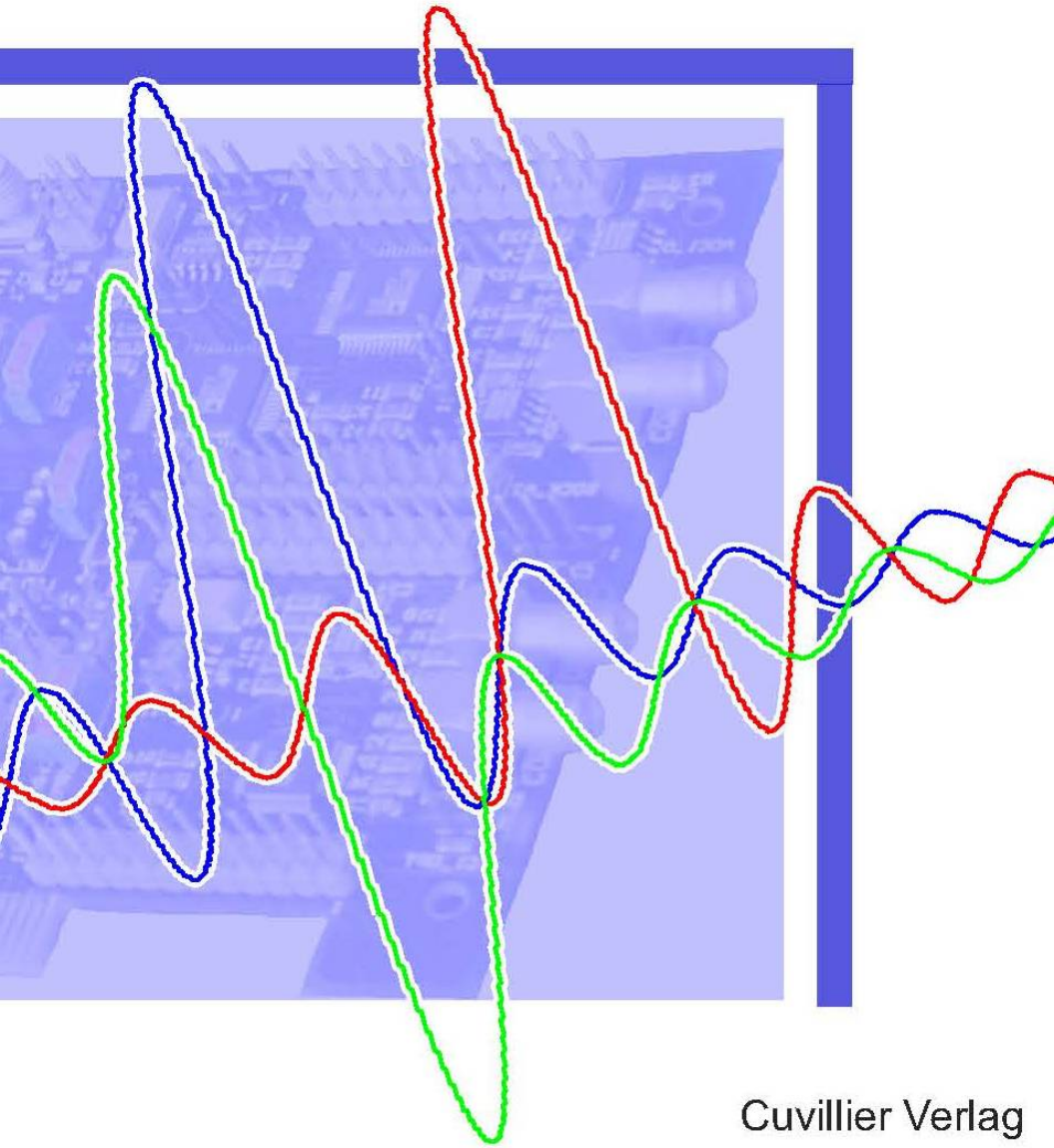
Marc Reinert

FPGA-basierte Realisierung eines OFDM-Funkübertragungssystems



Cuvillier Verlag

FPGA-basierte Realisierung eines OFDM-Funkübertragungssystems

Vom Promotionsausschuss der
Technischen Universität Hamburg-Harburg
zur Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation

von

Marc Reinert

aus

Leer (Ostfriesland)

2009

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.
1. Aufl. - Göttingen : Cuvillier, 2009
Zugl.: (TU) Hamburg-Harburg, Univ., Diss., 2009
978-3-86955-022-0

Vorsitzender:	Prof. Dr.-Ing. Wolfgang Krautschneider
1. Gutachter:	Prof. Dr. rer. nat. Dr. h. c. Herman Rohling
2. Gutachter:	Prof. Dr. Friedrich Mayer-Lindenberg

Tag der mündlichen Prüfung: 29. Mai 2009

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

1. Auflage, 2009
Gedruckt auf säurefreiem Papier

978-3-86955-022-0

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Nachrichtentechnik der Technischen Universität Hamburg-Harburg.

Für die Möglichkeit zur Promotion, die Betreuung meiner Arbeit, ein hervorragendes Arbeitsumfeld und das entgegengebrachte Vertrauen danke ich ganz herzlich Herrn Prof. Dr. Hermann Rohling. Für die Übernahme des Zweitgutachtens und das Interesse an meiner Arbeit gilt mein Dank Herrn Prof. Dr. Friedrich Mayer-Lindenberg. Weiterhin danke ich dem Vorsitzenden der Prüfungskommission, Herrn Prof. Dr.-Ing. Wolfgang Krautschneider.

Den Kolleginnen und Kollegen des Instituts für Nachrichtentechnik gilt mein Dank für die interessante und tolle Zusammenarbeit sowie die stets herzliche und freundschaftliche Atmosphäre.

Ich danke meinen Eltern und allen Freunden, die mich auf dem langen Weg zu dieser Arbeit begleitet haben und die während der Phase der Promotion sicher das eine oder andere Mal zu kurz gekommen sind. Schließlich danke ich den neuen Kolleginnen und Kollegen bei Airbus, die mit viel Verständnis und freudigem Interesse die Vollendung dieser Arbeit beobachtet haben.

Hamburg, im Juni 2009

Marc Reinert

Inhaltsverzeichnis

1 Einleitung

Die Entwicklung neuer und zukunftsweisender Produkte zur digitalen Signalverarbeitung ist eine wichtige Aufgabe in der Elektrotechnik. Viele unterschiedliche Standards zu Produktgruppen und Technologien versuchen, jederzeit die neuesten Möglichkeiten der Halbleiterbauelemente zugrunde zu legen und auf diese Weise immer leistungsfähigere Funktionseinheiten zu spezifizieren. Neue Ansätze im Bereich der digitalen Datenverarbeitung eröffnen viele Chancen auf neuartige Multimediaangebote und einen Entwicklungsschub bei der Kommunikations-Infrastruktur. Solche neuen Systeme müssen sich in bestehende oder ähnliche Technologien einfügen lassen. Die Umsetzung einer neuen Spezifikation unter Sicherstellung der geforderten Kompatibilitäten erfordert die Auswahl geeigneter Bauteile und eine den Vorgaben getreue detaillierte Entwicklung.

Ein wichtiges Produktfeld der Elektrotechnik mit hohen Ansprüchen an Hardware zur digitalen Signalverarbeitung ist die Nachrichtenübertragung. Bei der Planung eines digitalen Kommunikationssystems ist von wesentlicher Bedeutung, welche Übertragungstechnik eingesetzt werden soll. Ein Verfahren, das den hohen Ansprüchen moderner Multimediaanwendungen gerecht wird, ist *Orthogonal Frequency Division Multiplexing* (OFDM). Die zunehmende Verbreitung von OFDM in der kabelgebundenen und funkbasierten Datenübertragung resultiert daraus, dass es auch unter widrigen Ausbreitungsbedingungen eine sehr hohe Leistungsfähigkeit aufweist. Enorme Fortschritte in der Halbleitertechnologie machen es heutzutage möglich, die anspruchsvolle Datenverarbeitung von ODFM-Systemen effizient zu realisieren und so die Funkübertragung digitaler Daten immens zu verbessern. Neben der Robustheit bietet OFDM noch eine hohe Variabilität, was ein anwendungsbezogenes Einstellen bezüglich Datenrate, Fehlersicherheit, Zugriffsverfahren und Kanalparameter in optimaler Weise vorbereitet.

Ein OFDM-System wird gezielt für bestimmte zu erwartende Szenarien entworfen und eine Vielzahl von Parametern kann dahingehend vorab kalkuliert werden. Der Einfluss auf das Signal zwischen Sender und Empfänger wird dabei durch die Eigenschaften Funkkanals bestimmt. Ein Beispielszenario für die digitale Datenübertragung bilden drahtlose Netzwerke in Büroumgebungen (WLANs), bei denen hohe Datenraten wichtig sind. Eine direkte Sichtverbindung zwischen Sender und Empfänger ist nicht notwendig gegeben und es sind keine Bewegungen der Übertragungseinheiten mit mehr als Fußgängergeschwindigkeit zu erwarten. Vorschläge für

OFDM-Systeme dieser Art finden sich in den WLAN-Standards HiperLAN/2 und IEEE802.11a/g [HIP01][IEE99].

Nach der Festlegung auf eine geeignete Übertragungstechnik und der analytischen Bestimmung der wesentlichen Systemparameter ergibt sich eine Reihe von weiteren Entwurfsfragen, sobald eine Umsetzung in Hardware erfolgen soll. Zunächst ist die Auswahl geeigneter Bauelemente von entscheidender Bedeutung. Die in dieser Arbeit diskutierte Realisierung setzt dabei konsequent auf *Field Programmable Gate Arrays* (FPGAs) zur digitalen Signalverarbeitung. Die Technologie ist sehr leistungsfähig und bietet eine Vielzahl an Möglichkeiten für das Systemdesign. FPGA-Hardware ist rekonfigurierbar. Große Fortschritte aktueller und zukünftiger Generationen von FPGAs eröffnen neue Möglichkeiten, komplexe digitale Verarbeitungsketten in einem Chip abzubilden (*System on a Chip, SoC*) und bieten dabei ein Höchstmaß an Flexibilität.

FPGAs sind universelle Logik-Chips, die zunächst keine komplexen spezialisierten Verarbeitungseinheiten beinhalten. Sie beinhalten einen großen Vorrat an sehr kleinen identischen Grundzellen zum Schaltungsentwurf, die in einem zweidimensionalen Feld im Chip angeordnet sind. Die Funktionalität erhalten die Bauteile erst durch die Programmierung mit einer Schaltungsstruktur. Die zu entwickelnden Schaltungen können am PC z.B. grafisch als Schaltplan oder in einer für Schaltkreise geeigneten Programmiersprache, einer sogenannten Hardware Beschreibungssprache, eingegeben werden. Für große Designs wird häufig, wie auch in dieser Arbeit, die *VHSIC Hardware Description Language* (VHDL) eingesetzt. Die so beschriebenen Schaltungsdesigns sind nach geeigneter Übersetzung beliebig oft in FPGAs herunter ladbar, sodass Modifikationen und Designvarianten unmittelbar auf der Hardwareplattform erprobt werden können.

Der Schwerpunkt dieser Arbeit liegt auf Realisierungsaspekten, die sich bei der Implementierung eines WLAN-OFDM-Modems auf einer FPGA-Hardware ergeben. Kriterien wie Designgröße und Verarbeitungsgeschwindigkeiten sind dabei von zentraler Bedeutung. Diese Parameter sind wiederum abhängig von der Verarbeitungsgenauigkeit, der hardwareangepassten Implementierung und dem gewünschten Maß an Flexibilität.

Alle Basisband-Module des OFDM-Systems werden im Rahmen dieser Arbeit im FPGA realisiert. Diskutiert wird vor dem Kontext eines leistungsfähigen Gesamtsystems die Optimierung der einzelnen Komponenten. Ideal sind hardwareeffiziente Lösungen, die die theoretisch erzielbare Übertragungssicherheit oder die maximale Datenrate nicht bzw. nur unwesentlich beeinträchtigen. Bei vielen Parametern ist der Aufwand der Hardwarestruktur gegenüber dem Einfluss auf die Performance abzuwägen.

Die Designaspekte werden so weit wie möglich theoretisch beschrieben und anschließend durch Tests auf einer Experimentalhardware verifiziert. Durch Veränderungen der FPGA-Programme

können verschiedene Parameterkombinationen ausprobiert und hinsichtlich ihres Einflusses auf das Gesamtsystem untersucht werden.

Die Umsetzung eines theoretischen OFDM-Entwurfs in eine Hardwarerealisierung bringt zwangsläufig Ungenauigkeiten durch nicht ideale Verarbeitung mit sich. Diese Störeinflüsse werden in dieser Arbeit aufgezeigt, um sie in geeigneter Weise berücksichtigen und gegebenenfalls minimieren zu können. Wesentlich ist dabei die frühzeitige Abwägung, ob die Hardware den zu erwartenden Verarbeitungsaufwand bewältigen kann und ob gegebenenfalls eine aufwandsgünstige (suboptimale) Implementierung die Performance der Datenübertragung in inakzeptabler Weise reduzieren würde.

Dass sich ein OFDM-System mit Hilfe von FPGAs realisieren lässt, steht sicher außer Frage. Ziel dieser Arbeit ist es, darzustellen, wie gut dieser Hardwareansatz zu einer OFDM-Verarbeitung passt, was speziell bei der Realisierung zu beachten ist, wie die digitale Signalverarbeitung für FPGAs optimiert werden kann und wo das System zur Nachrichtenübertragung besonders von den programmierbaren Schaltungsstrukturen profitieren kann. Der aufgebaute OFDM-Demonstrator belegt durch Test- und Messergebnisse, dass sich die in dieser Arbeit diskutierten Techniken äußerst effizient realisieren lassen. Die Entwicklung des Experimentalsystems gibt eine Übersicht zu der erzielbaren Verarbeitungsperformance und erfasst die Leistungsfähigkeit der eingesetzten Hardwareressourcen.

2 OFDM-Übertragungstechnik

OFDM ist eine Technik zur Datenübertragung auf vielen nebeneinanderliegenden Trägerfrequenzen mit dem Ziel, selbst in stark gestörten Kanalsituationen noch hohe Datenraten zu ermöglichen. Im folgenden Kapitel wird die Bedeutung dieser Technologie von der Entstehung der Grundideen über die vielfältige Anwendung bis hin zu einem Ausblick auf zukünftige Chancen eingeordnet.

2.1 Historie und Grundlagen von OFDM

Die Mehrträger-Nachrichtenübertragung ist eine Möglichkeit zur Erweiterung der Übertragungskapazität digitaler Funkübertragungssysteme. Wenn die Kapazität einer einfachen Datenübertragungsstrecke nicht ausreicht, kann sie durch die Installation einer zweiten Strecke auf einer anderen Sendefrequenz relativ leicht verdoppelt werden. Somit ist ein einfacher Fall von Mehrträger-Übertragung realisiert. Die zweite Möglichkeit, die Kapazität der Einträger-Übertragung zu erhöhen, ist eine Verkürzung der Symboldauer. Sehr hohe Symbolraten sind jedoch in Kanälen mit langen Impulsantworten schwer zu verarbeiten und erfordern eine aufwendige Signalentzerrung.

Die Nutzung mehrerer Sender auf unterschiedlichen Trägern setzt in der Regel voraus, dass die Trägerfrequenzen weit genug auseinander liegen, da sonst die empfangsseitige Unterscheidung zwischen den Sendern unmöglich wird und sich die einzelnen Übertragungen irreparabel stören.

Diese Bedingung der unabhängigen Frequenzbänder ist ein essentieller Faktor bei der Entwicklung von Mehrträger-Übertragungsverfahren. Längst sind Techniken entworfen worden, bei denen aus einem integrierten Sender auf mehreren Frequenzen unabhängige Daten übertragen werden. Diese sogenannten Frequenzmultiplexverfahren (*Frequency Division Multiplex,* FDM) besitzen immer die Möglichkeit, Daten unterschiedlicher Nutzer auf die einzelnen Kanäle zu verteilen.

Die Entwicklung solcher Mehrträgersysteme hat gezeigt, dass die Trägerfrequenzen relativ weit auseinander liegen müssen. Die Spektren der einzelnen Kanäle dürfen sich nicht überlappen. Darüber hinaus ist sogar je nach Flankensteilheit das Bandpassfilter im Empfänger noch ein ungenutzter Sicherheitsabstand zwischen den einzelnen Spektren nötig.

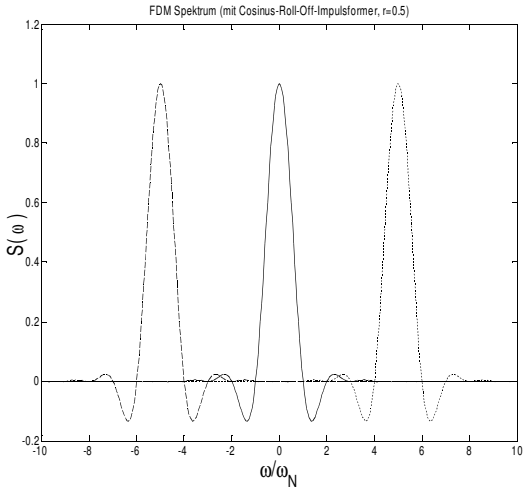


Abb. 2.1 FDM-Spektrum

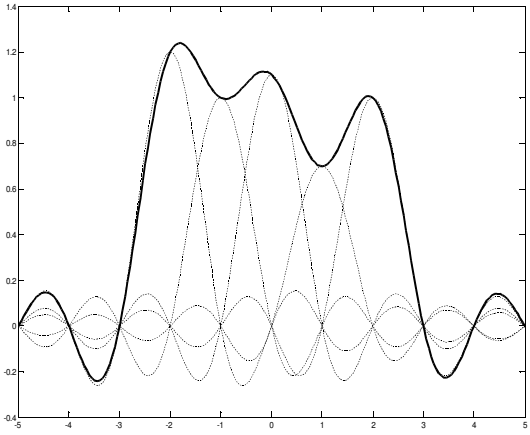


Abb. 2.2 OFDM-Spektrum

Hier setzt die Entwicklung der OFDM-Technologie an. Durch geeignete Wahl der Trägerfrequenzen wird trotz überlappender Trägerspektren die Orthogonalität der Sendesignale über die Symboldauer gewährleistet, sodass empfangsseitig eine vollständige Trennung der Einzelsignale möglich ist. Dieser Effekt wird in Kap. 2.1.2 noch genauer betrachtet.

Historisch gesehen wurde nach verschiedenen Veröffentlichungen zu orthogonalen Signalen oder Mehrträgerverfahren mit überlappenden Spektren 1971 das erste vollständige OFDM-System von Weinstein und Ebert [WEI71] konzipiert. Eine Einordnung der wichtigsten Meilensteine bei der Entwicklung der OFDM Übertragungstechnik vor und nach dieser Publikation findet sich z.B. in [MAY99].

Bemerkenswert bei der Entwicklungsgeschichte ist die Tatsache, dass die theoretischen Grundlagen dieser Übertragungstechnik sehr früh zur Verfügung standen. Es war jedoch technologisch zunächst nicht möglich, diese in reale Systeme umzusetzen. Erst mit der Entwicklung von schnellen Halbleiterbauelementen wie Digitalen Signalprozessoren (DSPs) oder FFT-Prozessoren konnten erste Übertragungseinheiten aufgebaut werden.

Inzwischen hat die OFDM-Technologie in vielfältigen Bereichen der Kommunikation Einzug gehalten. WLAN, *Digital Video Broadcasting-Terrestrial* (DVB-T), *Digital Subscriber Line* (DSL) oder *Digtal Audio Broadcasting* (DAB) sind wohl die bekanntesten kommerziellen Einsatzgebiete.

Eine wichtige Motivation für diese Arbeit ist die Tatsache, dass die Leistungsfähigkeit von OFDM-Übertragungssystemen sehr stark von den Möglichkeiten der Hardware abhängig ist. Sie lässt sich in Bezug auf Datenkapazität und Fehlersicherheit mit leistungsstarken Komponenten immer noch weiter verbessern. In realen Systemen hängt also die praktisch erzielbare Übertragungsgeschwindigkeit nicht zuletzt von der Verarbeitungsgeschwindigkeit der

Halbleiterbauelemente ab. Auf die verwendete FPGA-Hardware des im Rahmen dieser Arbeit entwickelten OFDM-Demonstrators wird in Kapitel 3 ausführlich eingegangen.

2.2 Mobilfunkkanäle

OFDM ist ein aussichtsreicher Kandidat für zukünftige Mobilfunksysteme. Um die Vorteile dieser Technik herausstellen zu können, ist es notwendig, einige Eigenschaften der bei mobiler Funkübertragung vorliegenden Übertragungskanäle zu charakterisieren.

Aus Messungen und praktischen Erfahrungen heraus ist deutlich geworden, dass es bei einer Datenübertragung mit Hilfe von Funktechnologie häufig zu Mehrwegeausbreitungen kommt. Insbesondere wenn keine direkte Sichtverbindung (*Non Line Of Sight*, NLOS) zwischen Sender und Empfänger besteht, überlagern sich am Empfänger signifikante Signalanteile von mitunter zahlreichen indirekten Ausbreitungspfaden.

Störungen wie additives Rauschen (z.B. AWGN[1]), Signaldämpfung oder zeitliche Verzögerung können unabhängig von der Mehrwegeausbreitung in allen kabelgebundenen und kabellosen Kanälen auftreten. Bei Letzteren kommt durch die Möglichkeit, dass sich die Kommunikationspartner relativ zueinander bewegen, noch der Dopplereffekt [DOP42] hinzu. Die speziellen Störungen durch Mehrwegeausbreitung ergeben sich aus der Tatsache, dass am Empfänger Signale aus unterschiedlichen Übertragungspfaden überlagert werden. Variierende Signallaufzeiten (engl. *Delay*), Dämpfungen, Interferenzen und, wenn sich Sender und Empfänger bewegen, auch unterschiedliche Frequenzverschiebungen durch den Dopplereffekt auf den Ausbreitungspfaden beeinflussen dann die Übertragung [PAE99]. Abb. 2.3 zeigt ein Beispiel mit zwei Pfaden, bei dem diese Effekte auftreten.

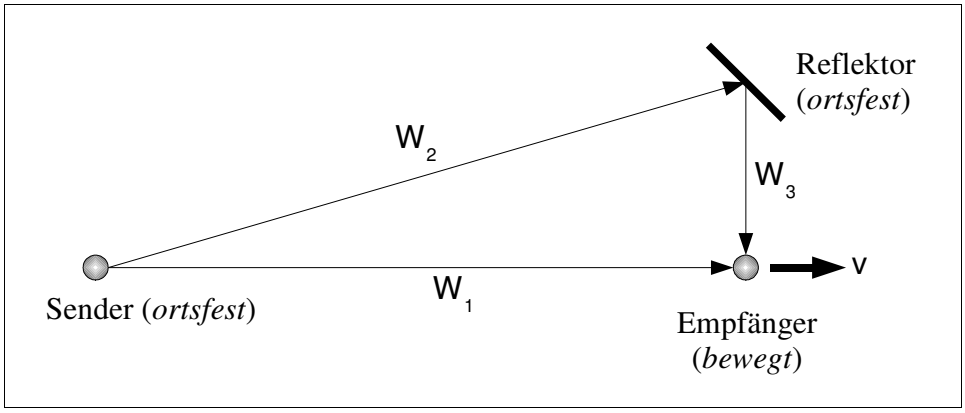


Abb. 2.3 Laufzeitunterschied und Dopplereffekt bei Mehrwegeausbreitung

1 AWGN (engl. *Additive White Gaussian Noise*) bezeichnet ein Rauschsignal, welches Gauß- bzw. normalverteilt ist. Dabei sind kleinere Rauschamplituden deutlich häufiger als große, sodass ein Histogramm dieser Störgröße die Gauß'sche Glockenform aufweist. Weißes Rauschen bezeichnet Störgrößen mit konstantem Leistungsdichtespektrum.

