \text{QPSK - } \frac{2}{3} \\ 47 \cdot T_{p200} & | \text{Modulationstyp} = \text{16 - QAM - } \frac{1}{2} \\ 63 \cdot T_{p200} & | \text{Modulationstyp} = \text{16 - QAM - } \frac{2}{3} \\ 71 \cdot T_{p200} & | \text{Modulationstyp} = \text{16 - QAM - } \frac{3}{4} \end{cases} \quad (61)$$

Mithilfe der soeben erfolgten Ableitung der allgemeinen Verarbeitungszeit des Kanaldekodierers und der für die Daten eines RS-Blockes erforderlichen Symbolanzahl lässt sich der durch T_{wrs} modellierte parameterabhängige Latenzanteil für T_{sof} bei der Verwendung der RS-Kodierung wie folgt darstellen:

$$T_{wrs} = \begin{cases} T_{\text{de}}(6) - T_{de1} & | \text{Modulationstyp} = \text{BPSK - } \frac{1}{2} \\ T_{\text{de}}(4) - T_{de1} & | \text{Modulationstyp} = \text{BPSK - } \frac{2}{3} \\ T_{\text{de}}(3) - T_{de1} & | \text{Modulationstyp} = \text{QPSK - } \frac{1}{2} \\ T_{\text{de}}(2) - T_{de1} & | \text{Modulationstyp} = \text{QPSK - } \frac{2}{3} \\ T_{\text{de}}(2) - T_{de1} & | \text{Modulationstyp} = \text{16 - QAM - } \frac{1}{2} \\ 0 & | \text{Modulationstyp} = \text{16 - QAM - } \frac{2}{3} \\ 0 & | \text{Modulationstyp} = \text{16 - QAM - } \frac{3}{4} \end{cases} \quad (62)$$

Für (62) wurde vereinfachend der Start des letzten Symbols eines Blocks verwendet, auch wenn die RS-Dekodierung erst nach dem Einschreiben einer bestimmten Datenmenge dieses Symbol beginnen kann. Da T_{de1} bereits in T_{sof} enthalten ist, wurde der Bezugszeitpunkt von $T_{de}(n_{sym})$ entsprechend angepasst. Aufgrund des Aufbaus des Dekodierers aus drei Einzelmodulen, die jeweils einen einzelnen Block verarbeiten, ergibt sich eine weitere Besonderheit. Für die in (54) angegebene Latenz T_{rs} des RS-Dekodierers wurde ein kontinuierliches Einschreiben vorausgesetzt. Verlängert sich das Einschreiben durch periodische Wartezeiten gegenüber dem kontinuierlichen Einschreiben eines Blocks um einen Faktor ≤ 4, hat diese Verlängerung durch die Anpassung des 32-Bit-Eingangsdatenwortes auf eine 8-Bit-Verarbeitungswortbreite keinen Einfluss auf die Gesamtlatenz der zusätzlichen RS-Dekodierung, also die Summe von T_{wrs} und T_{rs}. Dies gilt nur bei einer gleichmäßigen Verteilung von Daten und Pausen. Die in (62) gegebenen Wartezeiten stellen somit den abgeschätzten schlechtesten Fall dar.

Die zusätzliche Verlängerung der Frameausgabe bei Verwendung des RS-Dekodierers lässt sich unter Verwendung von $T_{de}(n_{sym})$ modellieren. Spätestens mit der Ausgabe des letzten Datenwortes aus dem Kanaldekodierer kann die RS-Dekodierung beginnen. Somit beträgt die maximale Framelänge:

$$\begin{aligned} T_{lf} &= T_{de}\left(N_{sym}\right) + T_{last} + T_{dfs} + T_{dd} + T_{rs} + T_{do} - T_{sof} = \\ &= T_{de}\left(N_{sym}\right) + T_{last} - T_{wrs} - T_{de1} \end{aligned} \tag{63}$$

T_{last} kennzeichnet dabei die Ausgabedauer für den letzten RS-Block und beträgt 480 ns für eine kontinuierliche Ausgabe. Werden nicht alle kanaldekodierten Datenwörter zusätzlich durch den RS-Dekodierer verarbeitet, verringert die Framedauer sich entsprechend der Differenz aus dem Zeitpunkt der Ausgabe des letzten kanaldekodierten Datenwortes und der Ausgabe des letzten durch den RS-Dekodierer zu verarbeitenden Datenwortes zuzüglich der Zeit für die anschließende Ausgabe dieser überzähligen Datenwörter. Die nicht von N_{sym} abhängigen Terme in (63) – mit Ausnahme von T_{last} – dienen zur Anpassung des Bezugszeitpunktes von $T_{de}(n_{sym})$ auf den Zeitpunkt des Beginns der Frameausgabe (T_{sof}). Wird keine RS-Kodierung verwendet, kann durch das Nullsetzen von T_{wrs} mit (63) auch die Framelänge für diesen Fall bestimmt werden, das Ergebnis ist identisch zur Formulierung (52).

Für (63) wurde vorausgesetzt, dass der RS-Dekodierer mindestens den gleichen Datendurchsatz wie das Interface zwischen Kanaldekodiererausgang und RS-Dekodierereingang bietet. Sollte dies nicht gegeben sein, resultiert die beschriebene, möglicherweise in der Endphase der Kanaldekodierung auftretende Ausgabe von Datenwörtern mit der maximalen Datenrate in einer größeren Verarbeitungszeit. Die schnellere Ausgabe von Datenwörtern muss über FIFOs gepuffert werden. Die Frameausgabe verlängert sich um die Verarbeitungszeit der zwischengespeicherten, nicht direkt verarbeiteten Datenwörter. Die Anzahl der zwischengespeicherten Datenwörter kann aus dem Verhältnis der Datenraten sowie der Dauer der schnellen Ausgabe bestimmt werden.

4.4.4 Zusammenfassung und Auswertung

Die in vorherigen Abschnitten 4.4.2 und 4.4.3 abgeleitete Modellierung der Basisbandlatenzen des EASY-A VHR-E Systems erfordert aufgrund der Systemkomplexität eine sehr umfangreiche mathematische Beschreibung. Die für den Sender essentiellen Zeiten des Starts einer Frameausgabe (T_{fs}) sowie der zeitlichen Dauer eines Datenframes (T_{fd}) lassen sich noch in einfachen Formulierungen zusammenfassen:

$$T_{fs} = \pm 1.85\,\text{ns} + \begin{cases} 46{,}85\,\text{ns} & \mid \text{Modulationstyp} = \text{BPSK} \vee \text{QPSK} \\ 905{,}25\,\text{ns} & \mid \text{Modulationstyp} = \text{16-QAM-}\frac{1}{2} \\ 1848{,}75\,\text{ns} & \mid \text{Modulationstyp} = \text{16-QAM-}\frac{2}{3} \\ 2144{,}75\,\text{ns} & \mid \text{Modulationstyp} = \text{16-QAM-}\frac{3}{4} \end{cases} \tag{64}$$

$$T_{fd} = 4502{,}9\,\text{ns} + N_{sym} \cdot 592\,\text{ns} + N_{mid} \cdot 1184\,\text{ns} \tag{65}$$

Für die in (64) und (65) angegebenen Zeiten wurden die Parameter des Beispielsystems mit einer Präambeldauer von rund 3,9 µs, einer OFDM-Symbolperiode von rund 593 ns sowie einer Midambeldauer von rund 1,2 µs zugrunde gelegt.

Die für den Sender wichtige Latenzzeit für den Start einer Frameausgabe T_{fs} beträgt je nach Modulations- und Punktierungsart zwischen ca. 45 ns und 2147 ns. Während die kleine Ausgabeverzögerung für BPSK und QPSK im Allgemeinen vernachlässigt werden kann, ist der große Anteil von ca. 0,9 - 2,1 µs für 16-QAM bereits signifikant, speziell bei der Verwendung kurzer Datenframes.

Im Gegensatz zur Latenzmodellierung des Senders ist die Modellierung des Empfängers erheblich komplexer. Zum einen wirken sich hier die unterschiedlichen Modulations-, Punktierungs- und Kodierungsschemata aus, zum anderen ergeben sich Wartezeiten aufgrund von Parameterabhängigkeiten. Durch einen Verarbeitungsleistungsüberschuss im digitalen Datenpfad in Verbindung mit einer Verkürzung der Symbolperioden für die ersten $N_{th} = 61$ Symbole (siehe (47) und (48)) wird versucht, die zusätzlichen Latenzbeiträge während der laufenden Frameverarbeitung zu verringern. Dies ist zum einen jedoch nur bei ausreichend langen Frames möglich, zum anderen hat eine solche Vorgehensweise keinen Einfluss auf die Verzögerungszeit bis zum Start der Datenausgabe.

Durch den gewählten Parallelisierungsansatz mit Streams als einer Variante der Blockparallelität ergeben sich im Zusammenspiel mit der äußeren RS-Kodierung weitere variable Latenzbeiträge. Wie später im Abschnitt 6.3 genauer ausgeführt wird, entstehen bei blockparallelen Verfahren zusätzliche Latenzen durch die Reihenfolgenwiederherstellung am Ausgang. Infolge der Abhängigkeiten der einzelnen Symbole eines Streams wird diese Auswirkung für den Streamansatz noch weiter verstärkt.

Aufgrund der soeben genannten Faktoren kann für den Empfänger keine einfache Gesamtformulierung der entstehenden Latenzen angegeben werden. Für den folgenden Überblick wurden die einzelnen Komponenten so weit wie möglich zusammengefasst. Gleichzeitig erfolgte eine Abschätzung des schlechtesten Falls für jitterbedingte Effekte wie dem ungenauen Zeitpunkt der Frameerkennung. Da sich die Zeiten der Signalfeldausgabe und der Ausgabe des ersten Datenwortes eines Frames im Allgemeinen auf den Start des Empfängereingangssignals beziehen, wurde der für die Modellierung gewählte Bezugszeitpunkt durch die Addition von T_{sync} entsprechend angepasst. Damit ergeben sich für die drei wichtigen Zeiten des Abschlusses der Signalfeldverarbeitung, der Ausgabe des ersten Datenwortes und der Ausgabe des letzten Datenwortes die folgenden Formulierungen:

$$T_{sync} + T_{sfv} = 12555{,}9\,\text{ns} + \begin{cases} 5\,\text{ns} & \text{| CRC - Fehler} \\ 150\,\text{ns} & \text{| RS - Kodierung aus} \wedge \text{kein CRC - Fehler} \\ 270\,\text{ns} & \text{| RS - Kodierung an} \wedge (N_{res} < 48) \wedge \text{kein CRC - Fehler} \\ 275\,\text{ns} & \text{| RS - Kodierung an} \wedge (N_{res} \geq 48) \wedge \text{kein CRC - Fehler} \end{cases} \tag{66}$$

$$T_{sync} + T_{sof} = 14942{,}4\,\text{ns} + \begin{cases} 0\,\text{ns} & \text{| RS - Kodierung aus} \\ 1664\,\text{ns} + T_{\text{de}}(6) & \text{| BPSK - } \frac{1}{2} \wedge \text{RS} - \text{Kodierung an} \\ 1664\,\text{ns} + T_{\text{de}}(4) & \text{| BPSK - } \frac{2}{3} \wedge \text{RS} - \text{Kodierung an} \\ 1664\,\text{ns} + T_{\text{de}}(3) & \text{| QPSK - } \frac{1}{2} \wedge \text{RS} - \text{Kodierung an} \\ 1664\,\text{ns} + T_{\text{de}}(2) & \text{| QPSK - } \frac{2}{3} \wedge \text{RS} - \text{Kodierung an} \\ 1664\,\text{ns} + T_{\text{de}}(2) & \text{| 16 - QAM - } \frac{1}{2} \wedge \text{RS} - \text{Kodierung an} \\ 2809\,\text{ns} & \text{| 16 - QAM - } \frac{2}{3} \wedge \text{RS} - \text{Kodierung an} \\ 2809\,\text{ns} & \text{| 16 - QAM - } \frac{3}{4} \wedge \text{RS} - \text{Kodierung an} \end{cases} \tag{67}$$

$$T_{lf} = T_{de}(N_{sym}) + \begin{cases} T_{last} - T_{de}(6) & \text{| BPSK - } \frac{1}{2} \wedge \text{RS} - \text{Kodierung an} \\ T_{last} - T_{de}(4) & \text{| BPSK - } \frac{2}{3} \wedge \text{RS} - \text{Kodierung an} \\ T_{last} - T_{de}(3) & \text{| QPSK - } \frac{1}{2} \wedge \text{RS} - \text{Kodierung an} \\ T_{last} - T_{de}(2) & \text{| QPSK - } \frac{2}{3} \wedge \text{RS} - \text{Kodierung an} \\ T_{last} - T_{de}(2) & \text{| 16 - QAM - } \frac{1}{2} \wedge \text{RS} - \text{Kodierung an} \\ T_{last} - 1145\,\text{ns} & \text{| 16 - QAM - } \frac{2}{3} \wedge \text{RS} - \text{Kodierung an} \\ T_{last} - 1145\,\text{ns} & \text{| 16 - QAM - } \frac{3}{4} \wedge \text{RS} - \text{Kodierung an} \\ T_{sym}(N_{sym}) - 1145\,\text{ns} & \text{| RS - Kodierung aus} \end{cases} \tag{68}$$

Für die Berechnung der variablen Symboldauer ist (50) heranzuziehen, $T_{de}(N_{sym})$ ergibt sich entsprechend des Verhältnisses aus Symbol- und Streamanzahl (siehe (55)) nach den Vorschriften (56) oder (60). Zur besseren Visualisierung der unterschiedlichen Latenzen sind in den Abb. 18 und 19 einige Ergebnisse für bestimmte Kombinationen aus Modulations- und Kodierungsart sowie verschiedenen Datenwortanzahlen dargestellt. Weiterhin wurde der Einfluss der variablen Streamanzahl mit erfasst. Für die Varianten ohne RS-Kodierung wurde die in Tabelle 3 (Seite 63) spezifizierte minimale Streamanzahl vorausgesetzt. Es ist zu beachten, dass die minimale Streamanzahl bei kurzen Frames unterschritten wird, sofern der Frame insgesamt aus weniger OFDM-Symbolen besteht. Daher beschränkt sich die Darstellung der 16-QAM-Modulation in Abb. 18 auf maximal 11 bzw. 13 Streams. Die Diagramme zeigen jeweils in der linken Säule die gesamte Empfängerlatenz, bestehend aus den drei Komponenten T_{sync}, T_{sof} und T_{lf}. Die rechte Säule zeigt die zeitliche Länge des entsprechenden Frames (T_{fd}) auf dem Kanal. Die Zahl oberhalb der Säulen gibt jeweils die Gesamtsumme der drei Komponenten T_{sync}, T_{sof} und T_{lf} an.

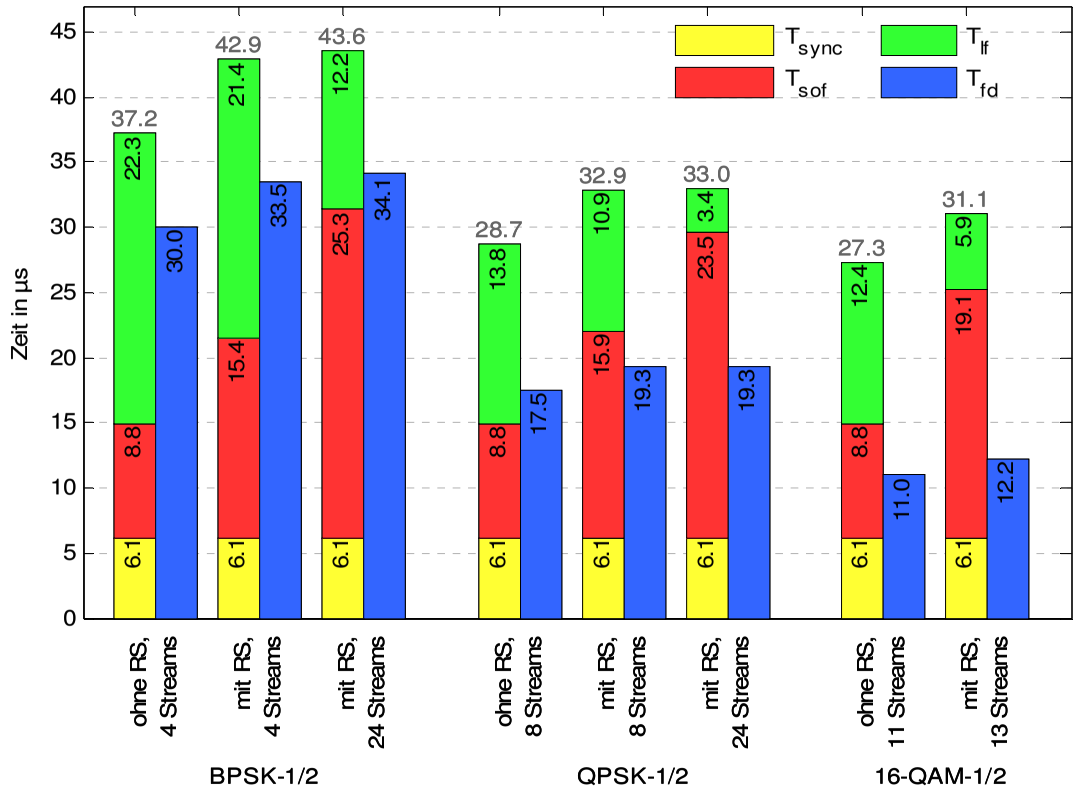

Abb. 18: Empfängerlatenz unterschiedlicher Übertragungsparameter, 2 kB Daten

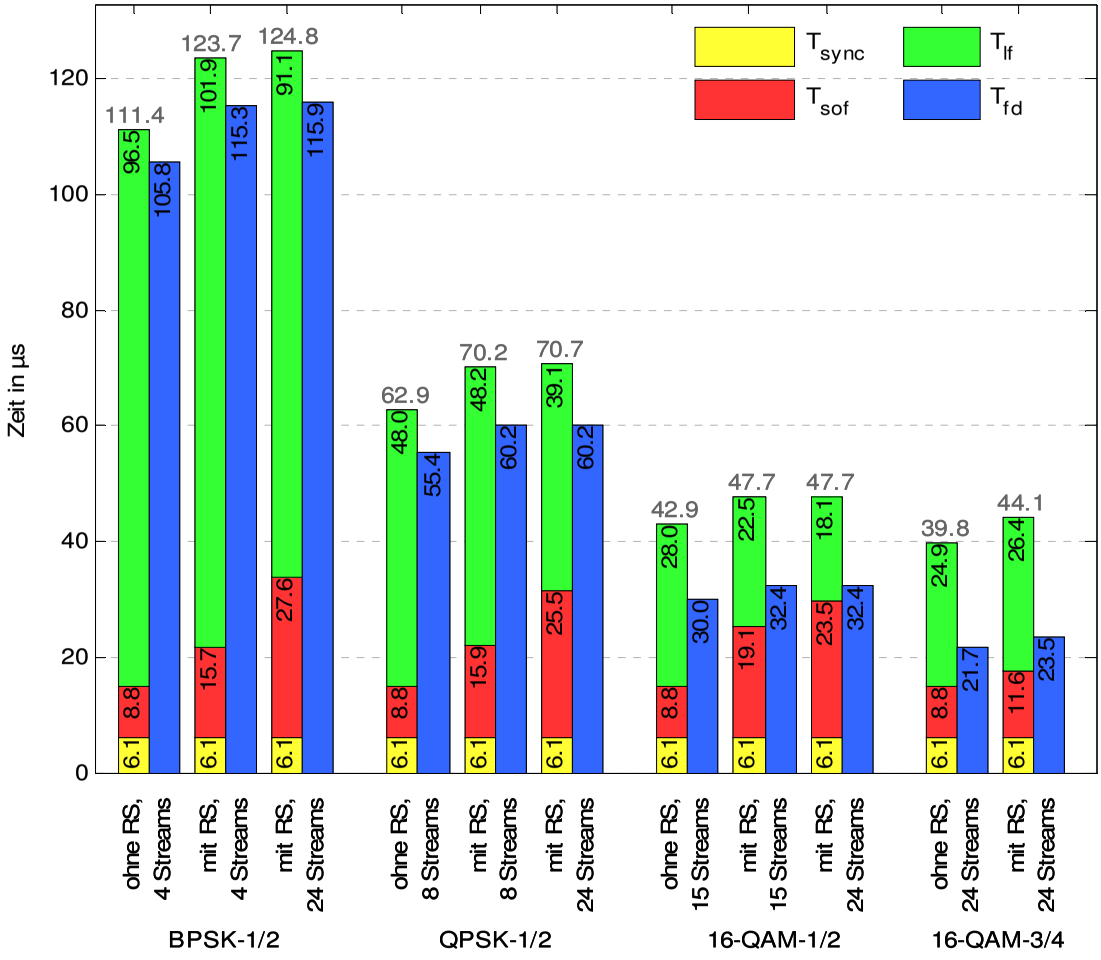

Abb. 19: Empfängerlatenz unterschiedlicher Übertragungsparameter, 8 kB Daten

Ohne Verwendung des äußeren RS-Codes wird beim Beispielsystem rund 15 µs nach dem Präambelbeginn am Empfängereingang das erste Datenwort an der Schnittstelle zum MAC ausgegeben. Für die Ausgabe des letzten Datenwortes ist nicht nur die zeitliche Länge der Datensymbolsequenz, sondern auch eine weitere Verzögerung durch die langsamere Verarbeitung innerhalb eines Streams zu berücksichtigen. Gleichzeitig ist die bereits genannte Verkürzung der Symboldauer zu beachten. So ist in den Diagrammen gut zu erkennen, dass bei kurzen Frames, also z. B. bei 2 kB Daten und einer 16-QAM-½-Modulation ohne RS-Kodierung bzw. bei 8 kB Daten und einer 16-QAM-¾-Modulation, T_{lf} in etwa T_{fd} entspricht oder sogar größer ist. Bei langen Frames, also der Verwendung einer niedrigeren Modulationsart oder einer größeren Datenanzahl, ist T_{lf} im Vergleich zu T_{fd} deutlich kleiner. Dies folgt direkt aus der beschriebenen Symboldauerverkürzung, die erst bei hinreichend langen Frames ein „Aufholen“ der initialen Verzögerung ermöglicht. Mit steigender Framelänge nähert sich die Gesamtsumme aus T_{sync}, T_{sof} und T_{lf} immer weiter der Framedauer T_{fd} an.

Bei der zusätzlichen Verwendung der RS-Kodierung vergrößert sich die Latenz bis zur Ausgabe des ersten Datenwortes zum einen um die Verzögerungszeit des RS-Dekodierers, zum anderen ergibt sich abhängig von der verwendeten Streamanzahl sowie der Modulations- und Punktierungsart eine weitere Wartezeit. Bei der Verwendung der minimal erforderlichen Streamanzahl entsteht für einen BPSK-½-modulierten Frame mit mehr als zehn Symbolen durch die RS-Kodierung eine weitere Verzögerung von rund 6,6 µs. Das erste Datenwort wird somit erst rund 21,5 µs nach dem Eingang des Framebeginns ausgegeben. Wird die maximale Anzahl von 24 Streams verwendet, vergrößert sich die Wartezeit auf die Ausgabe des ersten Datenwortes um ca. 65 %. Die Zeit T_{lf} verringert sich jedoch durch die höhere Ausgaberate des Dekodierers am Ende eines Frames, sodass die Ausgabe des letzten Datenwortes bei der Verwendung von 24 Streams nur geringfügig später erfolgt. Bei einer 16-QAM-¾-Modulation mit RS-Kodierung entsteht kein Beitrag durch eine Wartezeit auf das vollständige Blockeinschreiben, stattdessen entspricht die zusätzliche Latenz der Verzögerungszeit des RS-Dekodierers mit 2,8 µs.

Durch die Erhöhung der Streamanzahl lässt sich zwar der Durchsatz des Kanaldekodierers steigern, da aber gleichzeitig die maximale Eingangsdatenrate durch den Demapper begrenzt ist, kann diese Durchsatzsteigerung nicht zu einer weiteren Latenzreduktion für die bereits im Frontstage vorverarbeiteten Folgesymbole genutzt werden. Stattdessen entsteht durch die größere Streamanzahl eine weitere Latenz aufgrund der Blockabhängigkeiten. Es zeigt sich, dass nicht nur zur Reduzierung von T_{sof}, sondern auch zur Reduzierung der Gesamtlatenz zwischen dem Erkennen eines Frames und der Ausgabe des letzten Datenwortes eine geeignete Anzahl an Verarbeitungsstreams gewählt werden muss, die gerade nicht der maximal möglichen Streamanzahl entspricht. Dies ist insbesondere dann notwendig, wenn die Verarbeitungsleistung der dem Kanaldekodierer folgenden Stufen nicht der maximal möglichen Ausgabedatenrate des Kanaldekodierers, sondern nur der durchschnittlichen Datenrate an dieser Schnittstelle entspricht.

Basierend auf den bisherigen Erkenntnissen über die Latenzmodellierung wird im folgenden Abschnitt ein verallgemeinertes Latenzmodell entworfen. Mithilfe des gewählten Blackbox-Ansatzes für die einzelnen Basisbandverarbeitungsmodule, die nur anhand von wenigen globalen Parametern und Größen beschrieben werden, lassen sich einfach die Auswirkungen verschiedener Modulimplementierungen untersuchen.

4.5 Verallgemeinertes Latenzmodell

Im vorherigen Abschnitt 4.4 wurde ein Latenzmodell für das EASY-A VHR-E-System abgeleitet. Dieses Modell ist insbesondere für den Empfänger sehr komplex, da sich durch die Verkürzung der Symbolperioden und die parallele Streamarchitektur variable Wartezeiten ergeben. Die zusammengefassten Formulierungen (66), (67) und (68) für die Empfängerlatenzen ermöglichen zwar einerseits eine etwas einfachere Darstellung, andererseits sind die Beiträge der einzelnen Basisbandkomponenten nicht mehr zu erkennen.

Der modulare Aufbau des Basisbandprozessors legt nahe, ein verallgemeinertes Latenzmodell eines einzelnen Verarbeitungsmoduls zu definieren. In diesem Abschnitt wird als kurzer Exkurs ein solches verallgemeinertes Modell vorgestellt. Die Signalverarbeitung im Basisband ist im Allgemeinen datenflussorientiert, d. h. die Basisbandverarbeitung lässt sich als sequentielle Schaltung einzelner Module mit einem Datenein- und -ausgang modellieren. Ein entsprechend abstrahiertes Verarbeitungsmodell wurde bereits bei der Darstellung des detaillierten Aufbaus eines Kommunikationssystems in Abb. 2 (Seite 11) vorgestellt. Wie in dieser Abbildung zu erkennen ist, können in Abhängigkeit von der gewählten Abstraktionsebene auch in der Basisbandverarbeitung parallele Pfade und Module mit mehreren Eingängen auftreten, so z. B. die gezeigten parallelen, voneinander abhängigen Verarbeitungspfade für die Kanalschätzung und die Kanalentzerrung. Sofern jedoch nur der Datenfluss und insbesondere die Verzögerungen des Datenflusses beachtet werden, lässt sich eine solche parallele Struktur in eine sequentielle überführen. Die Kanalentzerrung kann erst mit dem Vorliegen der Ergebnisse der Kanalschätzung beginnen, dies entspricht einer Verzögerung des Datensymbolflusses um die für die Kanalschätzung benötigte Zeit. Die parallele Schaltung aus Kanalschätzungsmodul und Kanalentzerrungsmodul entspricht einer sequentiellen Schaltung dieser beiden Module, bei der der Datenfluss im Kanalschätzungsmodul verzögert wird. Daher wird das allgemeine Latenzmodell auf der Grundlage einer rein sequentiellen Basisbandmodellierung abgeleitet. Zum Ende dieses Abschnitts wird erläutert, wie sich das beschriebene Modell in ein Modell zur Berücksichtigung paralleler Pfade, also für Module mit mehreren Eingängen, erweitern lässt. Wie bereits aufgeführt, ist eine solche parallele Struktur in einem abstrakten datenflussorientierten Basisbandmodell unüblich.

Für die Latenzmodellierung wird ein Basisbandverarbeitungsmodul als blackboxbasierte Funktionseinheit abstrahiert. Die gesamte Basisbandverarbeitung besteht entsprechend der

vorgestellten Abstraktionsebene aus einer hintereinander geschalteten Kette solcher Funktionseinheiten. Zur Identifizierung werden die Funktionseinheiten mit dem Index m fortlaufend nummeriert, insgesamt besteht die Kette somit aus M einzelnen Funktionseinheiten.

Die abstrahierte Verarbeitung innerhalb einer Funktionseinheit erfolgt symbolbasiert. Ein Symbol kennzeichnet jeweils eine für den verwendeten Algorithmus zusammengehörende Menge an Daten. Es kann sich also um ein OFDM-Symbol, einen RS-Block oder auch nur um ein einzelnes Datenwort handeln. Damit lässt sich eine Funktionseinheit m für die Latenzmodellierung durch die Größen $T_{sof,m}$ und $T_{sym,m}$ charakterisieren. $T_{sof,m}$ bezeichnet die Verzögerungszeit zwischen dem Beginn des ersten Symbols am Einheit der Funktionseinheit und dem Beginn der Ausgabe des ersten Symbols durch diese Funktionseinheit. Die gesamte Verzögerungszeit T_{sof} der Basisbandverarbeitung ergibt sich als Summe aller Einzellatenzen:

$$T_{sof} = \sum_{m=1}^{M} T_{sof,m} \tag{69}$$

Die zweite Größe $T_{sym,m}$ bezeichnet die Symbolperiode innerhalb der Funktionseinheit und ist als Zeitdifferenz des Ausgabestarts von zwei aufeinanderfolgenden Symbolen definiert. Wartezeiten zwischen dem Ende einer Symbolausgabe und dem Beginn der Ausgabe des folgenden Symbols werden der ersten Symbolperiode hinzugerechnet. Für das letzte Symbol einer Ausgabesequenz bezieht sich die Symbolperiode nur auf die Ausgabezeit des letzten Symbols. $T_{sym,m}$ ist offensichtlich abhängig vom jeweils ausgegebenen Symbol $n_{sym,m}$. Mit $T_{sym,m}(n_{sym,m})$ werden unter anderem Wartezeiten aufgrund von Blockverarbeitung und Parameterabhängigkeiten modelliert. Gleichzeitig werden mögliche dynamische Symbolperiodenveränderungen erfasst. Die Symbole benachbarter Funktionseinheiten sind nicht notwendigerweise identisch. So können die Symbole aus einer unterschiedlichen Datenmenge bestehen. Dies ist z. B. beim Vergleich der Symbole des Kanaldekodierers mit den Symbolen des RS-Dekodierers der Fall. Die Symbolgröße des ersten entspricht der Datenmenge eines OFDM-Symbols, die des letzteren der RS-Blockgröße. Daraus folgt eine unterschiedliche Anzahl von Datensymbolen für die beiden Funktionseinheiten. Wird die Ausgangssymbolanzahl einer Funktionseinheit m durch $N_{sym,m}$ beschrieben, ergibt sich $q_{sym,m}$ als Anpassungsfaktor zwischen Eingangs- und Ausgangssymbolanzahl:

$$N_{sym,m} = q_{sym,m} \cdot N_{sym,m-1} \tag{70}$$

Des Weiteren können auch bei gleichen Datenmengen Symbole mit der gleichen Symbolnummer unterschiedliche Symbolperioden haben. Die gesamte Ausgabezeit T_{lf} der Basisbandverarbeitung ergibt sich als Summe der Symbolperioden der letzten Funktionseinheit M:

$$T_{lf} = \sum_{n=1}^{N_{sym,M}} T_{sym,M}(n) \tag{71}$$

Der letzte für die Beschreibung einer Funktionseinheit notwendige Parameter $q_{wait,m}$ kennzeichnet die Anzahl an Symbolen, die eingegeben werden muss, bevor ein Ausgabestart erfolgt. Über diesen Parameter wird zum einen eine initiale Verzögerung durch Symbolgrößenänderungen erfasst. Zum anderen können Speicherstrukturen wie z. B. Interleaver modelliert werden, die die Ausgabe um eine definierte Anzahl an Eingangssymbolperioden verzögern.