Die Skizze zeigt ein Szenario mit einem feststehenden Sender und einem bewegten Empfänger. Am Empfänger treffen zwei Signale von unterschiedlichen Pfaden ein: vom direkten (*Line Of Sight,* LOS) Pfad zum einen und von einem indirekten, durch Signalreflexion an einem stehenden Objekt entstandenen Pfad zum anderen. Dieses schematische Beispiel zeigt, wie Delay und Doppler-Einflüsse entstehen können.

Der Unterschied in der Laufzeit τ entsteht dadurch, dass die reflektierte Welle einen längeren Weg zurücklegt. In einem homogenen Ausbreitungsmedium wird das indirekte Signal proportional zur Verlängerung des Weges verzögert

$$\tau_{W_2} = \frac{\Delta l}{c} = \frac{W_2 + W_3 - W_1}{c} \qquad (2.1).$$

Hierbei wird angenommen, dass sich die Welle in Luft nahezu mit der Vakuum-Lichtgeschwindigkeit c ausbreitet. Δl bezeichnet die Wegdifferenz zwischen direktem und reflektiertem Pfad.

Im Allgemeinen wird die Übertragung mithilfe der Kanalimpulsantwort *h(t)* beschrieben. Bei der Mehrwegeausbreitung kann die Kanalimpulsantwort als Summe der N_P Einzelpfad-Impulsantworten in Abhängigkeit der jeweilige Laufzeitunterschiede τ beschrieben werden als

$$h(\tau, t) = \sum_{\nu=1}^{N_P} h_\nu \delta(t - \tau_\nu) \qquad (2.2).$$

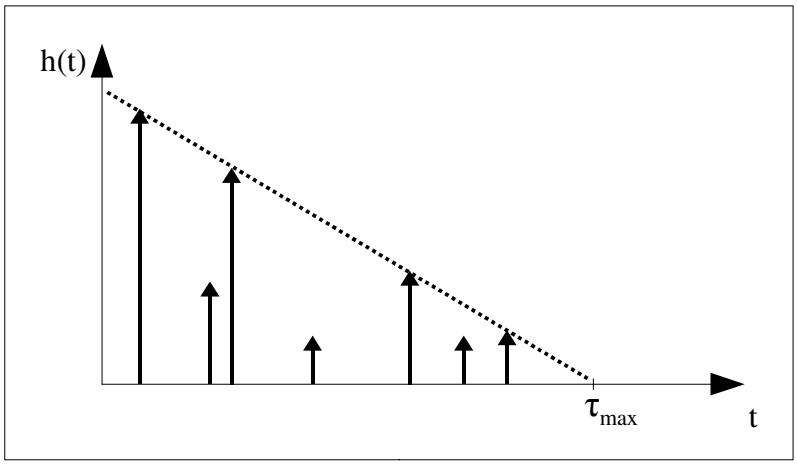


Abb. 2.4 Beispiel einer Kanalimpulsantwort bei meheren Ausbreitungspfaden (n=7)

Im deterministischen Fall, bei dem Dämpfungskoeffizienten α_ν, Phasenverschiebungen Φ_ν und Laufzeitunterschiede τ_ν bekannt sind, ergibt sich die Kanalimpulsantwort für zeitinvariante Systeme und die dazugehörige Übertragungsfunktion (Fouriertransformierte von *h(t)*) zu

$$h(t)=\sum_{\nu=1}^{N_P}\alpha_\nu e^{j\Phi_\nu}\delta(t-\tau_\nu)\circ\!\!-\!\!\bullet\sum_{\nu=1}^{N_P}\alpha_\nu e^{j\Phi_\nu}e^{-j2\pi f\tau_\nu}=H(f) \qquad (2.3).$$

Die Übertragungsfunktion in Gleichung 2.3 macht deutlich, dass ein zeitinvarianter Mehrwegekanal ein frequenzabhängiges Übertragungsverhalten aufweist. Diese Frequenzselektivität entsteht durch die je nach Frequenz konstruktiven bzw. destruktiven Überlagerungen. Dies bedeutet, dass einige Frequenzanteile in Abhängigkeit von der Kanalsituation verstärkt oder gedämpft werden. Ein Beispiel des Spektrums eines frequenzselektiven Kanals zeigt Abb. 2.5.

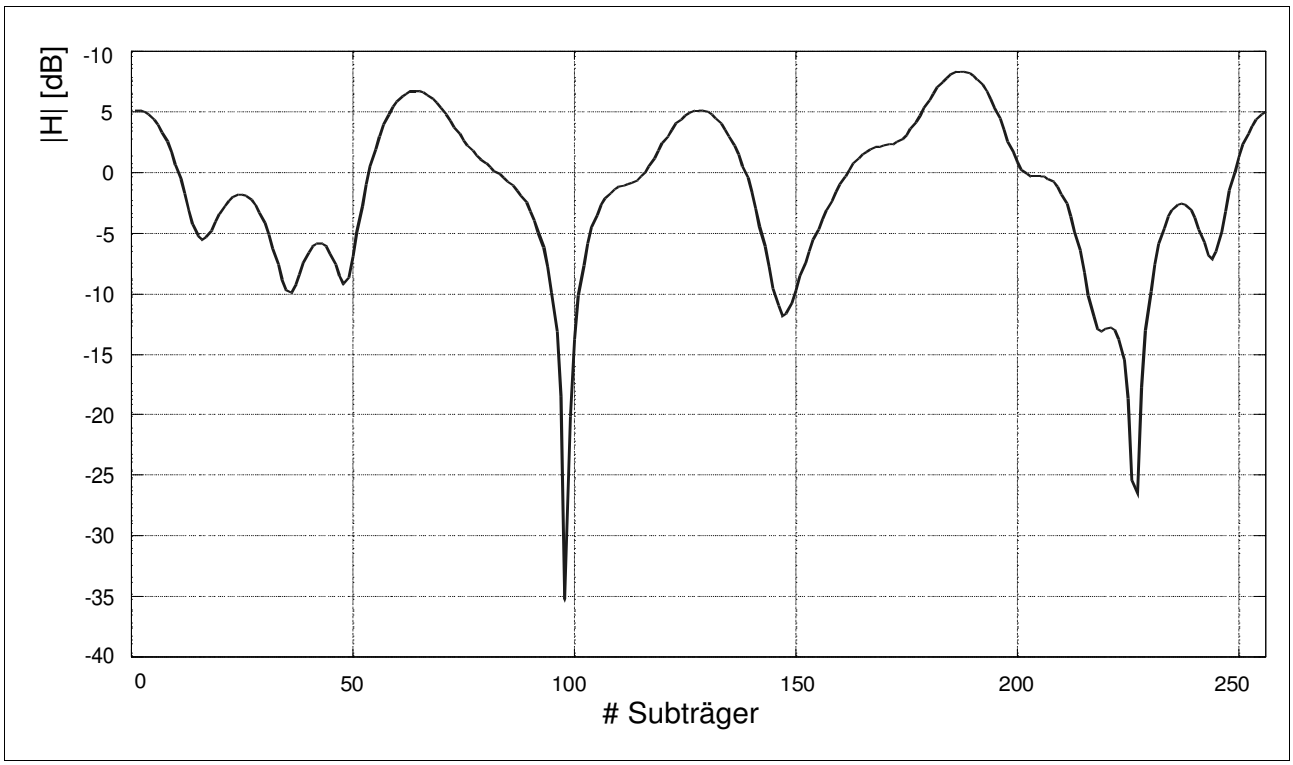


Abb. 2.5 Spektrum eines frequenzselektiven Kanals

OFDM-Übertragungssysteme nutzen daher meistens Techniken, welche die Dateninformation über mehrere Trägerfrequenzen spreizen, z.B. durch eine Kanalcodierung in Verbindung mit einem Interleaver. So kann verhindert werden, dass einzelne starke Dämpfungseinbrüche im

Übertragungsspektrum unmittelbar zu Datenverlust führen. Dahinter steht dann redundante Information, die sich statistisch über leistungsstarke und gedämpfte Subträger verteilt [COS01].

In dem in Abb. 2.3 skizzierten Beispiel wird jedoch zusätzlich von sich bewegenden Elementen ausgegangen. In diesem Fall handelt es sich um ein zeitvariantes System, bei dem die Kanalparameter Dämpfung, Phasenverschiebung und Laufzeitunterschiede zeitlich veränderlich sind. Dafür muss die Impulsantwort allgemeiner gefasst werden

$$h(\tau,t)=\sum_{\nu=1}^{N_P}\alpha_\nu(t)e^{j\Phi_\nu(t)}\delta(t-\tau_\nu(t)) \qquad (2.4).$$

Bei der Betrachtung von OFDM-Signalen kann häufig vereinfachend angenommen werden, dass die Dämpfung und die Phasendrehung für kurz Zeitabschnitte (in der Größenordnung der OFDM-Symboldauer) konstant bleiben. Es bleibt jedoch der Einfluss des Dopplereffekts, der bei sich relativ zueinander bewegten Objekten zu einer Frequenzverschiebung führt [STR92].

$$f_D=f_0\frac{v}{c} \qquad (2.5).$$

In Abb. 2.3 bewegt sich der Empfänger mit der Geschwindigkeit v exakt in Richtung von W_1 vom Sender weg. Relativ zur Ausbreitungsrichtung des reflektierten Signals W_3 bewegt sich der Empfänger nicht. Damit ist in diesem Fall der Frequenzunterschied der beiden Pfade gleich der Dopplerfrequenz aus der Bewegung in Richtung von W_1. Die Kanalimpulsantwort und die Übertragungsfunktion mit kurzzeitig konstanter Dämpfung und Phasendrehung unter Berücksichtigung der Dopplerverschiebung ist

$$h(\tau,t)=\sum_{\nu=1}^{N_P}\alpha_\nu e^{j\Phi_\nu}e^{-j2\pi f_d t}\delta(t-\tau_\nu)\circ\!\!-\!\!\bullet\sum_{\nu=1}^{N_P}\alpha_\nu e^{j\Phi_\nu}e^{-j2\pi f_d t}e^{-j2\pi f\tau_\nu}=H(f,t) \qquad (2.6).$$

Die hierbei getroffenen Annahmen (Kurzzeitbetrachtung) ermöglichen eine sehr gute Analyse von OFDM-Systemen. Es ist darüber hinaus auch möglich, allgemeinere Fälle von Funkübertragung mit einer wechselnden Anzahl von Pfaden, mit verschiedenen Laufzeiten und wechselnden Geschwindigkeiten mit jeweils unterschiedlichen Dopplereinflüssen zu beschreiben. Um solche widrigen Kanalsituationen nachzubilden, können statistische Größen genutzt werden. Dafür werden nach [LAM04][GRÜ00] der *Delay Spread* τ_S als Standardabweichung von τ

$$\tau_S = \sqrt{\int_{-\infty}^{\infty} (\tau - \tau_E)^2 \, p(\tau) \, d\tau} \qquad (2.7)$$

und der *Doppler Spread* $f_{D,S}$ als Standardabweichung von f_D

$$f_{D,S} = \sqrt{\int_{-\infty}^{\infty} (f_D - f_{D,E})^2 \, p(f_D) \, df_d} \qquad (2.8)$$

definiert. τ_E und $f_{D,E}$ sind jeweils die Erwartungswerte der Signallaufzeiten bzw. die Dopplerverschiebungen der einzelnen Ausbreitungspfade. Als wichtige Eingangsgröße müssen dafür die Wahrscheinlichkeitsdichtefunktion p(τ) und p(f_D) dieser Parameter durch Messungen ermittelt oder geschätzt werden.

Der Systementwurf ohne verfügbare konkrete Messwerte ist häufig einfacher mit Annahmen für die maximale relative Umweglaufzeit und die maximale relative Dopplerfrequenz, auch wenn diese Größen streng genommen nicht existieren. Es kann jedoch davon ausgegangen werden, dass je nach vorgesehenem Anwendungsfall Delays oder Dopplerfrequenzen oberhalb eines bestimmten Grenzwertes nur mit vernachlässigbarer Häufigkeit bzw. auf Pfaden mit vernachlässigbarem Beitrag zur Gesamtleistung auftreten.

Häufig werden in der Literatur für die Einflüsse durch bewegte Sender bzw. Empfänger und Laufzeitunterschiede die Begriffe **Kohärenzzeit**[2] und **Kohärenzbandbreite** eingeführt. Dabei ist es wichtig zu beachten, dass die Kohärenzzeit durch den Einfluss des Dopplereffekts und die Kohärenzbandbreite durch die unterschiedlichen Signallaufzeiten beeinflusst werden.

Die Kohärenzzeit beschreibt, wie schnell sich der Kanal über der Zeit ändert. Bewegen sich Sender und Empfänger relativ zueinander, ändert sich die Laufzeit der Signale über der Zeit (Zeitvarianz) und es kommt zum Dopplereffekt. Je geringer dieser Einfluss ist (f_D klein), umso größer ist die Zeit annähernd gleicher Bedingungen. Die Kohärenzzeit ist näherungsweise umgekehrt proportional zur Dopplerfrequenz

$$T_K \sim \frac{1}{f_{(D,max)}} \qquad (2.9).$$

2 Kohärenz (vom lat. cohaerere = zusammenhängen) bezeichnet in der digitalen Signalanalyse i. d. R. die Abhängigkeit des Ausgangssignals vom Eingangssignal. In dem hier aufgezeigten Kontext geht es jedoch um die Abhängigkeit von Signalanteilen aus den unterschiedlichen Ausbreitungspfaden zu verschiedenen Zeitpunkten bzw. bei verschiedenen Frequenzen.

Die Kohärenzbandbreite ist auch bei ortsfesten Systemen durch die Mehrwegeausbreitung definiert. Wie vorab beschrieben, kommt es durch die Laufzeitunterschiede verschiedener Pfade zu Interferenzen der Signalanteile und damit zu der in Abb. 2.5 gezeigten Frequenzselektivität. Dabei definiert die Kohärenzbandbreite eine Frequenzspanne mit annähernd konstanter Übertragungsfunktion und ist näherungsweise umgekehrt proportional zur maximalen Laufzeitdifferenz

$$B_K \sim \frac{1}{\tau_{max}} \qquad (2.10).$$

Die vollständige mathematische Herleitung dieser Kohärenzgrößen findet sich z.B. in [MAY99].

2.3 Mathematisches Model

Wie schon in der Einführung erwähnt, steht das „O“ bei OFDM für die Orthogonalität, eine der wesentlichen Eigenschaften der übertragenen Subträgersignale. Sie beschreibt die Möglichkeit, ein zusammengesetztes Sendesignal im Empfänger wieder eindeutig in seine einzelnen Komponenten zerlegen zu können. Mathematisch wird diese Bedingung wie folgt formuliert:

$$\int_a^b s_n(t) \cdot s_m(t) dt = 0 \ , \ falls: n \neq m \qquad (2.11)$$

Zwei Signale sind orthogonal, wenn das Integral ihres Produktes über einen definierten Bereich gleich Null ist. Das in OFDM-Übertragungssystemen erzeugte Sendesignal wird aus orthogonalen Sinus- und Cosinusfunktionen zusammengesetzt. Ein typisches OFDM-Signal wird im Basisband aus einer vorgegebenen Bandbreite gebildet, innerhalb derer die einzelnen Subträger äquidistant angeordnet sind. Häufig ist die Anzahl der Untergliederungen eine Potenz von zwei (dies erweist sich als Vorteil für die weitere Verarbeitung), wobei nicht alle der durch die äquidistante Aufteilung möglichen Frequenzen als Subträger genutzt werden müssen. Das Summensignal aus den einzelnen Subträgern erfüllt nicht nur die Bedingung aus Gleichung 2.11, sondern hat auch noch die wichtige Eigenschaft, dass es die Orthogonalität in Übertragungskanälen mit linearem zeitinvariantem Verhalten (LTI-Systeme) aufrechterhält, da die Sinus und Cosinuskomponenten des Sendesignals Eigenfunktionen des LTI-Kanals sind. Die Erfahrung hat gezeigt, dass die meisten Kanäle über die Symboldauer betrachtet hinreichend gut als LTI-System modelliert werden können und somit die Trennung der Subträgersignale im Empfänger sehr gut möglich ist.

Technisch realisiert wird die Erzeugung eines Basisband-OFDM-Sendesignals durch die Berechnung der inversen diskreten Fourier-Transformation (IDFT). Diese Funktion erzeugt aus den auf komplexe Sendesymbole abgebildeten Nutzdaten die entsprechenden komplexen Schwingungen im Zeitbereich (siehe Abb. 2.6 und Abb. 2.7).

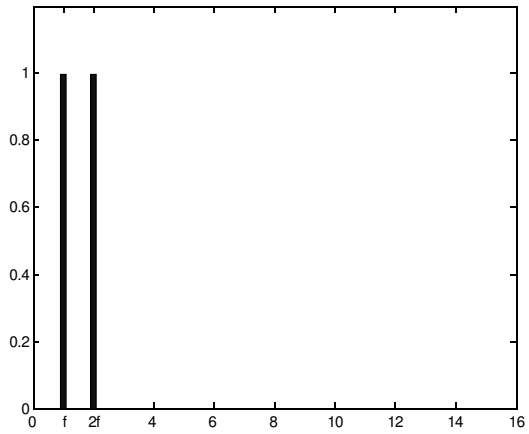


Abb. 2.6 OFDM-Beispiel mit zwei Trägern

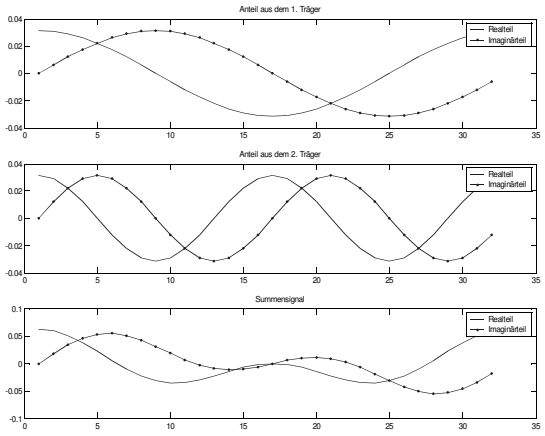


Abb. 2.7 Komplexes Zeitsignal mit zwei Frequenzen

Das in den Abbildungen dargestellte Beispiel zeigt die Konstruktion eines einfachen OFDM-Signals. Die Daten werden im Basisband erzeugt und nachfolgend in den gewünschten Trägerfrequenzbereich hochgemischt. Im Beispiel aus Abb. 2.6 sind im Basisband zwei Subträger bei den Frequenzen *f* und *2f* vorhanden. Daraus wird das Signal durch die Transformation in den Zeitbereich errechnet. Abb. 2.7 zeigt die zu den beiden Trägern gehörenden Grundschwingungen sowie das resultierende Summensignal. Im Allgemeinen sind bei einem OFDM-Signal sehr viel mehr Träger belegt, die jeweils unabhängig voneinander in Amplitude und/oder Phase moduliert werden können.

Ein weiterer wichtiger Punkt ist, dass die IDFT als Inverse Fast Fourier-Transformation (IFFT) sehr recheneffizient realisiert werden kann [COO65]. Nur dadurch ist es möglich, OFDM-Systeme mit sehr vielen Subträgern (bei DVB-T z.B. 2048 oder 8192) und hohen Datenraten platzsparend und wirtschaftlich in Hardware zu realisieren.

Die Verarbeitung der Daten geschieht dabei blockweise, je nach Länge der FFT *N*. Ein Block wird als OFDM-Symbol bezeichnet. Der Takt, in dem die Nutzdatenymbole aufeinander folgen, wird in Abhängigkeit vom Subträgerabstand Δf als $T_S = \frac{1}{\Delta f}$ gewählt, damit die Orthogonalitäts-bedingung in diesem Zeitintervall (Symboldauer) exakt erfüllt wird. So passt die kleinste mögliche Schwingung, also die der niedrigsten Frequenz (abgesehen vom Gleichanteil), genau mit einer

vollständigen Periode in diesen Zeitraum. Alle anderen Schwingungen sind mit Vielfachen ihrer Periode enthalten.

Die Berechnung des OFDM-Signals entspricht dabei der Umsetzung eines OFDM-Spektrums mit der zeitkontinuierlichen inversen Fouriertransformation auf zeitdiskrete Signale.

$$s(t)=\frac{1}{\sqrt{N}}\int_{-\infty}^{\infty} S(j\omega)\mathrm{e}^{j\omega t}\,d\omega \qquad (2.12)\quad \text{Inverse Fouriertransformation}$$

Das OFDM-Signals nutzt eine definierte Bandbreite B, in der die Modulationssymbole $S_{n,k}$ äquidistant auf Subträger verteilt angeordnet sind. Dadurch kann das Integral aus 2.12 auf die Summe der Subträger reduziert werden.

$$s_n(t) = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} S_n(k\,2\pi\Delta f)\mathrm{e}^{j2\pi k\Delta f t} \qquad (2.13)$$

Die Subträger liegen hier an den Stellen $\omega_k=k\ 2\pi\Delta f$. Das bandbegrenzte Signal erlaubt die in der digitalen Verarbeitung notwendige Betrachtung an zeitdiskreten Abtastpunkten Δt. Pro OFDM-Symbol gibt es N Abtastwerte. Die Länge des Abstastintervalls beträgt $\Delta t=1/B = 1/(N\ \Delta f)$. Durch Einsetzen der Abtastzeitpunkte $i\Delta t$ mit $i=0, 1, .. N-1$ erhält man

$$s_n(i\,\Delta t) = s_{n,i}=\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} S_{n,k}(k\,2\pi\Delta f)\mathrm{e}^{j2\pi k\Delta f\ i\Delta t}=\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} S_{n,k}(k\,2\pi\Delta f)\mathrm{e}^{j2\pi k\frac{i}{N}}$$

(2.14) IDFT, IFFT

Diese blockweise Verarbeitung bei OFDM entspricht somit exakt der IDFT und kann somit auch besonders recheneffizient als IFFT realisiert werden.

Im Empfänger wird das Eingangssignal wieder in den Frequenzbereich transformiert, um die einzelnen Subträger auswerten zu können. Dazu wird dann entsprechend die Diskrete Fourier-Transformation ausgeführt, die wieder als Fast Fourier-Transformation (FFT) realisiert wird.

$$R_{n,k}=\frac{1}{\sqrt{N}}\sum_{i=0}^{N-1} r_{n,i}\mathrm{e}^{-j2\pi ik/N} \qquad (2.15)\quad \text{DFT, FFT}$$

2.4 Parameter und Verhalten von OFDM im Mobilfunkkanal

Im vorhergehenden Kapitel wurde gezeigt, wie das Prinzip der OFDM-Übertragung im Falle eines störungsfreien Kanals funktioniert. Doch wie verhält sich eine solche Übertragungsstrecke in einer Mehrwegesituation, also mit Delay und Doppler Spread?

Ein großer Vorteil von OFDM ist die verhältnismäßig langsame Symbolfolge. Je nach Zahl *N* der genutzten Subträger verlängert sich die Symboldauer im Vergleich zu einem Einträgersystem, bei dem jedes Modulationssymbol für sich unabhängig und in einem entsprechend kürzeren Zeitschlitzschlitz übertragen wird. Der Aufwand zur Entzerrung[3] wird damit sehr viel geringer, weil sich die Intersymbolinterferenz (ISI) nur auf einen Bruchteil des Symbols auswirkt.

Dennoch ist dieser Effekt nicht zu vernachlässigen. In fast allen realisierten OFDM-Standards wird der Störeinfluss der Nachbarsymbole durch das senderseitige Hinzufügen eines Schutzintervalls (engl. Guard-Intervall) eliminiert bzw. minimiert.

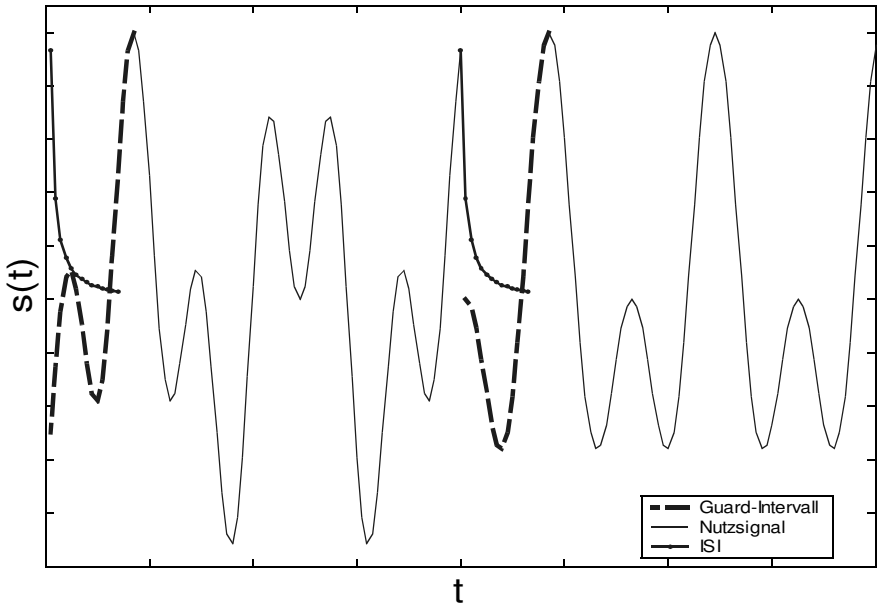


Abb. 2.8 OFDM-Zeitsignal mit Guard-Intervall (und symbolisch eingezeichneter ISI)

Dazu wird von dem jeweils aktuellen OFDM-Symbol der hintere Signalausschnitt kopiert und vorne an das Symbol angefügt. Abb. 2.8 zeigt beispielhaft ein so erweitertes Zeitsignal. Durch diese periodische Erweiterung bleiben die Orthogonalität und die Information auf den einzelnen Subträgern erhalten, und Signalanteile des Guard-Intervalls (durch Verzögerungen im

3 Vergleichbar ist hier die Frequenzbereichsentzerrung bei vorhandenem Guard-Intervall, wie es im Folgenden beschrieben wird.

Mehrwegekanal) bleiben ohne negativen Einfluss auf das eigentliche OFDM-Symbol. Die Länge des Guard-Intervalls sollte dafür so dimensioniert sein, dass es länger ist als der maximale relative Delay. Da es im Empfänger wieder verworfen wird, ist in diesem Fall der ISI-Einfluss vollständig eliminiert.

Innerhalb der ISI-freien Daten kann der Verarbeitungsblock (Länge T_S) beliebig ausgeschnitten werden. Die zeitliche Verschiebung des Blocks (Synchronisations-Offset) führt zwar zu einer konstanten zyklischen Verschiebung im Zeitbereich, diese kann aber durch eine Phasendrehung in der Kanalschätzung bzw. im Entzerrer wieder herausgerechnet werden. Eine Verschiebung des Verarbeitungsblocks nach vorne, also in den ISI gestörten Bereich, ist dabei weniger kritisch als eine Verschiebung nach hinten, also in das Folgesymbol. Im ersten Fall werden noch prinzipiell richtige Daten übernommen, die durch die in der Regel zeitlich exponentiell abfallende Überlagerung von Signalanteilen aus dem Vorgängersymbol gestört sind.

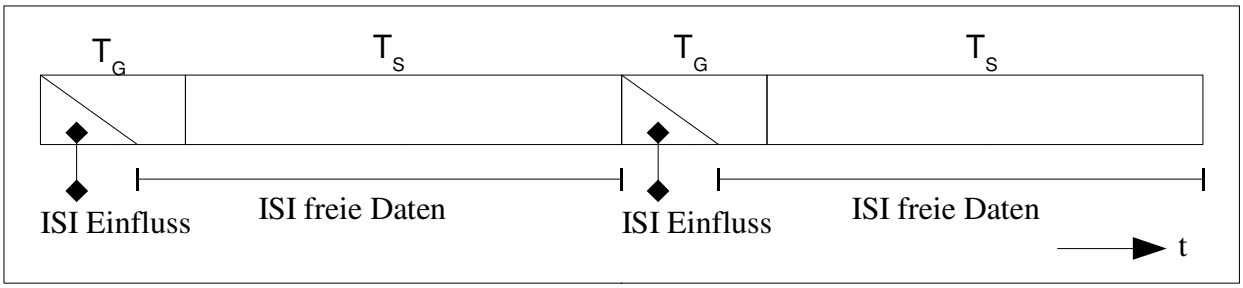


Abb. 2.9 Die Einflusstiefe der Impulsantwort auf OFDM-Symbole

Ein Nachteil dieser Methode ist jedoch offensichtlich: Die Gesamtsymboldauer verlängert sich um den Anteil des Guard-Intervalls

$$T = T_G + T_S \qquad (2.16)$$

Es muss also je nach erwartetem maximalen relativen Delay zusätzliche Übertragungszeit geopfert werden. Als Designregel gilt hier, dass das Guard-Intervall kurz sein sollte gegenüber der genutzten Symboldauer, also $T_G << T_S$.

Die Einflüsse des Dopplereffekts sind hingegen nur sehr schwer wieder zu eliminieren. Eine geeignete Wahl der OFDM-Parameter kann jedoch diesen Einfluss vernachlässigbar klein gestalten. Die Bedingung dafür ist, dass der Subträgerabstand deutlich größer ist als die maximale Dopplerfrequenz

$$\Delta f >> f_{\mathrm{D,max}} \qquad (2.17)$$

Die Überlagerung von Signalkomponenten mit verschiedenen Dopplerverschiebungen ist jedoch ein Effekt, der nur bei bewegten Sendern, Empfängern oder Reflektoren auftritt.

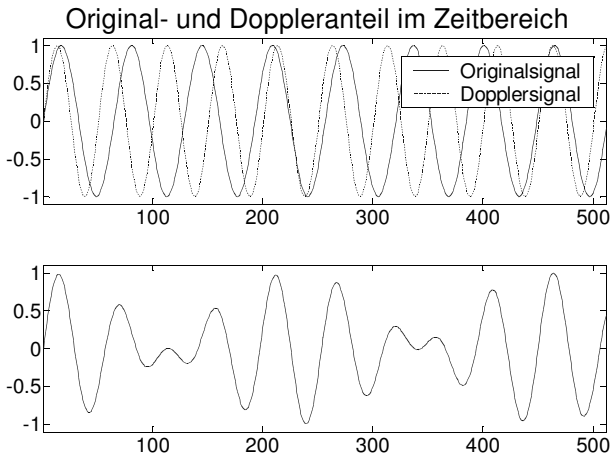


Abb. 2.10 Dopplereffekt im Zeitbereich (Einzelkomponenten und Überlagerung)

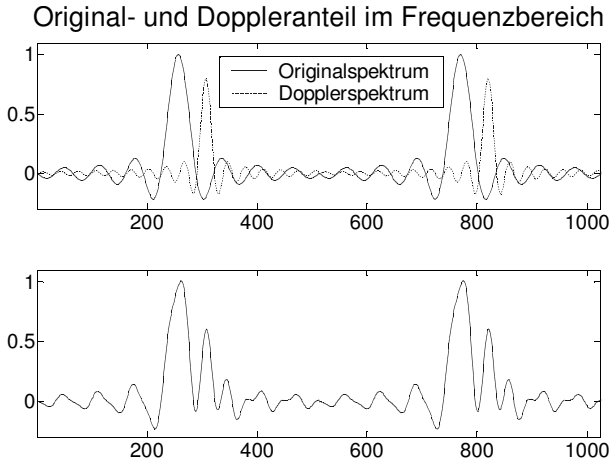


Abb. 2.11 Verzerrungen durch Dopplerüberlagerung im Frequenzbereich

In vielen Szenarien, wie z.B. den WLAN-Netzen HiperLAN/2 oder IEEE802.11a, sind kaum Bewegungen zu erwarten bzw. wird maximal Fußgängergeschwindigkeit angenommen. In diesen Fällen ist der Dopplereffekt kaum ein Entwurfskriterium. Erst bei Übertragungsstrecken zu Autos oder Flugzeugen muss dieser Punkt sehr genau beachtet werden. Störungen durch Dopplerverschiebungen werden als *Inter Channel Interference* oder *Inter Carrier Interference* (ICI) bezeichnet, weil sich die einzelnen Subträger gegenseitig stören. Hierbei wird dann auch direkt die Orthogonalitätsbedingung verletzt.

Es sind damit zwei Designregeln für OFDM-Parameter hergeleitet worden:

$T_G << T_S$, Guard-Intervall kurz gegenüber Nutzsymboldauer,

$\Delta f >> f_{D,max}$, Subträgerabstand groß gegenüber maximaler Dopplerfrequenz.

Beispiel: *Bei den sehr ähnlichen Standards HiperLAN/2 und IEEE802.11a sollen die OFDM-Parameter einmal überprüft werden. Dabei sind eine Übertragungsbandbreite von 20 MHz und die Sendefrequenz bei 5 GHz vorgegeben.*

Als Indoor-Netzwerk wird eine Reichweite von 200 m angestrebt und maximal Fußgängergeschwindigkeit (ca. 6 km/h bzw. 1.7 m/s) erlaubt. Angenommen wird weiter, dass Pfade, die mehr

als doppelt so lang sind wie der direkte Pfad, so stark gedämpft sind, dass sie keine Rolle mehr spielen. Daraus ergibt sich der maximale Delay zu

τ_{max} = 200 m / c = 0.67 µs (< T_G = 0.8 µs)

und die maximale Dopplerfrequenz

$f_D = (f_0 * v) / c$ = 28.3 Hz. (<< Subträgerabstand 312.5 kHz)

Die Standards haben folgende Parameter: T_G = 0,8 µs, T_S = 3,2 µs, Bandbreite = 20 MHz, 64 Subträger im Abstand von je 312,5 kHz, von denen 48 für Nutzdaten und 4 für Pilotsymbole verwendet werden und 12 ungenutzt bleiben. Es sind also alle Kriterien erfüllt, wobei immerhin 20% der Symboldauer für das Guard-Intervall aufgewendet werden muss, um das System robust gegen ungünstige Mehrwegesituationen zu machen.

Im obigen Beispiel ist der Subträgerabstand um mehr als den Faktor 1000 größer als die veranschlagte maximale Dopplerfrequenz. Dies ist von Vorteil, weil ggf. Subträgerüberlagerungen (ICI) das OFDM-System unmittelbar stören und kaum zu korrigieren sind. Sicherlich spielte bei dem Systementwurf auch eine Rolle, dass eine 64'er FFT (relativ geringer Rechenaufwand) in 4 µs ausreichen sollten und/oder die Symboldauer wegen der Latenzzeiten nicht größer sein sollte.

ICI kann jedoch nicht nur durch den Einfluss von Dopplerfrequenzen entstehen. Eine sehr lange Impulsantwort des Kanals ($\tau_{max} > T_G$) oder eine unpräzise Zeitsynchronisation mit sehr großem Offset führt dazu, dass der in der FFT verarbeitete Datenblock nicht mehr ungestört ist. Das Einbeziehen von Daten, die nicht zum aktuellen Symbol gehören, führt zu einer unsauberen Rechteckfensterung im Zeitbereich. Dadurch werden die si-Impulse $\left(si(x) = \frac{\sin(x)}{x} \right)$ im Frequenzbereich verzerrt und die einzelnen Träger beginnen, sich gegenseitig zu beeinflussen (ICI). Dieser Einfluss durch zu lange Impulsantworten oder falsch ausgeschnittenen Datenblöcken entspricht der durch den Dopplereinfluss hervorgerufenen ICI. Eine Berechnung dieser Störgrößen in Abhängigkeit vom Symboloffset ist in [BRÜ02] beschrieben.

Einer der wichtigsten Zusammenhänge bei der Betrachtung von OFDM-Systemen für Mobilfunkkanäle ist jedoch, dass die Orthogonalität des Übertragungssignals für fast alle Szenarien (nämlich lineare Kanäle) erhalten bleibt.

2.5 Verarbeitungsschritte realer OFDM-Übertragungssysteme

Auf der physikalischen Schicht (auch Bitübertragungsschicht) eines Nachrichtenübertragungssystems kann eine Vielzahl von funktionalen Modulen zusätzlich genutzt werden, um die

Datenübertragung leistungsfähiger zu machen. Beispiele dafür sind Quellen- und Kanalcodierung, Vorver- und Entzerrer oder Mehrantennenübertragung. Viele dieser Maßnahmen sind nicht OFDM spezifisch, d.h. sie finden sich auch bei anderen Übertragungstechniken wieder. Dennoch ist die Abstimmung der einzelnen Komponenten auf das eingesetzte Übertragungsverfahren sehr wichtig. Hier werden Maßnahmen zur Verbesserung der Übertragungssicherheit ebenso berücksichtigt wie Punkte, die die Flexibilität bei der Datenübertragung steigern. Dabei stehen die Möglichkeiten der adaptiven Anpassung an die aktuellen Übertragungsbedingungen im Vordergrund.

Abb. 2.12 zeigt eine typische OFDM-Verarbeitungskette der physikalischen Schicht vom Fehlerschutz bis zur Digital/Analog-Schnittstelle. Das zur Funkübertragung notwendige analoge Mischen auf eine HF-Frequenz wird hier nicht im Detail betrachtet, da es nicht Teil der digitalen Datenverarbeitung im FPGA ist.

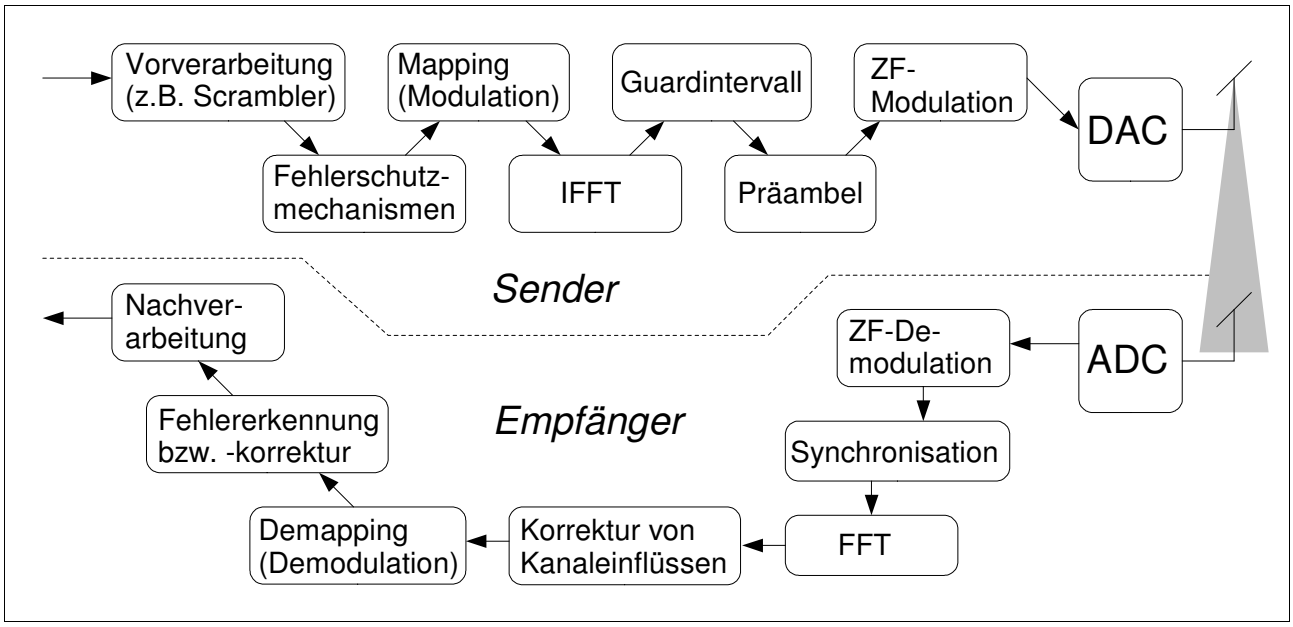


Abb. 2.12 Typische Verarbeitungsschritte der physikalischen Schicht von OFDM-Systemen

Die Vorverarbeitung steht hier allgemein für alle Maßnahmen, die die nachfolgenden Verarbeitungsschritte unterstützen. In vielen Fällen wird hier der Datenstrom zunächst verwürfelt (Scrambler). Diese Dekorrelation der Daten verbessert die Statistik der Leistungsverteilung, gemessen am der *Peak-to-Average Ratio* (PAR)[4] innerhalb des OFDM-Symbols.

Fehlerschutzmechanismen werden eingebaut, um die Sicherheit bei der Übertragung zu erhöhen. Hierbei gilt die Regel, dass zusätzliche Sicherheit mehr redundante Daten erfordert und somit die

4 Die Verbesserung des PAR durch den Scrambler ist nicht garantiert (im Einzelfall kann es sogar schlechter werden). Wichtig ist aber, dass dieselbe Datenfolge bei Übertragungswiederholungen dann zu einem anderen OFDM-Symbol mit wahrscheinlich besserem (jedenfalls anderem) PAPR führt. Bei langen Nullfolgen o.ä. wird das PAR aber unmittelbar verbessert.

Nettodatenrate reduziert. Eine Anpassung der Menge an Redundanzdaten an die vorhandene Kanalqualität ermöglicht dabei eine optimale Nutzung der Übertragungskapazität bei vorgegebener Anforderung an die Zuverlässigkeit der Übertragung, wie z.B. eine maximal zulässige Bitfehlerrate.

Das Mapping ist die Abbildung der Datenbits auf Modulationssymbole in der komplexen Ebene. Auch hier gibt es eine Vielzahl von Möglichkeiten, mit denen die Nutzdaten für jeden einzelnen Subträger des OFDM-Symbols moduliert werden können. Beispiele hierfür sind: unterschiedliche Modulationswertigkeiten (Anzahl Bits pro Modulationssymbol), differentielle oder adaptive Modulation. Auch hier besteht die Möglichkeit, die maximale Datenrate gegenüber der Sicherheit abzuwägen. Je mehr mögliche Punkte in der komplexen Ebene genutzt werden, um so kleiner ist der „Sicherheitsabstand“ beim Demapping im Empfänger. Dafür bildet der einzelne Konstellationspunkt dann mehr Datenbits ab.

In den bekannten OFDM-Standards werden vorab Parametersätze (zulässige Kombinationen von Modulations- und Kanalcodierverfahren, sogenannte *PHY-Modes*) definiert, aus denen je nach aktueller Kanalqualität der jeweils bestgeeignete ausgewählt werden kann. Dabei wird bei schlechten Übertragungsbedingungen beispielsweise der Fehlerschutz erhöht und gleichzeitig die Modulationswertigkeit herabgesetzt. So ist auch in extrem ungünstigen Szenarien eine Datenübertragung möglich, wenn auch mit sehr geringer Datenrate. In [HIP01] wird bei optimalen Bedingungen die höchste Datenrate (54Mbps) durch den Modus mit der Kombination 64-QAM/R=¾ erzielt. In ungünstigen Situationen kann die Datenrate in mehreren Stufen auf bis zu 6Mbps (BPSK, R=½) reduziert werden.

Nach dem Mapping folgt mit der blockweisen Berechnung der IFFT und dem Hinzufügen des Guard-Intervalls der OFDM-spezifische Teil, wie er in den vorangestellten Kapiteln beschrieben ist.

Das Einfügen einer Präambel und ein eventueller digitaler IQ-Modulator sind wiederum nicht OFDM spezifisch. Das Hinzufügen einer vordefinierten Datensequenz (wegen der Position vor den eigentlichen Daten „Präambel“ genannt) wird im Empfänger zum Erkennen des Beginns eines Datenblocks, zur Kanalschätzung und zur Synchronisation genutzt. Die digitale IQ-Modulation kann eingesetzt werden, um das analoge Signal hinter (bzw. im Empfänger vor) dem D/A- (bzw. A/D-)Wandler direkt auf einer Zwischenfrequenz zur Verfügung zu haben [COE01]. Die Vorteile dabei sind eine geringere Anfälligkeit gegen Störeinflüsse (keine IQ-Imbalance) und weniger hohe Anforderungen an die Filter im nachgeschalteten analogen Mischer. Ein solcher HF-Mischer (Trägerfrequenz 5 GHz bei IEEE802.11a und HiperLAN/2) bildet zusammen mit einem Umschalter zwischen Sende- und Empfangsmodus (bei bidirektionalen Funksystemen), verschiedenen Verstärkern und den Antennen das analoge HF-Frontend.

Der Empfänger wird neben den inversen Einheiten des Senders noch durch die Synchronisation und die Kanalkorrektur (Entzerrung) ergänzt. Eine zeitliche Synchronisation ist bei der Datenübertragung normalerweise unumgänglich, um Nutzdaten von Sendepausen oder Störsignalen

zu unterscheiden, das korrekte Herausschneiden der Datenblöcke und das Aktivieren aller Empfangsverarbeitungsmodule zum richtigen Zeitpunkt zu gewährleisten. Die Kanalschätzung bzw. die zugehörige Entzerrung ermöglicht eine qualitative Verbesserung des Empfangssignals durch das Herausrechnen bekannter Kanaleinflüsse. Das Prinzip basiert darauf, dass sich die Kanalsituation im Verhältnis zur Datenblocklänge nur langsam ändert. Damit ist es möglich, den Kanaleinfluss z.B. durch die vorangestellte Präambel zu schätzen und die nachfolgenden OFDM-Blöcke entsprechend zu entzerren.

Ein solches vollständiges OFDM-System ist im Rahmen dieser Arbeit auf einer Demonstrationsplattform implementiert worden. Die einzelnen Module werden im Detail in Kapitel 5 vorgestellt und hinsichtlich Gesamtperformance und Hardwaredesign genau untersucht.

2.6 Vorteile und Herausforderungen zukünftiger OFDM-Systeme

OFDM-Systeme bieten einen hohen Grad an Flexibilität. Die Eigenschaft, dass auf mehreren Frequenzen Symbole in einem festen Zeitraster gesendet werden, bietet den Vorteil, verschiedene Zugriffsverfahren einsetzen zu können. Das Zugriffsverfahren beschreibt die Möglichkeit, mehreren Teilnehmern Ressourcen der Übertragungsstrecke zuweisen zu können. Die wichtigsten drei Verfahren dabei sind *Time, Frequency* und *Code Division Multiple Access* (TDMA, FDMA und CDMA). Dabei wird die Datenkapazität wie folgt auf die einzelnen Teilnehmer verteilt:

- TDMA: Jedem Teilnehmer werden Zeitschlitze zugewiesen, in denen er exklusiv Daten übertragen kann.
- FDMA: Jeder Teilnehmer darf eine oder mehrere Frequenzen (*Subcarrier*) exklusiv nutzen.
- CDMA: Die Daten der verschiedenen Teilnehmer werden „überlagert“ auf mehrere Frequenzen und/oder Zeitsymbole verteilt. Dabei werden Spreizcodes eingesetzt, die es erlauben, im Empfänger die Anteile des jeweils gewünschten Teilnehmers herauszurechnen.