Mit den beschriebenen charakteristischen Größen und Parametern ergibt sich die in Abb. 20 dargestellte Modellierung einer Funktionseinheit.

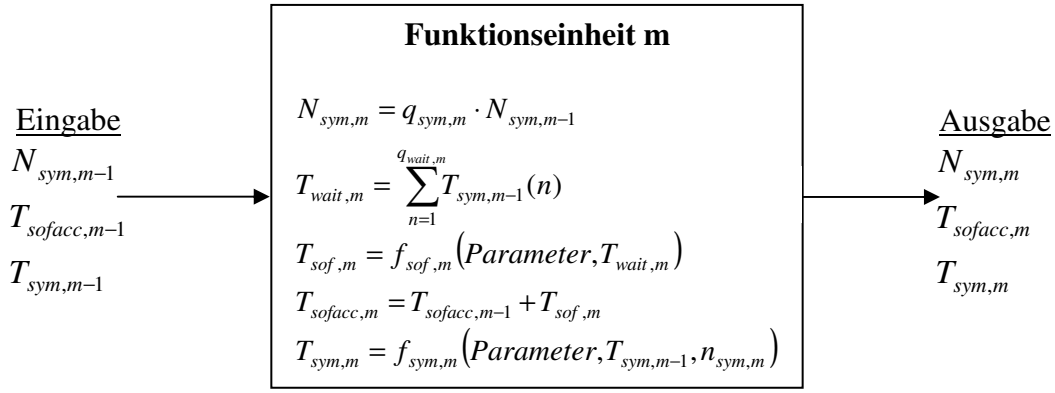

Abb. 20: Abstrahiertes Blackboxmodell eines Basisbandmoduls

Die Ausgabegröße $T_{sofacc,m}$ einer Funktionseinheit beinhaltet die akkumulierte Startverzögerung der vorhergehenden Instanzen. Für jede Funktionseinheit müssen $q_{sym,m}$ und $q_{wait,m}$ sowie die beiden Funktionen $f_{sof,m}$ und $f_{sym,m}$ spezifiziert werden. Mit *Parameter* werden in beiden Funktionen die Abhängigkeiten von globalen Übertragungsparametern, wie der Modulations- und Kodierungsart, erfasst. Die beiden Größen $q_{sym,m}$ und $q_{wait,m}$ sind unter Umständen auch von den Übertragungsparametern abhängig. Die Funktion $f_{sof,m}$ hängt neben den Übertragungsparametern auch von $T_{wait,m}$ ab. Im einfachsten Fall ist die Funktion $f_{sof,m}$ gegeben durch:

$$f_{sof,m}(Parameter, T_{wait,m}) = f_{sof,m}(Parameter) + T_{wait,m} \tag{72}$$

es kann sich aber auch um eine komplexere Abhängigkeit, z. B. in der Form:

$$f_{sof,m}(Parameter, T_{wait,m}) = \max(f(Parameter), T_{wait,m}) \tag{73}$$

handeln.

Zur Verdeutlichung der Anwendung des verallgemeinerten Modells wird dieses im Folgenden beispielhaft auf den Empfänger des beschriebenen EASY-A VHR-E-Systems angewendet. Der Empfänger besteht aus den Blöcken Frontstage (m=1), Demapper (m=2), Deinterleaver (m=3), Faltungsdekodierer (m=4) und Ausgabeeinheit (m=5). Die Eingabedaten in den ersten Block entsprechen den Daten des gesendeten Frames (m=0), wobei die Dauer der Präambel und des Signalfeldes mit in die Symbolperiode des ersten Datensymbols aufgenommen werden. Die Startverzögerungen ergeben sich entsprechend der abgeleiteten Latenzmodellierung für eine BPSK-Modulation mit Coderate ½ und ohne RS-Kodierung wie folgt:

$$\begin{aligned} T_{sof,1} &= 7025\,\text{ns} \\ T_{sof,2} &= 55\,\text{ns} \\ T_{sof,3} &= 25\,\text{ns} + T_{sym,2}(1) \\ T_{sof,4} &= 1145\,\text{ns} \\ T_{sof,5} &= 64\,\text{ns} \end{aligned} \tag{74}$$

Mit Ausnahme der Funktion $f_{sof,3}$ des Deinterleavers sind alle Funktionen $f_{sof,m}$ unabhängig von Wartezeiten, es gilt $q_{wait,m} = 0$ für $m \neq 3$. Für den Deinterleaver erfolgt die Ausgabe des ersten Symbols erst nach dem vollständigen Einschreiben des ersten Symbols: $q_{wait,3} = 1$. Für alle Blöcke stimmen die Symbolgrößen und somit die jeweiligen Symbolanzahlen überein, es gilt $N_{sym,1} = N_{sym,2} = \ldots = N_{sym,5} = N_{sym}$. Die Symbolperioden der Blöcke 1, 2 und 3 sind identisch und entsprechen der in (48) angegebenen Formulierung mit $T_{min} = 495$ ns, $T_{Nth} = 545$ ns und $T_{sym} = 595$ ns. Die wieder identischen Symbolperioden der Blöcke 4 und 5 lassen sich aus der Differenz:

$$T_{sym,4}(n_{sym}) = T_{sym,5}(n_{sym}) = T_{de}(n_{sym}+1) - T_{de}(n_{sym}) \quad \forall n_{sym} < N_{sym} \tag{75}$$

ermitteln (siehe (56), (59) und (60)). Damit ergeben sich die gesamte Startverzögerung T_{sof} und die Länge der Ausgabesequenz T_{lf} zu:

$$\begin{aligned} T_{sof} &= \sum_{m=1}^{5} T_{sof,m} = 8809\,\text{ns} \\ T_{lf} &= \sum_{n=1}^{N_{sym}} T_{sym,5}(n) = T_{de}(N_{sym}) + T_{sym,5}(N_{sym}) - T_{de}(1) = T_{de}(N_{sym}) + T_{sym,5}(N_{sym}) - 1145\,\text{ns} \end{aligned} \tag{76}$$

$T_{de}(1)$ entspricht der Startverzögerung des ersten Dekodierersymbols, also T_{de1}. Damit ergibt sich für T_{lf} eine zu (68) identische Formulierung.

Wird das Beispiel um einen RS-Dekodierer erweitert, ist eine neue Funktionseinheit hinter der Kanaldekodierung in die Kette einzufügen. Da aber die Ausgabeeinheit nur eine konstante Verzögerung vornimmt und keinerlei Abhängigkeiten von den Symbolgrößen zeigt, kann die neue Funktionseinheit auch als letzte Einheit der Kette hinzugefügt werden. Damit sind die

Indizes nicht neu zu vergeben, das Latenzmodell des RS-Dekodierers erhält stattdessen den Index m=6. Die Ermittlung der charakteristischen Latenzfunktionen des neuen Blockes erfolgt wieder am Beispiel der BPSK-Modulation mit Coderate ½. Für eine bessere Veranschaulichung des Beispiels sei angenommen, dass ein RS-Block aus der Datenmenge von genau fünf OFDM-Symbolen besteht. Der RS-Dekodierer muss also fünf Eingangssymbole abwarten, bevor er mit der Verarbeitung beginnen kann. Damit ergibt sich $T_{sof,6}$ zu:

$$\begin{aligned} T_{sof,6} &= 2809\,\text{ns} + T_{wait,6} = 2809\,\text{ns} + \sum_{n=1}^{5} T_{sym,5}(n) = 2809\,\text{ns} + \sum_{n=1}^{5} T_{sym,4}(n) = \\ &= 2809\,\text{ns} + T_{de}(6) - T_{de1} \end{aligned} \tag{77}$$

und die gesamte Startverzögerung T_{sof} beträgt:

$$T_{sof} = \sum_{m=1}^{6} T_{sof,m} = 8809\,\text{ns} + 2809\,\text{ns} + T_{de}(6) - T_{de1} = 10473\,\text{ns} + T_{de}(6) \tag{78}$$

Bei einem Frame mit 2048 Datenbytes und unter Verwendung von vier Streams beträgt die Startverzögerung also rund 15,4 µs. Da für jeden RS-Block erst fünf OFDM-Symbole eingeschrieben werden müssen, ergibt sich die Symbolperiode des RS-Dekodierers als:

$$T_{sym,6}(n) = \sum_{m=5\cdot n+1}^{5\cdot(n+1)} T_{sym,5}(m) \;\forall\; n \le N_{sym,6} - 1 \tag{79}$$

$T_{sym,6}(N_{sym,6})$ entspricht der Zeit für die kontinuierliche Ausgabe eines RS-Blocks, also T_{last}. Daraus folgt für die Länge einer Frameausgabe:

$$\begin{aligned} T_{lf} &= \sum_{n=1}^{N_{sym,6}} T_{sym,6}(n) = \sum_{n=6}^{N_{sym,5}} T_{sym,5}(n) + T_{sym,6}(N_{sym,6}) = \\ &= T_{de}(N_{sym,5}) - T_{de}(6) + T_{last} \end{aligned} \tag{80}$$

Für den Frame mit 2048 Datenbytes folgt daraus eine Framelänge von rund 21,4 µs.

Mithilfe des vorgestellten verallgemeinerten Latenzmodells lassen sich die Auswirkungen geänderter und neu hinzugefügter Module auf die Basisbandverarbeitung einfacher untersuchen, als dies mit einer Anpassung des spezialisierten Modells möglich wäre. Das Modell ist aufgrund der Modularisierung sowohl in objektorientierten Programmiersprachen als auch in blockorientieren Simulationsumgebungen gut umzusetzen. Das Hinzufügen eines neuen Moduls entspricht einer zusätzlichen Instanziierung des Objektes „Funktioneinheit“ mit Angabe der die Modullatenzen charakterisierenden Größen und Funktionen.

Soll, wie eingangs aufgeführt, eine Erweiterung des Modells um mehrere Eingänge zur Unterstützung paralleler Strukturen erfolgen, sind folgende Änderungen notwendig. Jede Funktionseinheit verfügt über i Eingänge, für die jeweils i unterschiedliche Parameter- und Funktionssätze abgeleitet werden. Dabei sind die charakteristischen Parameter und Funktionen nicht notwendigerweise voneinander unabhängig. Zur Ermittlung der akkumulierten Gesamtverzögerung darf nur ein Eingang mit seinen entsprechenden Größen verwendet werden. Dabei muss bei der Anordnung der Funktionseinheiten berücksichtigt werden, dass keine Schleifen entstehen.

Nach der bisher erfolgten Darstellung von Latenzursachen und -auswirkungen sowie der beispielhaften Latenzmodellierung einer Basisbandverarbeitung werden im zweiten Teil dieser Arbeit Verfahren zur Latenzverringerung vorgestellt und analysiert.

5. Überblick: Methoden zur Latenzverringerung

Die im Abschnitt 2.10 kategorisierten Ursachen für Latenzen im Basisband erfordern eine differenzierte Betrachtung im Hinblick auf eine mögliche Verringerung oder Vermeidung. So können Verzögerungszeiten aufgrund einer begrenzten Verarbeitungsgeschwindigkeit durch eine Erhöhung derselben einfach optimiert werden.

Latenzzeiten, die sich aus der Blockverarbeitung ergeben und in einer Wartezeit resultieren, können dagegen nicht durch höhere Verarbeitungsgeschwindigkeit verringert werden. Wie im Abschnitt 2.10 beschrieben, erfordert eine Blockverarbeitung, dass vor Verarbeitungsbeginn der gesamte Block in das Verarbeitungsmodul eingeschrieben wurde. Ein Beispiel dafür ist ein Blockinterleaver, dessen Latenz durch die Wartezeit für das Einschreiben eines vollständigen Blocks gegeben ist. Eine Erhöhung der internen Verarbeitungsgeschwindigkeit ändert die Latenz nicht, wenn nicht gleichzeitig die Eingangsdatenrate erhöht wird. Allein die Ausgabe könnte schneller erfolgen, dann entsteht aber automatisch eine Pause im Datenstrom zwischen dem aktuellen und dem folgenden Block. Zur Optimierung der Verzögerungszeit ist es also nicht ausreichend, das betreffende Modul losgelöst von seiner Umgebung zu betrachten.

Latenzverringerung im Basisband erfordert nicht nur einzelne Module und ihr Zusammenspiel zu optimieren, sondern sollte mit dem Systementwurf beginnen. Bereits bei der grundsätzlichen Konzeption und der Auswahl der zu verwendenden Algorithmen sind die beabsichtigte Implementierung sowie die Anforderungen an die Systemlatenz zu beachten. Ein Beispiel für eine mögliche Latenzverringerung durch die geeignete Systemkonzeption ist die im Abschnitt 2.7 vorgestellte Einfügung von Pilotunterträgern. Entsprechend der Spezifikation des EASY-A VHR-E-Systems wird nach jeweils 12 Datenunterträgern ein Pilotunterträger eingeführt. Da aufgrund der hohen Durchsatzanforderungen acht Datenunterträger parallel verarbeitet werden, ergibt sich die in Abb. 6 (Seite 28) dargestellte Struktur mit FIFOs zum Speichern der überzähligen Datenunterträger bei einem Mehrfachzugriff auf den gleichen Speicherblock. Wäre der Abstand zwischen den einzelnen Pilotunterträgern dagegen ein Vielfaches von acht, gäbe es beim Einschreiben von acht parallelen Datenunterträgern keine Mehrfachzugriffe auf denselben Speicherblock. Die zusätzliche Verzögerungszeit für das Leeren der FIFOs im Anschluss an das Einschreiben eines OFDM-Symbols könnte somit vermieden werden. Andererseits besteht bei einer solchen Änderung die Möglichkeit von negativen Auswirkungen auf die gesamte Systemperformance. Dies muss also im Rahmen einer Simulation untersucht werden. Für das vorgestellte VHR-E-System wurde ein solcher Vergleich der Framefehlerraten (FER[43]) zwischen zwei verschiedenen Pilotschemata durchgeführt. Das Ergebnis zeigt Abb. 21. Beim Schema 1 handelt es sich um das im VHR-E-System verwendete Pilotschema mit insgesamt 60 Pilotunterträgern, zwischen denen jeweils 12 Datenunterträger

[43] FER: Frame Error Rate

liegen. Im Schema 2 wurde die Anzahl der Pilotunterträger auf 50 verringert sowie der Abstand auf 16 Datenunterträger angehoben. Als Kanalmodelle fanden die von der IEEE 802.15.3c Task Group [59] entwickelten Kanalmodelle CM1.2 und CM2.3 Verwendung. Das erstgenannte modelliert eine LOS[44]-, das zweite eine NLOS[45]-Umgebung. Neben dem reinen Übertragungskanal wurden auch weitere Effekte wie eine zufällige Trägerfrequenzabweichung, ein zufälliger Phasenversatz sowie ein Phasenrauschen der Trägerfrequenz einbezogen. Für das Phasenrauschen wurde ein Wiener-Prozess mit -90 dBc/Hz bei 1 MHz Offset angenommen. Zusätzlich wurde der Phasenrauschprozess mit einem additiven weißen Grundrauschen von -120 dBc/Hz versehen. Als Modulationstyp wurde 16-QAM mit Coderate ½ verwendet, für jeden E_b/N_0-Wert wurden 5000 Frames mit je 8192 Datenbytes simuliert. Die dargestellte Framefehlerrate beinhaltet auch Synchronisationsfehler, wie nicht erkannte Präambeln sowie Dekodierungsfehler im Signalfeld.

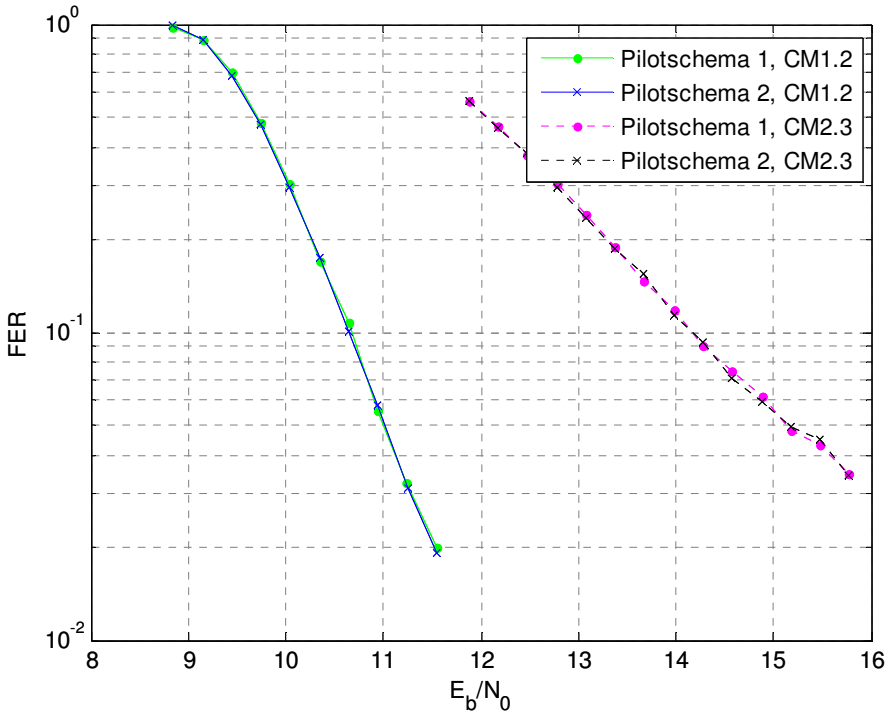

Abb. 21: Vergleich zwischen zwei unterschiedlichen Pilotschemata

Wie im obigen Diagramm zu erkennen, hat das geänderte Pilotunterträgerschema keinen Einfluss auf die Framefehlerrate. Durch eine Anpassung der Systemparameter wurde ohne Änderungen an der Systemperformance eine einfachere Hardwareumsetzung mit geringeren Ressourcenanforderungen erreicht und gleichzeitig die native Latenz des Piloteneinfügungsmoduls verringert. Weiterhin erhöht sich mit dem geänderten Schema die spektrale Effizienz des Systems, da durch den Verzicht auf zehn Pilotunterträger die insgesamt verwendete Bandbreite verringert werden kann bzw. bei gleichbleibender Bandbreite mehr Datenunterträger genutzt werden können.

[44] LOS: Line-of-sight, direkte Sichtverbindung zwischen Sender und Empfänger
[45] NLOS: Non-line-of-sight, keine direkte Sichtverbindung zwischen Sender und Empfänger

Ein weiteres Beispiel für die Latenzreduktion durch Algorithmenänderung ist die in [52] beschriebene Verringerung der Latenz für das Hinzufügen eines zyklischen Präfixes zu einem OFDM-Symbol. Durch eine Rotation der Symbole im Frequenzbereich wird das Voranstellen des Präfixes in ein Anfügen eines zyklischen Postfixes überführt.

Adaptive Systeme lassen sich zur Laufzeit an die sich ändernden Umgebungsbedingungen anpassen. Bei der Kombination mehrerer Kanalkodierungsvarianten, wie dem inneren Faltungs- und äußeren RS-Code im Beispielsystem, kann bei guten Übertragungsbedingungen eine der beiden Kodierungen abgeschaltet werden. Infolgedessen verringert sich die Latenzzeit. Kommunizieren auf Anwendungsebene unterschiedliche Dienste über ein solches Übertragungssystem, können Fehlerrate und Latenzzeit, abhängig von den Anforderungen des jeweiligen Dienstes, fallweise optimiert werden.

Neben dem dynamischen Verzicht auf ein bestimmtes Modul erlauben auch wiederholungsbasierte Verfahren, wie die iterative LDPC-Dekodierung, ein Einstellen der entstehenden Verarbeitungslatenz. Dazu wird unter Berücksichtigung der notwendigen Korrekturleistung die maximale Iterationszahl festgelegt. Gleichzeitig ist eine dynamische, d. h. blockweise, Verringerung der Dekodierungsverzögerung durch ein Überprüfen des Ergebnisses nach jeder Iteration möglich. Zwei entsprechende Verfahren sind in [53] und [54] vorgestellt. Ein Einstellen der Verarbeitungslatenz unter Berücksichtigung der Korrekturleistung ist auch bei der Viterbidekodierung über eine Variation der Traceback-Tiefe möglich.

Die vielleicht einfachste Variante, um die aus einer begrenzten Verarbeitungsgeschwindigkeit folgende Latenz zu verringern, ist die Erhöhung der Verarbeitungsgeschwindigkeit durch eine Steigerung der Taktrate. Dies gilt sowohl für DSP- / GPP-basierte Softwareumsetzungen als auch für in Hardware realisierte Module und Systeme. Die Erhöhung der Taktrate zieht dabei keine Änderungen an den Algorithmen und deren Umsetzung nach sich, geht aber üblicherweise mit einer Erhöhung des Stromverbrauchs einher. Gleichzeitig werden die Herstellungskosten des Systems durch die erhöhten Anforderungen an die zugrunde liegende IC[46]-Technologie vergrößert.

Sofern eine Steigerung der Taktrate nicht möglich ist, bietet sich eine Untersuchung auf alternative Implementierungsmöglichkeiten an. Dies schließt die Verwendung effizienterer Algorithmen sowie die Verlagerung von Verarbeitungsschritten aus der Software in eine dedizierte Hardware und umgekehrt ein. Bei der Verlagerung von Berechnungen in eine Hardwarerealisierung oder deren Optimierung ist eine nahe liegende Variante zur Steigerung der Systemleistung die Parallelisierung von Verarbeitungsschritten. Parallelverarbeitung eröffnet dabei noch weitere Möglichkeiten zur Reduzierung von Latenzen, die ausführlich im folgenden Kapitel 6 analysiert werden.

[46] IC: Integrated Circuit – integrierter Schaltkreis

Im Rahmen der Implementierung bzw. des Entwurfs der Systemarchitektur ist nicht nur die latenzeffiziente Umsetzung der einzelnen Verarbeitungsmodule, sondern auch die systemweite Datenflusssteuerung zu beachten. So wurde für die Implementierung des vorgestellten VHR-E-Systems auf eine bereits existierende Umsetzung des digitalen Frontstages zurückgegriffen. Wie im Rahmen der Latenzmodellierung im Abschnitt 4.4.3 ausgeführt, basiert die Datenflusskontrolle dieses digitalen Frontstages auf einem globalen Kontrollautomaten, der zyklusbasiert arbeitet. Nur zu einem fest definierten Zeitpunkt innerhalb eines Zyklus werden von außen vorgegebene Parameter übernommen. Damit verlängert sich die Reaktionszeit im schlechtesten Fall um eine Zyklusdauer, die im vorgestellten System einer OFDM-Symbolperiode entspricht. Wird dagegen die Steuerung, wie in der vorgestellten Architektur des digitalen Datenpfades des Empfängers, dezentralisiert innerhalb eines jeden Moduls vorgenommen, kann eine minimale Reaktionszeit auf äußere Eingaben erreicht werden.

Die Ergebnisse der Latenzmodellierung des Beispielsystems im Abschnitt 4.4 zeigen, dass sich ein großer Teil der Gesamtlatenz aufgrund von Wartezeiten ergibt. Diese resultieren nicht nur aus begrenzten Verarbeitungsgeschwindigkeiten in den vorhergehenden Modulen oder aus der langsamen Datenübertragung, sondern auch aus Abhängigkeiten. So ist die im Abschnitt 4.4.3 beschriebene Wartezeit aufgrund der Signalfeldverarbeitung ein Beispiel für eine Latenz aufgrund einer Parameterabhängigkeit. Die Verarbeitung der Datensymbole kann erst erfolgen, wenn die im Signalfeld festgelegten Übertragungsparameter bekannt sind. Wie bei der Kategorisierung (Abschnitt 2.10) erläutert, ist die Parameterabhängigkeit zwar ein sekundärer Effekt, dessen Ursache in der begrenzten Übertragungs- und Verarbeitungsgeschwindigkeit des Signalfeldes liegt. Trotzdem kann die Latenz durch Parameterabhängigkeit mithilfe spekulativer Verfahren verringert werden, ohne die Übertragungs- und Verarbeitungsgeschwindigkeit zu erhöhen.

Neben der Verringerung parameterabhängiger Latenzzeiten eignen sich spekulative Verfahren auch zur Reduktion von Latenzen aufgrund von begrenzter Verarbeitungsgeschwindigkeit sowie der Blockverarbeitung. Der Hintergrund spekulativer Verfahren, die bereits vorhandenen Anwendungen in der Datenverarbeitung, das genaue Prinzip der Anwendung zur Latenzverringerung in Basisbandprozessoren und die erzielten Ergebnisse werden im Kapitel 7 erläutert. Es sei hier kurz vorweggenommen, dass Spekulation auf einer Annahme über das mögliche Ergebnis der vorhergehenden Verarbeitung, also z. B. der Signalfelddekodierung, beruht. Mithilfe dieser Annahme werden die folgenden Verarbeitungsschritte ohne Wartezeit ausgeführt. Nach dem Abschluss der ersten Verarbeitungsschritte wird über den Spekulationserfolg entschieden. Bei einer erfolgreichen Spekulation wurde die Latenz verringert, während bei einer fehlgeschlagenen Spekulation keine Latenzverringerung erreicht werden kann.

Für die im Abschnitt 3.3 vorausgesetzte möglichst latenzfreie Kanalbelegungserkennung, die Verringerung der Zeit, in der ein System „blind“ für Änderungen auf dem Kanal ist, also

nichts über die aktuelle Kanalnutzung weiß, bietet sich die Nutzung eines RSSI[47]-Signals an. Dieses Signal gibt die Empfangsfeldstärke des eingehenden Signals an und ist ein essentieller Bestandteil vieler Drahtloskommunikationssysteme, wie z. B. WLAN[48] nach dem IEEE 802.11 Standard [45]. Üblicherweise wird es direkt im analogen Teil des Empfängers gebildet. Auch ein digitales RSSI hinter der Analog-Digital-Wandlung ist möglich. In diesem Fall darf allerdings keine vorhergehende AGC vorhanden sein, da diese ansonsten die Leistung auf einen konstanten Wert regelt. Mithilfe eines solchen digitalen RSSI-Signals wird im beschriebenen EASY-A-Beispielsystem die relativ lange Latenzzeit für die Kanalbelegungsermittelung über die Präambelerkennung verkürzt.

Latenzverringernde Maßnahmen sollten sich nicht nur auf das Basisband konzentrieren, sondern auch die höheren Schichten, speziell den MAC, mit berücksichtigen. So können durch eine gezielte Kombination bzw. Verlagerung von Aufgaben zwischen der MAC- und der PHY-Schicht die Auswirkungen von Latenzen aufgrund der Basisbandverarbeitung verringert werden. Wird ein Signalfeld verwendet, um Parameter und Einstellungen auf PHY-Ebene auszutauschen, können in diesem Signalfeld weitere Informationen, wie z. B. eine PHY-Adresse, übermittelt werden. Anhand dieser Adresse kann bereits im Basisbandempfänger bestimmt werden, ob das System der Adressat des Datenframes ist. Wenn nicht, werden alle einlaufenden Datensymbole des aktuellen Frames verworfen und es wird auf den nächsten Frame gewartet. Eine solche Vorgehensweise eignet sich auch wieder zur Energieeinsparung. Das gesamte Basisband mit Ausnahme des Synchronisators kann bis zum Erkennen des nächsten Frames in einen Ruhezustand geschaltet werden. Wenn die Länge des Datenframes im Signalfeld kodiert ist, kann dem MAC eine Information über die Dauer der Kanalbelegung mitgeteilt werden, auch wenn an ihn keine Empfangsdaten weitergereicht werden. Zugleich kann für die gegebene Dauer auch der Synchronisator in einen energiesparenden Ruhezustand geschaltet werden.

Eine weitere Möglichkeit zur Latenzverkürzung im Zusammenspiel zwischen MAC und PHY ist das Verschieben von Informationen aus dem MAC-Header in das Signalfeld [55]. Im MAC-Header sind unter anderem ein oder mehrere ACK-Bits enthalten, über die der fehlerfreie Empfang von Datenpaketen bestätigt wird. Müssen diese erst das gesamte Basisband durchlaufen, können sie erst nach einer teilweise hohen Latenzzeit ausgewertet werden – im EASY-A-System frühestens nach $T_{sync} + T_{sof} \approx 22{,}6\ \mu s$, wenn eine BPSK-½-Modulation und die optionale äußere RS-Kodierung bei einem Frame mit mindestens zehn Datensymbolen verwendet werden. Über die ACK-Bits wird auf MAC-Ebene die Freigabe der Sendepuffer geregelt, die im letztgenannten Fall erst nach einer relativ langen Zeit erfolgen kann. Werden dagegen die ACK-Bits mit im Signalfeld kodiert, verkürzt sich die Latenzzeit zwischen dem Eingang eines Frames bis zur Auswertung der ACK-Bits möglicherweise erheblich, im angeführten Beispiel von 22,6 µs auf $T_{sync}+T_{sfv} \approx 12{,}4\ \mu s$, also um rund 45 %.

[47] RSSI: Received Signal Strength Indicator
[48] WLAN: Wireless Local Area Network

Durch eine Übertragung redundanter Informationen im Signalfeld lässt sich dessen Auswertung in der Basisbandverarbeitung beschleunigen. Der Empfänger benötigt Informationen über die Anzahl an gesendeten Datenwörtern, zusätzlich aber auch über die Anzahl an daraus resultierenden Datensymbolen. Letztere kann aus der Datenwortanzahl berechnet werden, dazu ist jedoch möglicherweise eine signifikante Berechnungszeit notwendig. Wird die Symbolanzahl dagegen mit übermittelt, kann auf die Berechnung verzichtet werden. In diesem Fall ist allerdings eine Sicherstellung der Konsistenz zwischen übermittelter Datenwort- und Symbolanzahl durch einen Plausibilitätstest notwendig. Erfordert dieser weniger Zeit als die Berechnung der Symbolanzahl, kann die resultierende Gesamtverzögerung verringert werden.

Insgesamt lassen sich die folgenden vier Punkte zur Behandlung von Latenzen im Basisband ableiten:

1. Vermeidung durch ein der Problemstellung angepasstes Systemdesign
2. Verringerung der Auswirkungen durch Optimierung der Algorithmen
3. Verringerung durch Erhöhung der Verarbeitungsleistung
4. Verringerung durch Anwendung spekulativer Verfahren.

Im Kapitel 8 werden basierend auf den soeben angeführten Punkten allgemeine Richtlinien für den latenzarmen Entwurf einer Basisbandverarbeitung erarbeitet und in einen Gesamtablaufplan des Systementwurfs eingebunden. Zunächst erfolgt in den beiden folgenden Kapiteln eine detaillierte Vorstellung der Latenzverringerungsverfahren durch die Nutzung paralleler Strukturen (Kapitel 6) sowie die Anwendung spekulativer Verarbeitung (Kapitel 7).

6. Latenzverringerung durch Parallelität

6.1 Einführung

Wenn in digitalen Systemen eine Steigerung der Taktrate nicht möglich ist, ist Parallelisierung die übliche Vorgehensweise zur Erhöhung der Verarbeitungsleistung bzw. des Durchsatzes und gleichzeitig der Schlüssel zur Optimierung und Reduzierung der Verarbeitungszeiten. Hier muss darauf hingewiesen werden, dass Verarbeitungsleistung nicht gleichbedeutend mit Verarbeitungszeit bzw. Latenz ist. Die Verarbeitungszeit gibt die Zeit an, die zur kompletten Verarbeitung eines einzelnen Datums durch das betreffende Modul benötigt wird. Die Verarbeitungsleistung dagegen spezifiziert, wie viele Daten innerhalb einer definierten Zeit verarbeitet werden können. Durch eine Fließbandarchitektur erhöht sich der Datendurchsatz, also die Verarbeitungsleistung, aber nicht die für die Verarbeitung eines einzelnen Datums benötigte Zeit. Eine detaillierte Diskussion zu den verschiedenen Organisationsformen von Rechenwerken zur Minimierung der Bearbeitungszeit und zur Maximierung des Durchsatzes findet sich in [56].