Umfangreiche Informationen und Untersuchungen über Zugriffsverfahren in OFDM-Systemen sind in [GRÜ00] zusammengestellt.

Ein weiterer Vorteil von OFDM ergibt sich aus der breitbandigen Übertragung. Das Nutzband ist in komplexen Mehrwegeszenarien zwar häufig frequenzselektiv, weist also vereinzelt starke Leistungseinbrüche bei einzelnen Frequenzen auf. Da OFDM-Empfänger jedoch oft mit zuverlässigen Mechanismen zur Kanalschätzung ausgerüstet sind, kann für jeden einzelnen Subträger eine Qualität bestimmt werden um die Modulationswertigkeit entsprechend zu adaptieren. Beispielsweise wird ein stark gedämpfter Subträger nicht genutzt oder nur BPSK moduliert (ein Informationsbit), während Träger mit guter Empfangsleistung mit 64-QAM sechs Informationsbit

übertragen. Eine derartige effiziente adaptive Modulation kann die Leistungsfähigkeit der OFDM-Datenübertragung zusätzlich steigern, wobei die Herausforderung in einer geschickten Verteilung der Nutzdatenbits auf die einzelnen Subträger (*engl. Bitloading*) liegt [FIS96][LAM99a][GRÜ01]. Dies wird jedoch in den in dieser Arbeit betrachteten Referenzsystemen HiperLAN/2 und IEEE802.11a noch nicht unterstützt.

Noch weiterführend könnten die wichtigen Parameter des OFDM-Systems wie die Länge des Guard-Intervalls und/oder die Zahl der genutzten Subträger, die wie in 2.4 beschrieben für bestimmte angenommene Szenarien kalkulierbar sind, adaptiv verändert werden. Dazu ist jedoch eine sehr leistungsfähige Hardware notwendig, die z.B. in der Lage sein muss verschiedene FFT-Längen zu verarbeiten.

Dennoch ist die OFDM-Technik nicht in jedem Fall die ideale Übertragungstechnik. Sie erkauft sich die Robustheit im Mobilfunkkanal mit Mehrwegeausbreitung durch zusätzlichen Verarbeitungsaufwand (insbesondere die IFFT und FFT) und einen Datenoverhead (Guard-Intervall). Außerdem werden bei einem OFDM-System partiell sehr hohe Anforderungen an die Komponenten gestellt. So ist der Erhalt der Linearität von wesentlicher Bedeutung, woraus sich strenge Vorgaben beim Aufbau der analogen HF-Einheiten wie Mischer und Verstärker ableiten. Auch die Synchronisation muss notwendig sehr exakt arbeiten, um den Erhalt der Orthogonalität der Subträger zu gewährleisten.

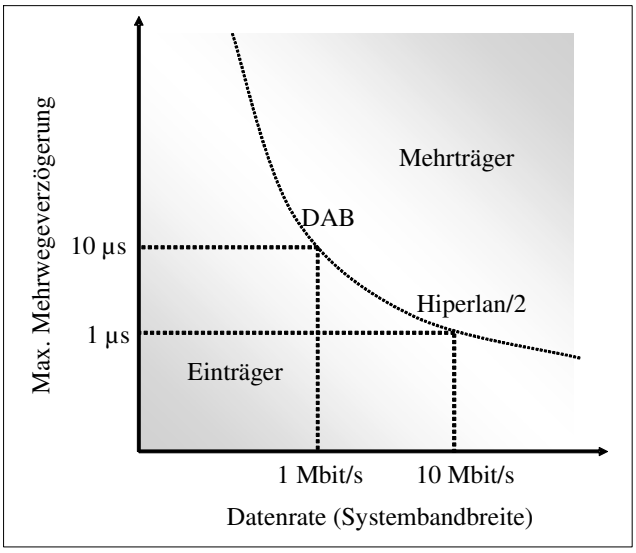


Abb. 2.13 Mehrträgerverfahren erzielen hohe Datenraten trotz großer Mehrwegverzögerung

Ein weiteres typisches Merkmal der OFDM-Verarbeitung ist das durch die IFFT generierte große Peak-to-Average-Verhältnis. Dies führt zu großen Sprüngen in der Signalamplitude und erfordert DA-Wandler mit guten Ausgangstreibern und sehr lineare Leistungsverstärker. Eine zusätzliche Vorverarbeitung kann diesen Effekt jedoch wirksam reduzieren. Untersuchungen dazu finden sich z.B. in [LAM99b][MAY98] [MÜL98].

Bei Kanälen mit guten Übertragungseigenschaften wie bei der kabelgebundenen Übertragung (z.B. Ethernet) oder stabilen LOS-Situationen können Einträgerverfahren möglicherweise mit geringerem Aufwand realisiert werden. Dies sind Kanalsituationen mit nur einem signifikanten Signalpfad oder mit geringen Laufzeitunterschieden (vgl. Abb. 2.13). In diesen Fällen kommen

einige Vorteile der ODFM-Verarbeitung, wie z.B. die Robustheit in Mehrwegeszenarien, ohnehin nicht zum Tragen.

Der allgemeine Trend zur kabellosen Kommunikation und hohen Verfügbarkeit von Informationen bietet jedoch viele Anwendungsfelder, in denen die Vorteile von OFDM sehr stark zur Geltung kommen und die diese Übertragungstechnik zu einem aussichtsreichen Kandidaten für zukünftige Mobilfunksysteme machen.

Die Tabelle 2.1 stellt noch einmal zusammenfassend die Vorteile eines OFDM-Systems denen einer herkömmlichen Einträger-Datenübertragung gegenüber.

Vorteile OFDM	Herausforderungen OFDM
Robust in Mehrwegesituation mit Fading	Höherer Rechenaufwand durch die IFFT/FFT
Entzerrung viel einfacher, kein bzw. kaum ISI	Empfindlich auf Trägerfrequenz-Offset
Technologie für hohe Datenraten bei hoher Mobilität	Tendenz zu einem ungünstigen Verhältnis von maximaler zu mittlerer Amplitude (PAR)
Geeignet für TDMA, FDMA und CDMA	Hohe Anforderung an die Linearität der analogen Baugruppen
Gute spektrale Effizienz	
Anpassungsfähig bei verschiedenen Kanal-situationen, insb. mit adaptiver Modulation	
Hohe Flexibilität unterstützt das Design von Mehrantennensystemen	

Tabelle 2.1 Die Charakteristik von OFDM-Systemen (mit Blick auf alternative Einträgerverfahren)

2.7 Ausblick und Alternativen zu OFDM

Der im Rahmen dieser Arbeit aufgebaute OFDM-Demonstrator ist mit Übertragungsraten von mehr als 50 Mbit/s schon sehr leistungsfähig und bietet aufgrund der flexiblen Programmierbarkeit und Erweiterbarkeit ein weites Spektrum an Möglichkeiten zur Untersuchung von Algorithmen und Techniken, die in aktuellen und zukünftigen Systemen eingesetzt werden können. An dieser Stelle soll kurz auf mögliche zukünftige Entwicklungen eingegangen werden, die in unmittelbarem Zusammenhang mit OFDM stehen und Denkanstöße zur Weiterentwicklung und Zukunft dieser Technologie geben können.

Neben einer Steigerung der Datenraten werden auch Kriterien wie Mobilität oder Reichweite wichtige Kriterien sein. Eine hohe Mobilität, wobei sich Sender und Empfänger relativ zueinander schnell bewegen, stellt, wie in den vorhergehenden Kapiteln diskutiert, unmittelbar Anforderungen

an die Übertragungstechnik. Eine Einordnung der wichtigsten aktuell definierten Mobilfunkstandards vor diesem Hintergrund ist in Abb. 2.14 dargestellt. Die erzielbaren Reichweiten sind stark von der Philosophie des Netzes abhängig, insbesondere bei zellularen Netzen mit definierter maximaler Sendeleistung, wie bei den aktuellen Handynetzen (GSM, UMTS).

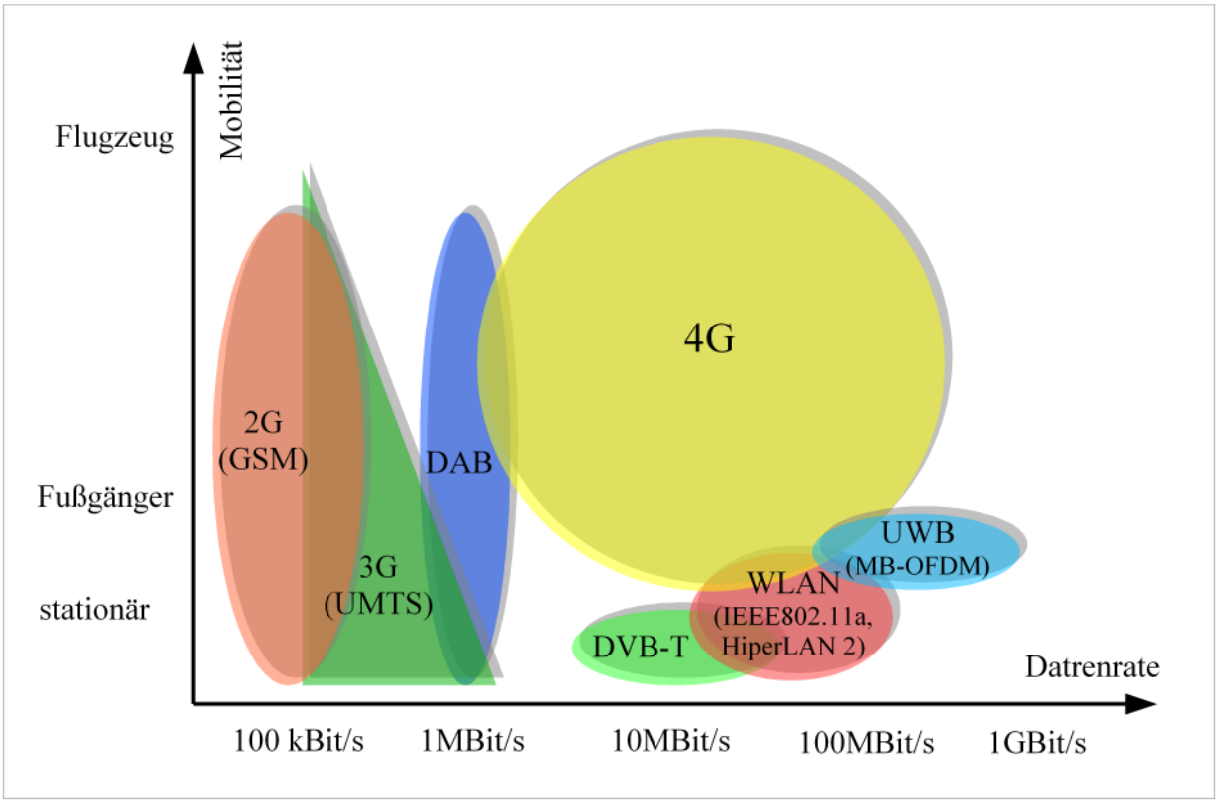


Abb. 2.14 Datenraten und Mobilität aktueller und zukünftiger Mobilfunksysteme

Aktuell werden große Hoffnungen in die Nutzung von Mehrantennensystemen gesetzt, die noch Potenzial zur Steigerung der Übertragungskapazität erwarten lassen. Grundsätzlich kommen dabei verschiedene Möglichkeiten in Betracht. Die naheliegenden Verfahren der Mehrantennenübertragung nutzen die Tatsache, dass sich zwischen den einzelnen Antennenpaaren statistisch teils bessere und teils schlechtere Kanäle einstellen. Der Gewinn wird durch eine gezielte Auswahl des besten Kanals oder durch eine gewichtete Kombination der Empfangssignale erzielt. Dies sind sogenannte *Diversity*-Verfahren [FOS96][STE03], weil sie die Verschiedenheit der Kanäle ausnutzen. Im Gegensatz zu aufwendigeren Verfahren werden hier nicht mehr Daten als bei Einantennensystemen übertragen, aber die Übertragungsqualität in Mehrwegekanälen wird erhöht. Im Übrigen funktioniert diese Technik empfangsseitig auch, wenn der Sender mit nur einer Antenne ausgestattet ist.

Ein weiterführender Ansatz bei der Nutzung von mehreren Antennen ist die Übertragung von unterschiedlichen Daten auf den einzelnen Kanälen zwischen den verschiedenen Antennen. Hierbei können entsprechend deutlich größere Datenmengen übertragen werden. Da jede Sendeantenne unterschiedliche Daten sendet, ist die Rekonstruktion der an den Empfangsantennen überlagerten Signale sehr anspruchsvoll. Diese Verfahren werden wegen der Nutzung räumlich verschiedener Einzelkanäle als *Spatial-Multiplexing* bezeichnet.

Schließlich gibt es einen dritten Ansatz zur Nutzung von mehreren Antennen, der sich weder den Diversity- noch den Spatial-Multiplexing-Verfahren zuordnen lässt. Durch eine gezielte Anordnung der Antennen in einem Array und eine phasenverschobene Ansteuerung kommt es durch die Überlagerung der Wellen zu einer stark richtungsabhängigen Abstrahlcharakteristik [KRI96]. Diese ursprünglich aus Radaranwendungen stammende Technik wird als *Beamforming* bezeichnet. Hierbei kann durch eine gerichtete Abstrahlung die Gesamtsendeleistung reduziert werden und es ist möglich, verschiedene Winkelbereiche unterschiedlich stark abzudecken. In der Regel wird die Richtwirkung durch eine gezielte Überlagerung der Einzelsignale eines Antennen-Arrays erzielt.

Einen weiteren Ansatz zur Datenübertragung ohne die Nutzung speziell zugewiesener Frequenzbänder bietet die Ultra-Wideband-Technik (UWB). Sie nutzt ein sehr breites Spektrum mit extrem geringer spektraler Sendeleistungsdichte. Das Signal kann anderen (ggf. auch lizenzierten) Signalen überlagert werden und wird auf den belegten Mobilfunkfrequenzen aufgrund der geringen spektralen Leistungsdichte allenfalls als geringes Rauschen wahrgenommen. Generell gibt es verschiedene Möglichkeiten die UWB-Technik im Einzelnen zu realisieren. Ein aussichtsreicher Kandidat ist jedoch Multiband-OFDM, welches von der Multiband OFDM Alliance[5] vorgeschlagen wurde. Geplant ist ein OFDM-System mit einer Bandbreite von mehr als 500 MHz und Übertragungsraten von bis zu 480 Mbit/s.

5 Homepage: www.multibandofdm.org

3 FPGAs – Aufbau und Eigenschaften

Im vorigen Kapitel wurde gezeigt, dass die Berechnung der Fouriertransformation einen Kernpunkt der OFDM-Signalverarbeitung darstellt. Der Einsatz dieser Übertragungstechnik ist direkt an die Möglichkeit gekoppelt, die FFT schnell und aufwandsgünstig berechnen zu können. Moderne FPGAs und digitale Signalprozessoren stellen inzwischen die dafür notwendigen Voraussetzungen bereit. Durch enorme Fortschritte in der Halbleitertechnologie werden hohe Verarbeitungsgeschwindigkeiten bei gleichzeitig geringer Leistungsaufnahme erreicht. Außerdem bieten diese Halbleiterbausteine die Voraussetzungen für eine sehr hohe Verarbeitungsgenauigkeit der digitalen Signale.

3.1 Historie der programmierbaren Logikbausteine

Die Geschichte der programmierbaren Logikbausteine (PLDs) schließt an eine Verbreitung und Nutzung von Bausteinen mit programmierbaren Logikfunktionen, den *Programmable-Array-Logic-ICs* (PALs) an. Während einfache Grundschaltungen wie Logikgatter, Flipflops oder nur lesbare Speicher (ROMs) schon verfügbar waren, bot die Technologie der PALs viel weitreichendere Möglichkeiten, diese Grundfunktionen flexibel miteinander zu verschalten. Ziel war es, Bausteine zu entwickeln, die allein durch ihre Programmierung jene Schaltungen ersetzen können, die bis dato üblicherweise als Platinen mit sehr vielen diskreten Logikbausteinen aufgebaut wurden.

PALs waren die ersten integrierten Schaltkreise, die programmierbare Logikfunktionen bereitstellen konnten. Sie beinhalteten einfache Logikelemente wie UND- und ODER-Funktionen, die jedoch über programmierbare Verbindungsleitungen vielfältig miteinander verschaltet werden konnten. Als weitere Ausbaustufe der PALs gelten die *Generic-Array-Logic-ICs (*GALs), die sich von Ersteren nur darin unterscheiden, dass sie im Gegensatz zum PAL mehrfach neu programmiert werden können.

Die sich daran anschließende Entwicklung der FPGAs und *Complex Programmable Logic Devices* (CPLDs) unterscheidet sich dann in einem wesentlichen Punkt von diesen programmierbaren Logikbausteinen: es sollen nun auch die internen Zustände ausgewertet und damit komplexe zustandsabhängige Schaltungen erzeugt werden. Dazu ist es nötig, sich

Eingangssignale oder interne Werte zu speichern, also Flipflops bzw. Register bereitzustellen. Und es erfordert eine Technologie, die internen Zusatzsignale in den Logikblock rückzukoppeln. Hierzu wird wiederum die programmierbare Verdrahtung genutzt, die schon in PALs oder GALs eingesetzt wurde.

Eine weitere Komponente leitet sich aus nur lesbaren Speichern (ROMs) als logische Schaltungselemente ab. Die Idee bei der Nutzung von Speichern zur Umsetzung von Logikfunktionen ist, dass zu jeder Kombination von Eingangssignalen, ausgewertet als Adresse des Speichers, ein beliebiges Ausgangssignal erzeugt werden kann. Mit dem Inhalt des Speichers kann so jede mögliche Logikfunktion realisiert werden.

Die Read-Only-Speicher konnten zwar als Logikschaltungen genutzt werden, waren aber bei frühen Bausteinfamilien nicht im System für verschiedene Funktionen konfigurierbar. Inzwischen gibt es modernere Versionen dieser Speicher, sogenannte PROMs, EPROMs oder EEPROMs (*Electrically Erasable Programmable ROMs*), die sich auf unterschiedliche Weise programmieren und auch überschreiben lassen.

Solche kleinen Speicherzellen bilden nach wie vor einen Kernpunkt in vielen modernen FPGA-Designs, bei denen der Logikblock durch den Einsatz von Look-Up-Tables (LUTs) implementiert wird.

Tabelle 3.1 zeigt in einem historischen Überblick die Entwicklung der jeweiligen Schaltungselemente.

Jahr	***Produkt***	***Erfinder/Hersteller***
1899	AND-Gatter	Nikola Tesla
1919	Flip Flop (*Eccles-Jordan trigger circuit)*	W. H. Eccles, F. W. Jordan
1948	Transistor	W. Shockley, J.Barden, W. Brattain
1956	PROM	Wen Tsing Chow
1977	PAL	Monolithic Memories Inc.
1980	GAL	Lattice
1983	CPLD	Altera
1985	FPGA, XC2064 (1200 Gatter)	Xilinx

Tabelle 3.1 Historie integrierter Halbleiterbauelemente zum Aufbau von Logikschaltungen

FPGAs sind also aus der Kombination von drei wesentlichen Komponenten aufgebaut: programmierbaren Logikfunktionen (häufig als LUT), Registern (Flipflops) und programmierbaren Verbindungsleitungen.

Die Herstellung von FPGAs war jedoch nicht nur von der Verknüpfung der beschriebenen Prinzipien abhängig. Vielmehr ist die Herstellung sehr eng an die Evolution der modernen Halbleiterfertigung gekoppelt. Erst mit Herstellungsprozessen zur Fertigung von Schaltungen mit sehr hohen Integrationsdichten waren die Grundlagen zum Aufbau von FPGAs gegeben. Und weil der relativ einfache strukturelle Aufbau dieser programmierbaren ICs quasi beliebig skalierbar ist, können in Zukunft wohl immer größere und schnellere FPGAs hergestellt werden. Dies geschieht praktisch im Gleichschritt mit den Fortschritten der Halbleiterfertigung.

3.2 Prinzipieller Aufbau von FPGAs

Der kleinste funktionale Grundbaustein im FPGA ist eine logische Zelle (LC). Sehr viele davon sind auf einem Chip in einem Feld (engl. *Array*) regelmäßig angeordnet. Durchzogen wird dieses LC-Feld von einem Netzwerk programmierbarer Verbindungen, das die Verschaltung zu komplexen Funktionseinheiten ermöglicht. Umgeben ist das Schaltungsfeld von I/O-Blöcken, die den Anschluss an externe Baugruppen bzw. die äußere Platine ermöglichen.

Eine logische Zelle[1] besteht aus einem Look-Up-Table-Speicher mit vier Eingängen und einem Flipflop. Damit auch logische Funktionen mit mehr als vier Eingangsbits realisiert werden können, gibt es die Möglichkeit, mehrere LUTs zusammenzuschalten. Die Flipflops können in Verbindung mit den LUTs oder auch einzeln als Register genutzt werden, wenn keine kombinatorische Auswertung des Eingangssignals notwendig ist. Moderne FPGAs lassen sich sehr flexibel strukturieren und ermöglichen so eine effiziente Nutzung der Ressourcen.

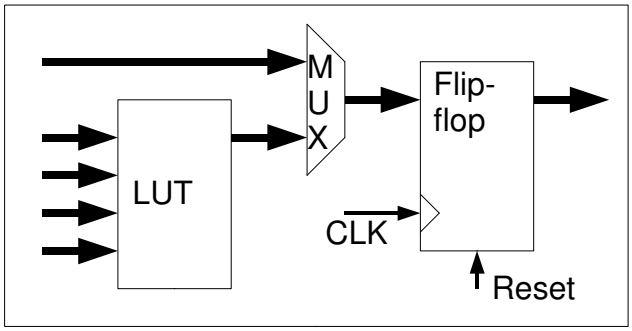


Abb. 3.1 Logische Zelle eines FPGAs

Die LCs sind je nach Hersteller und Baureihe etwas unterschiedlich aufgebaut. So kann z.B. die Anzahl der Eingänge in die LUTs variieren. Außerdem gibt es häufig zusätzliche Multiplexer und Querverbindungen zwischen den Zellen, die beispielsweise den Aufbau von schnellen Addierern durch eine zusätzliche Weitergabe von Carrybits ermöglichen [AUE01]. Diese Details sind jedoch baureihenspezifisch und daher schwer allgemein abzuhandeln. Die Hersteller fassen häufig mehrere logische Zellen zu kleinen Einheiten zusammen, die dann einen gesonderten Namen bekommen. So heißt z.B. bei Xilinx die Kombination aus zwei LCs ein SLICE, vier SLICEs ergeben wiederum ein CLB (*Configurable Logic Block*). Ein sinnvoller Vergleich von FPGAs unterschiedlicher Hersteller ist am zuverlässigsten anhand der Zahl der logischen Zellen möglich, weil sich diese in sehr ähnlicher Form in allen FPGAs wiederfinden. Eine die Gegenüberstellung von Gatterzahlen, die

1 Dies bezieht sich beispielhaft auf die Virtex-II Serie von Xilinx, gilt aber für eine Vielzahl von FPGAs.

von den Produzenten häufig beworben werden, ist wenig geeignet, da die Hersteller häufig unterschiedliche Definitionen für ein Gatter in den Halbleiterstrukturen wählen.

Neben der möglichen Anzahl der LCs ist auch das Netzwerk der programmierbaren Verdrahtungen abhängig von der Halbleiterfertigung. Je vielschichtiger ein Halbleiter aufgebaut werden kann, desto mehr Ebenen für Leiterbahnen stehen zur Verfügung. Ein aktueller Baustein[2] wird in einem Acht-Lagen-Halbleiterprozess gefertigt. Er stellt spezielle und exklusive Netzwerke zur Verteilung von Takt- oder Reset-Signalen bereit. Insbesondere die Taktleitungen in FPGAs müssen so konstruiert sein, dass alle Flipflops exakt synchron auf die gleiche Taktflanke schalten können.

Ein den FPGAs sehr ähnliches Anwendungsgebiet decken CPLDs ab. Sie unterscheiden sich von FPGAs dadurch, dass sie nur eine Spalte bzw. Matrix mit programmierbarer Logik und eine Reihe von nachgeschalteten Flipflops enthalten. In Abb. 3.2 ist der prinzipielle Aufbau dieser PLDs schematisch dargestellt. Die Logik des CPLD ist eine zweistufige UND/ODER-Matrix, die auch *Field Programmable Logik Array* (FPLA) genannt wird. Der große Vorteil dieses zweistufigen Aufbaus ist eine feste Durchlaufzeit. Diese bzw. der maximale Takt ist nur vom eigentlichen Chip abhängig, nicht jedoch von dessen Programmierung. Darüber hinaus werden die meisten CPLDs direkt nichtflüchtig programmiert, während FPGAs das Programm beim Start häufig aus externen Speichern ins eigene flüchtige SRAM laden.

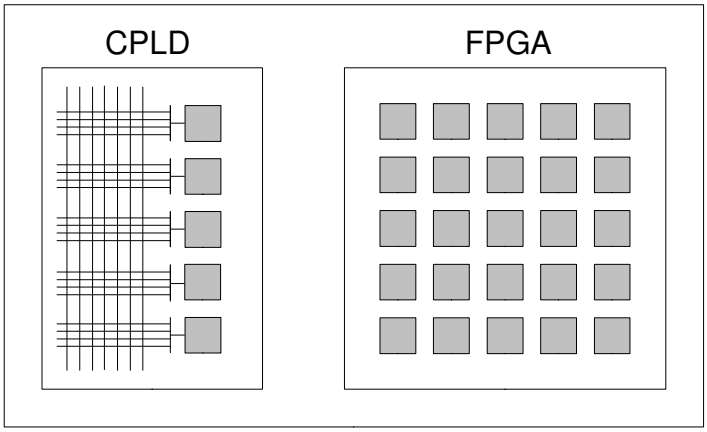


Abb. 3.2 Architektur von CPLD und FPGA

Der Nachteil von CPLDs besteht darin, dass die Struktur nur bis zu einer bestimmten Komplexität geeignet ist. Größere Schaltungsdesigns erfordern mehr Logik und mehr Flipflops, als sich in dieser zweireihigen Struktur aufbauen lässt.

Da CPLDs gegenüber FPGAs eine andere, PAL-ähnliche Logikmatrix nutzen, können sie bei der Implementierung von Anwendungen mit viel kombinatorischer Logik und wenigen Registern Vorteile bieten. Bei dem Aufbau von komplexen Signalverarbeitungsaufgaben sind jedoch in der Regel viele Register nötig. Insbesondere bei einer mehrstufigen Verarbeitung sind FPGAs geeigneter.

2 Diese Angabe bezieht sich auf die Xilinx Virtex-II Serie.

3.3 Besonderheiten der Xilinx Virtex-II Architektur

Die im Rahmen dieser Arbeit aufgebaute Experimentalhardware nutzt ein modernes FPGA der Xilinx Virtex-II-Serie. Diese Bausteine bieten sehr große Schaltungsarrays und können komplexe Signalverarbeitungsaufgaben übernehmen. Sie stellen neben der Grundstruktur noch einige zusätzliche Elemente bereit, welche die Leistungsfähigkeit für viele Einsatzbereiche deutlich erhöhen. Drei dieser Komponenten mit speziellen Funktionen werden im folgenden Abschnitt kurz vorgestellt: *Dual-Port Block RAMs* (DPRAM), Multiplizierer und *Digital-Clock-Manager* (DCM).

Die Dual-Port Block RAMs sind zusätzliche, jeweils 18 Kilobit große Speicher. Sie können entsprechend große Mengen an Daten vorhalten, ohne dass dafür die Flipflops der logischen Zellen belegt werden müssen. Die DPRAMs lassen sich manuell oder automatisch in das Schaltungsdesign integrieren und können über zwei unabhängige Schnittstellen (Dual-Port) angesprochen werden. Die Breite der gespeicherten Datenwörter kann zwischen 1, 2, 4, 9, 18 und 36 Bit gewählt werden.

Die FPGAs der Firma Xilinx enthalten ab der Virtex-II-Serie integrierte 18*18-Bit-Multiplizierer. Sie tragen der Erfahrung Rechnung, dass in der digitalen Signalverarbeitung sehr häufig Multiplikationen gebraucht werden. Diese Hardware-Multiplizierer arbeiten deutlich schneller als solche, die aus logischen Zellen, also der Standard-FPGA-Struktur, aufgebaut werden. Außerdem verkleinern sie wegen ihres funktionsoptimierten Konzepts die Schaltungsdesigns.

Die in der Virtex-II-Architektur enthaltenen *Digital-Clock-Manager* bieten ein sehr anpassungsfähiges Taktsignal-Management. Aus einem Eingangstakt können mit Hilfe einer rückgekoppelten Regelschleife verschiedene Ausgangstakte generiert werden. Das ursprüngliche Signal wird dabei in der Frequenz vervielfacht, geteilt oder phasenverschoben. Die so erzeugten Takte können innerhalb und außerhalb der Chips genutzt werden. Dadurch wird sichergestellt, dass auch externe Baugruppen, wie z.B. A/D-Wandler, taktsynchron angesteuert werden können.

Neben den speziellen Hardwareelementen zeigen sich die FPGAs der Virtex-II Serie in weiteren Punkten sehr flexibel. Die I/O-Blöcke, also die Treibereinheiten direkt an den Pins des FPGAs, erlauben die Ein- und Auskopplung von elektrischen Signalen verschiedener Pegel. Insgesamt können 19 asymmetrische und sechs differentielle Standards wie z.B. *Low Voltage TTL* oder *Low Voltage Differential Signaling* (LVDS) genutzt werden.

Die LUTs der einzelnen logischen Zellen können beim Virtex-II außer als Logikelement noch als Schieberegister konfiguriert werden. Diese können eine maximale Tiefe von 16 Bit[3] haben, was ohne die spezielle Nutzung eines LUT sechzehn Flipflops benötigen würde. Wenn Datenströme innerhalb des FPGA verzögert oder synchronisiert werden müssen, kann diese Konfiguration zu einer großen Platzersparnis führen.

Neben diesen beschriebenen Möglichkeiten bieten moderne FPGAs zahlreiche Detaillösungen zur Optimierung der Schaltungsstrukturen. Viele Einzelheiten werden durch die Entwicklungs-

3 Die LUT speichern bis zu 16 Bit, wobei die vier Eingänge eine Adresse des ein Bit breiten Speichers bilden.

software automatisch integriert, ohne dass der Anwender dies explizit programmieren muss. Eine Beschreibung der vielfältigen Möglichkeiten von Virtex-II-Bausteinen findet sich im zugehörigen Datenblatt [XIL00].

Sollen die oben beschriebenen chipspezifischen Schaltungselemente gezielt eingebunden werden, bietet die Programmierumgebung verschiedene Verfahren, die gewünschten Komponenten in die eigenen Designs einzubinden. Häufig werden dazu vorgegebene Programmabschnitte in den eigenen Source-Code eingebunden. Der Nachteil einer solchen „erzwungenen“ Nutzung von chipspezifischen Schaltungsteilen ist das Erschweren der Migration auf andere FPGA- oder CPLD-Baureihen, was prinzipiell bei den programmierten Designs ohne großen Aufwand möglich sein sollte.

3.4 Programmierung von Schaltungsstrukturen in VHDL

Die ersten computergestützten Entwicklungswerkzeuge für programmierbare Logik basierten auf einer grafischen Schaltplaneingabe (engl. *Schematics*). Damit wurde sehr ähnlich wie mit diskreten Bauteilen bei herkömmlichen Platinendesigns eine Baugruppe entworfen. Diese Tools waren besonders hardwarenah und verwalten direkt die Elemente der Chips, in der Regel die im Baustein vorhandenen Baugruppen wie Flipflops, Multiplexer, AND-Gatter usw. Eine Erweiterung hin zu einem hierarchischen Aufbau ermöglichte dabei das Einfügen komplexerer Funktionseinheiten als eigene Symbole. Ab einer gewissen Komplexität ist dieses Vorgehen aber nicht mehr effizient. Große grafische Designs werden schwer überschaubar, auch wenn mit Hilfe von Funktionsblöcken und der Nutzung von mehreren Hierarchieebenen einigermaßen modular programmiert wird.

Vielmehr haben sich textbasierte Entwicklungswerkzeuge auf breiter Front durchgesetzt, ähnlich wie bei der Programmierung von Computern, Mikrocontrollern und DSPs. Die für CPLDs und FPGAs verwendeten Programmiersprachen werden als Hardwarebeschreibungssprachen (*Hardware Description Language*, HDL) bezeichnet. Zunächst entwickelten viele Hersteller eigene HDLs , wie z.B. ABEL, PALASM oder MACH XL. Inzwischen haben jedoch zwei Dialekte die bei Weitem größte Bedeutung: VHDL und Verilog.

Welche dieser beiden Sprachen genutzt wird, hängt in der Regel mehr von persönlichen Vorlieben oder der Verfügbarkeit ab, als dass es sich durch technische Unterschiede begründen ließe. Markus Wannemacher schreibt in [WAN98], Verilog sei eher an die Programmiersprache C angelehnt, während VHDL mehr an ABEL oder Pascal erinnert. Die in dieser Arbeit betrachteten FPGA-Funktionen sind in VHDL programmiert. Diese Sprache wird zurzeit in Europa am verbreitetsten zur Hardware-Programmierung genutzt. Die grundsätzlichen Aussagen zur Optimierung von FPGA-Designs sind jedoch unabhängig von dem verwendeten Dialekt und lassen sich leicht auf andere HDLs übertragen.

VHDL steht für ***V****ery High Speed Integrated Circuit* ***H****ardware* ***D****escription* ***L****anguage* und wurde im Auftrag der US-Regierung in einem Halbleiterprojekt entwickelt. Seit 1987 ist die Sprache standardisiert und bis heute sind viele Teile erweitert und überarbeitet worden. Der große Vorteil von VHDL liegt in der breiten Einsetzbarkeit. So kann nicht nur das Schaltungsdesign darin beschrieben werden. Auch die Spezifikation, Dokumentation, Simulation und Synthese kann bei Bedarf VHDL-basiert erfolgen.

Um ein FPGA in einem solchen Design-Flow zu programmieren, wird zunächst das gewünschte Verhalten der Schaltung in VHDL beschrieben. Hinzu kommt die Definition der äußeren Schnittstellen, also welche Ein- und Ausgangssignale genutzt werden sollen. Das Programm benötigt keinerlei Informationen darüber, wie die interne Schaltung aufzubauen ist. Ein VHDL-Programm ist von der konkret verwendeten Zielhardware unabhängig[4]. Bei größeren Designs ist es unumgänglich, hierarchisch zu programmieren, also einzelne Funktionen in Unterprogramme auszulagern und evtl. auch einzeln zu prüfen. Diese Unterprogramme (engl. *Components*) sind eigenständige VHDL-Programme, die in der nächsthöheren Ebene über ihre definierten Schnittstellen eingebunden werden.

In einem speziellen Hardwareaufbau sind zusätzlich noch Informationen nötig, die den Anschluss der umgebenden (externen) Komponenten beschreiben. Diese Daten werden in einer eigenen Datei für Nebenbedingungen (engl. *Constraint File*) abgelegt. Hier werden die meisten hardwareabhängigen Parameter kontrolliert, wie z.B. die genutzten I/O-Standards, erforderliche Taktfrequenzen und eine Anschlusstabelle für die Signale an die PINs des Chips.

4 Solange keine chipspezifischen Komponenten eingebunden werden (vgl. Kapitel 3.3)

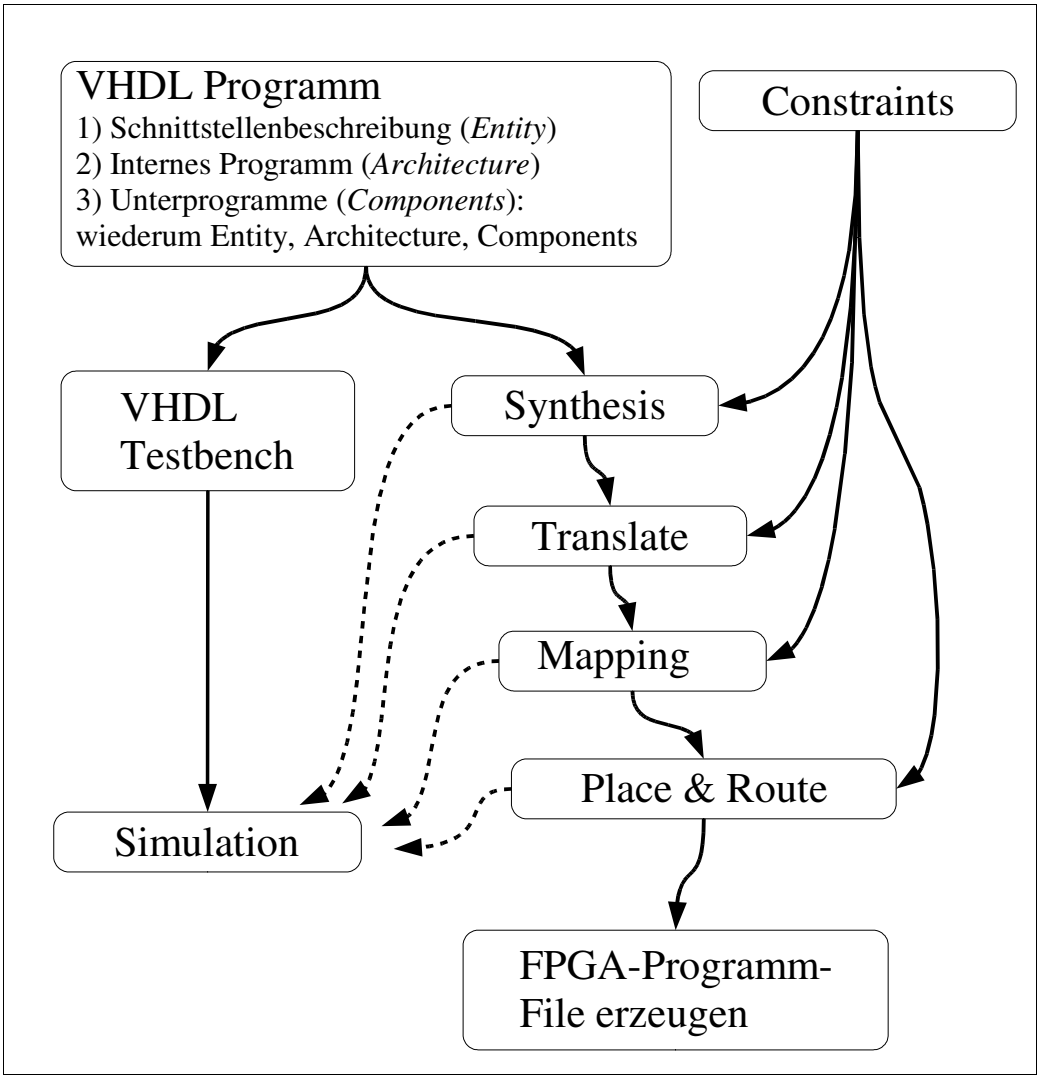


Abb. 3.3 VHDL-Design-Fluss

Die VHDL-Programme und das Constraint File bieten alle notwendigen Nutzerinformationen, die das Design in einer modernen Entwicklungsumgebung zum Schaltungsentwurf benötigt. Die in Abb. 3.3 dargestellten Schritte wie *Synthesis, Translate, Mapping, Place & Route* und das Erzeugen der programmierfähigen Datei können völlig automatisch durchlaufen werden. Bei größeren Projekten ist es jedoch unausweichlich, die Funktionalität der Programme während des Entwicklungszyklus zu testen. Dies erfolgt mit Hilfe von Simulationen.

Um eine Schaltung zu simulieren, muss sie von außen möglichst realistisch angeregt werden. Anschließend wird das Verhalten der Ausgangssignale beobachtet. Die Anregung von außen wird auch in einer VHDL-Datei beschrieben (engl. *Testbench*). Die Verwendung von VHDL-Testbenches erspart das Erlernen einer zusätzlichen Beschreibungssprache für die Durchführung von Simulationen. Die Möglichkeiten reichen von einfachen Tests bis hin zu umfangreichen

automatischen Simulationen mit Zugriff auf Messwerte oder mit Einbindung von Kanalmodellen. Es ist darüber hinaus häufig hilfreich, auch innere Zustände der Schaltung zu betrachten, um Fehler besser lokalisieren zu können. Bei einer Simulation ist das Verfolgen der internen Verarbeitung sehr gut möglich, während das fertige Chip-Design nur durch die Ein- und Ausgaben geprüft werden kann. Dabei ist es wichtig, das Verhalten der Programme in allen Phasen des Design-Zyklus zu testen, weil z.B. erst nach dem *Translate*-Prozess das zeitliche Verhalten der Schaltung berücksichtigt werden kann. Später im Design-Flow können noch Annahmen über die Platzierung der Schaltungselemente und die Längen der Verdrahtungen mit einbezogen werden. Je später im Implementierungsprozess simuliert wird, desto realistischer ist das Ergebnis. Leider erhöht sich dabei auch der Rechenaufwand sehr stark, weil immer mehr Details berücksichtigt werden müssen.

VHDL basiert auf einer sehr übersichtlichen Anzahl von Befehlen. Somit sind die Sprache und die Syntax recht schnell zu erlernen [LEH94][REI03]. Dennoch ist der Einsatz einer solchen abstrakten Programmiersprache nicht ohne Nachteil. Leider ist es für leistungsfähige Designs unumgänglich, auch zu wissen, wie bestimmte VHDL-Konstrukte auf die Hardware abgebildet werden. Dies wird jedoch in entsprechenden Lehrbüchern wenig behandelt, sodass ein VHDL-Entwickler hier auf selbst gesammelte Erfahrung zurückgreifen muss. Wann wird aus einem Schaltungsteil ein Flipflop oder ein Latch, wie wird eine S*tate Machine* implementiert usw.? Hilfe bieten hier eventuell Guidelines [SYN99][XIL06], die versuchen, den Stil der VHDL-Programmierung zu bewerten. Ansonsten hilft nur die Kontrolle mit Hilfe des Entwicklungswerkzeugs, ob die resultierende Schaltung sowohl in Aufbau als auch in der Funktion den Vorgaben entspricht.

Zu einer VHDL-Umgebung gehört auch eine wachsende Anzahl an *Packages* und *Libraries*, die bestimmte Deklarationen von Datentypen definieren. Hier ist es ratsam, auf die weitverbreiteten standardisierten Varianten zurückzugreifen. So sind z.B. in *IEEE.std_logic_1164.all* oder *IEEE.std_logic_arith.all* Datentypen wie vorzeichenbehaftete Zahlen und die resultierenden Rechenvorschriften definiert. Prinzipiell können *Packages* aber auch selbst angelegt werden.

3.5 Vorteile FPGA-basierter Hardware

Hardwareaufbauten in Forschung und Entwicklung können durch die Nutzung von programmierbaren Logikbausteinen sehr universell entworfen werden. Die hohe Flexibilität dieser Technik bietet vielfältige Möglichkeiten zur Weiterentwicklung und/oder zur Fehlerkorrektur. Darüber hinaus besteht die Chance, neue Verfahren oder Algorithmen in „echten“ Systemen, die also z.B. WLAN mit voller Datenrate bieten, auszuprobieren. Die Vor- und Nachteile einer FPGA-basierten Lösung werden im Vergleich zu anderen Technologien deutlich. Die wichtigsten alternativen Lösungen in der Signalverarbeitung sind Digitale-Signal-Prozessoren (DSPs) und anwendungsspezifische integrierte Schaltungen (ASICs).

DSPs sind heutzutage sehr weit entwickelt und extrem leistungsfähig. Hohe Taktraten und viele parallel zu nutzende Recheneinheiten führen zu großen Datendurchsätzen bei der digitalen Signalverarbeitung. FPGAs erreichen nicht so hohe Taktraten wie moderne DSPs. Sie haben aber den Vorteil, dass alle Operationen in Hardwarestrukturen abgebildet werden und somit prinzipiell parallel ausgeführt werden können. Dabei sind die Wortbreiten nicht vorgegeben, sodass Einheiten, die nur einzelne Bits verarbeiten, auch nur eine Datenleitung nutzen und nicht wie beim DSP den gesamten 16 oder 32 Bit breiten Bus. Gleiches gilt für Speicherzellen, die bei DSPs auch in der Regel die volle Wortbreite aufweisen. Ein Adresswerk zum Zugriff auf die Daten entfällt bei paralleler Abarbeitung im FPGA völlig. Da alle Verarbeitungsschritte in einer solchen programmierten Schaltungsstruktur gleichzeitig ablaufen, häufig sogar mit einem einheitlichen Takt, ist es sehr viel einfacher, Echtzeitanforderungen zu erfüllen. Der Hardwareaufbau und die Verifikation sind bei DSP-Platinen oft komplizierter, weil die Anschlusspins mit festen Funktionen belegt sind und nicht wie beim FPGA frei zuweisbar bleiben. Auch eine Fehlersuche oder -korrektur im DSP-Design ist damit erheblich aufwendiger.

Da die Schaltung (nicht die Programmierung) innerhalb des DSPs fest vorgegeben ist, besteht hier jedoch der Vorteil, dass diese getestet ist und fehlerfrei arbeitet. In einem FPGA muss die implementierte Schaltung nach jeder Programmierung auf richtige Funktionsweise geprüft werden. Hier können neben Fehlern in der Firmware auch noch Unzulänglichkeiten in der Schaltungsumsetzung auftreten. Hilfreich ist dabei ist die prinzipielle Möglichkeit, dass beliebige Signale zu Testzwecken durch ein Anpassen der Programmierung aus dem Inneren der Verarbeitungskette auf Pins nach außen (Stecker, Debug-Pins) geführt werden können.

Ein Kostenvorteil der einen oder anderen Realisierungsmöglichkeit ist inzwischen kaum festzustellen. Beide Chipformen sind in unterschiedlichen Leistungsklassen zu bekommen und auch entsprechend günstig oder kostspielig. Auch die Programmierung und die Entwicklungstools sind auf ähnlichem Niveau.

DSPs haben Vorteile, wenn zu unterschiedlichen Zeiten verschiedene Programme ausgeführt werden. Hier müssten FPGAs rekonfiguriert oder wenigstens zum Teil umprogrammiert werden (engl. *partial reconfiguration*), was sich zurzeit bei FPGAs noch im Entwicklungsstadium befindet. Wenn sehr hohe Genauigkeiten gefordert sind, insbesondere bei Fließkommaarithmetik, sind Signalprozessoren leichter zu handhaben. Die Verarbeitung von Fließkommazahlen wird in den Entwicklungsumgebungen für programmierbare Logik bisher kaum unterstützt.