In einem Basisbandprozessor für hohe Datenraten sind üblicherweise die einzelnen Module für eine parallele Verarbeitung ausgelegt, um den notwendigen Datendurchsatz sicherzustellen. Dabei kann die Parallelität in zwei verschiedenen Realisierungen erreicht werden: Parallelität auf Bitebene und Parallelität auf Blockebene. Beide Prinzipien basieren auf mehreren parallelen Funktionseinheiten, unterscheiden sich aber in der Einbindung in den gesamten Datenverarbeitungsfluss des Systems. Eine genaue Erläuterung der jeweiligen Organisation und Eigenschaften erfolgt in den beiden folgenden Abschnitten.

Im letzten Abschnitt dieses Kapitels wird untersucht, inwieweit das implizite Interleaving einer bitparallelen Verarbeitung zum Verzicht auf einen dedizierten Interleaver genutzt werden kann. Dieses implizite Interleaving ergibt sich durch die Aufteilung benachbarter Bits auf unterschiedliche Verarbeitungseinheiten.

6.2 Bitparallele Verarbeitung

Ein bitparalleles Modul ist dadurch gekennzeichnet, dass es über eine bitparallele Eingangs- und Ausgangsschnittstelle verfügt und eine Operation gleichzeitig auf die eingehenden Bits anwendet. Die Operation kann dabei aus einzelnen Schritten bestehen, die entsprechend einer ortssequentiellen Verarbeitung bzw. im Rahmen einer Fließbandarchitektur durch unterschiedliche, aufeinanderfolgende Funktionseinheiten realisiert werden. (Genau genommen besteht eine solche Architektur aus mehreren hintereinander geschalteten bitparallelen Modulen, die jeweils einen Verarbeitungsschritt als Operation durchführen.) Auch eine zeitsequentielle Verarbeitung innerhalb der Funktionseinheiten schränkt das übergeordnete Prinzip der Bitparallelität nicht ein.

Bitparallel bedeutet nicht notwendigerweise, dass die Verarbeitung jeweils auf einzelnen Bits erfolgt. Zur Unterscheidung von Blockparallelität wird der Begriff „Bit“ im Zusammenhang mit der Bitparallelität im Rahmen dieser Arbeit zur Kennzeichnung einer logischen Informationseinheit der Datenschnittstelle genutzt. Besteht ein Datenbus aus M Bits, die sich in N Datenwörter gruppieren lassen und werden die N Datenwörter parallel durch N Funktionseinheiten verarbeitet, handelt es sich auch um eine N-bitparallele Verarbeitung entsprechend der gewählten Definition.

Für das Prinzip der Bitparallelität sind die in den Abb. 22(a) und (b) dargestellten zwei Varianten zu unterscheiden. Ist die Verarbeitung eines Eingangsbits unabhängig von anderen Eingangsbits, erfolgt die parallele Verarbeitung in voneinander unabhängigen identischen Funktionseinheiten (FE). Ist dagegen die Verarbeitung eines eingehenden Bits von den anderen eingehenden Bits abhängig, besteht das bitparallele Modul aus einer einzigen Funktionseinheit mit einem parallelen Ein- und Ausgang.

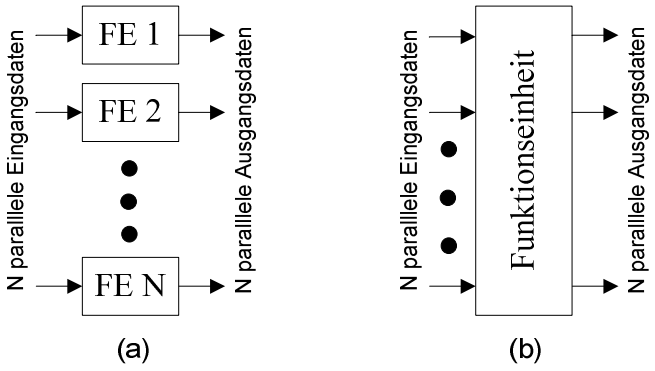

Abb. 22: Varianten für Bitparallelität

Ein Beispiel für die voneinander unabhängige bitparallele Verarbeitung ist das parallele Mapping von Datenbits auf Datensymbole im EASY-A-Beispielsystem. In diesem System werden aus jeweils acht aufeinanderfolgenden Gruppen von Bits (entsprechend acht Datenwörtern) parallel durch acht Mappereinheiten acht Datensymbole erzeugt. Eine parallele Faltungskodierung ist dagegen ein Beispiel für die voneinander abhängige bitparallele Verarbeitung.

Wenn im Rahmen einer Parallelverarbeitung entsprechend Abb. 23 an der Eingangsschnittstelle M unabhängige Bits angeliefert werden, aber nur N Funktionseinheiten mit $N < M$ zur parallelen Verarbeitung von N Bits zur Verfügung stehen, handelt es sich um eine bitparallele Verarbeitung innerhalb des gestrichelt gekennzeichneten Teilmoduls. Je nach Art der Zuordnung der M Eingangsbits auf die N Funktionseinheiten ist das übergeordnete Modul bit- oder blockparallel. Es ist bitparallel, wenn die Aufteilung der Eingangsbits B_0, ..., B_{M-1} auf die Funktionseinheiten F_0, ..., F_{N-1} entsprechend der Vorschrift:

$$F_{i \bmod N} = B_i \text{ mit } i = 0, ..., (M-1) \tag{81}$$

erfolgt. Eine Kommutation in der Zuordnung ist dabei zulässig. Bitparallelität ist also immer dann gegeben, wenn mehrere unabhängige benachbarte Bits der Eingangsschnittstelle parallel durch unterschiedliche Funktionseinheiten verarbeitet werden.

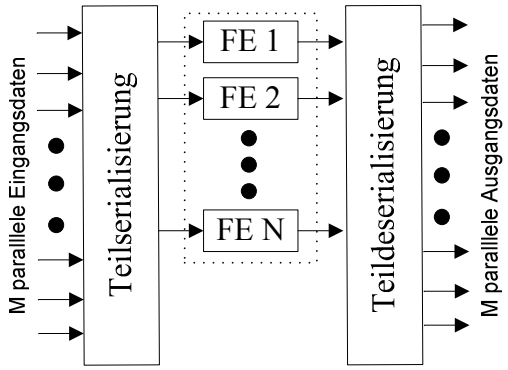

Abb. 23: Bitparallelität mit M > N Eingangsbits

Die bitparallele Verarbeitung findet sich in vielen Standards und Standardisierungsvorschlägen zur drahtlosen Kommunikation, wie z. B. in IEEE 802.15.3c [46] und 802.11ad [57]. Während entsprechend der obigen Erläuterung bei Bitparallelität mit unabhängigen Eingangsbits die gleiche Operation durch mehrere identische Funktionseinheiten parallel auf mehrere Bits angewendet wird, lässt sich auch die in den Standards definierte Unequal Error Protection (UEP) als Sonderform der bitparallelen Verarbeitung definieren. Bei dieser Fehlerschutzkodierungsart wird ein eingehendes Datenwort in mehrere verschieden priorisierte Bereiche unterteilt, für die eine unterschiedliche Fehlerschutzkodierung erfolgt. Damit können die verschiedenen Auswirkungen von Fehlern in den einzelnen Bereichen zur Optimierung der Datenübertragungsrate genutzt werden. So würde die Verfälschung des MSB[49] bei einer Videoübertragung einen stark sichtbaren Einfluss haben, während die Verfälschung des LSB[50] so gut wie keinen subjektiven Qualitätsverlust verursacht [58]. Je nach Videoquelle geht dieser Fehler im Bildrauschen unter. Für eine bitparallele UEP unterscheiden sich also die Funktionseinheiten für die höherwertigen und niederwertigen Bits voneinander. Da trotzdem mehrere benachbarte Bits parallel von mehreren Funktionseinheiten verarbeitet werden, handelt es sich um eine bitparallele Verarbeitung.

Es ist offensichtlich, dass bei N parallelen Funktionseinheiten der Gesamtdurchsatz um den Faktor N gesteigert wird, während die Verzögerung zwischen der Ein- und Ausgabe eines bestimmten Datenbits der entsprechenden Latenz einer einzelnen Funktionseinheit entspricht. Werden unterschiedliche Funktionseinheiten verwendet, z. B. im Rahmen einer UEP, entspricht die Latenz des gesamten Moduls der größten Latenz der einzelnen Funktionseinheiten. Bitparallelität verringert für sich genommen also nicht die Verarbeitungszeit für ein einzelnes Bit. Stattdessen wird die für die Verarbeitung mehrerer Bits benötigte Zeit durch die Erhöhung der Verarbeitungsleistung reduziert.

[49] MSB: Most Significant Bit, höchstwertigstes Bit in einem Datenwort
[50] LSB: Least Significant Bit, niedrigwertigstes Bit in einem Datenwort

Während durch die bitparallele Ausführung eines Moduls die Latenz für die Verarbeitung eines Bits nicht verringert werden kann, eignet sich dieses Vorgehen trotzdem zur Verringerung von Latenzen innerhalb des gesamten Systems. So können durch die höhere Verarbeitungsgeschwindigkeit innerhalb des Moduls Wartezeiten aufgrund einer Parameterabhängigkeit oder einer Blockverarbeitung verringert werden. Gleichzeitig ist Bitparallelität eine Möglichkeit zur Erhöhung der Übertragungsgeschwindigkeit zwischen einzelnen Modulen und demzufolge zur Latenzverringerung, sofern die Latenzursache eine begrenzte Übertragungsgeschwindigkeit ist.

Die im Abschnitt 2.3 genannte Möglichkeit zur Reduktion der Latenz eines Blockinterleavers durch eine parallele Ein- und Ausgabe von Daten ist ein Beispiel für die Verringerung der Wartezeit durch eine bitparallele Übertragung. Der Interleaver des EASY-A VHR-E-Systems nimmt mit jedem Takt 32 Eingangsbits entgegen und gibt nach der Verzögerung um einen Block mit jedem Takt 32 Bits aus. Der die Latenz bestimmende Beitrag durch das Warten auf einen komplett in den Speicher geschriebenen Block ist durch die höhere Schreibrate geringer. Dafür ist jedoch vorauszusetzen, dass die Daten vom vorhergehenden Modul auch in der notwendigen höheren Geschwindigkeit bereitgestellt werden können.

6.3 Blockparallele Verarbeitung

Analog zur Bitparallelität sind auch bei der in Abb. 24 dargestellten blockparallelen Verarbeitungsstruktur mehrere Instanzen eines Basismoduls in ein Gesamtsystem integriert. Diese verarbeiten im Gegensatz zur bitparallelen Verarbeitung jeweils einen Block von benachbarten Bits, also eine an der Eingangsschnittstelle zeitlich aufeinanderfolgende Reihe einzelner Bits. Das Basismodul selbst kann dabei bitsequentiell oder bitparallel arbeiten. Auch die Schnittstellen müssen nicht notwendigerweise bitparallel ausgeführt sein. Die sequentiell oder parallel einlaufenden Datenbits werden entsprechend der Abb. 25 auf einzelne Blöcke aufgeteilt, die entsprechend der Blockzugehörigkeit von den einzelnen Basismodulen verarbeitet werden.

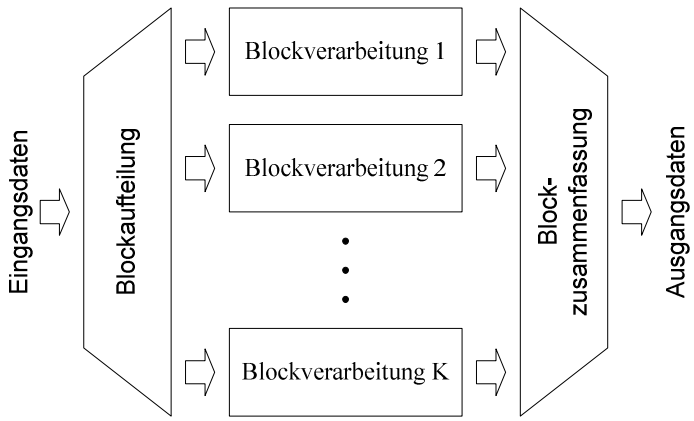

Abb. 24: Struktur der blockparallelen Verarbeitung

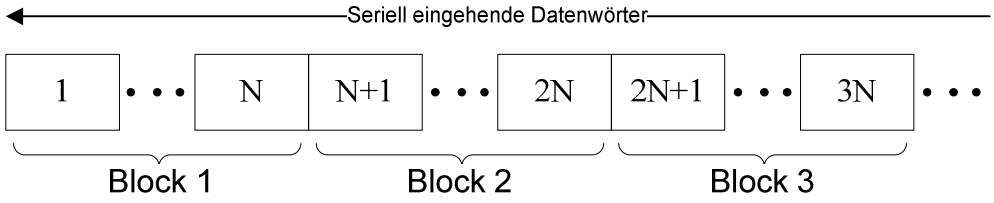

Abb. 25: Blockaufteilung

Eine blockparallele Verarbeitung bietet sich offensichtlich für die Implementierung blockbasierender Algorithmen an. Sofern die Verarbeitungsleistung des Basismoduls nicht ausreicht, können zeitlich aufeinanderfolgende Blöcke parallel durch verschiedene identische Instanzen des Basismoduls verarbeitet werden. Da die Eingangs- und Ausgangsschnittstelle des blockparallelen Moduls die erforderliche Datenrate des Systems bietet, müssen am Ein- und Ausgang der jeweiligen Instanzen Zwischenspeicher vorgesehen werden. Am Eingang puffern diese die hohe Datenrate, am Ausgang ermöglichen sie die Wiederherstellung der Reihenfolge.

Ein Beispiel für eine blockparallele Verarbeitung ist das im Abschnitt 4.3 vorgestellte RS-Dekodierungsmodul. Während die durchschnittliche Datenrate am Eingang des RS-Moduls bis zu 4,3 Gbit/s beträgt, bietet das Basismodul zur RS-Dekodierung nur einen maximalen Durchsatz von 1,6 Gbit/s. Deshalb sind drei parallele Instanzen vorgesehen, die zusammen einen maximalen Durchsatz von 4,8 Gbit/s bieten. Durch die Blockparallelität ist das RS-Modul transparent, d. h. die interne Parallelstruktur spiegelt sich nicht in den von außen erkennbaren Eigenschaften wider.

Für die blockparallele Verarbeitung müssen die zugrunde liegenden Algorithmen nicht auf einer Blockverarbeitung basieren. Es kann sich auch um stromorientierte Algorithmen, wie die Faltungskodierung, handeln. In diesem Fall wird für die Anwendung der Blockparallelität eine Blockbildung anhand einer wählbaren Blockgröße vorgenommen. Ein Beispiel hierfür ist der streambasierte Kanalkodierungs- und Kanaldekodierungsansatz des vorgestellten Beispielsystems. Dieser wurde bereits im Abschnitt 4.3 erläutert. Da es durch den Faltungskodierer keine Blockbildung in Form einer fixen Anzahl an zu kodierenden Datenwörtern gibt, wird diese durch eine Aufteilung anhand der OFDM-Symbolgrenzen vorgenommen. Die Bits des ersten Symbols werden durch den ersten Kodierer verarbeitet, die des zweiten durch den zweiten usw. Die nacheinander von einem Basismodul verarbeiteten Blöcke sind allerdings nicht unabhängig, sondern stellen einen fortlaufenden Teilstrom der zu kodierenden Daten dar. Deshalb wird in diesem Zusammenhang auch von einzelnen Streams gesprochen. Trotzdem handelt es sich hierbei um eine blockparallele Verarbeitung mit der Besonderheit, dass die Zuordnung der Blöcke zu den einzelnen Basismodulen nicht beliebig erfolgen kann, sondern ein bestimmtes Schema eingehalten werden muss. Im allgemeinen Fall ist es dagegen egal, welches Basismodul einen Block verarbeitet, sofern die richtige Reihenfolge am Ausgang wieder hergestellt wird.

Für die Analysen hinsichtlich Durchsatz und Latenz bei der blockparallelen Verarbeitung wird vorausgesetzt, dass die Geschwindigkeit der Schnittstellen am Ein- und Ausgang dem maximalen Gesamtdurchsatz R_{max} des parallelen Moduls entspricht, sodass die Übertragungsgeschwindigkeit kein begrenzender Faktor ist.

Mit R_{fe} als Durchsatz einer Basismodulinstanz und N_p als Anzahl der parallelen Instanzen ergibt sich analog zur bitparallelen Verarbeitung ein maximaler Gesamtdurchsatz von:

$$R_{max} = N_p \cdot R_{fe} \tag{82}$$

Wie im Abschnitt 2.10 erläutert, ergibt sich die Gesamtlatenz einer Blockverarbeitung aus der Verzögerungszeit T_{fe} sowie der Wartezeit auf das Einschreiben eines vollständigen Blocks. Ein nichtparalleles blockverarbeitendes Modul mit dem Durchsatz R_{max} erzeugt somit eine Latenz von:

$$T_{Lnpar} = \frac{N_{block}}{R_{max}} + T_{fe} \tag{83}$$

zwischen dem Einschreiben und Ausgeben des ersten Datenwortes eines Blocks der Größe N_{block}. Durch die geringere Datenrate entsteht bei einer blockparallelen Ausführung und der gleichen Blockverarbeitungzeit T_{fe} eine Verzögerung für das erste Datenwort des ersten Blocks von:

$$T_{Lpar} = \frac{N_{block}}{R_{fe}} + T_{fe} = \frac{N_p \cdot N_{block}}{R_{max}} + T_{fe} \tag{84}$$

Bereits unter Vernachlässigung der Verzögerung durch die Zwischenspeicherung zur Ratenanpassung vergrößert sich die Latenz durch die blockparallele Verarbeitung. Während bei der nichtparallelen Ausführung unter Voraussetzung einer Fließbandarchitektur und eines kontinuierlichen Einschreibens der zweite Block nach einer zusätzlichen Einschreibezeit $\frac{N_{block}}{R_{max}}$ ausgegeben wird, entsteht bei der blockparallelen Verarbeitung ein weiterer Latenzbeitrag durch die Wiederherstellung der Blockreihenfolge. Die Zeit zwischen dem Beginn des Einschreibens bis zur vollständigen Ausgabe des ersten Blocks beträgt:

$$T_{B1} = \frac{N_p \cdot N_{block}}{R_{max}} + T_{fe} + \frac{N_p \cdot N_{blout}}{R_{max}} \tag{85}$$

N_{blout} berücksichtigt dabei eventuelle durch die Verarbeitung entstehende Änderungen in der Blockgröße. Die Verzögerungszeit der Zwischenspeicherung wurde wieder vernachlässigt. Bei einem kontinuierlichen Einschreiben ist das erste Datenwort des zweiten Blocks in Bezug auf das Einschreiben des ersten Datenwortes des ersten Blocks nach der Zeit:

$$T_{B2} = \frac{N_{block}}{R_{max}} + \frac{N_p \cdot N_{block}}{R_{max}} + T_{fe} \tag{86}$$

verarbeitet und könnte ausgegeben werden. Ist $N_p \cdot N_{blout} > N_{block}$, wurde zum Zeitpunkt T_{B2} der erste Block noch nicht vollständig ausgegeben, es muss eine zusätzliche Wartezeit:

$$T_{B2w} = \frac{N_p \cdot N_{blout} - N_{block}}{R_{max}} \tag{87}$$

für die Ausgabe des zweiten Blocks eingehalten werden. Eine weitere Wartezeit kommt hinzu, wenn wie bei dem im Abschnitt 4.4 analysierten Streamansatz die einzelnen Blöcke innerhalb eines Basismoduls voneinander abhängig sind, der zweite Block also erst dann ausgegeben werden kann, wenn die erste Instanz des Basismoduls bereits ihren zweiten Block verarbeitet hat. Diese Wartezeit kann zum Schluss der gesamten Verarbeitung wieder aufgeholt werden, erfordert allerdings, dass die nachfolgenden Module nicht die gleiche mittlere Datenrate unterstützen, sondern stattdessen die Spitzendatenrate kontinuierlich verarbeiten können.

Im Gegensatz zur Bitparallelität entstehen durch die Blockparallelität zusätzliche Verzögerungszeiten. Im Hinblick auf die Vermeidung dieser ist eine bitparallele Ausführung der Blockverarbeitung, wie sie z. B. in den bereits genannten Standards IEEE 802.11ad [57] und 802.15.3c [46] vorgesehen ist, die günstigere Variante. Bei der bitparallelen Blockkodierung wird ein auf der Schnittstelle eingehendes Datenwort in mehrere Bereiche unterteilt, die dann als jeweils fortlaufender Teilstrom von einzelnen Blockkodierern verarbeitet werden. Benachbarte Bits werden somit unterschiedlichen Blöcken zugeteilt. Der Nachteil dieser Variante gegenüber der Blockparallelität ist das zwangsweise Entstehen mehrerer Blöcke, auch wenn die insgesamt einlaufenden Daten nur einen Block erfordern würden. Bei einer systematischen Blockkodierung kann eine Codeverkürzung durch das Einfügen von Nullen vorgenommen werden. Diese zusätzlichen Nullen werden nicht mit übertragen, sondern erst im Empfänger vor dem Dekodierer eingefügt. Ist eine solche Codeverkürzung nicht möglich, müssen die zusätzlich eingefügten Daten zum Auffüllen der Blöcke mit übertragen werden und verringern somit die erzielbare Nutzdatenrate.

Es wurde bereits aufgeführt, dass die sich aus der parallelen Streamverarbeitung ergebenden Latenzen aufgrund der Reihenfolgenwiederherstellung durch einen höheren Datendurchsatz in den folgenden Modulen zum Ende der Verarbeitung aufholen lassen. Allgemein gilt für Verzögerungen aufgrund von Reihenfolgenwiederherstellung oder Parameterabhängigkeit, dass sie durch einen höheren Durchsatz in den folgenden Modulen ausgeglichen werden können.

Sei T_1 die Zeit, die für die Verarbeitung der Daten ohne Berücksichtigung der eben genannten Verzögerungen benötigt wird. Werden durch eine Durchsatzsteigerung die während der Wartezeit gespeicherten Daten schneller verarbeitet, nähert sich die benötigte Gesamtverarbeitungszeit T_2 bei einer hinreichenden Menge an zu verarbeitenden Daten wieder der Zeit T_1 an. Diese Durchsatzerhöhung lässt sich sowohl durch Block- als auch Bitparallelität erreichen. Für die Blockparallelität werden die folgenden Module mehrfach instanziiert. Dazu muss sich

der Datenstrom in unabhängige Teilblöcke unterteilen lassen. Ist dies gegeben, bietet sich die Blockparallelität an, da diese aufgrund der Transparenz nach außen ohne Änderungen am restlichen System eingebaut werden kann. Bei der Verwendung einer bitparallelen Lösung, z. B. durch zwei parallele RS-Dekodierer mit der Aufteilung des parallelen Eingangsdatenwortes in eine obere und untere Hälfte, muss auch eine entsprechend gedoppelte Kodierstruktur im Sender vorgesehen werden.

Zusammenfassend lässt sich feststellen, dass blockparallele Lösungen zur Durchsatzsteigerung gegenüber einer entsprechenden bitparallelen Lösung zusätzliche Verzögerungszeiten erzeugen. Der Vorteil der Blockparallelität liegt in der Transparenz, die bei einer Bitparallelität mit mehreren unabhängigen Funktionseinheiten nicht unbedingt gegeben ist, z. B. bei der Durchführung einer parallelen Blockkodierung.

Mit der Überführung der blockparallelen Streamverarbeitung des Beispielsystems in eine bitparallele Variante lässt sich durch den dann möglichen Verzicht auf einen dedizierten Interleaver eine weitere Verringerung der Latenzzeit erreichen. Bitparallelität ermöglicht ein implizites Interleaving, welches entsprechend der Interleaverfunktion die Fehlerkorrektureigenschaften eines Kanalcodes erhöhen kann, ohne eine zusätzliche Verzögerung durch das Interleaving an sich zu erzeugen.

6.4 Implizites Interleaving

Im Abschnitt 2.3 wurde erläutert, dass ein Interleaver die Fehlerkorrekturleistung eines Systems verbessern kann. Durch eine Umordnung der Bits nach der Kodierung und der Wiederherstellung der ursprünglichen Reihenfolge vor der Dekodierung werden durch Kanalstörungen erzeugte Bündelfehler in Einzelfehler umgewandelt. Aufeinanderfolgende Bits bei der Kodierung bzw. Dekodierung werden mit einem möglichst großen Abstand auf dem Kanal übertragen. Speziell für die Viterbidekodierung eines Faltungscodes ist diese Vorgehensweise von Vorteil.

In den vorhergehenden Abschnitten wurde dargestellt, dass es für die Durchsatzerhöhung der Kanalkodierung und -dekodierung unterschiedliche Möglichkeiten gibt. Bei einer blockparallelen Implementierung wird der eingehende Datenstrom in aufeinanderfolgende Blöcke von Bits unterteilt, die von jeweils einem Kanalkodierungsmodul verarbeitet werden. Im EASY-A VHR-E-System entsprechen diese Blöcke den einzelnen OFDM-Symbolen, also der Menge an Bits, die in einem solchen Symbol übertragen werden können. Wie in OFDM-Systemen üblich, erfolgt das anschließende Interleaving nur über die Unterträger eines OFDM-Symbols, also innerhalb eines solchen Blocks.

Eine andere Möglichkeit zur Durchsatzsteigerung ist die Bitparallelität, bei der mehrere unabhängige Instanzen den einlaufenden Datenstrom parallel verarbeiten. Die erste Kodierer-

instanz verarbeitet das erste Bit des parallelen Datenwortes, die zweite Instanz das zweite Bit usw. Diese Verarbeitung ist unabhängig von etwaigen Symbolgrenzen. Wird das parallele Ausgangsdatenwort ohne Umordnung der Bitreihenfolge für die Übertragung verwendet, erfolgt die Dekodierung benachbart übertragener Bits durch unterschiedliche Dekodiererinstanzen. Sind N_p parallele Kodierer und Dekodierer vorhanden, werden Bündelfehler bis zu einer Länge von N_p Bits in Einzelfehler für den jeweiligen Dekodierer umgewandelt. Diese Aufteilung erfolgt ohne das Vorhandensein eines dedizierten Interleavers, das Interleaving wird implizit vorgenommen. Wird unter Berücksichtigung der maximalen Bündelfehlerlänge die Anzahl der unabhängigen parallelen Kodierer und Dekodierer bestimmt, kann auf einen Interleaver verzichtet werden. Damit ergibt sich eine Verkürzung der Gesamtlatenz um die Summe aus Interleaver- und Deinterleaververzögerung.

Die Diagramme in den Abbildungen 26, 27 und 28 zeigen einen Vergleich der Framefehlerraten zwischen dem EASY-A VHR-E-System und einer Variante mit einem impliziten Interleaving für die drei Modulationstypen BPSK, QPSK und 16-QAM mit Coderate ½ und ohne äußeren RS-Code. Für das implizite Interleaving wurden Interleaver und Deinterleaver entfernt und die 24 Kodierer- und Dekodiererinstanzen entsprechend des bitparallelen Ansatzes unabhängig voneinander für jeweils 24 eingehende Datenbits genutzt. Als Kanalmodelle fanden, wie schon bei der Simulation der Pilotunterträgerschemata im Kapitel 5, die Kanalmodelle CM1.2 (LOS-Umgebung) und CM2.3 (NLOS-Umgebung) Verwendung. Für jeden E_b/N_o-Wert wurden 4000 Frames mit je 8192 Datenbytes simuliert. Die dargestellte Framefehlerrate beinhaltet auch Synchronisationsfehler wie nicht erkannte Präambeln sowie Dekodierungsfehler im Signalfeld.

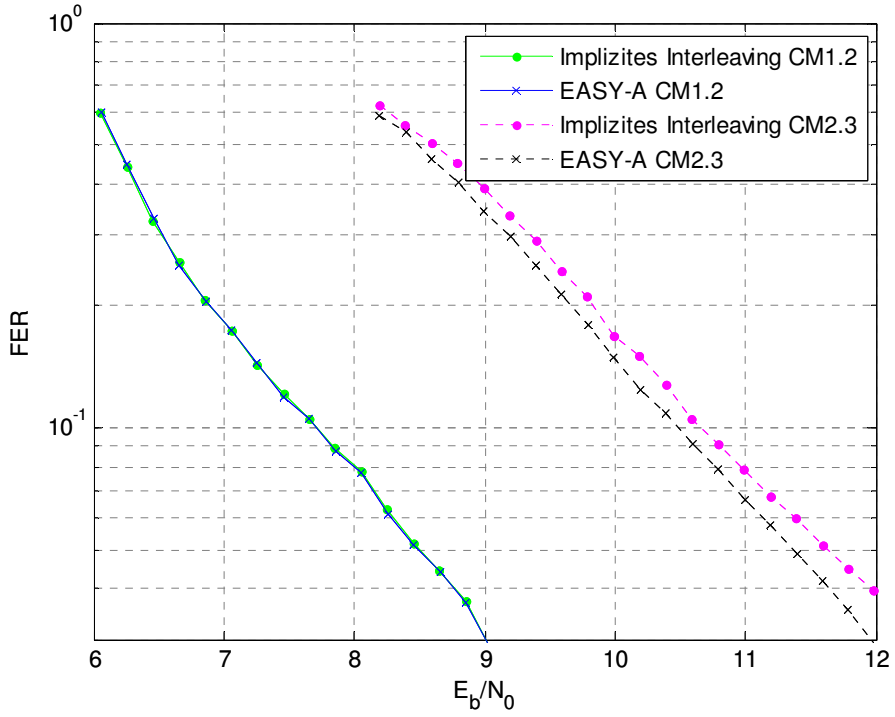

Abb. 26: Vergleich Framefehlerrate BPSK-½-Modulation

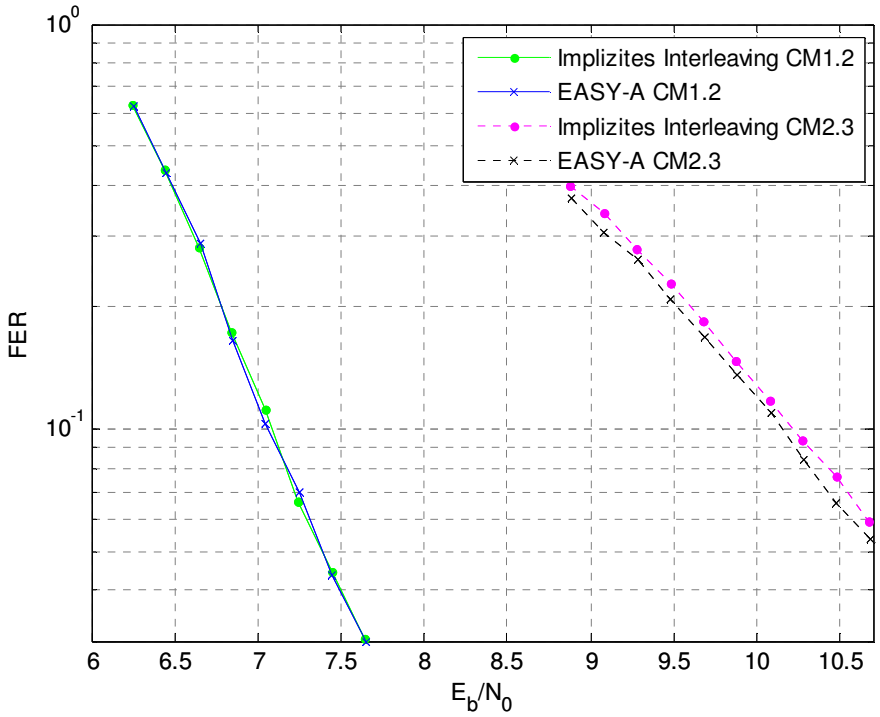

Abb. 27: Vergleich Framefehlerrate QPSK-½-Modulation

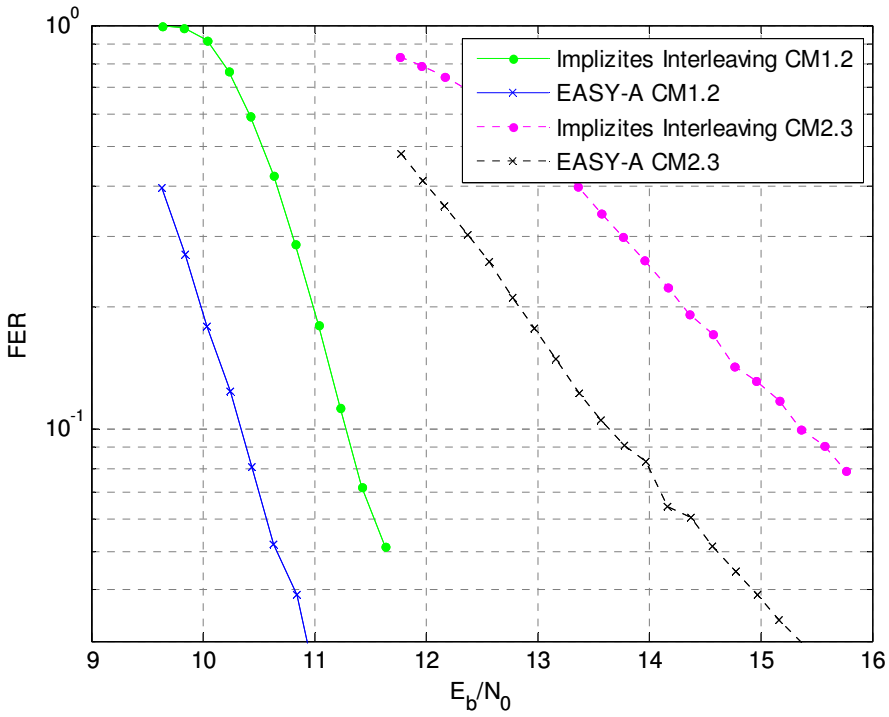

Abb. 28: Vergleich Framefehlerrate 16-QAM-½-Modulation

Während sich unter LOS-Bedingungen mit BPSK- oder QPSK-Modulation die Fehlerraten nicht unterscheiden, gibt es unter NLOS-Bedingungen für diese Modulationen einen leichten Performanceverlust von ca. 0,1 dB bei einer Framefehlerrate von 10 %. Dieser folgt aus der verkürzten Interleavingtiefe. Im EASY-A-System beträgt die Interleavingtiefe für BPSK und QPSK 64, im geänderten System aber nur 24 aufgrund der 24 Kodierer und Dekodierer. Wird die Interleavingtiefe des EASY-A-Systems manuell auf 24 verkürzt, sind die jeweiligen Ver-

läufe der Framefehlerrate für das EASY-A-System und das System mit impliziten Interleaving wie erwartet identisch.

Für die 16-QAM-Modulation zeigt sich dagegen auch unter LOS-Bedingungen mit einem Unterschied von ca. 1 dB bei einer Framefehlerrate von 10 % ein deutlicher Performanceeinbruch bei der Verwendung des impliziten Interleavings. Dieser begründet sich nicht nur in der verkürzten Interleavingtiefe, sondern auch in der unterschiedlichen Zuverlässigkeit der Bits. Für die 16-QAM-Modulation werden 4 Bits zu einem Sendesymbol zusammengefasst. Dabei ergeben sich unterschiedliche Zuverlässigkeiten für die MSBs und LSBs. Um für die aufeinanderfolgenden Bits eines Dekodierers eine Abwechslung zwischen hoher und geringer Zuverlässigkeit auch beim impliziten Interleaving zu erreichen, können für jedes zweite Datenwort jeweils zwei benachbarte Bits miteinander vertauscht werden. Die dadurch erzielbare Verbesserung zeigt Abb. 29. Unter LOS-Bedingungen ist der Performanceverlust nun vernachlässigbar, während er sich unter NLOS-Bedingungen bei einer Framefehlerrate von 10 % von vorher 2 dB auf nunmehr 1 dB halbiert.

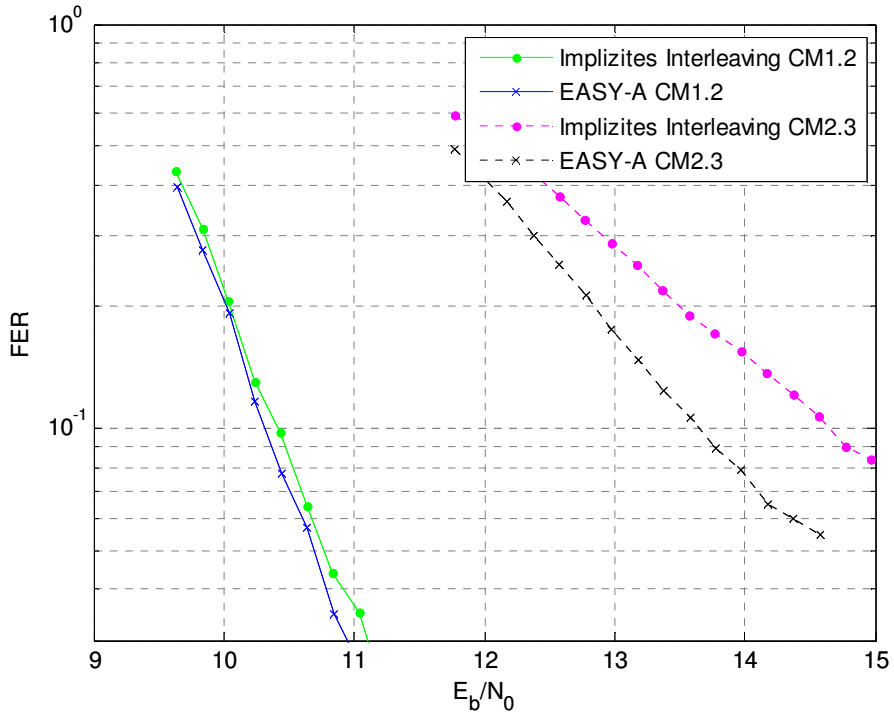

Abb. 29: Vergleich Framefehlerrate bei 16-QAM-½-Modulation mit Bitvertauschung

Ein Nachteil des impliziten Interleavings ist die direkte Abhängigkeit der Interleavingtiefe von der Anzahl an parallelen Modulen. Während bei einem Blockinterleaver die Interleavingtiefe durch die Blockgröße und ein Vertauschen der Auslesereihenfolge variiert werden kann, ist die Interleavingtiefe beim impliziten Interleaving durch die Anzahl an bitparallelen Modulen vorgegeben. Dieser Nachteil kann durch die Verwendung eines Interleavers für jeden der parallelen Zweige behoben werden. Da bereits ein implizites Interleaving erfolgt, kann die Interleavingtiefe und demzufolge die Blockgröße des für jeden Zweig hinzugefügten Interleavers entsprechend verringert werden. Damit reduziert sich die Latenz aufgrund der kleineren

Blockgröße, während die Performance im Hinblick auf die Framefehlerrate identisch gegenüber dem blockparallelen System ist.

Durch die Anwendung des impliziten Interleavings ergibt sich im Beispielsystem aus Kapitel 4 eine Latenzreduktion um rund 1,2 µs zwischen dem Eingang eines Frames am Empfänger und der Ausgabe des ersten Datenwortes an den MAC-Prozessor. Bei der Verarbeitung eines Frames ohne Verwendung des äußeren RS-Codes entspricht diese Zeit 7,7 % der Empfängerverzögerung. Aufgrund der größeren Gesamtverzögerung bei einer Anwendung des äußeren RS-Codes verringert sich die erzielbare Reduktion in diesem Fall auf 5,4 %. Durch eine geeignete Berücksichtigung der Parallelverarbeitung bereits während der System- und Algorithmenspezifikation lassen sich somit Latenzen in der Basisbandverarbeitung verringern oder sogar vermeiden.

7. Latenzverringerung durch Spekulation

7.1 Spekulation in der Datenverarbeitung

Laut Wörterbuch [60] bedeutet Spekulation: „eine Schlussfolgerung über etwas ohne eine gesicherte Erkenntnis“ bzw. „eine Erwartung, dass ein bestimmtes Ereignis oder ein Zustand in der Zukunft eintritt, ohne dafür eine ausreichende Basis zu haben“. Diese Definition legt auf den ersten Eindruck hin nahe, dass Spekulation kein Konzept für die Datenverarbeitung oder Datenkommunikation ist. Trotzdem sind spekulative Verfahren im technischen Bereich bereits seit längerer Zeit weit verbreitet und ihre Bedeutung wird in Zukunft noch weiter zunehmen.

Das erste und vielleicht beste Beispiel für die Anwendung ist die spekulative Programmcodeausführung moderner Prozessoren, auch Sprungvorhersage (Branch prediction) genannt. Den notwendigen hohen Verarbeitungsdurchsatz erreichen diese Prozessoren nur durch eine mehrstufige Pipeline, die z. B. bei der Nehalem-Prozessorarchitektur 16 Stufen umfasst [61]. Mehrere Befehle befinden sich zeitgleich in unterschiedlichen Stufen der Pipeline, bei einer vierstufigen Pipeline könnten das die Stufen „*Befehl holen*“, „*Befehl dekodieren*“, „*Befehl ausführen*“ und „*Ergebnis speichern*“ sein. Sofern das Programm eine lineare Abfolge von Befehlen ohne Sprünge beschreibt, ist die Pipeline immer gefüllt und ein hoher Durchsatz wird erreicht. Typischerweise enthalten jedoch Computerprogramme eine Vielzahl von bedingten Sprungbefehlen, sei es durch die Ausführung von Schleifen oder durch Verzweigungen aufgrund äußerer Einflüsse wie Benutzereingaben oder einem bestimmten Berechnungsergebnis. Sofern ein bedingter Sprungbefehl auftritt, können die nachfolgenden Befehle noch nicht in die Pipeline geladen werden. Die Position zur Fortsetzung des Programms steht erst nach der vollständigen Bearbeitung des Sprungbefehls fest.

Eine einfache Lösung dieses Problems ist das Hinzufügen von leeren Verarbeitungstakten (NOP – no operation) nach einem bedingten Sprungbefehl, sei es durch den Compiler oder den Codeinterpreter im Prozessor. Der Nachteil dieses Verfahrens ist der deutliche Verlust an Durchsatz, wenn häufig bedingte Sprünge ausgeführt werden.

Durch eine Analyse des typischen Sprungverhaltens lässt sich eine bessere Möglichkeit finden, die Sprungvorhersage [62]. Programmschleifen werden im Schnitt sehr häufig wiederholt, d. h. es erfolgt in nahezu allen Fällen ein Rücksprung zum Schleifenanfang [63]. Eine erste Annahme ist demzufolge: Beim Auftreten eines bedingten Sprungbefehls wird immer an die Sprungadresse (zum Schleifenanfang) gesprungen. Also werden die dort definierten Befehle direkt nach dem Sprungbefehl in die Pipeline geladen und verarbeitet. Ohne Kenntnis des tatsächlichen Sprungverhaltens wird auf ein bestimmtes Ergebnis spekuliert.

Was passiert, wenn das Ergebnis des Sprungbefehls nicht die Annahme bestätigt, sondern das Programm an der als unwahrscheinlich verworfenen Adresse fortgesetzt werden soll? In diesem Fall ist die Spekulation fehlgeschlagen, die Berechnungen der nachfolgenden Befehle sind irrelevant oder falsch. Also müssen die Ergebnisse verworfen werden, Datenveränderungen sind nicht erlaubt. Anschließend wird die tatsächlich auszuführende Befehlssequenz geladen. Dieses Umschalten mit Invalidierung der Berechnungsergebnisse bedeutet einen höheren Verwaltungsaufwand, der länger dauert als das Füllen der Pipeline mit NOPs. Erst wenn die Sprungvorhersage häufiger korrekt ist, ergibt sich ein Vorteil.

Moderne Prozessoren, wie die genannte Nehalem-Architektur, beinhalten komplexe Sprungvorhersageeinheiten, die nicht statisch, sondern dynamisch operieren. Für jeden Sprungbefehl wird in einer Tabelle das vorherige Sprungverhalten gespeichert. So kann aufgrund der Analyse der vergangenen Werte das Verhalten vorhergesagt werden, während das tatsächliche Ergebnis des Sprungbefehls zur Verbesserung der Datenbasis genutzt wird. Detaillierte Beschreibungen zu den verwendeten Algorithmen sind in [62] zu finden.

Ein weiteres Beispiel für ein spekulatives Verfahren ist das Prefetching [64], das Laden von Inhalten aus einem langsamen Speicher in einen schnelleren Zwischenspeicher, den Cache, ohne dass die Inhalte angefordert wurden. Im Falle des tatsächlichen Bedarfs kann so eine höhere Zugriffsgeschwindigkeit erzielt werden. Angewendet wird diese Methode in den mehrstufigen Cache-Strukturen moderner Prozessoren oder bei Festplatten. Letztere lesen bei der Anforderung eines bestimmten Blocks gleich die nächsten mit aus und speichern sie in einem DRAM[51]- oder SRAM[52]-basierten Speicher. Sofern die benötigten Daten in aufeinanderfolgenden Blöcken gespeichert sind, erhöht diese Vorgehensweise die Zugriffszeit deutlich.

Bereits im Abschnitt 4.4.2 wurde eine eher unbekannte Anwendung für Spekulation genannt, die in [51] beschriebene Anwendung von Spekulation zur Verringerung der Latenz bei der Synchronisation zwischen verschiedenen Taktdomänen.

In den folgenden Abschnitten wird untersucht, ob und wie sich Konzepte der spekulativen Datenverarbeitung für die Anwendung in der digitalen Basisbandverarbeitung eignen. Dabei steht die Verkürzung der Latenz, also der Verarbeitungszeit zwischen dem Empfang eines Frames an der Antenne und der Auslieferung der enthaltenen Daten an die MAC-Schicht im Mittelpunkt. In diesem Zusammenhang wurde die Anwendung von Spekulation nach bestem Wissen des Autors noch nicht in der Fachliteratur beschrieben.

Die in [49] beschriebene *blinde* Weiterverarbeitung von Symbolen wird dort als spekulativ bezeichnet. Nach Meinung des Autors der vorliegenden Arbeit trifft das dort beschriebene

[51] DRAM: Dynamic Random Access Memory
[52] SRAM: Static Random Access Memory

Verfahren jedoch nicht den Kerngedanken der Spekulation im Kontext der vorliegenden Arbeit.

Bei der *blinden* Weiterverarbeitung passiert Folgendes: Das digitale Frontstage hat einen Frame erkannt und beginnt mit der Verarbeitung der einlaufenden Datensymbole. Bevor jedoch das Signalfeld komplett verarbeitet, also auch demoduliert, dekodiert und interpretiert ist, verfügt das Frontstage über keinerlei Informationen, aus wie vielen Symbolen dieser Datenframe besteht. Würde der einlaufende Datenstrom dagegen bis zum Abschluss der Signalfeldauswertung gespeichert, entstünde eine zusätzliche erhebliche Verzögerung in der Verarbeitung. Um dies zu vermeiden wird angenommen, dass der einlaufende Datenframe aus mehr als den während der Wartezeit einlaufenden Symbolen besteht, also eine kontinuierliche Verarbeitung möglich ist.

Im Unterschied zur Spekulation muss hier bei einem „Misserfolg“ der Annahme, d. h. bei der Verarbeitung von mehr Symbolen als im Frame vorhanden sind, keine erneute Verarbeitung der Symbole erfolgen. Stattdessen werden die überzähligen Symbole aus dem Ausgangspuffer gelöscht. Identisch mit der Spekulation entsprechend dieser Arbeit ist die Notwendigkeit eines Speichers am Ausgang, der die verarbeiteten Symbole so lange puffert, bis über die Erfüllung der Spekulationannahme entschieden werden kann. Spekulation im vorliegenden Kontext bedeutet jedoch weiterhin, dass bei einem „Misserfolg“ der Annahme nicht nur ein Löschen der Ergebnisse, sondern eine erneute vollständige Ausführung der bereits spekulativ vorgenommenen Verarbeitungsschritte notwendig ist. Bei der im folgenden Abschnitt eingeführten *spekulativen Demodulation* ist neben der Modulationsart auch die Anzahl der Datensymbole unbekannt. Zusätzlich zur spekulativen Verarbeitung erfolgt gleichzeitig eine ähnliche *blinde* Weiterverarbeitung.

7.2 Spekulative Demodulation

7.2.1 Einführung

Viele drahtlose Kommunikationssysteme erlauben die Verwendung unterschiedlicher Modulations- und Kodierungsarten, um die Datenübertragung zur Laufzeit an geänderte Bedingungen und Anforderungen anzupassen. So kann bei einer Verschlechterung der Kanaleigenschaften z. B. auf eine niedere Modulationsart zurückgeschaltet oder die Coderate der Kanalkodierung verringert werden. Die Empfangsstation muss über diese Änderungen informiert werden, um eine korrekte Demodulation und Dekodierung vornehmen zu können. Dazu gibt es drei prinzipielle Möglichkeiten.

Zum einen kann ein Protokoll verwendet werden, bei dem durch den Austausch von Steuerungs- und Bestätigungsinformationen der beabsichtigte Wechsel angekündigt, bestätigt und anschließend zeitgleich durchgeführt wird. Auch wenn durch die Bestätigung sichergestellt

wird, dass beide Stationen gleichzeitig die Änderungen vornehmen, gibt es zusätzliche Fehlerfälle, die im Protokoll beachtet werden müssen. So ist denkbar, dass nach dem Umschalten auf eine höhere Modulationsart durch einen sich zeitgleich verschlechternden Kanal keine Pakete mehr fehlerfrei dekodiert werden können. In diesem Fall muss entweder vorgesehen werden, dass die Stationen nach einer gewissen Zeit selbstständig auf eine niedrigere Modulationsart zurückschalten, oder die Steuerungsinformationen müssen immer in der robustesten Modulationsart übertragen werden. Letzteres bedeutet allerdings, dass anhand einer weiteren Seiteninformation dem Empfänger mitgeteilt wird, ob es sich bei dem aktuell empfangenen Paket um Steuerungsinformationen oder normale Daten handelt.

Die zweite Möglichkeit greift auf die soeben angesprochenen zusätzlichen Seiteninformationen zurück. Mit jedem Frame wird ein Signalfeld übertragen, welches Informationen über die verwendete Modulationsart und weitere für die Dekodierung notwendige Parameter enthält. Dieses Signalfeld hat dabei eine vorher spezifizierte feste Position innerhalb des Frames und ist auch mit einer vorher festgelegten Kodierung und Modulationsart versehen. Der Empfänger ist also immer in der Lage, anhand des Signalfeldes die für das Datenfeld verwendeten Parameter zu extrahieren, ohne dass ein spezielles Protokoll für den Austausch dieser Informationen benötigt wird. Der Verzicht auf ein möglicherweise komplexes Protokoll wird erkauft durch eine Verringerung des theoretisch möglichen Datendurchsatzes aufgrund des Overheads für die Signalisierung. Diese erfolgt mit jedem Frame, unabhängig davon, ob tatsächlich Änderungen an den Parametern erfolgten oder nicht. Trotzdem ist diese zweite Vorgehensweise aufgrund der Universalität zur Übermittlung framespezifischer Zusatzinformationen sehr gebräuchlich.

Als dritte Möglichkeit bietet sich in manchen Fällen die Nutzung eines zusätzlichen Seitenkanals an. So könnten z. B. über eine ultrahochratige unidirektionale 60-GHz-Verbindung Videodaten von einem Server an einen Client übertragen werden, während der Austausch von Steuerungsinformationen über eine parallel bestehende klassische WLAN-Verbindung erfolgt.

Im Folgenden wird die zweite Variante zugrunde gelegt. Am Beispiel des bereits früher eingeführten 60-GHz-OFDM-Systems (Kapitel 4) wird untersucht, inwieweit durch eine *spekulative Demodulation* die Latenz der digitalen Basisbandverarbeitung verringert werden kann.

7.2.2 Prinzipbeschreibung

Spekulative Demodulation beschreibt, wie der Name schon sagt, ein Verfahren, bei dem die empfangenen Symbole demoduliert werden, ohne dass das System sichere Informationen über die verwendete Modulationsart besitzt. Diese stehen stattdessen erst zu einem späteren Zeitpunkt zur Verfügung. Daraus ergibt sich die Verwendung der im vorherigen Abschnitt aufge-

führten zweiten Möglichkeit zu Signalisierung der Übertragungsparameter mithilfe eines Signalfeldes. Verallgemeinernd wird in dieser Arbeit die Bezeichnung *spekulative Demodulation* für ein Verfahren verwendet, bei dem die Gesamtheit oder ein Teil der Übertragungsparameter beim Empfänger vor der Signalfeldauswertung unbekannt ist. Es wird also neben der Modulationsart auch auf das gewählte Punktierungsschema und den verwendeten Kanalcode spekuliert. Eine reine Spekulation auf die Modulationsart würde im Beispielsystem die Latenz des Gesamtsystems nur um die Interleaverlatenz verringern, das Demapping selbst erzeugt keine signifikante Latenz (siehe Abschnitt 4.4.3). Die spekulativ demodulierten Signale könnten zwar den Deinterleaver passieren, eine Dekodierung ist aufgrund des unbekannten Punktierungsschemas jedoch nicht möglich. Erst durch eine Spekulation auf die Gesamtheit der Übertragungsparameter lassen sich Datensymbole bereits parallel zur Signalfeldverarbeitung verarbeiten. Im Gegensatz dazu steht die *spekulative Dekodierung*, bei der nicht auf die Art der Kodierung, sondern auf eine fehlerfreie Übertragung der Daten spekuliert wird. Weiteres zur *spekulativen Dekodierung* findet sich im Abschnitt 7.3.

Die resultierende Framestruktur bei der Verwendung eines Signalfeldes ist in Abb. 30 gezeigt, der schematische Aufbau der zugrunde gelegten Basisbandverarbeitung in Abb. 31. Der Frame besteht aus einer Präambel, gefolgt von dem Signalfeld und den Datensymbolen. Die Verarbeitung eines empfangenen Frames läuft wie folgt ab: Nach einer erfolgreichen Synchronisation wird das Signalfeld an den Demapper übergeben. Dieser extrahiert die gesendete Bitsequenz und liefert sie an den Deinterleaver weiter, anschließend wird das Signalfeld dekodiert. Nachdem aus den dekodierten Signalfeldinformationen die Modulationsart der Datensymbole bestimmt wurde, kann diese an den Demapper weitergegeben werden, der daraufhin mit der Verarbeitung der Datensymbole beginnen kann. Da der einlaufende Symbolstrom kontinuierlich ist, werden die Datensymbole in einem Puffer bis zum Abschluss der Signalfeldauswertung zwischengespeichert.

Abb. 30: Framestruktur

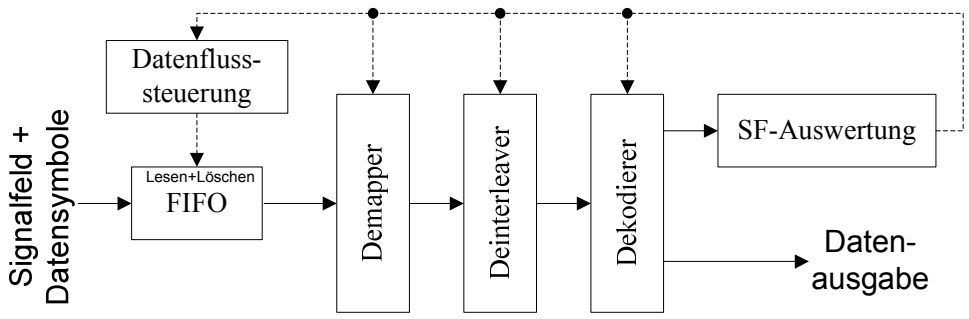

Abb. 31: Grundlegende Struktur der Basisbandverarbeitung

Bei einer Punkt-zu-Punkt-Verbindung werden sich die Modulationsart der Datensymbole sowie die gewählte Kodierung und Punktierung eher selten ändern. So wird der Verbindungsaufbau meist mit einer niedrigeren Datenrate vorgenommen und bei einem stabilen Verhalten anschließend auf eine Modulationsart für eine höhere Datenrate umgeschaltet.

Es ist demzufolge naheliegend anzunehmen, dass die Modulationsart der vorherigen Frames auch für den aktuellen Frame gültig ist, analog zur statischen Sprungvorhersage: *Sprung wird ausgeführt*. Mithilfe dieser Annahme müssen die Datensymbole nicht mehr bis zur Auswertung des Signalfeldes gespeichert werden, sondern können parallel dazu demoduliert und dekodiert werden.

Nach Fertigstellung der Signalfeldverarbeitung wird die angenommene Hypothese überprüft. Sofern das Ergebnis positiv ist, ist die Datendemodulation und -dekodierung korrekt und die Verzögerungszeit des Systems wurde reduziert. Trifft die Hypothese nicht zu, ist die Spekulation fehlgeschlagen. Die Datenverarbeitung muss abgebrochen und mit den gültigen Parametern neu gestartet werden. Die Behandlung von Spekulationsmisserfolgen sowie die Einflüsse der *spekulativen Demodulation* auf die Systemstruktur werden im folgenden Abschnitt 7.2.3 dargestellt.

Auch bei einer zutreffenden Hypothese kann es notwendig sein, die laufende Datenverarbeitung abzubrechen. Dieser Fall tritt dann auf, wenn die Zeit für die Signalfelddekodierung und -auswertung länger ist als die zeitliche Dauer der Datensymbole und gleichzeitig die Anzahl der Datensymbole im Signalfeld kodiert wurde. In diesem Fall werden aufgrund der blinden Weiterverarbeitung mehr Datensymbole verarbeitet als ursprünglich gesendet. Nach der Signalfeldauswertung müssen die überzählig verarbeiteten Daten aus der Verarbeitungspipeline gelöscht werden.

Nicht nur bei Punkt-zu-Punkt-Verbindungen, auch in komplexeren Netzwerken mit häufig wechselnden Übertragungsparametern für die eingehenden Frames, bietet Spekulation einen Vorteil in Bezug auf die mittlere Latenzverkürzung. Dazu ist eine Erweiterung der Hypothesenbestimmung notwendig. So können die Modulationsarten der vorher empfangenen Frames gespeichert und ausgewertet werden, um aufgrund einer Mehrheitsentscheidung oder einer Musteranalyse die Hypothese aufzustellen.

7.2.3 Basisbandstruktur zur Behandlung von Spekulationsmisserfolgen

Zur Ableitung der resultierenden Systemstruktur für eine *spekulative Demodulation* sind unterschiedliche Erfordernisse bei der Behandlung von Spekulationsmisserfolgen zu beachten. Bei einem Spekulationsmisserfolg ist zum einen die laufende Verarbeitung der Datensymbole abzubrechen und mit den gültigen Parametern neu zu starten. Zum anderen muss sicherge-

stellt werden, dass bei einem Misserfolg kein Teil der spekulativ vorverarbeiteten Datensymbole an nachfolgende Verarbeitungseinheiten weitergereicht wird.

Damit die Datensymbole bei einem Spekulationsmisserfolg erneut verarbeitet werden können, müssen die eingehenden Datensymbole parallel zur spekulativen Verarbeitung gespeichert werden. Ein solcher Speicher ist auch bei der nichtspekulativen Verarbeitung notwendig. Bei dieser müssen die Datensymbole so lange gepuffert werden, bis das Signalfeld dekodiert wurde. Dazu kann eine gemäß Abb. 31 in den Datenfluss eingebettete FIFO-Struktur verwendet werden. Während das Signalfeld gleich weitergegeben wird, werden die einlaufenden Datensymbole erst nach der Signalfelddekodierung und -auswertung aus der FIFO gelesen. Im Gegensatz zur nichtspekulativen Verarbeitung können die Daten bei der spekulativen Vorgehensweise nicht nach der Auslieferung aus dem Eingangspuffer gelöscht werden, sondern erst dann, wenn die Hypothese als zutreffend verifiziert wurde. In Bezug auf die Systemimplementierung bedeutet dies, dass nicht mehr eine in den Datenfluss eingebettete FIFO-Strukur verwendet werden kann. Stattdessen wird entsprechend Abb. 32 eine in den Datenfluss eingebettete Speicherstruktur benötigt, bei der das Lesen und Löschen von Datensymbolen getrennte Operationen sind.

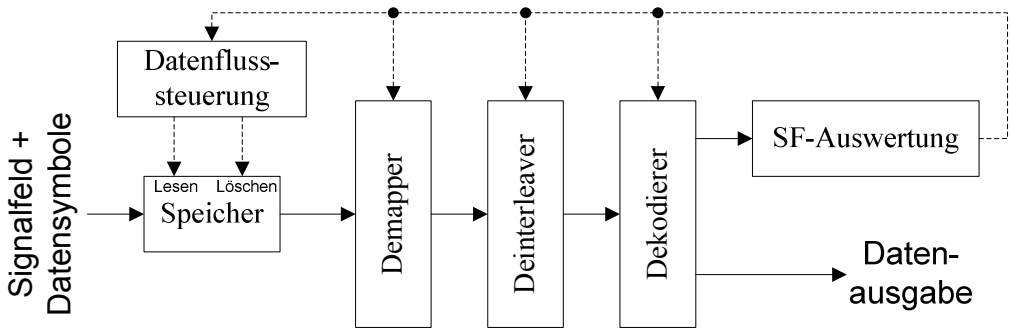

Abb. 32: Spekulative Basisbandverarbeitung mit Speicher am Eingang

Alternativ ist es durch die in Abb. 33 dargestellte Änderung der Speichereinbindung in den Datenfluss möglich, auch bei der *spekulativen Demodulation* eine FIFO-Struktur zur Speicherung der eingehenden Datensymbole für eine erneute Verarbeitung bei einem Spekulationsmisserfolg zu verwenden. Bei der *spekulativen Demodulation* können die Datensymbole direkt nach dem Signalfeld an das Verarbeitungsmodul weitergegeben werden. Es ist im Allgemeinen keine Pause im Datenstrom erforderlich. Somit muss der Speicher nicht mehr in den eigentlichen Datenfluss integriert werden, sondern kann parallel zur Eingangsschnittstelle des spekulativen Moduls aufgebaut werden. Eingehende Daten werden sowohl direkt verarbeitet als auch in der FIFO gespeichert. Bei einem Spekulationsmisserfolg wird auf den FIFO-Pfad umgeschaltet und die Verarbeitung erfolgt wie bei der nichtspekulativen Vorgehensweise.