FPGAs sind somit nicht immer die bessere Lösung. In einigen Fällen können verschiedene Lösungen ähnlich gute Ergebnisse liefern. Erst eine genaue Betrachtung der Datenverarbeitung zeigt die Vor- oder Nachteile der gewählten Technologie. Inzwischen können auch vollständige Prozessoren in leistungsfähigen FPGAs nachgebildet werden oder sind sogar als vorgefertigte Schaltungsblöcke (engl. *IP-Cores*) verfügbar.

ASICs spielen als weitere Alternative nur in professionellen Aufbauten mit hohen zu erwartenden Stückzahlen eine Rolle. Deren Schaltungsentwicklung ist in großen Teilen mit der FPGA-Programmierung identisch. Die entworfene Schaltung wird nach dem Designprozess, häufig auch in den Hardwarebeschreibungssprachen VHDL und Verilog, direkt als Halbleiter gefertigt und ist dadurch höher zu takten, platzsparender und verbraucht weniger Leistung. Es werden im Halbleiter nur die Strukturen aufgebaut, die gerade benötigt werden. Wortbreiten, Speicher und Datenbusse sind frei skalierbar und platzierbar. Die Schaltung wird nicht wie beim FPGA aus vorgegebenen Standardzellen, den LCs, aufgebaut.

Der Zeitaufwand, die Kosten und das Risiko von schwer korrigierbaren Fehlern eines Schaltungsentwurfs auf ASIC Basis sind jedoch ungleich höher als bei FPGA Prototypen. Nachdem ein Design fertig entwickelt wurde, dauert es wegen der aufwendigen Fertigung noch mehrere Wochen oder Monate, bis ein ASIC zur Verfügung steht. Die einmalig notwendige Erstellung von Belichtungsmasken sowie die Vielzahl der Prozessschritte in der Halbleiterproduktion erfordern viel Zeit und verursachen sehr hohe Grundkosten. Somit ist die Fertigung eines solchen Chips nur für höhere Stückzahlen wirtschaftlich sinnvoll.

Die größten Nachteile eines ASICs ergeben sich jedoch aus der Tatsache, dass ein Schaltungsdesign im Nachhinein kaum mehr veränderbar ist. Die Fehlersuche ist sehr schwer, weil es in der Regel keinen Zugriff auf die inneren Zustände gibt. Werden gravierende Fehler in der Funktion des Bauteils gefunden, ist eine kostspielige und langwierige neue Fertigung unumgänglich. FPGAs können bei Fehlern einfach neu programmiert werden. Eine Verbesserung bzw. Erweiterung der Funktionen ist jederzeit möglich (Updatefähigkeit).

Neben den speziellen Eigenschaften von verschiedenen Baugruppen ist jedoch der Aufwand von dem Entwurf einer technischen Anwendung bis hin zum fertigen Produkt ein wichtiges Kriterium.

Im klassischen Fall erfolgt diese Arbeit bei vorhandenem Konzept in drei Phasen: eine simulative Umsetzung auf Computern, die Konstruktion von Prototypen und anschließend eine Produktentwicklung. Die Simulationsumgebung dient zur Verifikation der nötigen Algorithmen und prüft deren Funktion. Ein Prototyp testet die praktische Umsetzbarkeit und bildet dieses System dafür in geeigneter Hardware nach. Um daraus nun ein wettbewerbsfähiges Produkt zu kreieren, muss der Prototyp noch hinsichtlich der Größe, Kosten und Verfügbarkeit der Komponenten optimiert werden.

Im Vergleich dazu verfolgt ein FPGA-basierter Entwicklungszyklus den Ansatz, diese drei Phasen zu verschmelzen und so eine schnellere und günstigere Umsetzung zu ermöglichen.

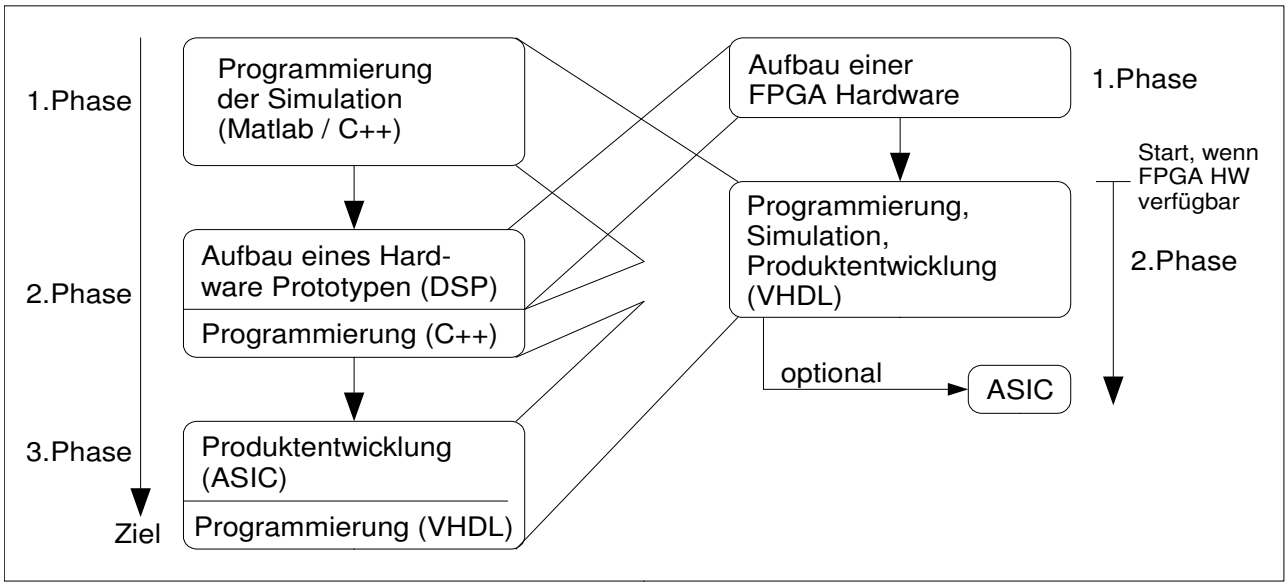


Abb. 3.4 Vergleich eines konventionellen und eines FPGA-Designflows

Eine FPGA-Plattform bietet nun die herausragende Eigenschaft, den gesamten Entwicklungszyklus auf Basis derselben Programmierung durchzuführen. Alle Elemente werden z.B. in VHDL oder Verilog programmiert und können so schon ausführlich am PC simuliert und getestet werden. Anschließend werden diese Programme für die FPGAs übersetzt und können so auf einem FPGA basierten Prototypen ausgeführt werden. Häufig ist die FPGA-Lösung schon klein und günstig genug, um sie im späteren Produkt einzusetzen. Soll jedoch z.B. bei zu erwartenden extrem großen Stückzahlen dennoch ein ASIC entworfen werden, ist dies inzwischen auf Basis von FPGA-Programmierungen ein sehr einfacher standardisierter Prozess.

Weil FPGAs keine vordefinierten Schnittstellen haben und alle Schaltungsstrukturen erst durch die Programmierung entstehen, ist es oft möglich, den gleichen Hardwareaufbau für verschiedene Aufgaben zu nutzen. Die Flexibilität ist größer als bei DSP oder ASIC Designs. Dadurch ist es leichter möglich, vorhandene oder kommerziell verfügbare FPGA-Platinen einzusetzen und so den Aufbau eigener Hardware zu vermeiden. Dies führt zu einer großen Zeit- und Kostenersparnis.

Abgesehen von den beschriebenen Vorteilen sind auch bei der Programmierung von FPGAs einige Regeln einzuhalten. Die Programme dürfen nicht die Kapazität der Baugruppen übersteigen. Außerdem sind im FPGA die Funktionseinheiten, Speichergrößen, Laufzeiten oder maximale Geschwindigkeiten[5] vorgegeben. Inzwischen ist die Leistungsfähigkeit moderner Chips jedoch insgesamt sehr hoch und die Bausteine sind in verschiedenen Leistungsklassen verfügbar.

In Forschung und Entwicklung sind programmierbare Baugruppen unbedingt empfehlenswert. Die zu erwartende Entwicklungszeit ist kürzer und das Risiko eines notwendigen Redesigns ist

5 Gemeint sind hier die Laufzeiten und maximale Geschwindigkeiten der diskreten Schaltungsteile im FPGA. Die Laufzeiten des programmierten Design ergibt sich erst aus dem generierten Schaltungsaufbau.

deutlich geringer als bei ASICs. Darüber hinaus bieten die großen Anbieter von FPGAs inzwischen standardisierte Verfahren an, um FPGA-Designs in ASICs zu übertragen. Damit können dann auch große Volumina aus einem FPGA basierten Entwicklungszyklus heraus kostengünstig produziert werden.

3.6 Optimierungsmöglichkeiten im FPGA

Die wichtigsten Maßstäbe beim Lösen von Aufgaben der digitalen Signalverarbeitung in einem FPGA sind die maximale Verarbeitungsgeschwindigkeit und der interne Schaltungsaufwand, der die notwendige Komplexität des einzusetzenden Bauteils bestimmt. Generell ist es immer von Vorteil, möglichst kleine (vereinfachte) Schaltungsstrukturen in FPGAs zu implementieren, weil sie neben den gesparten Ressourcen i. d. R. auch höhere Taktraten als komplexe Logikkonstrukte ermöglichen. Daher wird im Rahmen dieser Arbeit jeder verwendete Algorithmus dahingehend untersucht, wie weit sich die einzelnen Schaltungsteile verkleinern lassen. Oft geht diese Verkleinerung einher mit einer Verringerung der Verarbeitungsgenauigkeit und erfordert somit eine genaue Abwägung zwischen Schaltungsaufwand und Verarbeitungsqualität. Solche Optimierungsmaßnahmen sind anwendungs- bzw. algorithmusspezifisch, d.h., sie sind abhängig von den jeweiligen Funktionen oder Verarbeitungsschritten.

Auf der anderen Seite gibt es Möglichkeiten, einen auch exakt definierten Verarbeitungsteil bzw. Algorithmus in einem FPGA besonders effizient zu implementieren. Dabei kann häufig zwischen hoher Verarbeitungsgeschwindigkeit und minimalem Ressourcenverbrauch abgewogen werden. In einigen Fällen ist darüber hinaus eine Mehrfachnutzung von Grundschaltungen möglich, sodass z.B. ein aufwendiger Dividierer verschiedenen Modulen (zwingend zu unterschiedlichen Zeiten) zur Verfügung gestellt werden kann. Optimierungsansätze dieser Art sind unabhängig von dem programmierten Algorithmus und somit schaltungsspezifisch.

In diesem Kapitel werden diese Möglichkeiten zur Optimierung sortiert nach anwendungs- und schaltungsspezifischen Ansätzen aufgelistet. Hier werden zunächst die Optimierungsansätze im Allgemeinen, also einige generelle Tricks für FPGA-Designs, erläutert. Die folgende Liste gibt zunächst einen Überblick über anwendungs- bzw. rechenspezifische Optimierungsmöglichkeiten:

- Anpassung der Verarbeitungsgenauigkeit,
- Minimieren der Rechenschritte von Algorithmen,
- Rechenschritte mit kleinen Wortbreiten zuerst ausführen,
- Vergleichsoperationen vereinfachen,
- Bitshiftoperationen nutzen (anstelle von Multiplikationen oder Divisionen),
- Rechnungen mit Konstanten bevorzugen.

Diese Gruppe von Maßnahmen ist unabhängig von hardwarespezifischen Ausstattungselementen innerhalb des FPGAs. Das Festlegen der minimal benötigten Rechengenauigkeit oder das Verwenden von gleichen Funktionen in unterschiedlichen Schaltungsteilen, die zu unterschiedlichen Zeiten genutzt werden, sind Beispiele dafür. Das Minimieren von Rechenschritten kann ein Design unter Umständen erheblich schneller machen. So wird z.B. bei einer Korrelationsberechnung ein zusammenhängender Datenblock aufsummiert. Im nächsten Takt wird wiederum eine Summe benötigt, bei der der Datenblock um einen Schritt „weitergerückt“ ist. Hier kann in aller Regel viel schneller die Vorgängersumme verändert werden, wobei der neue Wert hinzuaddiert und der herausfallende Wert subtrahiert wird, anstatt die gesamte Summe neu zu berechnen. Diese Art der Berechnung ist weit verbreitet. Es ist jedoch ein gutes Beispiel dafür, wie die gleiche Formel unterschiedlich aufwendig implementiert werden kann. Auch die Umsetzung der DFT als FFT ist eine solche verarbeitungsangepasste Funktion. In Kapitel 5 werden die Anpassungen der einzelnen OFDM-Module aufgezeigt und gegebenenfalls zur Performance-Steigerung genutzt.

In der digitalen Signalverarbeitung innerhalb eines FPGAs kann es durchaus sinnvoll sein, verschiedene Prozesse mit unterschiedlichen Wortbreiten auszuführen. Wenn nun mit Daten variierender Breite weitergerechnet werden soll, ist es sinnvoll, die Rechenschritte mit kleinen Wortbreiten zuerst auszuführen.

Das folgende einfache Beispiel soll diesen Aspekt verdeutlichen: Drei Zahlen sollen addiert werden. *Zahl1* sei 6 Bit breit, *Zahl2* und *Zahl3* jeweils 3 Bit breit. Das *Ergebnis* = *Zahl1* + *Zahl2* + *Zahl3* wurde im betrachteten Fall mit zwei 7-Bit-Addierern aufgebaut und erlaubte einen maximalen Takt von ca. 100 MHz. Folgende Änderung hat das Ergebnis signifikant beeinflusst: *Ergebnis* = *Zahl2* + *Zahl3* + *Zahl1* wurde im betrachteten Fall mit einem 4-Bit-Addierer und einem 7-Bit-Addierer aufgebaut und erlaubte einen maximalen Takt von ca. 130 MHz.

Dieses Beispiel zeigt, dass es möglich ist, die gleiche Rechnung unterschiedlich effizient zu implementieren. Hier hängt der Schaltungsaufbau sehr stark von der Leistungsfähigkeit der Entwicklungsumgebung ab. Moderne Software kann die oben gezeigte Optimierung unter Umständen eigenständig ausführen.

Auch bei Vergleichsoperationen kann die Programmierung eine sehr einfache Abbildung in Hardware ermöglichen. Immer dann, wenn es erlaubt ist, Wertevergleiche an Schwellen von Zweierpotenzen auszuführen, kann das Logiknetzwerk sehr klein und einfach aufgebaut werden. Diese Operationen können dann allein auf den höherwertigen Bits ausgeführt werden, während bei anderen Vergleichen zusätzliche Bits betrachtet werden müssen.

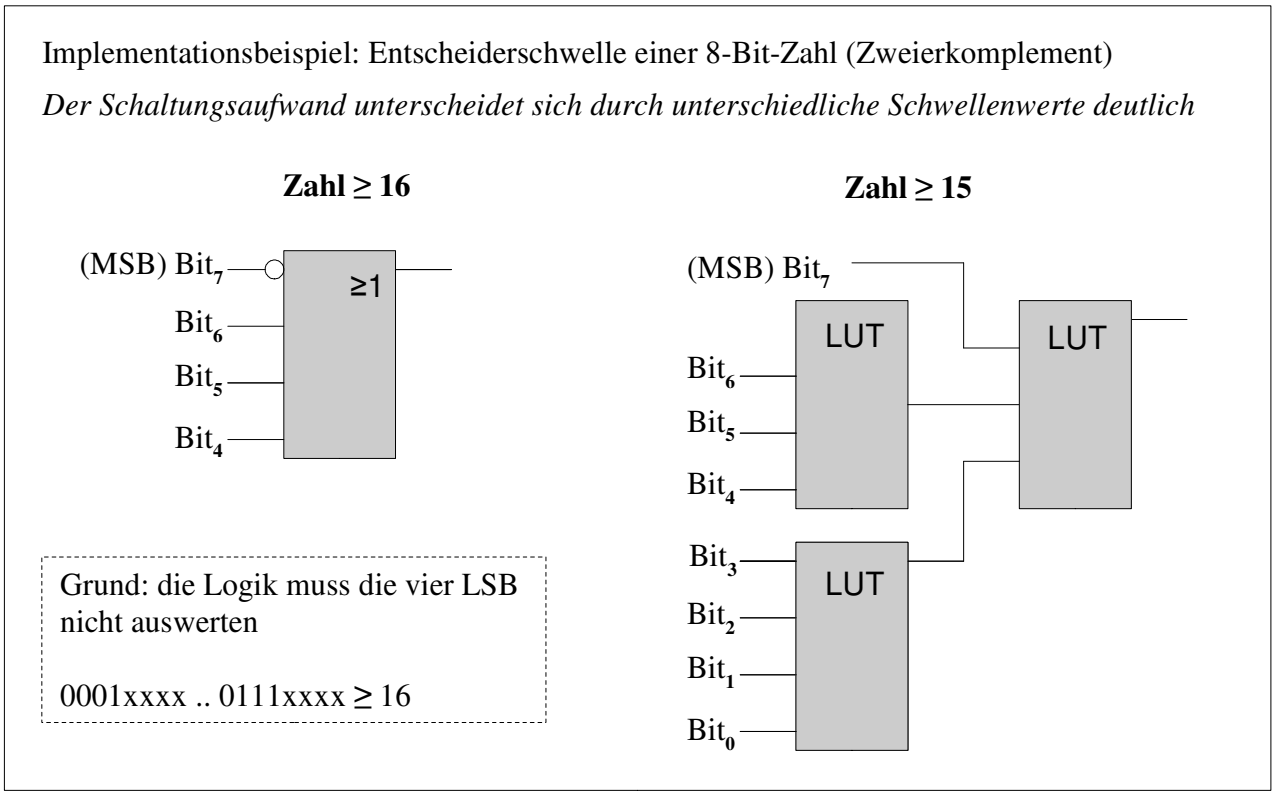


Abb. 3.5 Das Beispiel einer anwendungsspezifischen Optimierung anhand einer Größer-Gleich-Logik. Diese Variation ist nur möglich, wenn die Vergleichsschwelle in gewissem Rahmen frei gewählt und daher entsprechend günstig festgelegt werden kann.

Das in Abb. 3.5 dargestellte Beispiel einer Größenabschätzung zeigt, dass die geschickt gewählte Entscheidergrenze zu einer deutlich kleineren (und auch schnelleren) Logik führt. Es wird nur eine Logikstufe (LUT) benötigt, weil die Betrachtung der unteren Bits (LSBs) vollständig entfällt. Solche Anpassungen sind in der Regel nur dann möglich, wenn keine exakte Wertbestimmung erfolgt sondern nur eine Größenordnung ermittelt werden muss. Außerdem zeigt die Schaltung, dass die resultierende Logik umso kleiner ist, je größer dieser Schwellenwert sein kann.

Eine ähnliche Möglichkeit für eine solche Optimierung ist das Addieren einer festen Konstante. Auch hier wird die Schaltungslogik klein, wenn bei der Konstante möglichst viele LSBs Null sind. Wiederum müssen dann die niederwertigen Bits des Eingangswertes nicht betrachtet werden.

Die zweite Kategorie der Optimierung ist unabhängig von der Art des Rechnens und somit rein schaltungsspezifisch. Folgende Maßnahmen fallen in diese Gruppe:

- Pipelining (Verkürzung des kritischen Pfads),
- Übertaktung einzelner Schaltungsteile,
- Einfachheit der Programmierung,

- Wiederverwendbarkeit der Komponenten (z.B. IFFT, FFT),
- Puffer bei Eingangssignalen (insb. bei *state machines*),
- Schieberegister ggf. durch LUTs realisieren,
- Unterstützung der Implementierungstools.

Pipelining ist eine wichtige Möglichkeit, FPGA-Designs hinsichtlich der maximal möglichen Taktfrequenz zu optimieren. Generell gilt in einer getakteten Schaltung, dass ein Signal am Eingang der Flipflops während der maßgeblichen Taktflanke stabil anliegen muss. Der Zeitpunkt dieser Stabilität wird bestimmt durch die Signallaufzeit der Schaltungslogik zwischen dem vorgeschalteten und dem betrachteten Flipflop. Je mehr Logikstufen, Multiplexer oder Verbindungsleitungen in einem solchen Pfad liegen, umso länger wird die Signallaufzeit und umso kleiner die erlaubte Taktfrequenz.

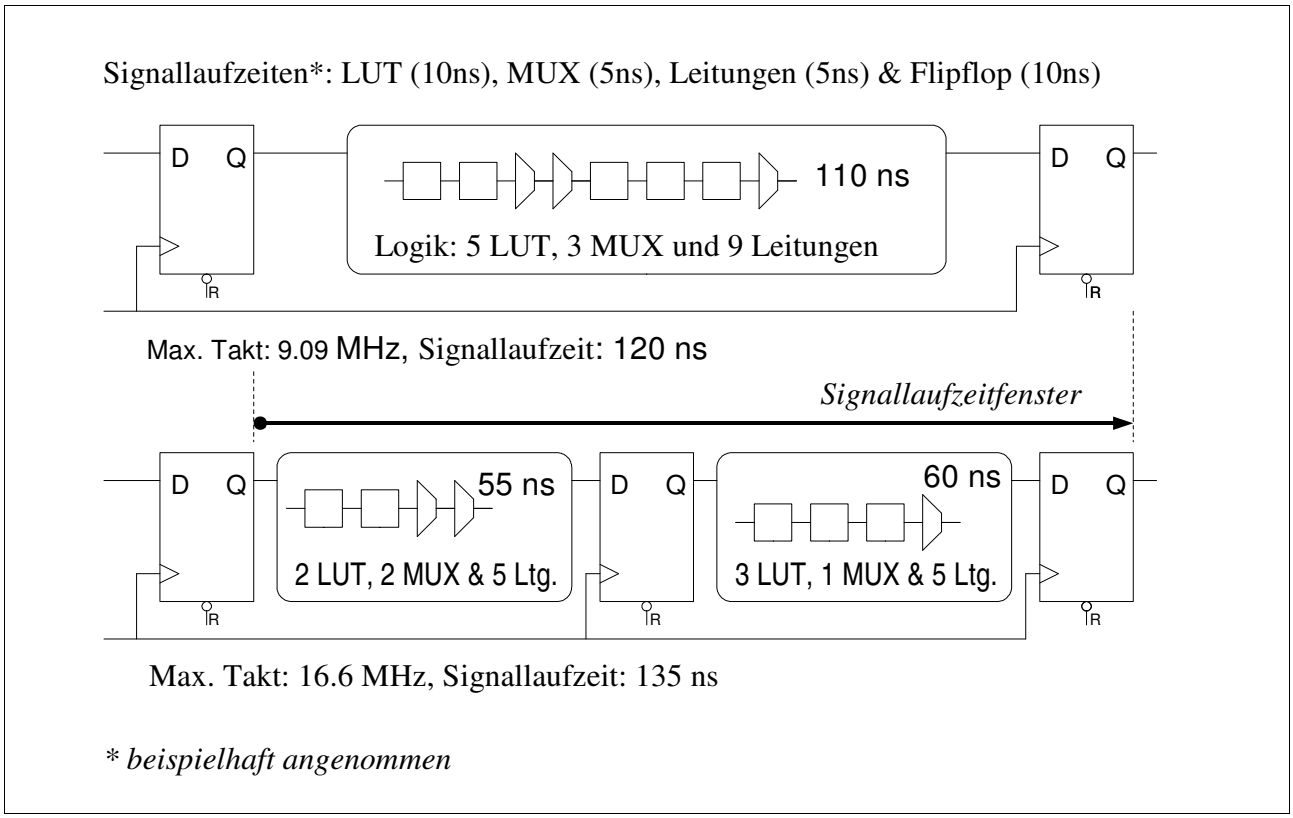


Abb. 3.6 Beispiel für Pipelining zur Takterhöhung

Pipelining nutzt den Ansatz, bei seriellen Verarbeitungsschritten diese Pfade zu verkürzen, indem zusätzliche Flipflops eingefügt werden. Dadurch wird ein langer Pfad durch mehrere kurze Pfade ersetzt. So kann der maximale Takt gesteigert werden, wobei der ursprüngliche Verarbeitungsschritt nun mehr als einen Taktzyklus erfordert.