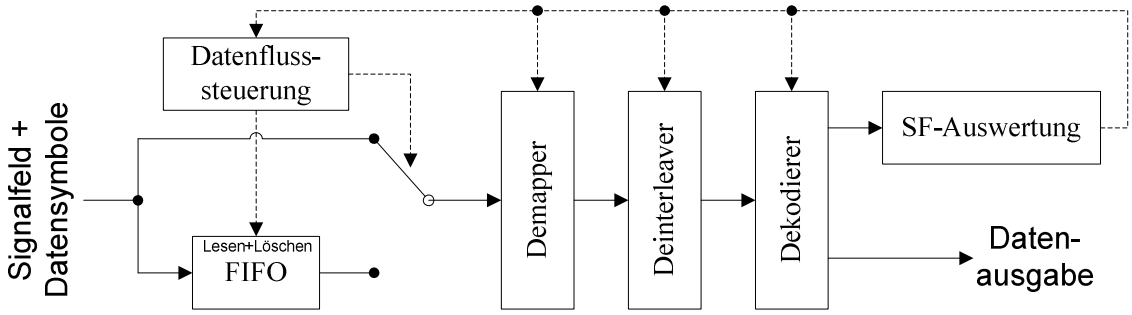

Abb. 33: Spekulative Basisbandverarbeitung mit parallelem FIFO-Pfad am Eingang

Die *spekulative Demodulation* benötigt demzufolge nicht mehr Speicher als die nichtspekulative Verarbeitung. Je nach verwendetem Integrationsschema des Speichers ändert sich nur die Ansteuerung. Bei der direkten Einbindung in den Datenfluss kann keine einfache FIFO-Struktur verwendet werden.

Für die Sicherstellung der Nichtweitergabe spekulativer Ergebnisse an die nachfolgenden Module bei einem Spekulationsmisserfolg ist die Verarbeitung innerhalb des spekulativen Moduls zu beachten. Erfolgt wie in Abb. 34 dargestellt die entsprechende Verarbeitung symbolsequentiell für Signalfeld und Daten, werden also die einlaufenden Symbole nacheinander in ihrer ursprünglichen Reihenfolge verarbeitet, reicht der Abbruch der laufenden Datensymbolverarbeitung aus, sofern für die Überprüfung der Spekulationsannahme keine zusätzliche Verarbeitungszeit benötigt wird. Ist dies nicht gegeben, muss entsprechend Abb. 35(a) bzw. (b) eine FIFO am Eingang oder ein Zwischenspeicher am Ausgang vorgesehen werden. Bei der Variante (a) wird durch eine geeignete Leseansteuerung der Eingangs-FIFO zwischen dem Signalfeldsymbol und dem ersten Datensymbol eine Pause in den fortlaufenden Symbolstrom eingefügt, die mindestens so groß wie die Zeit für die Überprüfung der Spekulationsannahme ist. Durch diese Pause wird sichergestellt, dass während der Hypothesenüberprüfung keine Datensymbole fertig verarbeitet werden. Somit ist kein Speicher am Ausgang notwendig. Bei einem Spekulationsmisserfolg wird die laufende Verarbeitung abgebrochen und neu gestartet. Ist der Eingangsdatenstrom dagegen ohne Pause, können durch die Zwischenspeicherung am Ausgang entsprechend der Variante (b) bereits verarbeitete Datensymbole bei einem Spekulationsmisserfolg gelöscht werden. Die Ausgabe der Datensymbole an die nachfolgenden Einheiten wird bis zur Auswertung der Spekulationshypothese verzögert.

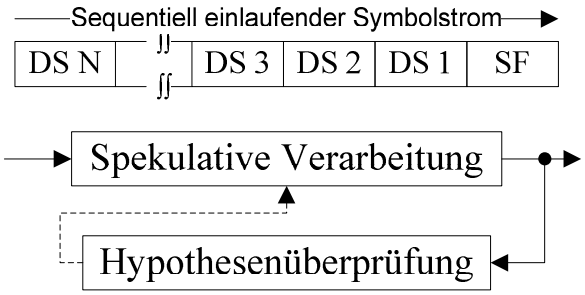

Abb. 34: Symbolsequentielle spekulative Verarbeitung

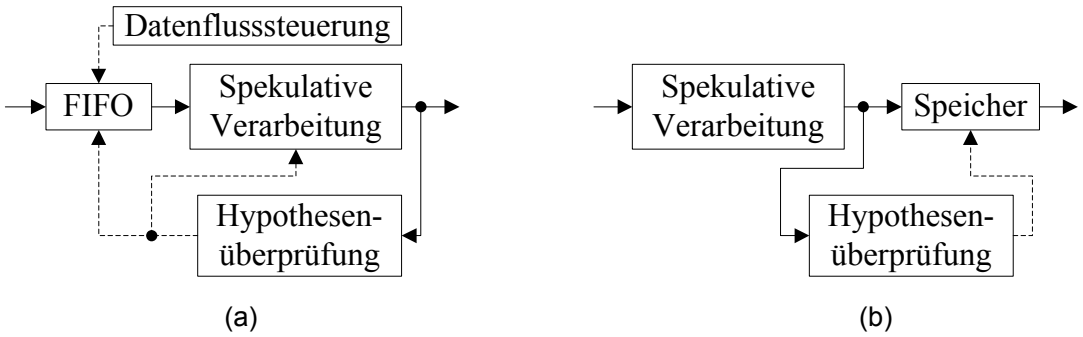

Abb. 35: Symbolsequentielle spekulative Verarbeitung für lange Hypothesenüberprüfung

Wie bereits erläutert, muss für die *spekulative Demodulation* ein Speicher am Eingang vorgesehen werden, um bei einem Spekulationsmisserfolg eine erneute Verarbeitung durchführen zu können. Damit empfiehlt sich für die symbolsequentielle Verarbeitung die Variante nach Abb. 35(a) zur Sicherstellung des korrekten Abbruchs der Datenverarbeitung. Wie mit einer zwischengeschalteten FIFO durchläuft der gesamte Symbolstrom das Speichermodul. Nach dem Signalfeld wird der Ausleseprozess für die benötigte Zeit der Spekulationshypothesenauswertung gestoppt.

Obwohl bei der vorgestellten Variante analog zur Verarbeitung ohne Spekulation zwischen Signalfeld und Datensymbolen eine Pause eingefügt wird, verkürzt sich die Latenzzeit der gesamten Verarbeitung. Wird für die Verarbeitung eines Symbols die Zeit T_s benötigt, entspricht dies der zusätzlichen Latenz für die Signalfeldverarbeitung im Fall ohne Spekulation. Mit Anwendung der *spekulativen Demodulation* ist nicht mehr T_s, sondern $T_{chkspec}$ als Zeit für die Überprüfung der Spekulationshypothese ausschlaggebend. Im Allgemeinen kann davon ausgegangen werden, dass $T_{chkspec} << T_s$ ist. Bei einem Spekulationserfolg entspricht die zusätzliche Latenz $T_{chkspec}$ und im Falle eines Spekulationsmisserfolges $T_{chkspec} + T_s \approx T_s$. Der Latenzbeitrag $T_{chkspec}$ für den Spekulationserfolg kann auch nicht vermieden werden, wenn statt der Pause im Symbolstrom ein Zwischenspeicher am Ausgang vorgesehen wird. Dieser muss die Datensymbole ebenfalls für die Zeit $T_{chkspec}$ zwischenspeichern und verzögert somit die Datenausgabe aus dem Modul um ebendiese Zeit.

Bei der sequentiellen Verarbeitung von Signalfeld und Datensymbolen innerhalb eines Verarbeitungspfades ist sichergestellt, dass zuerst das Signalfeldsymbol vollständig verarbeitet wurde, bevor der erste Teil des folgenden Datensymbols aus dem Modul ausgegeben wird. Erfolgt die Verarbeitung von Signalfeld und Datensymbolen dagegen wie in Abb. 36 dargestellt parallel in unterschiedlichen Pfaden, kann nicht mehr davon ausgegangen werden, dass während der laufenden Signalfeldverarbeitung keine Datensymbole fertig verarbeitet wurden. Um trotzdem bei einem Misserfolg der Spekulation die spekulativ berechneten Daten löschen zu können und diese nicht an die nachfolgenden Stufen weiterzugeben, ist ein Speicher im Anschluss an die Verarbeitung notwendig. Dieser speichert die berechneten Daten so lange, bis über den Spekulationserfolg entschieden werden kann.

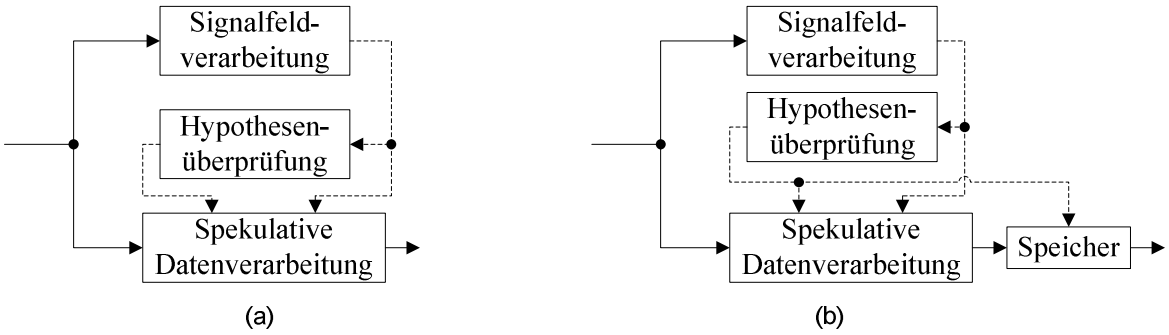

Abb. 36: Struktur spekulative Verarbeitung bei Parallelverarbeitung

Da sich im Allgemeinen die Übertragungsparameter von Signalfeld und Datensymbolen unterscheiden, ergeben sich in beiden Pfaden unterschiedliche Verarbeitungszeiten. Abhängig von den jeweiligen Verarbeitungszeiten müssen zwei Fälle unterschieden werden. Mit T_{sfv} als Verarbeitungszeit des Signalfeldes und T_{ds} als Verzögerungszeit zwischen der Ein- und Ausgabe eines Datensymbols ergeben sich die zwei folgenden Varianten:

Ist

$$T_{sfv} + T_{chkspec} < T_{ds} \tag{88}$$

werden während der Signalfeldverarbeitung und -auswertung keine Datensymbole fertig verarbeitet. Die Systemstruktur nach Abb. 36(a) mit einem Abbruch der laufenden Datenverarbeitung ist in diesem Fall ausreichend. Gilt dagegen:

$$T_{sfv} + T_{chkspec} > T_{ds} \tag{89}$$

muss entsprechend Abb. 36(b) ein Speicher am Ausgang vorgesehen werden, der die verarbeiteten Datensymbole bis zur Überprüfung der Spekulationshypothese speichert.

Während sich in der ersten Variante nach Abb. 36(a) die Latenz bei einem Spekulationserfolg um T_{sfv} verringert, bleibt dieser Betrag in der zweiten Variante nach Abb. 36(b) durch die Verzögerung der berechneten Daten am Ausgang bestehen. Nur der zusätzliche Latenzbeitrag durch die Verarbeitung der Datensymbole wird eliminiert. Die detaillierte Latenzmodellierung erfolgt im nächsten Abschnitt.

In realen Systemen für hohe Datenraten, wie dem im Kapitel 4 vorgestelltem EASY-A VHR-E-System, wird bereits ein hoher Grad an Parallelität genutzt. So werden im Kanaldekodierungsmodul mehrere Datensymbole mit dem erläuterten Streamansatz parallel verarbeitet. Der maximale Datendurchsatz eines einzelnen Verarbeitungspfades für einen Stream ist dabei kleiner als die Datenrate am Ausgang des Dekodierers. Erst durch die gleichzeitige Nutzung mehrerer Streams ergibt sich eine höhere Rate. Damit am Ausgang die Ursprungsreihenfolge der Symbole wiederhergestellt werden kann, sind für jeden einzelnen Stream Ausgangs-FIFOs notwendig. Diese speichern die Symbole aus den anderen Verarbeitungspfaden, bis das

Symbol aus dem aktuellen Pfad komplett ausgegeben wurde. Wird ein solches System um die *spekulative Demodulation* erweitert, muss kein zusätzlicher Speicher in das System integriert werden. Stattdessen werden die Ausgangs-FIFOs gegen Speichereinheiten ausgetauscht, die ein Löschen von vorverarbeiteten Symbolen ermöglichen.

7.2.4 Latenzmodell für die spekulative Demodulation

Bei Anwendung spekulativer Methoden wie der *spekulativen Demodulation* ist die Verarbeitungszeit im digitalen Basisband nicht fix, sondern hängt vom Spekulationserfolg ab. Die Latenz ist unter Vernachlässigung anderer variabler Verzögerungsbeiträge eine diskrete statistische Größe mit zwei möglichen Werten. Deshalb ist eine Erweiterung des im Kapitel 4 eingeführten Latenzmodells notwendig.

Die *spekulative Demodulation* verringert im Erfolgsfall die Latenz aufgrund der Parameterabhängigkeit zwischen Signalfeld und Datensymbolen. Im eingeführten Latenzmodell zeigt sich diese in der Zeit T_{sof}, also der Verzögerung zwischen der Framesynchronisation und der Ausgabe des ersten Datenwortes aus dem Basisband. Dabei ist der Beitrag der Signalfeldverarbeitung T_{sfv} nur implizit durch T_{dfs} in der Formulierung (37) für T_{sof} enthalten. Wie erläutert, bedingt eine geringe Variabilität von T_{sfv} keine Änderungen von T_{dfs}.

Die *spekulative Demodulation* im Basissystem ergibt sich aus einer Kombination von symbolsequentieller und symbolparalleler Verarbeitung. Erstere ist durch das Demapping und Deinterleaving gegeben, zweitere durch die streambasierte Dekodierung. Da die vollständige Signalfeldverarbeitung länger als die Verarbeitung des ersten Datenwortes des ersten Datensymbols dauert, verringert sich die Latenz $T_{sofspec}$ bei einem Spekulationserfolg um die Zeit für das Demapping und Deinterleaving sowie die Dekodierung des ersten Datenwortes:

$$T_{sofspec} = \begin{cases} T_{sof} - T_{dd} - T_{de1} & \text{| Spekulation erfolgreich} \\ T_{sof} & \text{| Spekulation fehlgeschlagen} \end{cases} \tag{90}$$

7.2.5 Ergebnisse

Zur Diskussion der Ergebnisse durch die Anwendung der *spekulativen Demodulation* wird das 60-GHz-OFDM-Kommunikationssystem aus Kapitel 4 vorausgesetzt. Für dieses wurden die erforderlichen Änderungen im eingeführten Latenzmodell im vorherigen Abschnitt dargestellt. Die für die vollständige Verarbeitung des Signalfeldes benötigte Zeit ist größer als die Verzögerungszeit bei der Verarbeitung eines Datensymbols, es gilt die Bedingung (89). Die Zeit für den Test der Spekulationshypothese kann dabei gegenüber T_{sfv} vernachlässigt werden.

Entsprechend der im Abschnitt 4.4.3 vorgenommenen Latenzmodellierung beträgt die Zeit zwischen der Framesynchronisation und der Ausgabe des ersten Datenwortes ohne Verwendung des äußeren RS-Codes rund 8,8 µs. Dies entspricht damit auch der Latenz, die bei der *spekulativen Demodulation* im Falle eines Spekulationsmisserfolgs auftritt. Die notwendige Zeit für den Abbruch der Datenverarbeitung ist vernachlässigbar klein, die gespeicherten spekulativ verarbeiteten Daten werden durch die erneut verarbeiteten Datensymbole überschrieben. Ist das Überschreiben bei der Verwendung von FIFO-Strukturen nicht möglich, kann das Löschen der entsprechenden FIFO-Einträge parallel zur erneuten Verarbeitung erfolgen. Auch für das Löschen entsteht keine zusätzliche Verzögerung.

Bei einer erfolgreichen Spekulation verringert sich die Latenz um die Zeit T_{dd} für das Demapping und Deinterleaving sowie die Zeit T_{de1} für die Dekodierung des ersten Datensymbols, also insgesamt um ca. 1,7 µs. Dies entspricht einer Reduktion von rund 19 % gegenüber der Variante ohne Spekulation. Der für die Anwendung der *spekulativen Demodulation* entstehende höhere Ressourcenbedarf bei der Systemimplementierung aufgrund der notwendigen Steuerung ist unter Berücksichtigung der bereits bestehenden Systemkomplexität vernachlässigbar. Wie bereits ausgeführt wurde, sind am Ausgang des streambasierten Dekodierungsmoduls bereits FIFOs zur Zwischenspeicherung der Datensymbole vorhanden. Da gleichzeitig auch am Ausgang des Frontstages ein Zwischenspeicher für die Datensymbole existiert, muss nur die Speicheransteuerung geändert werden, es ist kein zusätzlicher Speicher erforderlich.

Durch die *spekulative Demodulation* ergeben sich abhängig vom Spekulationserfolg variable Latenzzeiten. Diese stellen besondere Anforderungen an die MAC-Schicht, sofern ein Vorteil aus der framespezifischen Latenzreduktion gewonnen werden soll. Wird dagegen nur der schlechteste Fall, also das Fehlschlagen der Spekulation, für das Design des MACs in Betracht gezogen, ergibt sich kein Vorteil für das Gesamtsystem. Einige der genannten Anforderungen und Auswirkungen spekulativer Verfahren auf die MAC-Schicht werden im Abschnitt 7.5 kurz diskutiert. Beim Beispielsystem treten zusätzlich auch variable Latenzen in Abhängigkeit des gewählten Kodierungsschemas auf. So vergrößert die Verwendung des äußeren RS-Codes die Latenzzeit bis zur Datenausgabe. Auch diese Variabilität muss beim Design der MAC-Schicht beachtet werden.

7.3 Spekulative Dekodierung

7.3.1 Einführung und Prinzipbeschreibung

Neben der *spekulativen Demodulation* gibt es eine weitere Möglichkeit zur Latenzverkürzung durch die Anwendung von Spekulation, die *spekulative Dekodierung*. Wie bereits angeführt wurde, bedeutet *spekulative Dekodierung* innerhalb dieser Arbeit nicht die Spekulation auf die Art der Kanalkodierung, sondern die Spekulation auf Fehlerfreiheit der empfangenen Da-

ten. So ist eine *spekulative Depunktierung*, also die Spekulation auf das verwendete Punktierungsschema, ein Unterpunkt des s*pekulative Demodulation* genannten Verfahrens.

Die hier eingeführte *spekulative Dekodierung* eignet sich für systematische Blockcodes, d. h. Kodierungsschemata, bei denen das ursprüngliche Nachrichtenwort ein unveränderter Bestandteil des kodierten Datenwortes ist. Gleichzeitig sollten für die *spekulative Dekodierung* die einzelnen unabhängig kodierten Blöcke möglichst groß sein. Diese Bedingung ergibt sich zum einen dadurch, dass ein Vorteil im Hinblick auf die Latenzreduktion nur bei einer signifikanten Verarbeitungszeit für die Blockdekodierung entsteht. Zum anderen ergibt sich, wie später in diesem Abschnitt ausgeführt, ein Overhead für die Überprüfung der Spekulationshypothese. Damit dessen Einfluss auf die Nettodatenrate möglichst klein ist, ist die Verwendung von großen Blöcken notwendig. Eine genauere Spezifikation zur minimal sinnvollen Blocklänge findet sich im Abschnitt 7.3.3. Die vorgegebenen Bedingungen – systematischer Code und möglichst große Blocklänge – treffen auf den im Beispielsystem EASY-A verwendeten RS-Code zu.

Im Rahmen der Latenzanalyse (siehe Abschnitt 2.10) wurde bereits aufgeführt, dass Blockcodes in einer datenstrombasierten Verarbeitungsumgebung einen signifikanten Beitrag zur Gesamtlatenz leisten. Dies liegt unter anderem daran, dass für eine Dekodierung erst der komplette Datenblock im Dekodierungsmodul vorhanden sein muss. Um die durch die Blockkodierung entstehende Latenz im Empfänger zu verringern, wird durch die *spekulative Dekodierung* angenommen, die Daten wären fehlerfrei. Damit werden sie gleichzeitig mit der Eingabe in das Dekodierungsmodul in einen Ausgangspuffer übergeben. Dieser speichert die Daten bis zur Überprüfung der tatsächlichen Fehlerfreiheit. Nachdem über ein geeignetes Verfahren die Fehlerfreiheit der übertragenen Daten getestet wurde, kann über die Spekulationshypothese entschieden werden. Trifft diese zu, d. h. die Daten sind fehlerfrei, wird die laufende Verarbeitung der Daten im Dekodierungsmodul gestoppt. Gleichzeitig startet die Ausgabe der spekulativ in den Ausgabepuffer übertragenen Daten.

Sind dagegen Übertragungsfehler aufgetreten, werden die spekulativ in den Ausgabepuffer übertragenen Daten invalidiert bzw. gelöscht. Parallel dazu wird die Dekodierung weiter durchgeführt. Nach der erfolgten Dekodierung werden die nun fehlerkorrigierten Daten ausgegeben.

Die Entscheidung über den Erfolg der Spekulationsannahme kann auf zwei Weisen vorgenommen werden. Zum einen kann parallel zum Einschreiben des Blocks eine Berechnung der Prüfbits erfolgen. Diese werden dann mit den am Schluss des Blockes angehängten empfangenen Prüfbits verglichen. Zum anderen kann jedem Datenblock sendeseitig vor der Kodierung eine einfach auszuwertende Prüfsumme (z. B. eine CRC) hinzugefügt werden. Der entstehende erweiterte Block wird als Basis für die Blockkodierung verwendet. Im Empfänger wird parallel zur Eingabe der Daten in das Dekodierungsmodul die Prüfsumme fortlaufend

berechnet und durch einen Vergleich mit der empfangenen Prüfsumme ausgewertet. Stimmt die Prüfsumme mit der erwarteten überein, kann die Übertragung der an die Nutzdaten durch die Blockkodierung angehangenen Prüfbits an das Dekodierungsmodul gestoppt werden, die Spekulationsannahme der Fehlerfreiheit ist zutreffend. Die Gesamtlatenz verringert sich um die Verarbeitungszeit der Blockdekodierung sowie die Zeit für das Einschreiben der zusätzlichen Prüfbits. Stimmen die Prüfsummen nicht überein, wird die Blockdekodierung weiter durchgeführt. Die Wahl des Prüfsummenalgorithmus unterliegt dabei Randbedingungen, die im Abschnitt 7.3.3 erläutert werden.

Durch das Hinzufügen einer Prüfsumme verringert sich die Nettodatenrate, da der Anteil der Nutzdaten pro Kodierungsblock um die Größe der Prüfsumme verringert wird. Werden dagegen die durch die Kodierung hinzugefügten Prüfbits zur Auswertung der Spekulationshypothese verwendet, verringert sich die Nettodatenrate nicht. Eine detaillierte Darstellung des Einflusses verschiedener CRC-basierter Prüfsummen auf die Nettodatenrate findet sich im Abschnitt 7.3.3.

Im Vergleich zur Bildung der Prüfbits ist die Berechnung einer Prüfsumme im Allgemeinen einfacher und mit weniger Ressourcenaufwand zu implementieren. So erfordert die Berechnung einer 32-Bit-CRC für einen 8-bit-breiten Datenbus auf einem FPGA 13 LUT[53]s, während das im Beispielsystem verwendete RS-Kodierungsmodul mit einem 8-bit-breiten Datenbus insgesamt 206 LUTs benötigt.

Durch die Verwendung einer zusätzlichen Prüfsumme ergeben sich weitere Vorteile. So können die fehlerkorrigierten Daten auf Fehlerfreiheit getestet werden, die Sicherheit in Bezug auf nicht oder falsch korrigierte Übertragungsfehler wird erhöht. Durch die im Abschnitt 7.5 beschriebene Zusammenfassung von MAC und PHY im Hinblick auf die Prüfsummen lassen sich Einschränkungen in der Nettodatenrate verringern oder vermeiden.

7.3.2 Basisbandstruktur

Die Struktur des Basisbands für eine *spekulative Dekodierung* unterscheidet sich von der bereits vorgestellten Struktur für die *spekulative Demodulation*. Zum einen ist am Eingang kein Speicher notwendig, um die Daten bei einer fehlgeschlagenen Spekulation erneut verarbeiten zu können. Die Daten werden stattdessen gleichzeitig in einen Ausgangspuffer und an das Dekodierungsmodul weitergegeben. Weiterhin wird die Verarbeitung nicht bei einem Spekulationsmisserfolg, sondern bei einem Spekulationserfolg abgebrochen. Deshalb ist bei der *spekulativen Dekodierung* der Ausgangspuffer auch unabhängig von der für die Dekodierung benötigten Verarbeitungszeit notwendig. Während der bei der *spekulativen Demodulation* notwendige Eingangspuffer die Datensymbole speichert, um diese beim Spekulationsmisser-

[53] LUT: Look-up table

folg erneut verarbeiten zu können, speichert der bei der *spekulativen Dekodierung* notwendige Ausgangspuffer die Daten, um diese bei einem Spekulationsmisserfolg löschen zu können.

Für die in Abb. 37 dargestellte Struktur eines spekulativen Dekodierungsmoduls wurde zur Überprüfung der Spekulationsannahme, entsprechend der im vorhergehenden Abschnitt 7.3.1 vorgestellten zweiten Variante, eine blockweise Prüfsummenberechnung vorgesehen,.

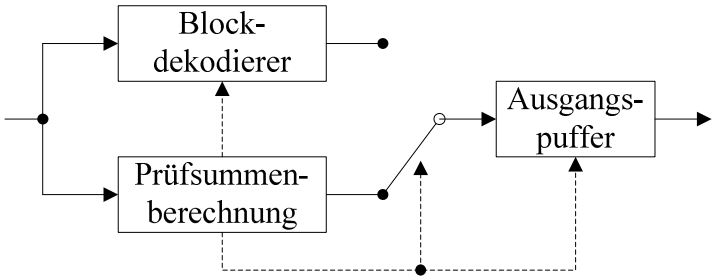

Abb. 37: Struktur spekulatives Dekodierungsmodul

Neben der Speicherung der spekulativ in den Ausgangspuffer übertragenen Daten muss dieser eine weitere Aufgabe erfüllen, die Wiederherstellung der ursprünglichen Blockreihenfolge. Ein Blockdekodierer arbeitet zwar mit einzelnen abgeschlossenen Datenblöcken, der gesamte übertragene Frame besteht im Allgemeinen aber aus mehreren Blöcken. Die Notwendigkeit einer Umsortierung der verarbeiteten Blöcke soll am folgenden Beispiel erläutert werden.

Angenommen sei eine Sequenz von fünf Blöcken B_1 bis B_5. Der Blockdekodierer ermöglicht ein kontinuierliches Einschreiben von Blöcken und benötigt die Zeit T_{block} zur internen Verarbeitung nach dem vollständigen Einschreiben eines Datenblocks. Werden die Datenblöcke kontinuierlich eingeschrieben, wird so nach einer anfänglichen Latenz von T_{block} ein kontinuierlicher Datenstrom ausgegeben. (Der Ausgabedatenstrom ist bei einem kontinuierlichen Eingangsstrom nur dann kontinuierlich, wenn die Datenrate entsprechend der durch den Dekodierer entfernten Redundanz verringert wird, sonst entstehen Lücken zwischen der Ausgabe der einzelnen Blöcke.)

Das Einschreiben in den Blockdekodierer erfordert die Zeit T_{wd}, das Einschreiben in den Ausgangspuffer die Zeit T_{wa}. Da die Paritybits nicht mit in den Ausgangspuffer übertragen werden müssen, gilt:

$$T_{wa} < T_{wd} \tag{91}$$

unter der Annahme, dass die vom Kodierer hinzugefügten Prüfbits hinter dem Datenblock angefügt werden. Ohne Einschränkung der Allgemeinheit wird diese Annahme für folgende Diskussion zur Wiederherstellung der Blockreihenfolge vorausgesetzt. Der Blockdekodierer benötigt zur Ausgabe der Daten und damit gleichzeitig zum Einschreiben in den Ausgabepuffer die Zeit T_{rd}. Diese Zeit entspricht im Allgemeinen T_{wa}, da das gleiche Interface verwendet und die gleiche Anzahl an Datenwörtern ausgegeben wird. Die zur Spekulationsprüfung not-

wendige Prüfsumme braucht nicht in den Ausgabepuffer übertragen werden, sofern sie nicht noch weiter verwendet wird. Die Prüfung der Spekulationshypothese erfordert die Zeit $T_{chkspec}$, die je nach Implementierung gegenüber T_{wa} vernachlässigt werden kann. Für die folgende Diskussion wird angenommen:

$$T_{wa} + T_{chkspec} < T_{wd} \tag{92}$$

d. h. die Zeit für das Prüfen der Spekulationshypothese mithilfe der Prüfsumme ist kürzer als die Zeit für das Einschreiben der Prüfbits in den Blockdekodierer. Zur Ausgabe eines Datenblocks aus dem Ausgabepuffer wird die Zeit T_{ra} benötigt.

Der Blockdekodierer wird anhand der oben gezeigten Struktur um eine *spekulative Dekodierung* erweitert. Die Blöcke B_1, B_2, B_4 und B_5 seien fehlerfrei (die Prüfsummenberechnung ergibt einen Spekulationserfolg), während der Block B_3 fehlerhaft sei und einen Spekulationsmisserfolg bewirkt. (Es ist dabei unerheblich, ob die Fehler innerhalb des Blockdekodierers korrigiert werden können, ohne Einschränkung der Allgemeinheit sei dies jedoch angenommen.) Im Falle eines Spekulationserfolgs wird für die komplette Übertragung eines Blocks vom Eingang in den Ausgangspuffer die Zeit:

$$T_{succ} = T_{wa} \tag{93}$$

benötigt, im Falle eines Spekulationsmisserfolges die Zeit:

$$T_{fail} = T_{wd} + T_{block} + T_{rd} \tag{94}$$

Entsprechend des vorgestellten Verfahrens werden die einzelnen Blöcke in ihrer Ursprungsreihenfolge sowohl in den Ausgangspuffer als auch in den Blockdekodierer übertragen. Am Ende des dritten Blocks ergibt die Prüfsummenberechnung einen Fehler, sodass die Daten aus dem Ausgangspuffer gelöscht werden. Während der dritte Block im Dekodierer verarbeitet wird, wird parallel der vierte in den Ausgangspuffer und den Blockdekodierer eingeschrieben. Entsprechend (94) und (92) gilt mit $T_{rd} = T_{wa}$ auch bei $T_{block} = 0$ für Blöcke, die durch den Blockdekodierer laufen:

$$T_{fail}(T_{block}=0) = (T_{wd} + T_{rd}) = (T_{wd} + T_{wa}) > (2 \cdot T_{wa} + T_{chkspec}) \tag{95}$$

Ist zusätzlich

$$T_{chkspec} << T_{wd} - T_{wa} \tag{96}$$

gilt

$$T_{fail}(T_{block}=0) > 2 \cdot T_{wa} + n \cdot T_{chkspec} \tag{97}$$

auch für eine Menge von n ∈ ℕ mit:

$$2 \le n \le \left\lfloor \frac{T_{wd} - T_{wa}}{T_{chkspec}} \right\rfloor . \tag{98}$$

Damit kann für den vierten Block die Spekulationsannahme überprüft werden, bevor der dritte Block vollständig im Ausgangspuffer liegt. Trotz Spekulationserfolg darf der Block B_4 nicht ausgegeben werden, da sonst die Ausgabesequenz B_1, B_2, B_4, B_3, B_5 lauten würde. Mit $T_{block} > T_{wd}$ kann sogar bereits die Überprüfung für den fünften Block erfolgen, der vierte Block wurde überprüft, bevor überhaupt mit dem Einschreiben des dritten Blocks in den Ausgangspuffer begonnen wurde.

Die Ausgabe des dritten Blocks aus dem Ausgabepuffer kann direkt während des Einschreibens vorgenommen werden. Es stellt sich die Frage, ob die durch den Blockdekodierer verarbeiteten Blöcke überhaupt in den Ausgangspuffer geschrieben werden müssen oder ob sie nicht direkt ausgegeben werden können. Eine solche Struktur zur direkten Ausgabe der durch den Blockdekodierer verarbeiteten Blöcke ist in Abb. 38 dargestellt. Die Zeit T_{ra} ist nun nur noch für die Blöcke mit Spekulationserfolg ausschlaggebend. In der folgenden Analyse muss berücksichtigt werden, dass T_{rd} in dieser Struktur nicht mehr unbedingt der Zeit T_{wa} entspricht.

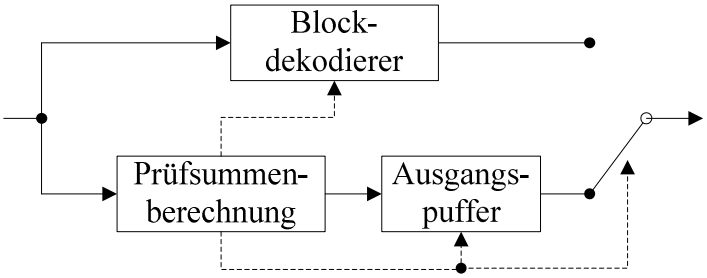

Abb. 38: Spekulativer Blockdekodierer mit direkter Ausgabe

Die Gesamtlaufzeit bei einem Spekulationserfolg ergibt sich zu:

$$T_{succ2} = T_{wa} + T_{chkspec} + T_{ra} \tag{99}$$

und bei einem Spekulationsmisserfolg zu:

$$T_{fail2} = T_{wd} + T_{block} + T_{rd} \tag{100}$$

Sind die entstehenden Zeiten für beide Pfade identisch, gilt also $T_{succ2} = T_{fail2}$, müssen die Ausgabedaten des Blockdekodierers nicht zwischengespeichert werden. Stattdessen wird zur Wiederherstellung der Blockreihenfolge zwischen den beiden Pfaden am Ausgang über einen

Multiplexer hin- und hergeschaltet. Unter dieser Bedingung ergibt sich jedoch kein Vorteil durch die Spekulation, die Verzögerungszeit bleibt gleich.

Mit $T_{fail2} > T_{succ2}$ ergibt sich eine zusätzliche Verzögerung im oberen Pfad, die den eigentlichen Anwendungsfall für die *spekulative Dekodierung* darstellt. Da alle Blöcke im Dekodiererpfad diese zusätzliche Verzögerung erfahren, können die Blöcke aus dem Ausgabepuffer direkt in den Datenstrom eingefügt werden. Für $T_{fail2} < T_{succ2}$ ist dies nicht gegeben. Diese Bedingung bedeutet jedoch, dass der spekulative Pfad eine höhere Latenz erzeugt als der eigentliche Verarbeitungspfad, sodass auf die Spekulation generell verzichtet werden sollte.

Für die *spekulative Dekodierung* ist eine Struktur nach Abb. 38 ausreichend, der Ausgabepuffer muss nur im spekulativen Pfad vorgesehen werden. Ein gemeinsamer Ausgabepuffer ist möglich, vergrößert jedoch den Umsetzungsaufwand. Es müsste eine Speicherstruktur mit zwei Schreibports und einem Leseport implementiert werden. Während eine Speicherstruktur mit zwei unabhängigen Leseports und einem Schreibport einfach aus zwei parallelen Speichern mit jeweils einem unabhängigem Lese- und Schreibport aufgebaut werden kann, erfordert eine Speicherstruktur mit zwei unabhängigen Schreibports und einem Leseport andere Konzepte.

Die notwendige Speichertiefe des Ausgabepuffers lässt sich aus T_{fail2} und T_{succ2} ermitteln. Mit N_{block} als Anzahl der Datenwörter pro Block ergibt sich die minimale Speichertiefe D_{smin} zu:

$$D_{smin} = \frac{N_{block}}{T_{wd}} \cdot \left(T_{fail\,2} - T_{succ\,2}\right) \tag{101}$$

Der erste Faktor gibt die maximale Einschreiberate in den Ausgangspuffer an. Für den gesamten Dekodierer entspricht diese nicht dem Kehrwert aus T_{wa}, sondern dem Kehrwert von T_{wd}. Die Blöcke können in das spekulative Dekodierungsmodul nicht schneller eingeschrieben werden, als der eigentliche Dekodierer diese verarbeiten kann.

7.3.3 Wahl des Prüfsummenalgorithmus

Wie im Abschnitt 7.3.1 erläutert, ist zur Bestimmung des Spekulationserfolges die Verwendung einer zusätzlichen Prüfsumme sinnvoll. Die Verwendung der durch die Blockkodierung hinzugefügten Prüfbits erfordert dagegen im Allgemeinen einen höheren Ressourcen- und Implementierungsaufwand.

Die Wahl des Prüfsummenalgorithmus unterliegt zwei Bedingungen. Die erste wurde bereits genannt, die Überprüfung sollte auf der vorgegebenen Zielplattform einfach und effizient umzusetzen sein, sodass gegenüber der Dekodierung Verarbeitungszeit eingespart werden kann und der entstehende höhere Ressourcenverbrauch für das Gesamtsystem vernachlässigbar ist.

Günstigstenfalls ist die Verarbeitungszeit $T_{chkspec}$ gegenüber den Einschreibezeiten T_{wd} bzw. T_{wa} vernachlässigbar. Zweitens muss die Prüfsumme eine Fehlererkennungsleistung bieten, die mindestens der Fehlerkorrekturleistung der verwendeten Blockkodierung entspricht. Ist die Fehlererkennungsrate geringer, wird möglicherweise ein verfälschtes Datenpaket als fehlerfrei deklariert und verfälscht ausgegeben, obwohl die Kanaldekodierung die enthaltenen Fehler hätte korrigieren können. Sofern die Fehlererkennungsleistung der Prüfsumme und die Korrekturleistung der Blockdekodierung nicht direkt miteinander verglichen werden können, muss die statistische Wahrscheinlichkeit für korrigierbare fehlerhafte Blöcke, die durch die Prüfsumme als fehlerfrei deklariert werden, anhand der Systemsimulation überprüft werden.

Für das im Abschnitt 2.2 eingeführte CRC-Prüfsummenverfahren ist kein direkter Vergleich der Fehlererkennungsrate mit der Fehlerkorrekturrate eines RS-Dekodierers möglich. Stattdessen ist ein statistischer Vergleich mithilfe der Blockfehlerrate (BLER[54]) notwendig. Für den in Abb. 39 dargestellten Blockfehlerratenvergleich wurde ein einfaches Modell, bestehend aus einem Blockkodierer, einem BPSK-Modulator, einem AWGN[55]-Kanal, einem BPSK-Demodulator und einem Blockdekodierer zugrunde gelegt. Als Blockcode kam der (255,239)-RS-Code des EASY-A-Beispielsystems zum Einsatz. Zusätzlich wurde die *spekulative Dekodierung* in das Modell aufgenommen, indem jeder Block vor der Kodierung um eine CRC erweitert wurde. Sofern diese CRC nach dem Demodulator als gültig erkannt wurde, erfolgte keine RS-Dekodierung des entsprechenden Blocks. Stattdessen wurde der Dateninhalt des empfangenen Blocks direkt mit den gesendeten Daten verglichen. Sofern Fehler auftraten, wurde der Block entsprechend als fehlerhaft registriert. Als CRCs wurden verschiedene gebräuchliche Varianten verwendet, die allgemein als CRC-4, CRC-8 und CRC-16 bekannt sind. Für jeden E_b/N_0-Wert wurde ein fortlaufender Datenstrom aus 500.000 Blöcken simuliert. Zum Vergleich sind im Diagramm zusätzlich die Blockfehlerraten für das gleiche System ohne *spekulative Dekodierung* sowie für ein System ohne RS-Kodierung dargestellt.

Im Diagramm ist gut zu erkennen, dass die Fehlererkennungsleistungen der CRC-4 und CRC-8 für eine Verwendung als Spekulationsprüfsumme bei der gewählten Blockkodierung zu gering sind. Es werden Blöcke als gültig gekennzeichnet, obwohl diese fehlerhaft sind und durch den Dekodierer korrigiert werden könnten. Eine 16-Bit-CRC zeigt dagegen praktisch keinen Unterschied gegenüber der Variante ohne *spekulative Dekodierung*, die Blockfehlerraten sind annähernd gleich. Dies gilt auch für längere CRCs wie eine CRC-24 und CRC-32, die aus Gründen der Übersichtlichkeit nicht im Diagramm dargestellt wurden.

[54] BLER: Block Error Rate
[55] AWGN: Additive White Gaussian Noise

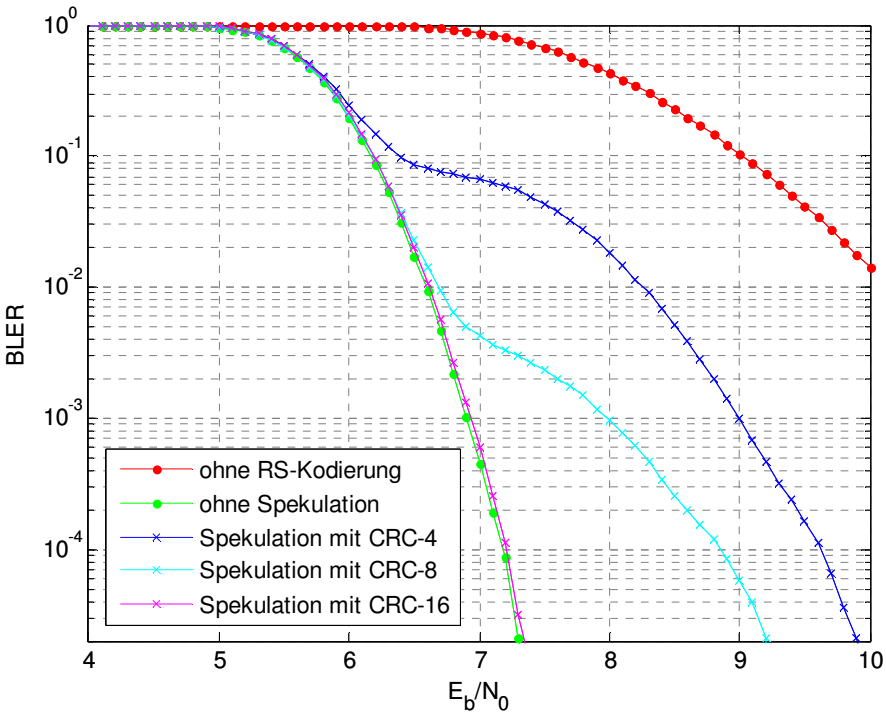

Abb. 39: Blockfehlerrate bei spekulativer Dekodierung für verschiedene CRCs

Eine den Blöcken hinzugefügte Prüfsumme ermöglicht über die Kontrolle der Spekulationsannahme hinaus den Test der dekodierten Daten auf Fehlerfreiheit. So kann kontrolliert werden, ob alle durch die Prüfsumme erkennbaren Fehler beseitigt und ob durch die Blockdekodierung neue Fehler eingefügt wurden. Die Ausgabedaten der Blockdekodierung sind somit verlässlicher.

Durch die Hinzufügung einer Prüfsumme verringert sich jedoch die bei gleichbleibenden Übertragungsparametern erzielbare Datenrate entsprechend:

$$R_{spec} = R_{nospec} \cdot \left(1 - \frac{N_{chkbit}}{N_{nutz}}\right) \tag{102}$$

mit R_{nospec} als Nutzdatenrate des Systems ohne Spekulation, N_{nutz} als der Anzahl der Nutzdatenbits pro Block ohne Spekulation, N_{chkbit} als Größe der Prüfsumme und R_{spec} als resultierender Datenrate. Bei dem im EASY-A-Beispielsystem verwendeten (255,239)-RS-Code und einer 32-Bit-CRC als Prüfsumme ergibt sich also eine Verringerung der Nutzdatenrate um 1,7 %.

Im gesamten Beispielsystem wird die erzielbare Datenrate jedoch auch durch den Overhead für Präambel und Signalfeld sowie die mögliche Datenwortanzahl innerhalb eines OFDM-Symbols beeinflusst. Diese Faktoren verringern den Einfluss der zusätzlichen CRC und somit deren prozentualen Anteil an der gesamten Reduktion der Datenrate.

Entspricht die Menge an zu sendenden Daten nicht exakt dem Produkt aus der durch die Modulations-, Kodierungs- und Punktierungsart vorgegebenen Datenwortanzahl pro OFDM-Symbol und einer ganzzahligen Anzahl an OFDM-Symbolen, müssen den zu sendenden Daten zusätzliche Füllwörter hinzugefügt werden. Dadurch ist es möglich, dass die zusätzlichen Bytes durch eine CRC teilweise überhaupt keinen Einfluss auf die Nettodatenrate haben. Das Diagramm in Abb. 40 zeigt anhand des Beispielsystems die erzielbare Nettodatenrate für verschiedene CRC-Größen als Funktion der Anzahl von Datenbytes innerhalb eines Frames. Zum Vergleich ist gleichzeitig die erzielbare Datenrate für ein nichtspekulatives System ohne äußere RS-Kodierung dargestellt. Für die Berechnung der Nettodatenrate wurden alle weiteren Einflussfaktoren wie die Präambel und das OFDM-Präfix berücksichtigt, als Modulationstyp fand BPSK Verwendung. Während für kurze Frames die zusätzliche CRC noch keine signifikante Auswirkung auf die Nettodatenrate hat, ergibt sich bei größeren Datenbyteanzahlen eine Reduktion um etwa 2 % bei Verwendung einer 32-Bit-CRC gegenüber der Variante ohne zusätzliche CRC in jedem RS-Block. Im Vergleich zur Datenratenreduktion durch die Verwendung der RS-Kodierung von 7 % fällt dieser Anteil gering aus.

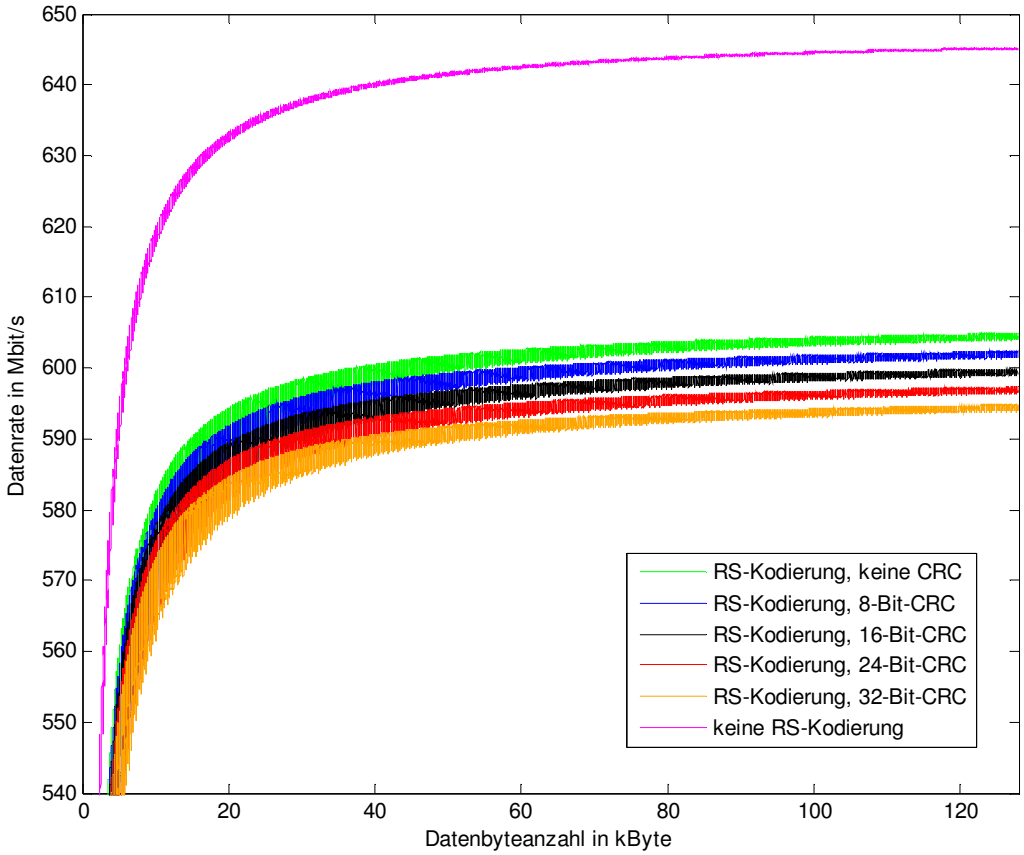

Abb. 40: Datenratenvergleich bei spekulativer Dekodierung für verschiedene CRCs

Die dargestellten Kurven sind nicht monoton, die Werte einer lokalen Umgebung springen stattdessen zwischen einer oberen und unteren Schranke. Dieses Verhalten begründet sich in der treppenfunktionsförmigen Abhängigkeit zwischen der Framedauer und der Anzahl an enthaltenen Datenwörtern. Die Abhängigkeit ist in Abb. 41(a) dargestellt, die Diagramme (b)

und (c) zeigen zwei Ausschnitte der in (a) dargestellten Kurven. In Tabelle 4 sind die jeweiligen Werte für gebräuchliche Datenbytenanzahlen aufgeführt.

Der Overhead durch die Verwendung einer blockweisen CRC zeigt sich in der unterschiedlichen Steigung der Kurven. Das Verhältnis von RS-Blockgröße zur Datenwortanzahl pro Symbol bestimmt den jeweiligen Stufenabstand, während die Stufengröße durch die RS-Blockgröße festgelegt ist. Durch das nicht ganzzahlige Verhältnis zwischen RS-Blockgröße und Datenwortanzahl pro Symbol ergeben sich teilweise kleinere Zwischenstufen. Während im unteren Bereich die Kurven teilweise übereinanderliegen und es somit keinen Unterschied in der resultierenden Symbolanzahl gibt, differieren die Verläufe für größere Datenwortanzahlen aufgrund der unterschiedlichen Steigung. Dies entspricht dem in Abb. 40 gezeigten Einschwingen auf einen annähernd konstanten Datenratenverlust für sehr große Frames.

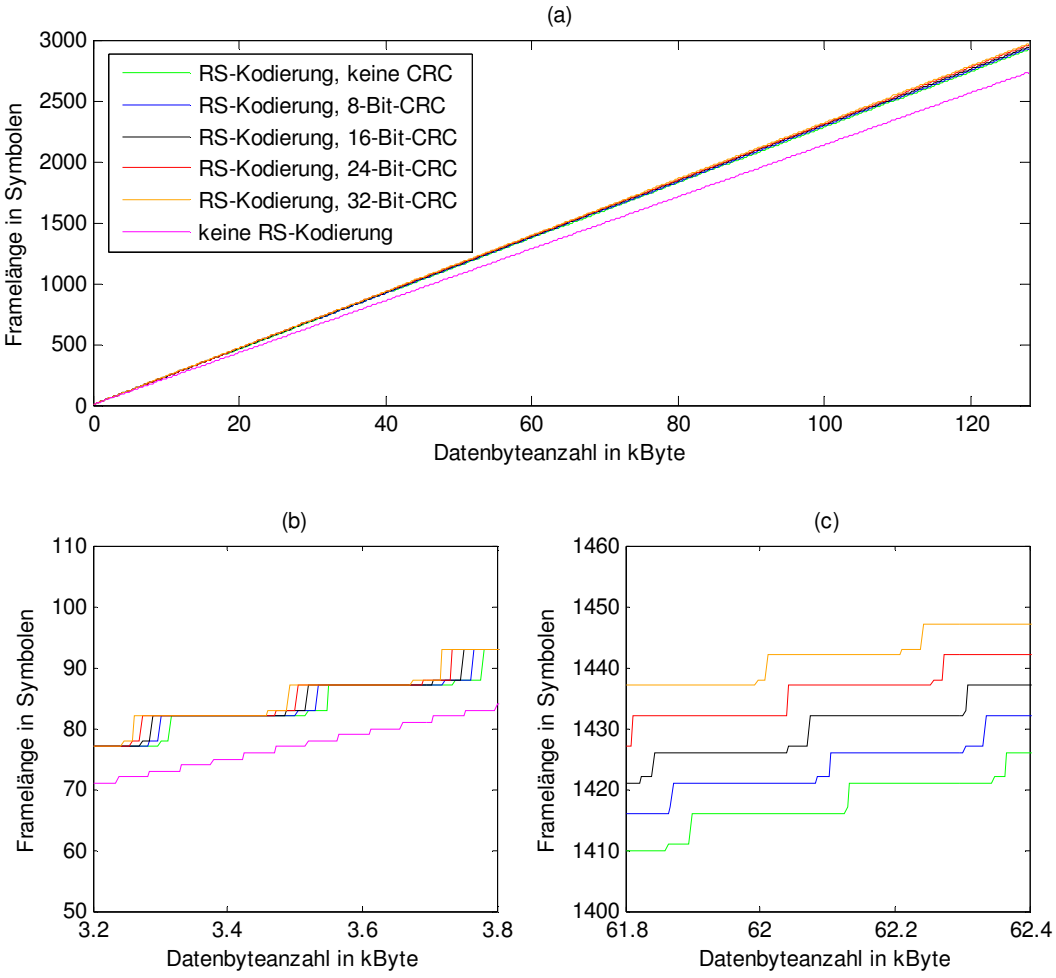

Abb. 41: Symbolanzahl eines Frames als Funktion von Datenbyteanzahl und CRC-Größe

Datenbyte-anzahl	ohne RS	mit RS, ohne CRC	RS + CRC-8	RS + CRC-16	RS + CRC-24	RS + CRC-32
2 kB	52	57	57	57	57	57
4 kB	95	100	105	105	105	105
8 kB	180	195	195	195	195	195
16 kB	351	376	376	377	381	381
32 kB	692	738	743	748	748	753
64 kB	1375	1470	1476	1481	1486	1492
128 kB	2740	2926	2937	2948	2963	2974

Tabelle 4: OFDM-Symbolanzahl für verschiedene Datenbyteanzahlen und CRCs

7.3.4 Ergebnisse

Wie im Abschnitt 7.3.2 ausgeführt, verringert sich im Falle einer erfolgreichen Spekulation die Verzögerungszeit um:

$$T_{fail2} - T_{succ2} \approx T_{block} \tag{103}$$

Beim RS-Dekodierer des EASY-A-Beispielsystems beträgt die Zeit zwischen dem Start des Einschreibens des ersten Datenblocks und der Ausgabe des ersten dekodierten Datenworts 2,8 µs. Dafür wurde ein kontinuierlicher Datenstrom am Eingang vorausgesetzt. Das Einschreiben eines Datenblocks von 239 Byte über den 32-bit-parallelen Datenbus in einen Speicher dauert bei einer Taktrate von 200 MHz 300 ns, das Einschreiben eines Datenblocks von 255 Byte 320 ns. Im Beispielsystem wird für die Ausgabe ein Takt von 125 MHz verwendet, sodass für das Auslesen 480 ns benötigt werden. Aufgrund der Architektur des RS-Dekodierers mit drei parallelen byteverarbeitenden Blockdekodierern (siehe Kapitel 4) ist für den ersten Block eine größere Ausgabezeit von 1,2 µs notwendig, erst mit einer Folge von mehreren kontinuierlich eingeschriebenen Blöcken verringert sich diese auf die angegebenen 480 ns. Damit beträgt die gesamte Zeit für die Verarbeitung eines Blocks inklusive der vollständigen Ein- und Ausgabe rund 3,3 µs unter Voraussetzung einer kontinuierlichen Verarbeitung mehrerer Blöcke.

Bei einer *spekulativen Dekodierung* kann der Prüfsummentest innerhalb eines Taktes erfolgen, $T_{chkspec}$ beträgt also 5 ns. Wird für die notwendige Steuerung des Einschreibens und Auslesens des Speichers ein zusätzlicher Overhead von maximal zehn Takten in Betracht gezogen, beträgt die Blockverarbeitungszeit bei einer erfolgreichen Spekulation 835 ns. Die Latenz verringert sich um 74,7 %. Wird nur die Verzögerung bis zur Ausgabe des ersten Datenwortes betrachtet, verringert sich die Latenz sogar um 87,3 %.

Durch eine erfolgreiche *spekulative Dekodierung* ergibt sich eine große Reduktion der Latenzzeit des Verarbeitungssystems, ohne auf die zusätzliche Korrekturleistung im Fehlerfall zu verzichten. Durch die Hinzufügung einer Prüfsumme verringert sich jedoch im Allgemeinen die bei gleichbleibenden Übertragungsparametern erzielbare Datenrate.

Gegenüber einer reinen Weitergabe der Eingangsdaten wird auch im Erfolgsfall bei der *spekulativen Dekodierung* die Datenausgabe um einen Datenblock verzögert. Durch eine Erweiterung der spekulativen Verarbeitung auf den gesamten Basisband-Verarbeitungspfad lässt sich diese Verzögerung teilweise weiter optimieren. Die notwendigen Änderungen werden im Abschnitt 7.4 diskutiert.

Bei der kontinuierlichen Verarbeitung einer Serie von Blöcken ergibt sich durch die Wiederherstellung der Blockreihenfolge nur für die erste Serie von fehlerfreien Blöcken eine Latenzreduktion. Ist die Spekulation für einen Block fehlgeschlagen, werden auch Blöcke mit Spekulationserfolg um die Verarbeitungszeit des Blockdekodierers verzögert. Durch eine Steigerung der Ausgabedatenrate ließe sich hier analog zur *blinden* Verarbeitung während der Signalfelddekodierung dieser Zusatzbeitrag für die folgenden Blöcke wieder sukzessive verringern.

7.4 Basisbandoptimierung zur Vermeidung zusätzlicher Speicher

Die bisherigen Erläuterungen zur Struktur für die spekulative Verarbeitung in den Abschnitten 7.2.3 und 7.3.2 bezogen sich stets auf die einzelnen Module. So wurden für die Betrachtungen über die Notwendigkeit eines Ausgangsspeichers die Verarbeitungszeiten im spekulativen und im nichtspekulativen Pfad miteinander in Beziehung gesetzt. Ist die Verarbeitung im spekulativen Pfad schneller, ist ein Speicher am Modulausgang erforderlich. Im Gesamtsystem entsteht durch diesen Speicher eine zusätzliche Latenz. Diese Systemstruktur ist stark modulorientiert.

Durch eine modulübergreifende Datenflusssteuerung kann die Latenzverkürzung durch spekulative Verfahren weiter optimiert werden. Das System nach Abb. 42, bestehend aus einem Demapper, Deinterleaver, Faltungsdekodierer und Blockdekodierer, verwendet die *spekulative Demodulation* für ein Gesamtmodul aus den ersten drei Einheiten. Der folgende Blockdekodierer kann mit *spekulativer Dekodierung* arbeiten, dies wird jedoch nicht vorausgesetzt. Die Verarbeitungszeit im Signalfeldzweig sei größer als die Verarbeitungszeit für die Datensymbole. Entsprechend Abschnitt 7.2.3 muss ein Speicher am Ausgang der spekulativen Verarbeitungsgruppe vorgesehen werden. Stattdessen kann jedoch der Blockdekodierer mit in den spekulativen Verarbeitungspfad aufgenommen werden, indem er durch die Möglichkeit eines Abbruchs der laufenden Verarbeitung erweitert wird. Durch die Verlängerung der Verarbeitungszeit wird nun möglicherweise die Bedingung (88) erfüllt und auf einen Speicher kann verzichtet werden.

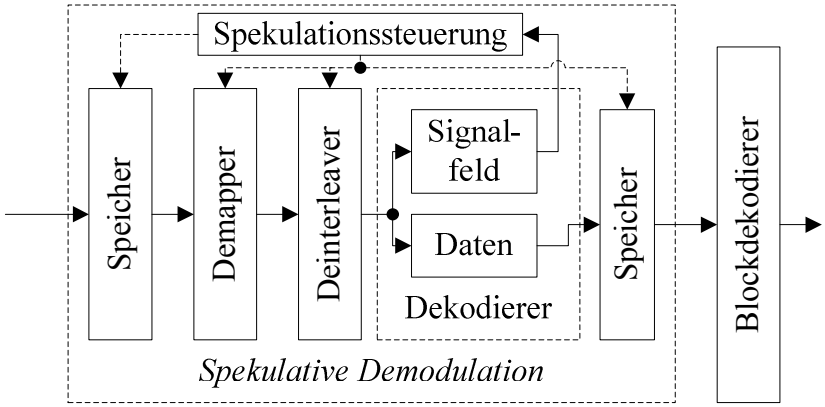

Abb. 42: Gesamtsystem mit spekulativer Demodulation

Ist die Bedingung (88) nicht erfüllt, arbeitet der Blockdekodierer aber mit dem Verfahren der *spekulativen Dekodierung*, kann dessen Ausgangspuffer als Speicher für die *spekulative Demodulation* verwendet werden. Dazu muss auch der nichtspekulative Pfad des Blockdekodierers mit an diesen Speicher angeschlossen werden.

Bei der Verwendung spekulativer Verfahren sollte nicht nur das einzelne betroffene Modul betrachtet werden, sondern die Gesamtheit der Basisbandverarbeitung, um den größtmöglichen Nutzen aus der Spekulation zu erreichen. Im folgenden Abschnitt 7.5 wird aufgeführt, wie durch das System eine geeignete Kombination von PHY- und MAC-Schicht noch weiter optimiert werden kann. Gleichzeitig wird analysiert, welche Änderungen spekulative Verfahren in der Basisbandverarbeitung für die MAC-Schicht nach sich ziehen.

7.5 Anforderungen an die MAC-Verarbeitung

7.5.1 Systemstruktur und Schnittstellen

Im Abschnitt 2.1 wurde die im Rahmen dieser Arbeit verwendete Gruppierung der Aufgaben eines Übertragungssystems in die beiden Schichten PHY und MAC eingeführt. Die PHY-Schicht umfasst dabei die Basisbandverarbeitung, also insbesondere die Kanalschätzung und Kanalkorrektur, die Kanalkodierung und -dekodierung sowie die digitale Modulation und Demodulation. Die MAC-Schicht übernimmt die Steuerung der Datenübertragung durch Paketisierung des Datenstroms, Adressierung des Empfängers der einzelnen Pakete sowie die Medienzugriffssteuerung, auf der insbesondere in drahtlosen Systemen ein hoher Schwerpunkt liegt. Es wurde bereits angesprochen, dass für den größtmöglichen Nutzen aus der Anwendung der beiden vorgestellten spekulativen Verfahren Änderungen gegenüber der üblichen Implementierung von PHY und MAC notwendig sind. Diese werden in diesem und dem folgenden Abschnitt erläutert. Da die beiden vorgestellten spekulativen Verfahren zur Latenzverringerung im Empfänger angewendet werden, beschränken sich die Erläuterungen zur Systemstruktur und den Schnittstellen zwischen PHY und MAC auf den Empfangsteil des gesamten Kommunikationssystems.

Die beiden Gruppen PHY und MAC stellen nicht nur eine logische Trennung in der Systemarchitektur dar, sondern sind in üblichen Kommunikationssystemen auch auf unterschiedlichen physikalischen Modulen untergebracht. So ist der PHY meist ein eigenständiger integrierter Schaltkreis, während der MAC als Softwarelösung auf einem angeschlossenen Mikroprozessor realisiert wird. Auch im Zuge einer fortschreitenden Integration hin zu einem Ein-Chip-System lässt sich diese Trennung feststellen. Wie in Abb. 43 dargestellt, ist das Interface zwischen PHY und MAC dabei datenstrombasiert, d. h. der PHY gibt die empfangenen und verarbeiteten Daten als seriellen oder parallelen Datenstrom aus. Dies entspricht der Verarbeitung innerhalb des PHYs. Der kontinierliche Eingangsdatenstrom vom analogen Empfangsteil wird fortlaufend verarbeitet. Auch bei der Verwendung von Blockkodierungen lässt sich das übergeordnete Verarbeitungsprinzip durch die sequentielle Folge von Blöcken als datenstromorientiert bezeichnen.