In dem in Abb. 3.6 gezeigten Beispiel ist eine größere Logikschaltung durch ein zusätzliches Flipflop erweitert worden. Die längste Signallaufzeit zwischen den Flipflops, der sogenannte kritische Pfad, bestimmt dabei die maximal erlaubte Taktfrequenz des Designs. Die gesamte Durchlaufzeit der Schaltung wird durch Pipelining zwar aufgrund der zusätzlichen Elemente minimal erhöht, jedoch wird der Durchsatz wegen des höheren Taktes beinahe verdoppelt.

Das Pipelining kann somit helfen, die Verarbeitungszeit zu verringern. Die Herausforderung liegt dabei aber oft in dem Auffinden der entsprechenden zeitkritischen Schaltungsteile. Außerdem ist diese Methode nicht immer einsetzbar, weil zwingend ein Zwischenergebnis innerhalb des Pfades zur Ablage in Flipflops abgreifbar sein muss. Ist jedoch z.B. das Resultat der Schaltung direkt auf den Eingang rückgekoppelt, wird ein Pipelining unmöglich.

Als Übertaktung wird die Nutzung von Schaltungsteilen bezeichnet, die mit einem Vielfachen des Taktes in Bezug auf die umgebende Schaltung betrieben werden. So kann beispielsweise ein Dividierer, der mehrere Takte benötigt, bevor ein gültiges Ergebnis am Ausgang anliegt, in einigen Fällen mit dem doppelten oder vierfachen Takt betrieben werden. Dadurch wird die Latenzzeit verkleinert und häufig kann die Größe des Schaltungselementes verringert werden, weil interne Rechenwerke unter Umständen mehrfach genutzt werden können.

Einen Teil der Optimierungsmöglichkeiten kann eine leistungsfähige Entwicklungsumgebung automatisch erkennen und umsetzen. Aber insgesamt wird es immer Punkte geben, an denen der Entwickler durch Vorgaben ein Design weiter verbessern kann. Wenn in einem großen Schaltnetzwerk eine Vielzahl von Multiplikationen ausgeführt werden muss, ist es für ein automatisches Implementierungstool beinahe unmöglich, zu erkennen, an welchen Stellen ein Design besonders zeitkritisch ist. Eine Entscheidung, wo im Schaltungsnetzwerk dafür die vorhandenen Hardwaremultiplizierer eingesetzt werden sollen, kann eine Entwicklungsumgebung in der Regel nicht optimal treffen. An zeitkritischen Elementen kann ein Design durch eine gezielte Vorgabe von Schaltungsteilen (im VHDL-Code oder mit Constraints) verbessert werden.

3.7 Was wird die zukünftige FPGA-Technologie bieten?

Eine weitere Erhöhung der FPGA-Komplexitäten ist leicht vorhersehbar. Dieser Prozess setzt sich seit den letzten Jahren kontinuierlich fort und er wird höchstwahrscheinlich auch weiter Bestand haben. Ob die Art der logischen Zellen in dieser Form erhalten bleibt oder ob sich bei noch größeren Halbleiterdesigns ein anderes Optimum durchsetzt, ist schwer abzuschätzen. Immerhin sind FPGAs mit 4-Bit-Logikelementen pro Flipflop schon sehr etabliert.

Als weiterer Trend werden ergänzende Halbleiterbaugruppen mit in den FPGA-Baustein integriert. Während jetzt schon Multiplizierer, Taktmanager oder Speicher eingebunden sind, ist zu erwarten, dass noch weitere häufig verwendbare Elemente hinzukommen. Mögliche Kandidaten

dafür sind integrierte A/D- und D/A-Wandler oder auch schnelle Datenschnittstellen. Die Vorteile der Integration sind das einfachere Einbinden dieser Elemente, kleinere Platinenbaugruppen, eine bessere Datenintegrität (z.B. keine Probleme mit unsauberen Signalen und weniger elektromagnetische Abstrahlung) sowie höhere mögliche Datenraten zwischen den chipinternen Modulen. Der Nachteil ist sicher, dass solche FPGAs sehr komplex und somit teuer werden, auch wenn diese Extras nicht genutzt werden.

Mit der zunehmenden Verbreitung wird sich daher auch die Vielfalt der Produkte deutlich erhöhen, sodass FPGAs sehr anwendungsspezifisch ausgewählt werden können. Die hohe Integration wird den Aufbau von kleineren Platinen mit hoher Funktionalität ermöglichen, was auch durch die modernen Chipgehäuse, wie die *Ball Grid Arrays* (BGA) mit geringen Pin- bzw. Ballabständen, weiter unterstützt wird. Die Chance zur Miniaturisierung mit diesen Baugruppen wird jedoch durch einen höheren technischen Aufwand bei der Verarbeitung erkauft, weil sich diese Chips nicht mehr ohne professionelle Bestückungsmaschinen verarbeiten lassen.

Zusammenfassend lässt sich sagen, dass der Trend der FPGAs parallel zu der allgemeinen Computer- und Platinentechnik verläuft: mehr Performance bei weniger Platz und Leistungsaufnahme. Eine hohe Rechenkapazität und die Möglichkeit, Mikrocontroller und Prozessoren innerhalb von programmierbaren Logikbausteinen als Schaltungsstruktur oder als IP-Core zu integrieren, bieten die Chance, mit FPGAs solche Bauelemente mit einem Zugewinn an Flexibilität zu ersetzen.

4 Analog/Digital Schnittstelle

Der Schwerpunkt dieser Arbeit liegt in der digitalen Signalverarbeitung innerhalb von FPGAs. Dennoch gehört zu einem OFDM-Übertragungssystem zwingend eine Umsetzung auf ein analoges Signal am Senderausgang bzw. die Digitalisierung des Eingangssignals am Empfänger. Die Prinzipien dieser Umsetzung werden im Folgenden ebenso erläutert wie die Darstellung der Werte im digitalen Bereich. Wichtig in diesem Zusammenhang ist die Auflösung von Zahlen bzw. die Wortbreite in der digitalen Signalverarbeitung. Je mehr Bits zur Verfügung stehen, umso präziser werden die Eingangsdaten erfasst, weiterverarbeitet und umso genauer sind auch die Resultate. Leider wächst damit auch unmittelbar der Schaltungsaufwand.

Ein bei solchen digitalen Verarbeitungsketten besonders wichtiges Thema ist außerdem die Wortbreitenanpassung der Zahlenwerte nach Operationen wie z.B. Multiplikationen oder Additionen. Der Einfluss dieser Skalierungen ist ganz ähnlich zu der Quantisierung in einem A/D-Wandler und wird in dem nachfolgenden Abschnitt 4.3 behandelt. Die große Bedeutung der Zahlenbereichsanpassung ergibt sich daraus, dass sie in der Verarbeitungskette vielfach vorkommt, während die A/D-Wandlung in jedem Empfangszweig nur einmal durchgeführt wird.

Abschließend werden noch mögliche suboptimale Einflüsse der A/D- bzw. D/A-Wandler aufgezeigt, also die Unterschiede zwischen einer theoretisch idealen Funktion und dem Verhalten von realen Baugruppen.

4.1 Digital/Analog- und Analog/Digital-Wandlung

In einem OFDM-Sender wird das Nutzsignal am Ende der digitalen Verarbeitungskette in ein analoges Signal gewandelt. Da es sich um ein komplexes Basisbandsignal handelt, werden zwei D/A-Wandler benötigt, um den Real- und Imaginärteil für den nachfolgenden Mischer bereitzustellen. In der Nachrichtentechnik spricht man von der Inphase- und der Quadraturkomponente (IQ-Signal).

Alternativ kann auch ein reelles Ausgangssignal berechnet werden, wenn das Signal digital auf eine Zwischenfrequenz moduliert wird. Dazu ist vorab eine Erhöhung der Abtastrate (*Upsampling*)

mit anschließender Filterung notwendig. In diesem Fall kommt der Sender mit einem einzigen D/A-Wandler aus, welcher aber mit einem schnelleren Takt arbeiten muss.

Unbedingt wichtig ist die Einhaltung des ersten Nyquisttheorems (Abtasttheorem), wonach der Kehrwert des Wandlertakts mindestens doppelt so hoch sein muss wie die obere Grenzfrequenz des Nutzsignals [FLI91]. In einem IEEE802.11a bzw. HiperLAN/2-System ist der Sampletakt der Wandler für IQ-Signale 20 MHz. Das genutzte Spektrum liegt hier zwischen -8,125 MHz und 8,125 MHz[1]. Die Nyquistbedingung ist in diesem Fall also erfüllt.

Der D/A-Wandler verfügt über eine Referenzspannung, welche den maximalen Ausgangspegel definiert. Dieser Spannungsbereich wird durch die Anzahl der möglichen digitalen Eingangszahlen geteilt, sodass anschließend jedem Eingangswert ein analoger Ausgangsspannungspegel zugewiesen wird. Dies erfolgt in der Regel linear, d.h. die Spannung am Ausgang ist proportional zur Größe des digitalen Eingangswertes.

Das analoge Ausgangssignal ist jedoch noch stufenförmig, weil sich die Werte am Wandler nur mit dem anliegenden Takt und nur auf diskrete Größen ändern. Im Frequenzbereich betrachtet zeigen sich dadurch neben dem Nutzsignal noch unendlich viele Wiederholspektren [FLI93], die vor der Weiterverarbeitung durch ein geeignetes Tiefpassfilter unterdrückt werden müssen. Würde auf ein solches Filter verzichtet, wären die strengen Anforderungen zur Unterdrückung von Außerbandabstrahlung nicht einzuhalten.

Das Gegenstück zum D/A-Wandler ist der Analog/Digital-Wandler, bei dem das Empfangssignal innerhalb eines Referenzintervalls quantisiert wird und als digitaler Zahlenwert (i. d. R. im Zweierkomplement) zur Verfügung gestellt wird. Soll ein OFDM-Basisbandsignal eingelesen werden, müssen auch hier zwei Umsetzer für Inphase- und die Quadraturkomponente genutzt werden. Dabei kommt es wiederum aufgrund der endlichen Zahl von möglichen Digitalisierungsstufen zu Quantisierungsrauschen.

Ein abgetastetes analoges Signal ist nur dann eindeutig rekonstruierbar, wenn auch hier das Nyquistkriterium eingehalten wird. Gegebenenfalls muss durch ein vorgeschaltetes Tiefpassfilter die Bandbreite des Eingangssignals auf die Abtastfrequenz reduziert werden. Damit können dann auch die Rauschanteile außerhalb des Nutzbandes eliminiert werden.

4.1.1 D/A-Wandler am Beispiel AD9767

Auf dem für diese Arbeit entwickelten Hardwareaufbau wird ein D/A-Umsetzer von Analog Devices eingesetzt. Dieser Baustein AD9767 integriert zwei Wandlerelemente in einem Gehäuse. Beide Einheiten arbeiten mit 14 Bit genauen Eingangswerten, die mit einem maximalen Wandeltakt

1 Die reservierte Bandbreite von 20 MHz reduziert sich etwas aufgrund der ungenutzten äußeren Subträger. Diese Verkleinerung hilft die Anforderungen der Seitenbandsignaldämpfung zu erfüllen (vgl. [HIP01], Kap. 5.8.2.2).

von 125 MHz in analoge Werte umgesetzt werden können. Die digitalen Eingangswerte werden (für diese hohe Geschwindigkeit beinahe unumgänglich) parallel an den Baustein geleitet.

Abb. 4.1 zeigt den prinzipiellen Aufbau eines D/A-Umsetzers. Diese Struktur macht deutlich, dass als Ausgangsgröße ein zum Eingangswert äquivalenter Strom erzeugt wird.

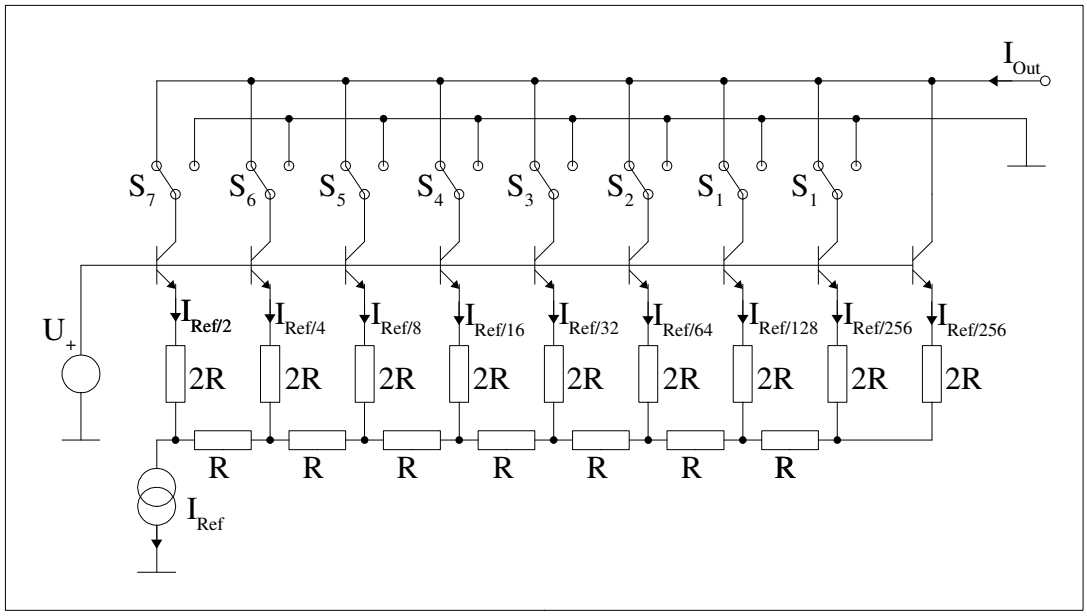


Abb. 4.1Prinzipieller Aufbau eines 8-Bit-D/A-Wandlers

Mit der für alle Transistoren gemeinsamen Basisspannung wird erreicht, dass die Emitter auf gleichem Potenzial liegen. Der Ausgangsstrom I_{out} ist eine Addition aus den Teilströmen des Netzwerkes aus Abb. 4.1, die je nach Schalterstellung Null oder ein Bruchteil des Referenzstroms sind.

Soll das Ausgangssignal eine Spannungsgröße sein, ist ggf. eine Ausgangsstufe zur Umsetzung nötig. Die in dieser Arbeit verwendete Hardware nutzt dafür eine Treiberstufe mit einem Operationsverstärker. Abb. 4.2 zeigt eine in dem Datenblatt vorgeschlagene Spannungsausgangs-stufe, wobei die differenziellen Ausgangsströme des Wandlers in einen einzelnen Spannungswert transformiert werden. Diese massebezogenen Einzelsignale werden häufig mit dem englischen Ausdruck *single ended* bezeichnet.

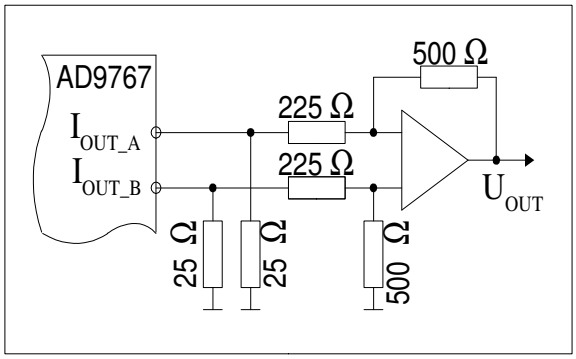


Abb. 4.2 D/A-Wandler Ausgangstreiber

Die wichtigen Kenngrößen von D/A-Wandlern sind die Genauigkeit, Ausgangsstufengeschwindigkeit und ihre Stabilität [HEI82][ZAN83]. Diese Faktoren beschreiben die Abweichungen von einem idealen Bauelement.

Die Genauigkeit wird durch die Größen Nullpunktverschiebung (*Offset*), Verstärkungsfehler (*Gain Error*) und Linearität bestimmt. Abb. 4.3 stellt dar, wie sich die jeweiligen Fehler auf den Ausgangswert auswirken.

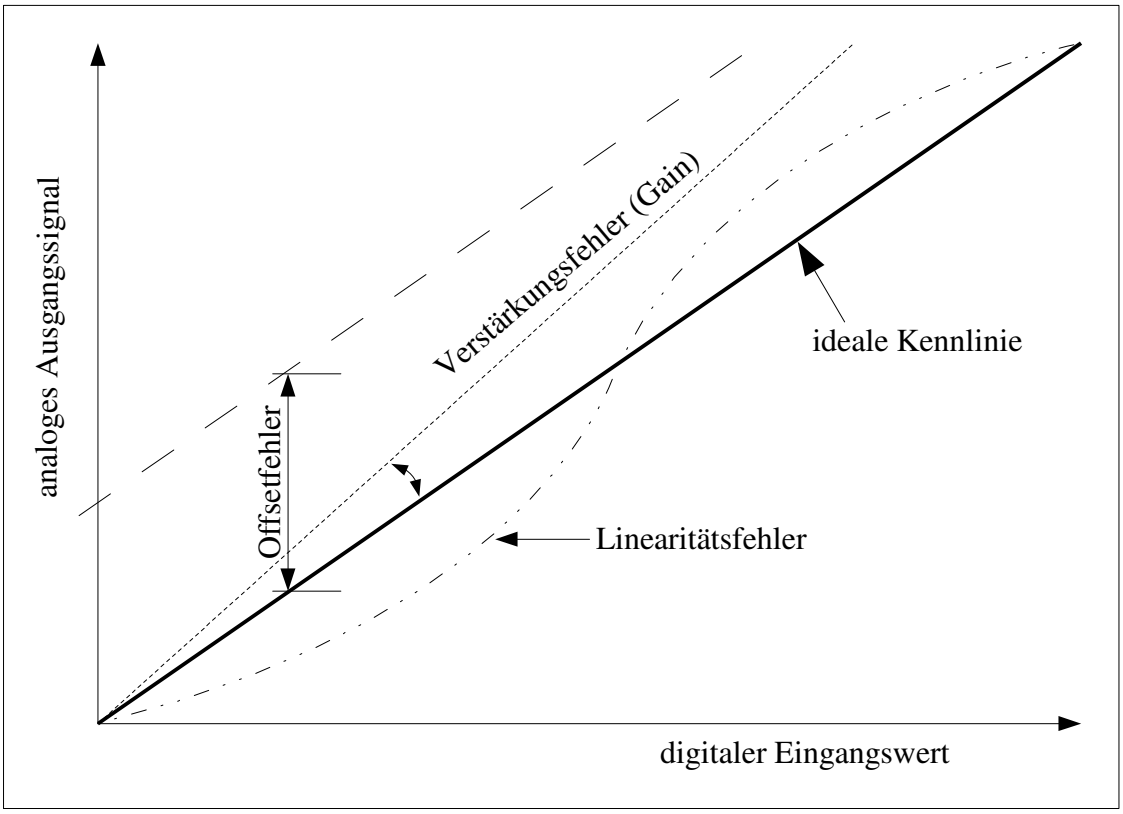


Abb. 4.3 Kenngrößen von D/A-Wandlern in Bezug auf ein ideales Bauelement

Die Ausgangsgeschwindigkeit beschreibt das Einschwingverhalten der Ausgangsstufe, also innerhalb welcher Zeit der gewünschte analoge Spannungswert stabil am Ausgang anliegt.

Als letzter Faktor ist die Stabilität von D/A-Wandlern gegenüber Temperaturschwankungen und gegenüber Schwankungen in der Versorgungsspannung ein möglicher Störfaktor.

Der AD9767 weist in Bezug auf die besprochenen Qualitätsmerkmale folgende Kenngrößen auf:

- Verstärkergenauigkeit[2]: typisch +/-1% der Vollaussteuerung (max. 5%),

2 Bei interner Spannungsreferenzquelle.

- Offsetfehler: 0,02% der Vollaussteuerung,
- Linearität (*Spurious Free Dynamic Range,* SDFR): -82 dBc (dB Carrier beschreibt das Verhältnis im Spektrum von Nutzfrequenz zu größter Störfrequenz),
- Ausgangs-Regelzeit (auf 0,1%): 35ns,
- Temperaturkoeffizient[2] : $+/-10^{-4}$ der Vollaussteuerung pro Grad Celsius,
- Stabilität in Abhängigkeit von der Eingangsspannung[3]: 0,4% der Vollaussteuerung.

Speziell für OFDM-Systeme ist die Linearität der Komponenten von großer Bedeutung, da sie als Grundvoraussetzung gilt, damit die Orthogonalität erhalten bleibt.

4.1.2 A/D-Wandler am Beispiel AD9433

Für Umsetzung einer analogen Spannung in einen binären Zahlenwert gibt es A/D-Wandler, die nach unterschiedlichen Prinzipien arbeiten. Als wichtigste wären die Zähl-, Integrations-, Iterations- sowie Parallelverfahren zu nennen [KUE90][DEM91]. Für die hohen Abtastraten, die für moderne Kommunikationssysteme nötig sind, kommen aber in erster Linie die schnellen Parallelwandler in Frage.

Abb. 4.4 zeigt ein Blockschaltbild eines A/D-Umsetzers mit acht parallel arbeitenden Komparatoren.

Leider ist der Schaltungsaufwand dieser Wandler sehr groß. Eine Erhöhung der Auflösung um nur ein Bit benötigt die doppelte Anzahl an Komparatoren und verdoppelt somit auch in etwa die Größe des integrierten Schaltkreises. Neben FPGAs sind die A/D-Wandler damit ein weiteres Beispiel von Komponenten, die unmittelbar von den modernen und leistungsfähigen Techniken in der Halbleiterentwicklung profitieren.

Der AD9433 ist ein monolithischer A/D-Konverter mit einer Auflösung von 12 Bit. Er ist erhältlich für Abtastraten von 105 und 125 MHz. Ein integriertes vorgeschaltetes Abtast-/Halteglied (engl. *Sample/Hold* oder *Track/Hold*) ist vor den

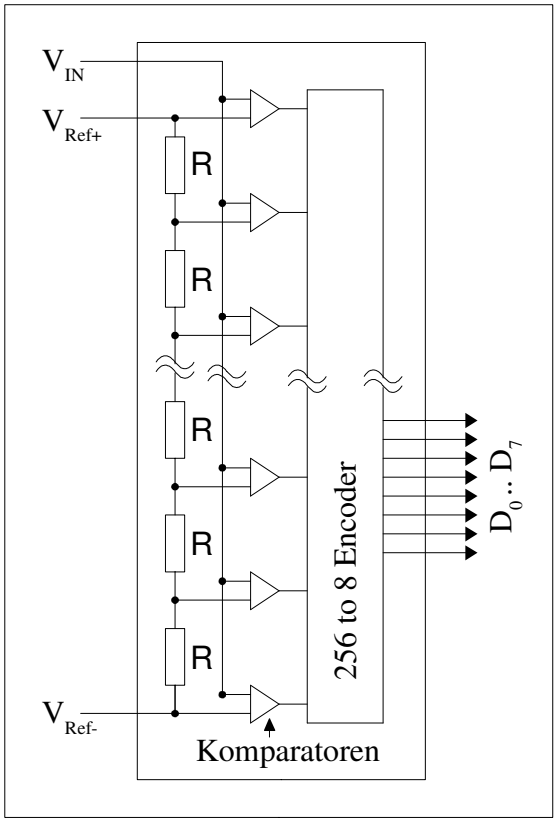


Abb. 4.4 Blockschaltbild eines parallelen A/D-Wandlers

3 Bei +/-10% Spannungsschwankung.

eigentlichen Wandler geschaltet. Dieser wiederum ist mehrstufig (*pipelined)* aufgebaut (vgl. Abb. 4.6).