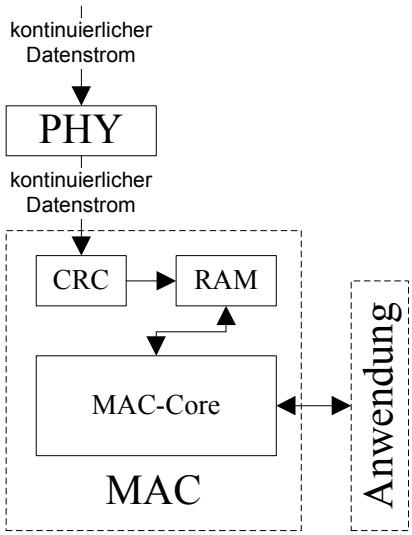

Abb. 43: Anbindung PHY und MAC

Der MAC dagegen arbeitet in paketbasierten Übertragungssystemen speicherbasiert, d. h. mit Puffern, die ein oder mehrere Datenpakete beinhalten. Entsprechend der im Abschnitt 2.1 eingeführten Aufgaben des MACs, also insbesondere der Adressierung und der Medienzugriffssteuerung, verwendet der empfangsseitige MAC die vom sendeseitigen MAC den Datenpaketen hinzugefügten MAC-Header-Informationen. Die eigentlichen Daten werden nicht verarbeitet, sondern nur im Puffer gespeichert und den höheren (Anwendungs-) Schichten des Kommunikationssystems im Speicher bereitgestellt oder über eine weitere Schnittstelle an diese weitergegeben. Wird das System nur als Brücke zwischen verschiedenen Übertragungskanälen, wie einem drahtgebundenen und einem drahtlosen Netzwerk, verwendet, kann der MAC die empfangenen Daten auch wieder direkt an einen (anderen) PHY ausgeben.

Für die Nichtverarbeitung des Datenbereichs eines Pakets im MAC gibt es eine Ausnahme: Der MAC ist eine Unterebene der Sicherungsschicht entsprechend des OSI-Modells und soll somit die fehlerfreie Datenübertragung sicherstellen. Dafür wird auf MAC-Ebene eine Prüf-

summe über den Datenbereich des Pakets oder das gesamte Paket inklusive MAC-Header verwendet. Für die empfangenen Daten muss also eine einmalige Prüfsummenberechnung erfolgen. Bei der Schnittstellenarchitektur nach Abb. 43 kann dies direkt im einlaufenden Datenstrom erfolgen, während die Daten in den Puffer übertragen werden. Sofern am Ende des Einschreibens die Prüfsumme einen Fehler anzeigt, wird der Pufferinhalt gelöscht und der Puffer für das nächste Datenpaket verwendet.

In den vorhergehenden Abschnitten zu notwendigen Änderungen in der PHY-Struktur für die spekulative Verarbeitung wurde bereits erläutert, dass abhängig von den auf die spekulative Verarbeitung folgenden Schritten und deren Verarbeitungsgeschwindigkeit ein Speicher vorgesehen werden muss, damit bei einem Misserfolg der Spekulationsannahme die Daten vor der datenstrombasierten Ausgabe an den MAC gelöscht werden können. Durch diesen Speicher verkleinert sich jedoch der Gewinn in Bezug auf die Latenzverringerung. Zwar bietet die spekulative Verarbeitung damit immer noch einen Vorteil gegenüber der rein nichtspekulativen, der Nutzen ist jedoch verringert.

Um die maximale Leistungsfähigkeit der spekulativen Latenzverringerung zu erhalten, muss die zusätzliche Zwischenspeicherung vermieden werden. Dies kann erreicht werden, indem die Schnittstelle zwischen PHY und MAC nicht mehr rein datenstrombasiert arbeitet, sondern der PHY direkt auf den Speicher des MACs zugreifen kann. Einerseits kann dies über einen DMA[56]-ähnlichen Zugriff geschehen, andererseits kann ein geteilter Speicher vorgesehen werden. Der MAC teilt dem PHY stets den aktuell als Paketpuffer zu verwendenden Speicherbereich mit, während der PHY nach der Überprüfung der Spekulationsannahme den Puffer als gültig markiert oder anderenfalls den Inhalt mit den erneut verarbeiteten Daten überschreibt.

Dieser Vorgehensweise steht jedoch der notwendige Datenintegritätstest über die Prüfsummenberechnung auf MAC-Ebene entgegen. Damit diese in den PHY verlagert werden kann, muss dem PHY der Aufbau der Pakete auf MAC-Ebene bekannt sein. So besteht ein Frame auf PHY-Ebene nicht notwendigerweise aus einem einzigen MAC-Paket. Stattdessen kann ein MAC zur Durchsatzsteigerung mehrere Pakete innerhalb eines PHY-Frames aggregieren. Jedes der Pakete wird auf MAC-Ebene mit einer eigenen Prüfsumme abgesichert, sodass im Fehlerfall nicht der gesamte PHY-Frame, sondern nur das oder die falschen Pakete wiederholt werden müssen.

Die Paketlängeninformation ist jeweils im MAC-Header enthalten. Dieser Header kann auch direkt auf PHY-Ebene gelesen und ausgewertet werden. Somit kann die Prüfung des jeweiligen Pakets auch bereits im PHY erfolgen. Eine Verlagerung der Paketintegritätsprüfung in den PHY bietet weiterhin den Vorteil, dass der MAC-Header bereits im PHY separiert werden kann. Während der Datenbereich des Pakets in den dafür vorgesehenen Puffer geschrie-

[56] DMA: Direct Memory Access

ben wird, ist es möglich, die MAC-Header-Informationen dem MAC in einem anderen Speicherbereich oder über eine andere Schnittstelle zur Verfügung zu stellen. Nur diese werden vom MAC für die weitere Verarbeitung benötigt. In Abb. 44 ist die resultierende Struktur von PHY und MAC dargestellt. Die notwendigen Informationen über den Aufbau eines MAC-Paketes sollten bei einer PHY-Implementierung nicht hartkodiert, sondern konfigurierbar ausgelegt werden. Ansonsten ziehen Änderungen im MAC-Paketaufbau auch Änderungen im PHY-Design nach sich.

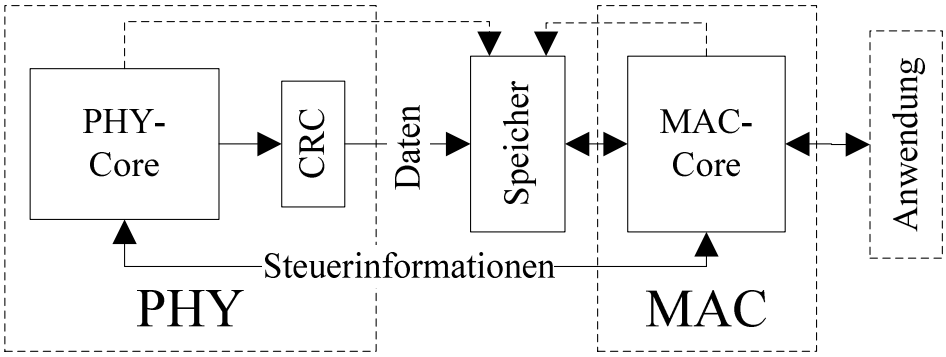

Abb. 44: Schnittstelle PHY-MAC mit geteiltem Speicher

Bei der Anwendung der *spekulativen Dekodierung* liegt der Gedanke nahe, die zusätzlich eingefügte Prüfsumme der einzelnen Blöcke zum Test auf Fehlerfreiheit zu verwenden und auf eine Prüfsummenberechung auf MAC-Ebene zu verzichten. Prinzipiell ist eine solche Vorgehensweise möglich, es muss aber beachtet werden, dass die Blockkodierung im Allgemeinen unabhängig von den Paketgrenzen über den gesamten zu übertragenden Frame erfolgt. Ist ein einziger Block des gesamten Frames nicht fehlerfrei dekodierbar, kann dies bedeuten, dass zwei Pakete erneut übertragen werden müssen, wenn die Paketgrenze genau innerhalb des fehlerhaften Blockes lag.

Eine Ausrichtung der Blöcke an den Paketgrenzen dagegen könnte insbesondere bei einer Aggregation von vielen kleinen Paketen zu einem Leistungseinbruch führen, da ein hoher Overhead zum Auffüllen der Kodierungsblöcke entsteht. Zur Vermeidung dieses Overheads wäre eine Verkürzung der Kodierung des letztens Blocks eines Pakets möglich, d. h. der Block wird vor der Kodierung mit Nullen aufgefüllt, die zusätzlichen Nullen werden jedoch nicht mit übertragen, sondern erst am Empfänger wieder eingeführt. Nun entsteht jedoch ein höherer Signalisierungsaufwand auf PHY-Ebene, da die jeweilige Paketlänge abhängig von der gewählten Kodierung nicht mehr aus dem MAC-Header extrahiert werden kann, sondern bereits vor der Blockdekodierung bekannt sein muss.

Zusammenfassend lässt sich feststellen, dass zur Anwendung spekulativer Verfahren Aufgaben des MACs durch den PHY übernommen werden sollten. Dies entspricht der auch in anderen Bereichen vorgenommenen Vereinheitlichung und Zusammenführung von PHY und MAC. So ermöglicht das vorgestellte EASY-A VHR-E-System eine Adressierung auf PHY-

Ebene. Nicht für die Empfangsstation bestimmte Frames werden direkt in der Basisbandverarbeitung nach der Signalfeldauswertung aussortiert und nicht weiter verarbeitet, identisch zu einem Frame mit einem fehlerhaften Signalfeld. Auch die Verlagerung der ACK-Signalisierung aus dem MAC-Header in das Signalfeld als Maßnahme zur Latenzverringerung wurde bereits im Kapitel 5 angeführt.

7.5.2 Medienzugriffssteuerung

Die Steuerung des Kanal- bzw. Medienzugriffs ist entsprechend des OSI-Modells (siehe Abschnitt 2.1) eine grundlegende Aufgabe der MAC-Schicht. Es existieren unterschiedliche Zugriffsprotokolle, z. B. die bereits eingeführten Verfahren CSMA/CD und CSMA/CA. Diese werden als wettbewerbsbasierte Protokolle[57] bezeichnet. Bei der Verwendung eines solchen Protokolls gibt es im Netzwerk keine ausgezeichnete Station, die die zentrale Steuerung des Zugriffs übernimmt. Damit können durch den gleichzeitigen Zugriff verschiedener Stationen auf den Kanal Kollisionen auftreten. Bei CSMA/CA wird versucht, diese durch das auf zufälligen Wartezeiten basierende Zugriffsverfahren weitestgehend zu verhindern, sie können jedoch nicht ausgeschlossen werden.

In wettbewerbsfreien Zugriffsprotokollen[58] werden durch eine geeignete Steuerung gleichzeitige Kanalzugriffe generell vermieden. Dies ist z. B. mit einer zentralen Koordinationsstation und einem TDMA[59]-basierten Verfahren möglich. Die zentrale Station teilt jedem Teilnehmer einen bestimmten Zeitschlitz zu, in dem dieser exklusiven Zugriff auf den Kanal erhält. Die aufeinanderfolgenden Zeitschlitze aller Stationen werden zu einem Block zusammengefasst, welcher durch ein spezielles Steuerungspaket durch die zentrale Station begrenzt wird. Anhand dieses Steuerungspakets können sich die einzelnen Stationen auf die vorgegebene Zeitperiode synchronisieren.

Die in den MAC-Protokollen festgelegten Zeiten für den Abstand einzelner Frames sind mit den Latenzzeiten der Basisbandverarbeitung verknüpft. So ist bei WLAN nach dem IEEE 802.11 Standard [45] mit SIFS[60] die Zeit zwischen dem Ende eines eingehenden Frames und dem Sendestart des ACK-Frames durch den Empfänger definiert. Die SIFS-Periode beträgt dabei in der Unterversion 802.11b 10 µs. Ist die Verarbeitungszeit des empfangenen Frames größer, kann die Bestätigung nicht mehr in der definierten Zeitspanne gesendet werden. Beim Entwurf und der Implementierung einer Basisbandverarbeitung für eine bestehende MAC-Spezifikation müssen die maximalen Verzögerungszeiten berücksichtigt werden. Umgekehrt müssen auch beim Entwurf eines MAC-Protokolls für eine bestehende Basisbandimplemen-

[57] contention-based protocols
[58] contention-free protocols
[59] TDMA: Time Division Multiple Access
[60] SIFS: Short Interframe Spacing

tierung die resultierenden Verzögerungszeiten in Betracht gezogen werden. Z. B. ist die SIFS-Periode abhängig von der maximalen Latenz der Basisbandverarbeitung.

Durch die Anwendung spekulativer Verfahren in der Basisbandverarbeitung ergeben sich teilweise stark unterschiedliche Verzögerungszeiten, je nachdem, ob die Spekulation erfolgreich oder nicht erfolgreich war. Wie im Abschnitt 7.3.4 präsentiert, verkürzt sich durch eine erfolgreiche Spekulation die Latenzzeit des RS-Blockdekodierers aus dem EASY-A-System um 87,5 %. Die Latenzzeit des Basisbandes kann also nicht mehr als eine feste Größe in das Design des MAC-Zugriffsprotokolls eingehen. Die MAC-Spezifikation muss flexibel auf die unterschiedlichen Verzögerungszeiten reagieren, um die maximale Durchsatzsteigerung durch die Spekulation zu erreichen.

Es sei als Beispiel eine einfache wettbewerbsbasierte Zugriffssteuerung nach dem CSMA/CA Prinzip angenommen. Im Anschluss an einen Frame muss eine feste Wartezeit T_{ifs} eingehalten werden. Danach steht eine Zeitspanne T_{ack} exklusiv für das Senden von Kontrollinformationen, wie einer Empfangsbestätigung, zur Verfügung. Ohne Einschränkung der Allgemeinheit sei angenommen, dass für jedes empfangene Datenpaket eine unmittelbare Empfangsbestätigung zu senden ist. (Auch bei einem Block-Acknowledgement, also des Sendens eines einzigen Bestätigungsframes für eine größere Anzahl von empfangenen Frames, muss die Zeit T_{ifs} zwischen den einzelnen Frames eingehalten werden.)

Das Basisband verwendet eine spekulative Blockdekodierung. Die einzelnen Daten- und ACK-Frames bestehen jeweils aus einem einzigen Block. Wird wie im genannten Standard verlangt, dass eine Bestätigung direkt nach dem Ablauf der Zeit T_{ifs} gesendet wird, ergibt sich diese Zeit aus der Basisbandlatenz bei einem Spekulationsmisserfolg zuzüglich der notwendigen Verarbeitungszeit auf der MAC-Ebene. Bei einem Spekulationserfolg mit daraus resultierender Verkürzung der Verarbeitungslatenz ergibt sich keine Durchsatzsteigerung, da die Zeit T_{ifs} vollständig abgelaufen sein muss, bevor der Bestätigungsframe gesendet wird.

Durch eine einfache Änderung kann der Datendurchsatz bei einem Spekulationserfolg gesteigert werden. Dazu wird der auf den Frame und die Zeit T_{ifs} folgende Zeitschlitz mit der Dauer T_{ack} immer noch exklusiv für die Framebestätigung reserviert. Die Bestätigung muss nun jedoch nicht direkt mit Ablauf der Zeit T_{ifs} gesendet werden, sondern kann zu einem beliebigen Zeitpunkt innerhalb der Periode T_{ack} erfolgen. Damit definiert sich T_{ifs} nun aus der kürzestmöglichen Latenz, also der Verzögerung bei einem Spekulationserfolg. In diesem Fall erfolgt die Bestätigung direkt nach der kurzen Wartezeit, und die nächste Datenübertragung kann früher beginnen. Bei einem Spekulationsmisserfolg ändert sich nichts gegenüber dem initialen Zustand. Die Summe aus T_{ifs} und T_{ack} muss mindestens der Latenz bei einem Spekulationsmisserfolg entsprechen.

Spekulative Verfahren in der Basisbandverarbeitung ermöglichen durch die Verkürzung der Latenzzeit beim Spekulationserfolg eine Durchsatzsteigerung. Damit diese auch den tatsächlichen Gesamtdurchsatz des Systems steigert und nicht nur den theoretisch möglichen Durchsatz der Basisbandverbindung, müssen die verwendeten MAC-Protokolle mit den variablen Latenzzeiten für Spekulationserfolge und -misserfolge flexibel umgehen können.

7.6 Spekulation zur Reduktion des Energieverbrauchs

In den vorherigen Abschnitten wurden die Verfahren *spekulative Demodulation* und *spekulative Dekodierung* unter dem Gesichtspunkt einer Latenzreduktion, hauptsächlich zur Steigerung des Datendurchsatzes, analysiert. Bei der Darstellung der Auswirkungen von Latenzen im Kapitel 3 konnte festgestellt werden, dass mithilfe einer Latenzreduktion unter Umständen auch der Energieverbrauch eines Systems optimiert werden kann. In diesem Abschnitt wird untersucht, inwieweit sich die vorgestellten spekulativen Verfahren für eine Verringerung des Energieverbrauchs eignen.

Die spekulativen Verfahren sind dadurch gekennzeichnet, dass abhängig vom Spekulationserfolg bzw. -misserfolg unterschiedliche Vorgänge innerhalb des spekulativen Verarbeitungsmoduls erfolgen. So muss bei der *spekulativen Demodulation* bei einem Spekulationsmisserfolg eine erneute Demodulation der Daten erfolgen. Bei der *spekulativen Dekodierung* wird die Dekodierung dagegen nicht erneut, sondern nur im Falle eines Spekulationsmisserfolges durchgeführt. Aus diesen Vorgehensweisen ergeben sich verschiedene Verzögerungszeiten bei einer erfolgreichen und einer fehlgeschlagenen Spekulation. Die Latenzreduktion bei einer erfolgreichen Spekulation ist eine gemeinsame Eigenschaft beider Verfahren. Im Hinblick auf die Reduktion des Energieverbrauchs unterschieden sie sich jedoch signifikant. Wird vereinfachend vorausgesetzt, dass sich der Energieverbrauch aus der für die Durchführung eines bestimmten Algorithmus benötigten Energiemenge ergibt, ermöglicht die *spekulative Demodulation* im Allgemeinen keine Energieverbrauchsreduktion. Stattdessen resultiert sie insgesamt in einer Erhöhung des gesamten durchschnittlichen Energieverbrauchs, sofern auch Spekulationsmisserfolge auftreten. Bei einem solchen Misserfolg ist die Verarbeitung der vorverarbeiteten Datensymbole erneut durchzuführen, sodass noch einmal die gleiche Energiemenge benötigt wird. Auch bei einem Spekulationserfolg ergibt sich im Allgemeinen keine Reduktion des Energieverbrauchs, da die gleiche Verarbeitung durchgeführt wird. Zur Latenzverringerung wird die Wartezeit zwischen der Signalfeldverarbeitung und dem Beginn der Datensymbolverarbeitung vermieden.

Im Gegensatz zur soeben erfolgten Darstellung wurde im Abschnitt 3.4 behauptet, dass sich durch die Verringerung der Verarbeitungszeit eines Moduls der Energieverbrauch desselben reduzieren lässt. Dies ist jedoch kein Widerspruch, wie sich bei einer genauen Betrachtung feststellen lässt. Die spekulative Demodulation verringert nämlich nicht die Latenzzeit für das

Datenverarbeitungsmodul, sondern beeinflusst das Zusammenspiel mehrerer Module durch die Behebung von Wartezeiten aufgrund von Parameterabhängigkeiten. Bei einem Spekulationserfolg könnte das Datenverarbeitungsmodul zwar direkt nach der früher fertig gestellten Datenverarbeitung in einen Energiesparmodus geschaltet werden. Zur Energieeinsparung wäre es aber auch möglich, auf Spekulation zu verzichten und das Datenverarbeitungsmodul während der Wartezeit auf die Signalfeldauswertung in den Energiesparmodus zu schalten.

Im Unterschied zur *spekulativen Demodulation* muss bei der *spekulativen Dekodierung* keine erneute Verarbeitung erfolgen. Sie ist dadurch gekennzeichnet, dass bei einem Spekulationserfolg überhaupt keine Dekodierung der Daten notwendig ist. Eine Datendekodierung wird nur bei einem Spekulationsmisserfolg durchgeführt, die für die Datendekodierung notwendige Energiemenge wird nur bei einer fehlgeschlagenen Spekulation verbraucht. Somit eignet sich die *spekulative Dekodierung* generell für eine Energieverbrauchsreduktion, unabhängig davon, ob sich durch die gewählte Systemstruktur und die Eigenschaften der MAC-Schicht insgesamt eine Latenzreduktion des Gesamtsystems ergibt. Hierfür muss vorausgesetzt werden, dass der Energiebedarf für die bei einer *spekulativen Dekodierung* zusätzlich notwendigen Strukturen sowie für den Test der Spekulationshypothese kleiner als der Energiebedarf des Dekodierungsmoduls an sich ist. Gleichzeitig müssen entsprechend der Relation zwischen Energieverbrauch des Dekodierungsmoduls und der zusätzlichen Strukturen ausreichend viele Spekulationserfolge auftreten, da sonst der für die Spekulation zusätzlich notwendige Energiebedarf den gesamten Energieverbrauch erhöht. Es ist weiterhin zu beachten, dass nicht das gesamte Dekodierungsmodul während der Spekulationsphase abgeschaltet werden kann, stattdessen muss ein paralleles Einschreiben ermöglicht werden. Sofern jedoch nur der Energieverbrauch reduziert werden soll und hierfür auch eine höhere Latenz toleriert werden kann, wäre es möglich, auf das parallele Einschreiben vollständig zu verzichten und vor Ausführung der Dekodierung zu überprüfen, ob die Daten dekodiert werden müssen. Damit wird die *spekulative Dekodierung* in eine nichtspekulative datenabhängige Verarbeitung überführt.

Zusammenfassend lässt sich feststellen, dass mithilfe der *spekulativen Dekodierung* neben der Latenz- auch eine Energieverbrauchsreduktion erreicht werden kann. Daher sollte die Anwendung dieses Verarbeitungsprinzips bei einem Systementwurf auch dann erwogen werden, wenn keine Latenzreduktion, sondern nur eine Energieverbrauchsoptimierung notwendig ist. In der im folgenden Kapitel vorgestellten Entwurfsmethodik für latenzarme Systeme wurde die Anwendung der *spekulativen Dekodierung* entsprechend eingearbeitet.

8. Methodik des latenzarmen Systementwurfs

Bereits zum Ende des Kapitels 5 wurden vier wichtige Punkte für die Vermeidung von Latenzen beim Entwurf eines Basisbandprozessors aufgeführt, unter anderem die Latenzvermeidung durch ein angepasstes Systemdesign und die Latenzverringerung durch Algorithmenoptimierung. Basierend auf den beiden genannten Punkten sowie den bereits in dieser Arbeit ausführlich vorgestellten Verfahren der Latenzreduktion durch Nutzung paralleler Strukturen und spekulativer Verarbeitung wird in diesem Kapitel eine allgemeine Methodik zum latenzarmen Entwurf eines digitalen Basisbandprozessors vorgestellt. Diese berücksichtigt dabei die Randbedingungen, die sich einerseits aus vorgegebenen Protokollen auf höheren Ebenen, andererseits aus den zur Verfügung stehenden (Hardware)-Ressourcen ergeben.

Den Einstiegspunkt des in Abb. 45 dargestellten Ablaufdiagramms eines Basisbandsystementwurfs stellt die Idee zur Entwicklung der Basisbandverarbeitung eines Kommunikationssystems dar. Mit der Konkretisierung der Idee startet ein iterativer Entwurfs- und Implementierungsprozess. Dabei besteht der globale Prozess aus mehreren einzelnen Teilprozessen, die wiederum iterativ sein können. Durch den iterativen Durchlauf können Ergebnisse eines (Teil-)Prozesses zur Anpassung der Anforderungen und Randbedingungen genutzt werden. Auf diese Weise lassen sich zunächst unklare Anforderungen genauer spezifizieren. Gleichzeitig können Anforderungen, die eine Umsetzung unmöglich machen oder stark erschweren, durch die Rückkopplung für einen erneuten Entwurf angepasst werden.

Zunächst wird ausgehend von der Idee unter Beachtung der allgemeinen Anforderungen, wie z. B. Datenrate und Datendurchsatz, Umgebungs- und Kanalbedingungen sowie evtl. notwendiger Standardkonformität, die Systemspezifikation erstellt. Diese kann bereits beim ersten Durchlauf Informationen über die geplante Hardwareumsetzung enthalten, dies ist jedoch nicht zwingend notwendig. Hardwareumsetzung bedeutet dabei nicht, dass die vorgestellte Methodik nur für die Konzeption von Basisbandverarbeitungssystemen geeignet ist, die vollständig hardwarebasiert in einem ASIC oder FPGA implementiert werden sollen. Stattdessen lässt sich die Methodik allgemein für beliebige software- und / oder hardwarebasierte Umsetzungen verwenden. Die Festlegung der Hardware entspricht dabei der Auswahl der Zielplattform sowie der Aufteilung der einzelnen Algorithmen in eine Software- oder Hardwareimplementierung. Da für die Basisbandverarbeitung im Bereich der hochratigen Drahtloskommunikationssysteme jedoch hauptsächlich nur hardwarebasierte Umsetzungen erfolgen, wurde die Darstellung der Methodik daraufhin angepasst.

Sofern sich aus der Systemspezifikation und den allgemeinen Anforderungen keine ausreichenden Informationen über die Hardware und deren Kenngrößen ableiten lassen, schließt sich nach der Systemspezifikation der blau gekennzeichnete Teilprozess der allgemeinen Sys-

temkonzeption und Simulation an. Als Ergebnis dieses Teilprozesses werden die Zielhardware ausgewählt und die entsprechenden Kenngrößen in der Spezifikation erfasst.

Abb. 45: Entwurfsmethodik für die latenzarme Basisbandverarbeitung

Während die Spezifikation die Systemparameter festlegt, ohne – mit Ausnahme der Konformität zu vorgegebenen Standards – eine bestimmte Systemarchitektur vorzugeben, wird im Rahmen der Systemkonzeption die Systemarchitektur bestimmt. Anhand der vorgegebenen Parameter werden geeignete Algorithmen für die Basisbandverarbeitung ausgewählt. Auch die Modularisierung des gesamten Systems erfolgt in der Konzeptionsphase. Die Systemkonzeptionsphase endet mit einer Simulation, anhand derer die Einhaltung der Spezifikation überprüft wird. Sofern die Simulation nicht erfolgreich ist, ist der Konzeptionsprozess zu wiederholen. Im Gegensatz zur allgemeinen Systemkonzeption werden bei der detaillierten Systemkonzeption die Zielplattform sowie deren Kenngrößen beachtet. So umfasst die detaillierte Systemkonzeption den rot gekennzeichneten Teilprozess der Parallelisierung. Sofern eine solche Parallelisierung notwendig ist, sollte die gesamte Datenverarbeitung an den parallelen Strukturen ausgerichtet werden. Ein Beispiel hierfür ist die bereits in den Kapiteln 2 und 5 eingeführte Einfügung von Pilotunterträgern, deren Abstand optimalerweise einem Vielfachen der parallel verarbeiteten Datenunterträgern entspricht. Auch das im Kapitel 6 vorgestellte implizite Interleaving sollte während der detaillierten Systemkonzeption evaluiert werden. Unabhängig von dem Erfordernis einer Latenzreduktion empfiehlt sich generell die Evaluation des impliziten Interleavings, da sich bei geeigneter Umsetzung eine Ressourceneinsparung bei gleichbleibender Systemperformance ergibt.

Neben der Vorstellung des impliziten Interleavings wurde im Kapitel 6 auch ausgeführt, dass eine bitparallele Verarbeitung der blockparallelen im Hinblick auf die entstehenden Verarbeitungszeiten vorzuziehen ist. Dabei ist jedoch zu beachten, dass blockparallele Strukturen einfach transparent ausgelegt werden können. Dies ist bei bitparallelen Strukturen nicht unbedingt gegeben. Abhängig von den vorgegebenen Anforderungen, speziell der Standardkonformität, ist hier eine geeignete Parallelisierungsart zu wählen.

Während des iterativen Durchlaufs der detaillierten Systemkonzeptions- und Simulationsphase können notwendige Algorithmenoptimierungen vorgenommen werden. Sofern sich nach mehreren Durchläufen herausstellt, dass die Spezifikation durch Änderungen an der Konzeption nicht oder nicht mit vertretbarem Aufwand erfüllt werden kann, ist die Spezifikation unter Beachtung der Ergebnisse anzupassen.

Der erfolgreichen Systemkonzeption und Simulation folgt die Implementierung auf der vorgegebenen Hardware. Die Einhaltung der Systemspezifikation wird durch eine Verifikation überprüft. Sofern das Ergebnis der Verifikation negativ ist, sind die Implementierung oder die Systemkonzeption und Implementierung zu wiederholen. Welche Schritte erneut zu durchlaufen sind, ergibt sich aus der Art der nicht erfüllten Parameter. Einige, wie z. B. die Verzögerung aufgrund einer beschränkten Verarbeitungsgeschwindigkeit, können durch rein implementierungsspezifische Aspekte variiert werden, während andere Parameter nur durch eine geänderte Systemkonzeption erfüllt werden können. Nach einer erfolgreichen Verifikation

werden die finalen Systemparameter extrahiert und im Datenblatt festgehalten. Das entworfene und implementierte System erfüllt die Anforderungen und kann eingesetzt werden.

Die soeben dargestellte grundlegende Methodik des Systementwurfs unter Beachtung der Kenngrößen der Zielplattform wurde in Abb. 45 um zwei zusätzliche Teilprozesse (grün und gelb gekennzeichnet) erweitert. Beide erfassen die Anwendung spekulativer Verfahren zur Systemoptimierung. Im grün gekennzeichneten Teilprozess wird bestimmt, ob sich der Energieverbrauch des Systems durch eine spekulative Blockdekodierung optimieren lässt. Daher kann dieser Prozess zum einen als Ergebnis einer fehlgeschlagenen Verifikation aufgerufen werden. Zum anderen ist es denkbar, diesen Prozess nach einer erfolgreichen Verifikation zu starten, um zusätzliche Energieeinsparungen zu erzielen. Unter Beachtung des für den Entwurfszyklus entstehenden Aufwands sollte stets auch nach einer erfolgreichen Verifikation überprüft werden, ob eine weitere Systemoptimierung durch eine erneute Iteration der Konzeptions- und Implementierungsphase sinnvoll ist.

Der gelb gekennzeichnete Teilprozess umfasst die Evaluation spekulativer Verfahren, also der *spekulativen Demodulation* und der *spekulativen Dekodierung*, im Hinblick auf eine Reduktion der Latenzzeit und eine damit einhergehende Durchsatzsteigerung des Gesamtsystems. Im Rahmen dieser Evaluation wird berücksichtigt, dass sich durch spekulative Verfahren nur dann eine Durchsatzsteigerung des Gesamtsystems ergibt, wenn die MAC-Schicht mit den entstehenden variablen Latenzen umgehen kann.

Zusammenfassend ist festzustellen, dass sich die Systemkonzeption einer Basisbandverarbeitung nicht losgelöst von der Systemimplementierung und den sich ergebenden Kenngrößen der Zielplattform durchführen lässt. Zur Gewährleistung der optimalen Systemleistung müssen alle Randbedingungen und Hardwarekenngrößen beachtet werden. Sofern diese beim initialen Start der Systemkonzeption noch nicht zur Verfügung stehen, entsteht ein iterativer Prozess, bei dem die Implementierungsergebnisse einen direkten Einfluss auf die geänderte Systemkonzeption nehmen. Durch die im Rahmen dieser Arbeit entworfenen spekulativen Verarbeitungsverfahren ergeben sich mit den drei neuen Teilprozessen weitere Freiheitsgrade innerhalb der Entwurfsmethodik. Diese neuen Teilprozesse sind zwar eine gute Ergänzung zur Optimierung des Systems, können jedoch nicht die sorgfältige detaillierte Systemkonzeption unter Beachtung der Hardwarekenngrößen ersetzen.

9. Fazit

Die zunehmende, teilweise mobile Nutzung digitaler Datendienste geht mit einem ständig steigenden Bedarf nach höheren Datenraten einher. Die notwendige Datenratensteigerung erfordert speziell im Bereich drahtloser Systeme mit Datenraten von mehreren Gbit/s sehr komplexe und aufwändige Algorithmen in der Basisbandverarbeitung. Aus der Umsetzung dieser Algorithmen ergeben sich große Verarbeitungslatenzen. Dies lässt sich in dem in Abb. 46 dargestellten Spannungsdreieck zwischen hoher Datenrate, niedriger Latenz und hoher Fehlersicherheit aufzeigen. Eine hohe Datenrate mit hoher Fehlersicherheit lässt sich im Allgemeinen nur durch Algorithmen mit einer höheren Latenz erreichen. Sollen dagegen gleichzeitig die Fehlersicherheit erhöht und die Latenz verringert werden, reduziert sich die erzielbare Datenrate.

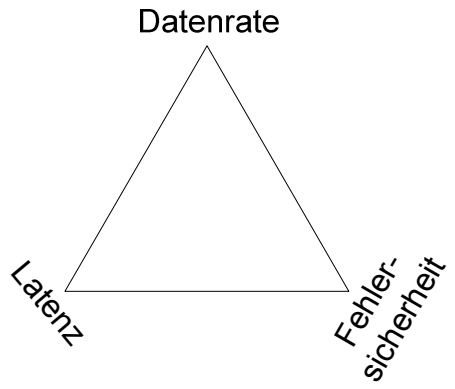

Abb. 46: Spannungsdreieck Datenrate, Latenz, Fehlersicherheit

Im Rahmen dieser Arbeit wurde ausgehend von einer Untersuchung der Latenzursachen innerhalb der Basisbandverarbeitung eine Kategorisierung vorgenommen. Es lassen sich grob zwei Gruppen unterscheiden, die entweder in einer aktiven Verarbeitungszeit oder in einer passiven Wartezeit resultieren. In der folgenden Diskussion der Latenzauswirkungen wurde aufgezeigt, dass Latenzen nicht nur Einfluss auf den Datendurchsatz innerhalb des Systems, sondern auch auf die Echtzeitfähigkeit, den Medienzugriff und den Energiebedarf haben.

Anhand der Latenzmodellierung des EASY-A VHR-E-Systems konnte festgestellt werden, dass sich ein Teil der signifikanten Latenzen der Basisbandverarbeitung nicht nur aus einer begrenzten Verarbeitungsgeschwindigkeit, sondern auch aufgrund von Parameter- und Datenabhängigkeiten während der Verarbeitung ergibt.

Im zweiten Teil dieser Arbeit wurden Verfahren zur Verringerung der Basisbandlatenzen vorgestellt. Ein besonderes Augenmerk galt hierbei zum einen der Anwendung paralleler Verarbeitungsstrukturen, zum anderen den neu entworfenen Verfahren der *spekulativen Demodulation* und der *spekulativen Dekodierung*.

Parallele Verarbeitungsstrukturen sind meist zur Steigerung des Durchsatzes bereits im System vorhanden, ihr Beitrag zu einer Latenzverringerung wurde jedoch in der Literatur noch nicht detailliert untersucht. Mithilfe der Unterscheidung zwischen bitparallelen und blockparallelen Verfahren wurde festgestellt, dass sich durch blockparallele Verfahren, wie dem im Beispielsystem verwendeten Streamkonzept, zusätzliche Latenzen ergeben. Bitparallele Verfahren ermöglichen demgegenüber eine Latenzverringerung durch die Nutzung des sich ergebenden impliziten Interleavings und dem daraus folgenden möglichen Verzicht auf einen dedizierten Interleaver.

Die vorgestellte *spekulative Demodulation* eignet sich zur Verringerung der Latenz aufgrund einer Parameterabhängigkeit zwischen Signalfeld und Datensymbol. Während im Allgemeinen die Verarbeitung der Datensymbole erst nach Abschluss der Signalfeldverarbeitung erfolgen kann, wird bei der *spekulativen Demodulation* eine Vermutung über die Übertragungsparameter der Datensymbole aufgestellt, um diese parallel zum Signalfeld verarbeiten zu können. Im Anschluss an die Signalfeldauswertung wird die aufgestellte Spekulationshypothese überprüft. Bei einer erfolgreichen Spekulation ergibt sich eine signifikante Verringerung der Verzögerungszeit bis zur Ausgabe der Datenwörter. Im vorgestellten OFDM-Basisbandprozessor des Beispielsystems beträgt die Reduktion 1,7 µs, dies entspricht 8 % der gesamten Empfängerlatenz.

Als weiteres spekulatives Verfahren zur Latenzverringerung wurde die *spekulative Dekodierung* eingeführt. Diese bietet sich zur Reduktion der bei einer Blockdekodierung auftretenden Latenzen an. Im Rahmen dieses Verfahrens wird auf eine fehlerfreie Datenübertragung spekuliert, die aufgestellte Hypothese wird anhand von (zusätzlichen) Prüfdaten ausgewertet. Zusätzliche Prüfdaten verringern die mögliche Nutzdatenrate, haben jedoch im Vergleich zu den Auswirkungen der Blockkodierung auf die Nutzdatenrate nur einen kleinen Anteil. Mit diesem Verfahren ergibt sich im Beispielsystem eine Latenzreduktion für den Beginn der Datenwortausgabe um 2,9 µs bzw. 13,4%.

Spekulative Verfahren zur Latenzverringerung sind dadurch gekennzeichnet, dass sich bei einem Spekulationserfolg die Latenz verringert, während bei einem Spekulationsmisserfolg die Latenz im Allgemeinen der Latenz eines identischen Systems ohne Spekulationsanwendung entspricht. Der zusätzliche Implementierungsaufwand ist größtenteils vernachlässigbar.

Im Rahmen dieser Arbeit wurden kurz die sich aus den variablen Latenzen der spekulativen Verfahren ergebenden Auswirkungen auf den MAC diskutiert. Gleichzeitig wurden Möglichkeiten aufgezeigt, wie durch eine Zusammenfassung der Aufgaben von PHY und MAC bzw. durch ein Verschieben von Aufgaben zwischen den beiden Schichten weitere Latenzreduktionen erreicht werden können. Mithilfe der Verschiebung des Verfahrens zur Sicherstellung einer korrekten Übertragung aus dem MAC in die PHY-Schicht können die Auswirkungen der zusätzlichen Prüfdaten einer *spekulativen Dekodierung* verringert werden.

Neben der Latenzverringerung eignet sich die *spekulative Dekodierung* auch zur Reduktion des Energieverbrauchs. Eine solche Energieverbrauchsreduktion ist genau dann gegeben, wenn das eigentliche Dekodierungsmodul während der Spekulationsphase in einen Energiesparmodus geschaltet wird und der gesamte zusätzliche Energiebedarf für die spekulative Verarbeitung kleiner als die durch den Dekodierungsverzicht entstehende mittlere Energieeinsparung ist. Folglich kann eine Energieeinsparung nur dann erfolgen, wenn auch Spekulationserfolge auftreten.

Mithilfe der vorgestellten Verfahren zur Latenzreduktion wurde, basierend auf der allgemeinen Vorgehensweise beim Entwurf und der Implementierung einer digitalen Basisbandverarbeitung, eine Methodik für ein latenzarmes Systemdesign entworfen. Der zugrunde liegende iterative Systementwurfs- und Implementierungsprozess wurde um zusätzliche Teilschritte und -prozesse für die Anwendung der latenzverringernden Verfahren erweitert. Gleichzeitig wurde die von der Latenzverringerung unabhängige Möglichkeit der Energieverbrauchsreduktion durch eine spekulative Dekodierung mit in diese Entwurfsmethodik aufgenommen.

Insgesamt lässt sich feststellen, dass bereits beim Systementwurf hochratiger Datenübertragungssysteme die sich ergebenden Latenzen der Basisbandverarbeitung beachtet werden müssen. Durch eine Berücksichtigung der Implementierungsbedingungen lassen sich einige Quellen zusätzlicher Verzögerungen vermeiden. Sofern spekulative Verfahren zur Latenzverringerung zum Einsatz kommen sollen, muss die MAC-Konzeption des Systems auf die resultierenden variablen Latenzen ausgelegt werden.

Abkürzungsverzeichnis

A/D-Wandler	Analog-Digital-Wandler
ACK	Acknowledge
ADSL	Asymmetric Digital Subscriber Line
AFE	Analoges Frontend
AGC	Automatic Gain Control
ARQ	Automatic Repeat Request
ASIC	Application-Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BCH-Code	Bose-Chaudhuri-Hocquenghem-Code
BEC	Backward Error Correction
BLER	Block Error Rate
BPSK	Binary Phase-Shift Keying
CRC	Cyclic Redundancy Check
CSMA / CA	Carrier Sense Multiple Access / Collision Avoidance
CSMA / CD	Carrier Sense Multiple Access / Collision Detection
D/A-Wandler	Digital-Analog-Wandler
DAB	Digital Audio Broadcasting
DC	Direct Current
DDR	Double Data Rate
DMA	Direct Memory Access
DMT	Discrete Multi-Tone
DRAM	Dynamic Random Access Memory
DS	Datensymbol
DSP	Digital Signal Processor
DV	Datenverarbeitung
DVB-T	Digital Video Broadcasting - Terrestrial
EDAC	Error Detection and Correction
EOF	End of Frame
FE	Funktionseinheit
FEC	Forward Error Correction
FER	Frame Error Rate
FIR	Finite Impulse Response
FFT	Fast-Fourier-Transform
FIFO	First-In First-Out
FPGA	Field-Programmable Gate Array
GPP	General Purpose Processor
HDMI	High-Definition Multimedia Interface
HW	Hardware
IC	Integrated Circuit
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
ISDN	Integrated Services Digital Network

ISI	Intersymbolinterferenz
ISO	International Organization for Standardization
LDPC-Code	Low-Density-Parity-Check-Code
LOS	Line-of-sight
LSB	Least Significant Bit
LUT	Look-up table
MAC	Medium Access Control
MSB	Most Significant Bit
NLOS	Non-line-of-sight
NOP	No Operation
OFDM	Orthogonal Frequency Division Multiplex
OSI	Open Systems Interconnetion
PLL	Phase-locked Loop
ppm	parts per million
QAM	Quadraturamplitudenmodulation
QPSK	Quadrature Phase-Shift Keying
RAM	Random Access Memory
ROM	Read-Only Memory
RS-Code	Reed-Solomon-Code
RSSI	Received Signal Strength Indicator
SDR	Single Data Rate oder auch Software-Defined Radio
SF	Signalfeld
SIFS	Short Interframe Spacing
SOF	Start of Frame
SRAM	Static Random Access Memory
TCP/IP	Transmission Control Protocol / Internet Protocol
TDMA	Time Division Multiple Access
UEP	Unequal Error Protection
UHR-C	Ultra-High-Rate - Cordless
USB	Universal Serial Bus
VGA	Video Graphics Array
VHR-E	Very-High-Rate - Extended Range
WLAN	Wireless Local Area Network

Symbolverzeichnis

a(n)	Zuordnungsfunktion des Interleavers / Deinterleavers
B_o, ..., B_n	Eingangsbitnummerierung, Blocknummerierung
D_{smin}	Minimale Speichertiefe des Ausgabepuffers
E_b/N_0	Verhältnis von Bitenergie zu Rauschleistung, normalisierter Signal-Rausch-Abstand
F_0, ..., F_n	Funktionseinheitennummerierung
f()	Allgemeine Kennzeichnung einer Funktion mit Abhängigkeiten
f_{clk}	Taktfrequenz
$f_{sof,m}$	Funktion für die Startverzögerung der m-ten Funktionseinheit
$f_{sym,m}$	Funktion zur Bestimmung der Symbolperiode der m-ten Funktionseinheit
N_{block}	Blockgröße
N_{blout}	Blockgröße bei der Ausgabe aus blockverarbeitendem Modul
N_c	Spaltenanzahl Blockinterleaver
N_{chkbit}	Anzahl an Prüfbits pro Datenblock, Größe einer Prüfsumme in Bits
N_{fft}	Anzahl FFT-Stützstellen
N_{mid}	Anzahl an gesendeten Midambeln
N_{nutz}	Anzahl an Nutzbits pro Datenblock
N_p	Parallelitätsfaktor
N_{pad}	Anzahl an Pad-Bytes im letzten OFDM-Symbol
N_{ps}	Größe eines Datenpakets
N_r	Zeilenanzahl Blockinterleaver
N_{res}	Anzahl Nutzdatenbytes im letzten RS-Block
N_{stream}	Anzahl tatsächlich verwendeter Streams
$N_{streammin}$	Anzahl minimal benötigter Streams
N_{sym}	OFDM-Symbolanzahl
$N_{sym,m}$	OFDM-Symbolanzahl der m-ten Funktionseinheit
n_{sym}	Zählvariable für OFDM-Symbole
$n_{sym,m}$	Zählvariable für OFDM-Symbole der m-ten Funktionseinheit
$N_{symfifo}$	Anzahl der in Frontstage-Ausgabe-FIFO gespeicherten Symbole
N_{takt}	Anzahl an benötigten Takten
N_{th}	Schwellwert für die Änderung der variablen Symboldauer
$q_{sym,m}$	Verhältnis von Ein- und Ausgangssymbolanzahlen für die m-te Funktionseinheit
$q_{wait,m}$	Anzahl an Eingangswarteperioden für die m-te Funktionseinheit
R, R_{neu}	Datenraten allg.
R_{in}, R_{out}	Datenraten am Ein- und Ausgang einer FIFO
R_{spec}, R_{nospec}	Datenraten mit und ohne spekulative Dekodierung
R_{fe}	Datendurchsatz einer Funktioneinheit bei Blockparallelität
R_{max}	Maximaler Gesamtdurchsatz eines blockparallelen Moduls
T_{ack}	Exklusive Zeitspanne für das Senden einer Empfangsbestätigung
T_b	Taktperiode sequentielles Schreiben
T_{B1}, ..., T_{Bn}	Vollständige Verarbeitungszeit des Blockes B_n
T_{B2w}	Wartezeit für die Ausgabe des 2. Blockes bei Blockdekodierung
T_{block}	Verarbeitungszeit Blockdekodierer

T_{by}	Zeit für die Berechnung der Pad-Bytes
T_{cdc270}	Jitter bei der Umsynchronisation zwischen zwei Taktdomänen
$T_{cdcfifo}$	Jitter bei der Taktumsynchronisation in einer asynchronen FIFO
$T_{chkspec}$	Zeit für das Testen der Spekulationshypothese
T_{crcchk}	Zeit für den Signalfeld-CRC-Test
T_{dd}	Verzögerungszeit für Demapping und Deinterleaving der Datensymbole
$T_{de}(n_{sym})$	Dekodiererverzögerung zwischen Eingabe des 1. und Ausgabe des n_{sym}-ten OFDM-Symbols
T_{de1}	Zeit für die Dekodierung des 1. Datenwortes des 1. OFDM-Symbols
T_{deo}	Zeit für die vollständige Dekodierung des 1. OFDM-Symbols
T_{dfs}	Verzögerung zwischen Framesynchronisation und Ausgabebeginn 1. OFDM-Symbol aus Frontstage
T_{do}	Verzögerungszeit für die Datenausgabe
T_{ds}	Verzögerungszeit für die Datensymbolverarbeitung
T_{dvs}	Allg. Verzögerungszeit für die Symboldekodierung
$T_{dw}(n_{sym})$	Verzögerung zwischen Framesynchronisation und Ausgabebeginn n_{sym}-tes Symbol aus Frontstage
T_{fail}, T_{fail2}	Gesamtverzögerung bei einem Spekulationsmisserfolg
T_{fd}	Framedauer bei der Ausgabe durch das Sender-Basisband
T_{fe}	Verzögerungszeit einer Funktioneinheit bei Blockparallelität
T_{fs}	Verzögerungszeit bis zum Frameausgabe am Sender
T_{ifs}	Zeitlicher Abstand zwischen zwei Frames
T_L, T_{Lneu}	Latenzzeit für die Paketübertragung
T_{last}	Ausgabedauer des letzten RS-Blockes bzw. Symbols
T_{lf}	Zeitliche Länge des Datenbereichs bei der Ausgabe zum MAC
T_{Lnpar}	Verzögerungszeit nichtparalleles blockverarbeitendes Modul
T_{Lpar}	Verzögerungszeit paralleles blockverarbeitendes Modul
T_{mid}	Midambeldauer
T_{Nth}	Symbolperiode für die Ausgabe des N_{th}+1 OFDM-Symbols
T_{min}	Minimale Zeit für die Ausgabe eines OFDM-Symbols aus Frontstage
T_{ns}, T_{ns2}	Zeit für die Berechnung der Symbolanzahl
T_{out}	Zeit für die Ausgabe eines terminierten Symbols über Dekodiererausgangsschnittstelle
$T_{outfifo}$	Zeit für die Ausgabe von $N_{symfifo}$-gespeicherten Symbolen
T_{ov}	Überlappungsbereich der einzelnen OFDM-Symbole
T_p, T_{pneu}	Paketübertragungszeit
T_{p125}, T_{p200}, T_{p270}	Taktperioden für 125 MHz, 200 MHz und 270 MHz
T_{pchk}	Zeit für den Gültigkeitstest der Parameter
T_{pd}	Zusatzverzögerung vor der Präambelausgabe
T_{plchk}	Zeit für den Signalfeld-Plausibilitätstest
T_{pp}	Zeit für die Präambelverarbeitung
T_{pre}	Präambeldauer
T_{pv}	Verzögerungszeit zwischen Startsignals und Parameterrückgabe
T_{ra}	Zeit für das Auslesen des Ausgabepuffers bei spekulativer Dekodierung
T_{rd}	Zeit für das Auslesen eines Datenblocks aus einem Blockdekodierer
T_{rs}	Verzögerungszeit des RS-Dekodierers
T_{rsb}	Zeit für die Berechnung der RS-Blockanzahl
T_{rso}	Zeit für die Bestimmung des Overheads durch die RS-Kodierung
T_s	Verarbeitungszeit eines Symbols

T_{sc}	Zeit für die Berechnung der notwendigen Streamanzahl
T_{sd}	Verzögerungszeit bis zur Bereitstellung eines OFDM-Symbols am Ausgabepuffer
T_{sfchk}	Zeit für den Signalfeld-Gültigkeitstest
T_{sfdd}	Verzögerungszeit für das Signalfeld-Demapping und -Deinterleaving
T_{sfde}	Zeit für die Signalfelddekodierung
T_{sffs}	Zeit für die Signalfeldverarbeitung im Frontstage
T_{sfv}	Verzögerungszeit für die Signalfeldverarbeitung
T_{sof}	Verzögerungszeit bis zur Ausgabe des ersten Datenwortes
$T_{sof,m}$	Verzögerungszeit bis zur Ausgabe des ersten Datenwortes für die m-te Funktionseinheit
$T_{sofacc,m}$	Akkumulierte Verzögerungszeit bis zur Ausgabe des ersten Datenwortes der m-ten Funktionseinheit
$T_{sofspec}$	T_{sof} unter Berücksichtigung der spekulativen Dekodierung
T_{sp}	Verzögerungszeit zwischen Startsignal und Präambelbereitstellung am Ausgabepuffer
T_{ss270}	Verzögerung für die Synchronisation des Startimpulses zwischen zwei Taktdomänen
$T_{streamov}$	Zusätzliche Verzögerungszeit durch den Streamoverhead
T_{succ}, T_{succ2}	Gesamtverzögerung bei einem Spekulationserfolg
T_{sw}	Zeit für die Summation der Datenwörter
T_{sym}	OFDM-Symbolperiode
$T_{sym,m}$	Symbolperiode für die m-te Funktionseinheit
$T_{symvar}(n_{sym})$	Variable OFDM-Symbolperiode
T_{sync}	Verzögerungszeit bis zum Abschluss der Framesynchronisation
T_{v}	Verzögerungszeit Blockinterleaver
T_{wa}	Einschreibezeit Ausgabepuffer bei spekulativer Dekodierung
T_{wait}	Zusätzliche Wartezeit für die Ausgabe eines Teilsymbols aus der Frontstage-Ausgabe-FIFO
$T_{wait,m}$	Zusätzliche Wartezeit für den Start der Ausgabe der m-ten Funktionseinheit
T_{wd}	Zeit für das Einschreiben eines Datenblocks in einen Blockdekodierer
T_{wrs}	Zusätzliche Wartezeit für das vollständige Einschreiben eines RS-Blocks
x(n)	Eingangsfolge Interleaver
y(n)	Ausgangsfolge Interleaver
z(n)	Ausgangsfolge Deinterleaver

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] H. Bruns, “Medienanstalt Berlin-Brandenburg gibt DAB keine Chance mehr,” Link: http://www.heise.de/newsticker/meldung/Medienanstalt-Berlin-Brandenburg-gibt-DAB-keine-Chance-mehr-117870.html, abgerufen am: 23.08.2011

[2] A. Börnsen, “Bericht zur Untersuchung der Digitalen Dividende,” 2009

[3] “Wachsender Datenverkehr,” Link: http://www.welt.de/wirtschaft/webwelt/article11790130/USA-errichten-das-Zwei-Klassen-Internet.html, abgerufen am: 22.07.2011

[4] Y. Kwon, Y. Fang und H. Latchman, “Performance analysis for a new medium access control protocol in wireless LANs,” in *Wireless Networks*, Bd. 10, Nr. 5, S. 519–529, 2004

[5] S. Abraham, A. Meylan und S. Nanda, “802.11n MAC design and system performance,” in *Proc. of 40th annual IEEE International Conference on Communications - ICC'05*, S. 2957-2961, 2005

[6] A. Acharya, A. Misra und S. Bansal, “High-performance architectures for IP-based multihop 802.11 networks,” in *IEEE Wireless Communications*, Bd. 10, Nr. 5, S. 22-28, 2003

[7] S. Trautmann, “Verfahren zur Reduzierung der Latenzzeit in Discrete-Multitone-Systemen,” Dissertation Fridericiana Karlsruhe, 2002

[8] D. H. Traeger und A. Volk, *LAN: Praxis lokaler Netze*, 4. Aufl., Stuttgart: Teubner, 2002

[9] C. E. Shannon, “The mathematical theory of communication,” *Bell System Technical Journal*, Bd. 27, S. 379-423 und 623-656, 1948

[10] M. Werner, *Nachrichtentechnik: Eine Einführung für alle Studiengänge*, 5. Aufl., Wiesbaden: Vieweg Verlag, 2006

[11] M. Werner, *Information und Codierung*, 2. Aufl., Wiesbaden: Vieweg+Teubner, 2008

[12] W. P. Kowalk und M. Burke, *Rechnernetze*, Stuttgart: Teubner, 1994

[13] C. Meinel und H. Sack, *Digitale Kommunikation: Vernetzung, Multimedia, Sicherheit*, Berlin, Heidelberg: Springer, 2009

[14] R. H. Morelos-Zaragoza, *The art of error correcting coding*, Chichester: John Wiley & Sons, Ltd., 2002

[15] B. Friedrichs, *Kanalcodierung*, Berlin, Heidelberg: Springer, 1996

[16] D. Conrads, *Telekommunikation: Grundlagen, Verfahren, Netze*, 5. Aufl., Wiesbaden: Vieweg Verlag, 2004

[17] S. P. Dandamudi, *Fundamentals of Computer Organization and Design*, New York: Springer, 2003

[18] W. C. Vermillion, *End-to-End DSL Architectures*, Indianapolis: Cisco Press, 2003

[19] M. Hein, *TCP/IP*, 6. Aufl., Bonn: mitp, 2002

[20] "FAQ Fastpath," Link: http://www.fastpath.de/fastpath/allgemeine-fragen-zu-fastpath/, abgerufen am: 20.07.2011

[21] A. Schemberg, *PC-Netzwerke*, 3. Aufl., Bonn: Galileo Press, 2006

[22] M. Werner, *Netze, Protokolle, Schnittstellen und Nachrichtenverkehr*, 1. Aufl., Wiesbaden: Vieweg Verlag, 2005

[23] M. G. Parker, K. G. Paterson und C. Tellambura, "Golay Complementary Sequences," in *Encyclopedia of Telecommunications*, J. G. Proakis, Hrsg., Hoboken: John Wiley & Sons, Inc., 2003

[24] R. L. Myers, *Display Interfaces*, 1. Aufl., Chichester: John Wiley & Sons, Ltd., 2002

[25] W. Schiffmann, H. Bähring und U. Hönig, *Technische Informatik 3*, Berlin, Heidelberg: Springer, 2011

[26] H.-P. Messmer und K. Dembowski, *PC-Hardwarebuch*, 7. Aufl., München, Boston: Addison-Wesley, 2003

[27] J. Siegl, *Schaltungstechnik*, 2. Aufl., Berlin, Heidelberg: Springer, 2005

[28] J. P. Alegre Pérez, S. C. Pueyo und B. C. López, *Automatic Gain Control*, New York: Springer, 2011

[29] J. G. Proakis und M. Salehi, *Digital Communications*, 5. Aufl., New York: McGraw-Hill, 2008

[30] H. Nuszkowski, *Digitale Signalübertragung im Mobilfunk*, 1. Aufl., Dresden: Vogt Verlag, 2010

[31] H.-Ju Kang und I.-Cheol Park, "FIR filter synthesis algorithms for minimizing the delay and the number of adders," in *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Bd. 48, Nr. 8, S. 770-777, 2001

[32] P. Kalivas, P. Bougas, A. Tsirikos, G. Economakos und K. Z. Pekmestzi, "New Systolic And Low Latency Parallel FIR Filter Schemes," in *International Journal of Applied Mathematics and Computer Sciences*, Bd. 2, Nr. 1, S. 322-325, 2006

[33] T. Schmid, O. Sekkat und M. B. Srivastava, "An experimental study of network performance impact of increased latency in software defined radios," in *Proc. of the the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization - WinTECH'07*, S. 59, 2007

[34] “Intel Xeon Processor X5690,” Link: http://ark.intel.com/products/52576/Intel-Xeon-Processor-X5690-(12M-Cache-3_46-GHz-6_40-GTs-Intel-QPI), abgerufen am: 20.07.2011

[35] W. Riggert, *Rechnernetze*, 3. Aufl., München: Fachbuchverlag Leipzig im C.-Hanser-Verlag, 2005

[36] D. Yohannes, *Fundamental Concepts on Wireless LAN & The IEEE 802.11 Protocol*, Nürnberg: dod.com nbg Printing Press, 2011

[37] S. Henzler, *Power management of digital circuits in deep sub-micron CMOS technologies*, Springer, 2007

[38] “EASY-A-Projekt,” Link: http://www.easy-a.de/, abgerufen am: 24.08.2011

[39] “WIGWAM-Projekt,” Link: http://www.wigwam-project.de, abgerufen am: 24.08.2011

[40] H. Xu, V. Kukshya und T. S. Rappaport, “Spatial and temporal characteristics of 60-GHz indoor channels,” in *IEEE Journal on Selected Areas in Communications*, Bd. 20, Nr. 3, S. 620-630, 2002

[41] S. Glisic et al., “Fully integrated 60 GHz transceiver in SiGe BiCMOS, RF modules, and 3.6 Gbit/s OFDM data transmission,” in *International Journal of Microwave and Wireless Technologies*, Bd. 3, Nr. 2, S. 139-145, 2011

[42] S. Glisic, K. Schmalz, F. Herzel, M. Elkhouly und J. C. Scheytt, “A fully integrated 60 GHz transmitter front-end in SiGe BiCMOS technology,” in *Proc. of IEEE 11th Topical Meeting on Silicon Monolithic Integrated Circuits in RF Systems - SiRF'11*, S. 149-152, 2011

[43] M. Piz, “EASY-A Baseband Specification (Physical Layer) v1.03,” internes Arbeitspapier, IHP GmbH, 2010

[44] M. Piz, E. Grass und M. Peter, “A simple OFDM physical layer for short-range high data rate transmission at 60 GHz,” in *Proc. of 11th International OFDM-workshop - InOWo'06*, S. 303–307, 2006

[45] IEEE Std 802.11-2007: *IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, New York: IEEE Computer Society, 2007

[46] IEEE Std 802.15.3c-2009: *IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 15.3: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPANs) - Amendment 2: Millimeter-wave-base Alternative Physical Layer Extension* , New York: IEEE Computer Society, 2009

[47] IEEE Std 802.16-2009: *IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Broadband Wireless Access Systems*, New York: IEEE Computer Society, 2009

[48] M. Petri, "Configurable, Modular and Scalable OFDM Baseband Processor for Data Rates up to 4 Gbps," in *Proc. of IASTED International Conference on Wireless Communications - WC'11*, S. 1-5, 2011

[49] M. Piz, "Wideband OFDM System for Indoor Communication at 60 GHz," Dissertation BTU Cottbus, 2010

[50] S. Churiwala und S. Garg, *Principles of VLSI RTL Design*, New York: Springer, 2011

[51] D. J. Kinniment und A. Yakovlev, "Low latency synchronization through speculation," in *Integrated Circuit and System Design*, E. Macii, V. Paliouras und O. Koufopavlou, Hrsg., Bd. 3254, S. 278-288, Springer, 2004

[52] S. Anikhindi und V. K. Jones, "Low memory and low latency cyclic prefix addition," U.S. Patent US 6687307 B1, Feb. 2004

[53] W. Stoye, "LDPC iteration control by partial parity check," in *Proc. of 2009 IEEE International Conference on Ultra-Wideband - ICUWB'09*, S. 602-605, 2009

[54] M. Ismail, I. Ahmed, J. Coon, S. Armour, T. Kocak und J. McGeehan, "Low latency low power bit flipping algorithms for LDPC decoding," in *Proc. of 21st IEEE International Symposium on Personal Indoor and Mobile Radio Communications - PIMRC'10*, S. 278–282, 2010

[55] M. Ehrig und M. Petri, "Diskussion ACK-Signalisierung," persönliche Gespräche 1./2. Quartal 2011

[56] H. Liebig, *Rechnerorganisation*, 3. Aufl., Berlin, Heidelberg: Springer, 2003

[57] IEEE P802.11ad/D3.0: *Draft Standard for Information Technology – Telecommunications and Information Exchange Between Systems – Local and Metropolitan Area Networks – Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amendment 4: Enhancements for Very High Throughput in the 60 GHz band*, New York: IEEE Computer Society, 2011

[58] H. Singh, X. Qin, H.-R. Shao, C. Ngo, C. Y. Kwon und S. S. Kim, "Support of Uncompressed Video Streaming Over 60GHz Wireless Networks," in *Proc. of 5th IEEE Consumer Communications and Networking Conference - CCNC'08*, S. 243-248, 2008

[59] TG3c, "IEEE 802.15.3c WPAN Task Group (TG3c) Contributions and documents," Link: http://www.ieee802.org/15/pub/TG3c_contributions.html, abgerufen am: 20.07.2011

[60] "Wiktionary," Link: http://de.wiktionary.org/wiki/Spekulation, abgerufen am: 29.08.2011

[61] M. E. Thomadakis, *The architecture of the Nehalem processor and Nehalem-EP smp platforms*, Technical Report, Texas A&M University, 2011

[62] M. Menge, “Sprungvorhersage in Fließbandprozessoren,” in *Informatik-Spektrum*, Bd. 21, Nr. 2, S. 73-79, 1998

[63] J. F. K. Lee und A. J. Smith, “Branch Prediction Strategies and Branch Target Buffer Design,” in *Computer*, Bd. 17, Nr. 1, S. 6-22, 1984

[64] U. Brinkschulte und T. Ungerer, *Mikrocontroller und Mikroprozessoren*, 3. Aufl., Berlin, Heidelberg: Springer, 2010