A/D-Wandler sind sehr empfindlich gegenüber unsauberen Abtasttakt-Eingangssignalen. Rauschen, Störungen oder Clock-Jitter (siehe auch Kap. 4.4) beeinflussen unmittelbar das Ergebnis der Wandlung. Um ein möglichst sauberes und unempfindliches Taktsignal in den Baustein einkoppeln zu können, besitzt der betrachtete Baustein einen differentiellen Takteingang.

Auch der Nutzsignaleingang des AD9433 ist differentiell. Wenn ein einzelnes massebezogenes Spannungssignal verarbeitet werden soll, muss dieses durch einen geeigneten Umsetzer in den Baustein gespeist werde. Abb. 4.5 zeigt beispielhaft eine solche Schaltung mit einem differentiellen Operationsverstärker. Dieser arbeitet jedoch rund um eine Mittenspannung, sodass der kleinste Wert bei Null Volt liegt, die digitale Null bei +2,5V und der maximale Wert bei +5V erreicht wird. Das Eingangssignal sollte also mit einem Gleichspannungsoffset von 2,5V beaufschlagt sein bzw. werden.

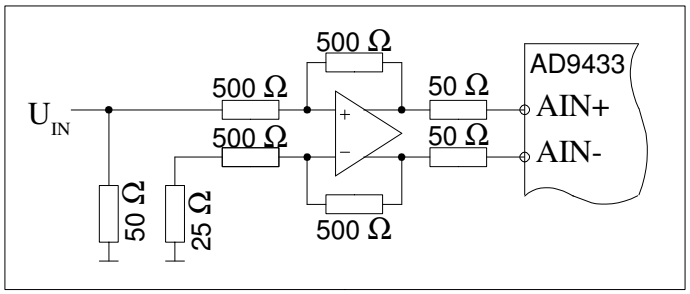


Abb. 4.5 A/D-Wandler Eingangstreiber

Wie schon oben erwähnt, ist der AD9433 als stufenweiser Wandler implementiert. Diese Bauweise (Abb. 4.6) ermöglicht es, die Komplexität des Bausteins erheblich zu reduzieren. Während prinzipiell jedes zusätzliche Bit bei einem A/D-Konverter zu einer Verdoppelung des integrierten Schaltkreises führt, kann aufgrund dieser Bauweise z.B. mit zwei 6 Bit genauen Wandlern eine Genauigkeit von 12 Bit erreicht werden. Das entspricht praktisch der Schaltungsgröße eines 7 Bit Konverters, wobei jedoch ein zusätzlicher D/A-Wandler und ein Subtrahierer zur Erzeugung des Differenzsignals vor der LSB[4]-Stufe notwendig sind.

4 LSB: *Least significant Bit*, also die Stufe mit der feinsten Auflösung.

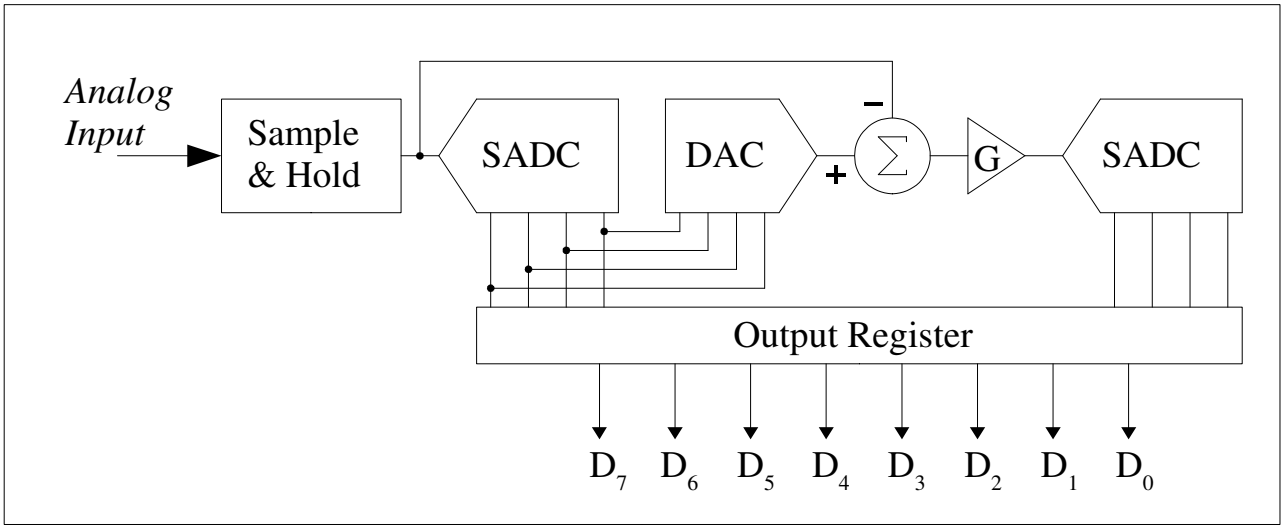


Abb. 4.6 Die Pipelined-Architektur zur Reduzierung des Schaltungsaufwandes bei A/D-Konvertern

Ähnlich wie bei den D/A-Wandlern gibt es auch hier neben den Leistungsdaten wie maximale Abtastrate und Auflösungsvermögen einige wichtige Kennwerte zu beachten. Die Eigenschaften Genauigkeit, Eingangsstufengeschwindigkeit (Eingangsbandbreite) und Stabilität [ZAN85] gelten hier in gleicher Weise wie im vorangehenden Kapitel beschrieben. Hinzu kommt aufgrund der großen Komplexität des Bauelements noch das Kriterium Leistungsaufnahme. Mehr als ein Watt sind hier durchaus üblich, was beim Platinendesign berücksichtigt werden muss – sowohl eine stabile Spannungsversorgung als auch eine ausreichende Wärmeabfuhr sind essentiell.

Die Kennwerte des AD9433:

- Leistungsaufnahme: 1,35W bei 125 MHz Abtastrate,
- Verstärkergenauigkeit: typisch +/-1% der Vollaussteuerung (max. -7% bis +3%),
- Offsetfehler: +/- 5mV (also 0,1% bei einer Referenz von 5V),
- Linearität: **+/-** 0,5 LSB (typisch),
- analoge Eingangsbandbreite: 750 MHz,
- Stabilität in Abh. von der Eingangsspannung[5]: 0,4% der Vollaussteuerung,
- Temperaturkoeffizient: $+/-125*10^{-6}$ (Gain) und $+/-50*10^{-6}$ (Offset) der Vollaussteuerung pro Grad Celsius.

Das beschriebene Bauelement erreicht einen Signal- zu Rauschabstand von 65,4 dB, was unter dem theoretisch idealen Wert (also nur Quantisierungsfehler, siehe Kap. 4.2) von 74dB eines 12-

5 Bei +/-10% Spannungsschwankung.

Bit-Wandlers liegt. Damit entspricht das reale S/N-Verhältnis 10,6 effektiv genutzten Bits. Eine Zahl, die für die Qualität der aufgebauten Schaltung aussagekräftiger ist als die vom Wandler ausgegebene Wortbreite.

4.2 Quantisierung

Die digitale Signalverarbeitung ist generell darauf beschränkt, nur endlich viele Zahlen aus einem bestimmten Wertebereich zu erlauben. Dazu wird für ein kontinuierliches Signal zunächst ein hinreichend großes Intervall definiert, in dem eine Digitalisierung durchgeführt werden soll. Dies ist bei einer realen Schaltung häufig problemlos möglich. Wenn z.B. ein Operationsverstärker über einen maximalen Ausgangspegel von ±5 V verfügt, sollte ein nachgeschalteter A/D-Wandler auf dem gleichen Intervall arbeiten, damit es nicht zu Verfälschungen durch abgeschnittene Signale kommt. Dieses vorab definierte Intervall wird anschließend in Stufen unterteilt, um die unendlich vielen Werte in einen endlichen Zahlenbereich abzubilden. In der Regel sind die Stufen dabei äquidistant gewählt und ihnen werden binär codierte Werte zugewiesen. Dies ergibt dann eine lineare Quantisierung. Für Sonderfälle können auch nichtlineare Quantisierungsstufen nützlich sein. Dies sind aber Ausnahmefälle, die schon bestimmte Annahmen über die Ein- bzw. Ausgangswerte berücksichtigen. Beispiele dafür finden sich in [KAM92], sollen hier aber nicht weiter betrachtet werden.

Die Wortbreite der Signale sollte bei der Quantisierung so gewählt werden, dass nachfolgende Verarbeitungsschritte mit hinreichender Genauigkeit durchgeführt werden können. Moderne 32- oder 64-Bit-Prozessoren arbeiten in einem dementsprechend großen Bereich von Fließkommazahlen mit sehr großer Genauigkeit, sodass Überläufe nur selten möglich sind und interne Quantisierungseffekte vernachlässigbar klein bleiben.

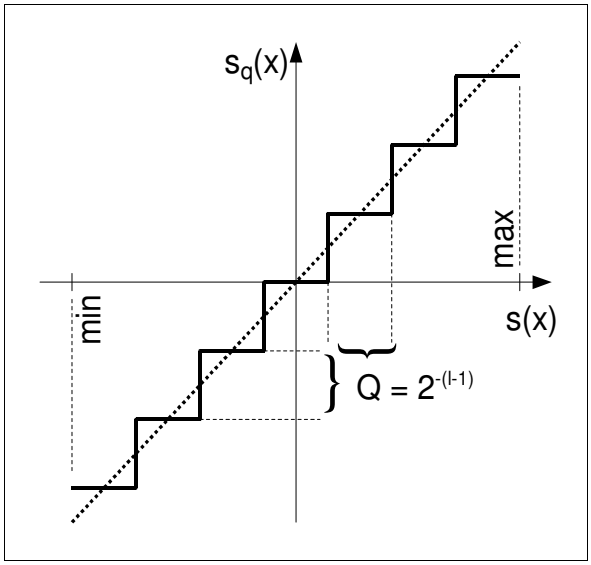


Abb. 4.7 Lineare Quantisierungskennlinie

In einer Verarbeitungsumgebung mit frei wählbaren Wortbreiten und Festkommaarithmetik ist das Einbeziehen dieser Effekte jedoch sehr wichtig. FPGAs rechnen mit Werten von beliebiger Genauigkeit, wobei Entwickler immer bestrebt sein sollten, die Wortbreiten so klein wie möglich zu wählen. Dies führt zu kleineren und schnelleren Schaltungselementen.

Die Anzahl von Bits (l), mit denen ein Signal aufgelöst wird, bestimmt das Quantisierungsrauschen. Je mehr Bits zur Verfügung stehen, desto feiner sind die Abstufungen und desto geringer ist das Rauschen durch Rundungsfehler. Quantisierungsrauschen spielt längst nicht nur bei der D/A- und A/D-Wandlung eine Rolle. In jedem Rechenschritt innerhalb der Signalverarbeitung, bei dem Zahlenbereiche angepasst werden müssen, kommt es zu einem ähnlichen Effekt, der im folgenden Kapital als Zahlenbereichsanpassung detailliert betrachtet wird.

Die Rauschleistung für die übliche lineare Quantisierung (Abb. 4.7) kann leicht bestimmt werden, wenn das Signal innerhalb der Quantisierungsstufen gleichverteilt (Abb. 4.8) ist

$$p_s(x) = \frac{1}{Q}\, rect\left(\frac{x}{Q}\right) \qquad (4.1)$$

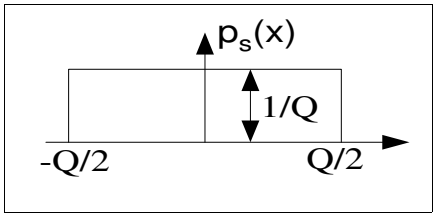


Abb. 4.8 Verteilungsdichtefunktion des Quantisierungsfehlers

und jede Rundung mathematisch korrekt ausgeführt wird

$$\frac{-Q}{2} \leqslant s(kT) < \frac{+Q}{2} \quad , \quad Q = \frac{1}{2^{(l-1)}} \qquad (4.2)\ .$$

Die mittlere Leistung durch den Quantisierungsfehler ist dann

$$E\{s^2\} = \int_{-\infty}^{\infty} p_s(x)\, x^2\, dx = \int_{-Q/2}^{Q/2} x^2\, dx = \frac{Q^2}{12} \qquad (4.3)\ .$$

Das S/N-Verhältnis ergibt sich beispielhaft für ein voll ausgesteuertes Sinussignal in Abhängigkeit der Wortlänge l zu

$$\frac{S}{N} = \frac{1}{2}\,\frac{1}{Q^2/12} = \frac{3}{2}\,2^{2l} \qquad (4.4)$$

oder als logarithmische Größe in Dezibel zu

$$\left[\frac{S}{N}\right]_{dB} = 10\,lg\left(\frac{3}{2}\,2^{2l}\right) = 1{,}77 + 6{,}0206 * l \qquad (4.5)\ .$$

Neben der Auflösungsgenauigkeit muss bei der digitalen Signalverarbeitung darauf geachtet werden, dass der maximale Wertebereich nicht überschritten wird. Im Allgemeinen sind jedoch an jeder Stelle der Verarbeitungskette die Pegel bzw. die Wortbreiten bekannt, sodass es in der Verantwortung des Entwicklers liegt, Überläufe zu vermeiden.

Es kann in seltenen Fällen vorkommen, dass überwiegend sehr kleine Werte auftreten. Dann kann es sinnvoll sein, den Wertebereich nicht mehr für die theoretisch möglichen Maximalwerte auszulegen, sondern einen Bereich zu wählen, in dem die Ergebnisse mit sehr hoher Wahrscheinlichkeit fallen. Wichtig ist, dass die Werte außerhalb des darstellbaren Bereichs korrekt abgeschnitten (engl. *Clipping*) und damit dem nächstgelegenen möglichen Wert zugeordnet werden.

Bei der Rechnung mit Binärzahlen in FPGA-Schaltungen muss das *Clipping* explizit berücksichtigt werden. Ein Überlauf ohne diese Funktion führt normalerweise zur Vorzeichenumkehr und damit zu massiven Fehlern in der Signalverarbeitung. Aber auch das Abschneiden der Werte ist immer eine Ungenauigkeit, die nach Möglichkeit vermieden werden sollte.

Der Wertebereich sollte immer so groß gewählt werden, dass keine Überläufe auftreten. Ist dies nicht möglich, sollte unbedingt ein *Clipping* implementiert werden.

4.3 Zahlenbereichsanpassung

Eine Addition zweier Acht-Bit-Zahlen erzeugt ein neun Bit breites Ergebnis, eine Multiplikation sogar ein 16-Bit-Resultat. Da in einer Verarbeitungskette wie dem betrachteten OFDM-System sehr viele dieser Operationen durchgeführt werden, muss anschließend jeweils eine Verkleinerung der Wortbreite des Ergebnisses durchgeführt werden, damit diese Werte nicht zu inakzeptabel breiten Zahlen wachsen. Die Reduzierung der Wortbreite mit einer mathematisch korrekten Rundung (Zahlenbereichsanpassung bzw. Requantisierung) fügt dem Signal jedoch jedes Mal (ähnlich wie bei der im vorigen Kapitel eingeführten Quantisierung) ein Quantisierungsrauschen hinzu. Eine sehr lange Kette aus Rechenschritten kann somit durch das häufige Runden schon einen erheblichen Einfluss auf das S/N-Verhältnis haben.

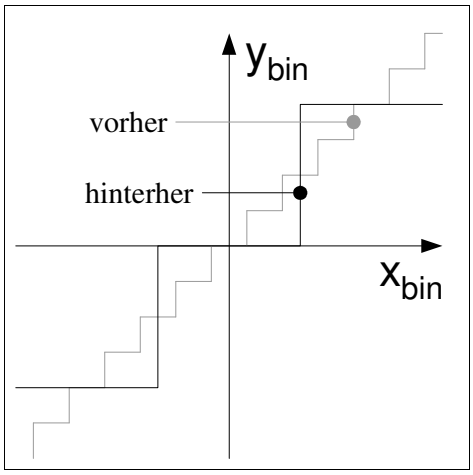


Abb. 4.9 Wortbreitenanpassung

Für die Berechnung des Quantisierungsfehlers bei der Wortbreitenanpassung wird angenommen (analog zu 4.1 bis 4.3), dass das Signal in der höheren Auflösung gleichverteilt ist. Die Breite der Quantisierungsstufen ist vorher Q_A und wird durch die Reduktion der Auflösung auf Q_N vergrößert.

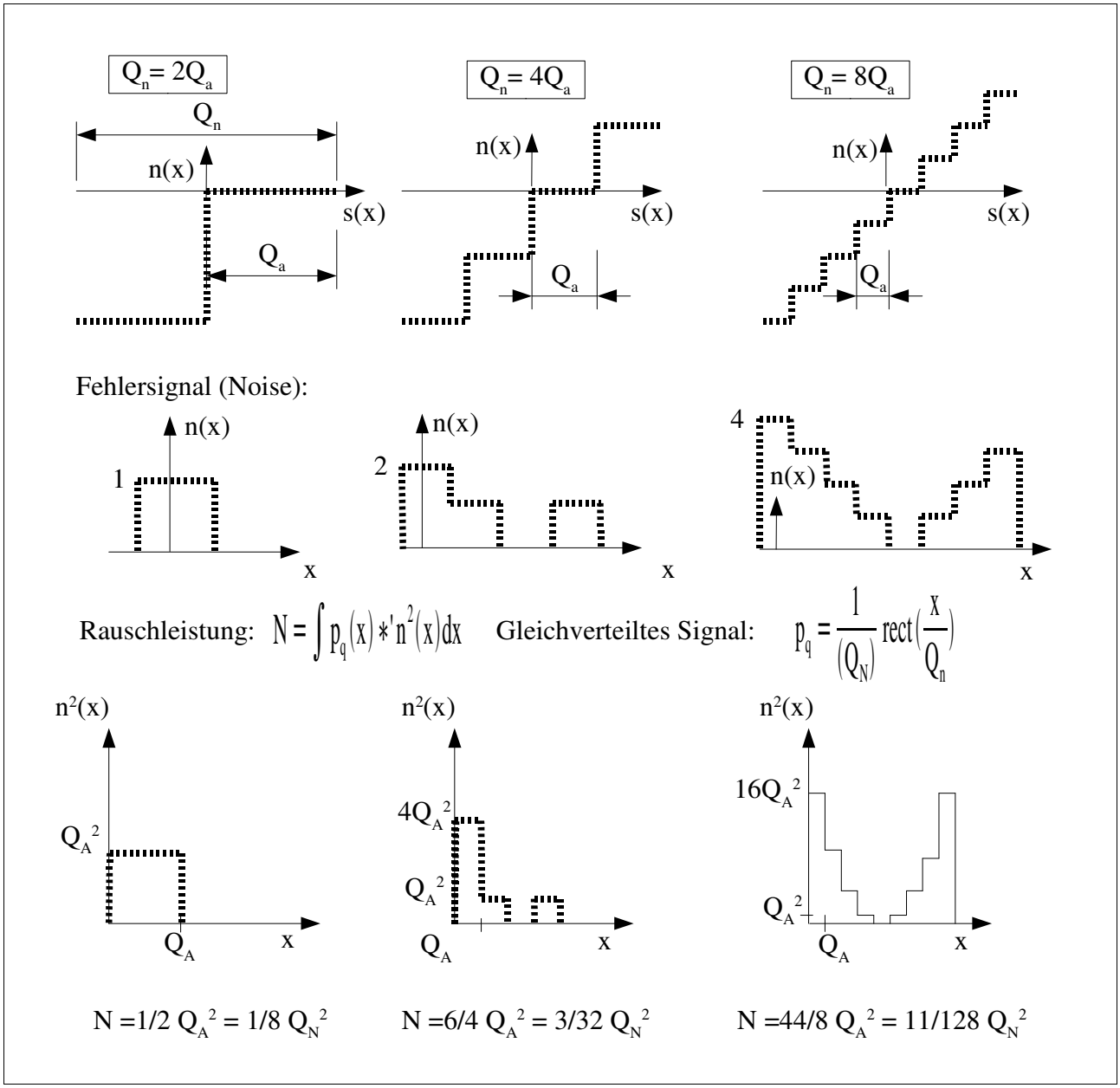


Abb. 4.10 Beispiele der Rauschleistung bei der Reduzierung der Wortbreite um ein Bit (links), zwei Bit (Mitte) und drei Bit (rechts).

In Abb. 4.10 sind Beispiele aufgeführt, die eine Verringerung der Wortbreite um ein bis drei Bits zeigen. Hier wird deutlich, dass das Rauschen der durch die Wortbreitenanpassung von Binärzahlen immer größer ist als das klassische Quantisierungsrauschen. Allerdings nähert sich diese Größe

asymptotisch dem aus Gleichung 4.3 an, wenn die Auflösung vor der Wortbreitenanpassung viel größer ist als nach der Operation. In den Beispielen ist jeweils nur eine Stufe der neuen Wortbreite Q_N dargestellt.

Da Q_A ein 2^n-Vielfaches von Q_N ist, kann die Größe des Rauschens in Abhängigkeit von diesen beiden Werten angegeben werden. Analog zu Gleichung 4.3 ergibt sich das Quantisierungsrauschen für Veränderungen der Wortbreiten $\Delta l = (l_A - l_N) \geq 2$ zu

$$E\{s^2\} = \int_{-\infty}^{\infty} p_s(x)\, n^2(x)\, dx = \left(1 + \sum_{2}^{y=\Delta l} 2^{y-1}\right) \frac{Q_N^2}{2^{2\Delta l + 1}} \qquad (4.6)\,.$$

Beträgt die Differenz der Wortbreiten nur $\Delta l = 1$, ist der Summenterm gleich Null und das Rauschen $N = \frac{Q_N^2}{2^3} = \frac{Q_N^2}{8}$. Bei großem Δl nähert sich $E(s^2)$ dem in Gleichung 4.3 für die Quantisierung eines kontinuierlichen Signals ermitteltem Wert $Q^2/12$.

Eine Einschätzung des Einflusses der internen Wortbreitenanpassungen auf das Gesamt-SNR in einer Übertragungsstrecke kann jedoch auch bei Kenntnis dieser Rauschleistung nicht gegeben werden. Allenfalls kann eine obere Schranke für das SNR ermittelt werden, wobei der Rauscheinfluss der Quantisierung und Wortbreitenanpassung klein gegenüber dem Rauschen der Übertragungsstrecke sein sollte.

$$SNR = \frac{S}{N} = \frac{S}{N_{Quantisierung} + N_{Channel}} \ll \frac{S}{N_{Quantisierung}} \qquad (4.7)\,.$$

An der Gleichung ist abzulesen, dass der absolute Einfluss auf das S/N-Verhältnis des Rauschens durch die Zahlenbreitenanpassung nicht direkt ersichtlich ist, weil sich dieser zu dem Rauschen des Kanals im Nenner addiert. Wenn jedoch $N_{quant.}$ deutlich kleiner als $N_{channel}$ ist, hat dieser Effekt aus der digitalen Signalverarbeitung kaum Bedeutung. Dieser Fall sollte bei dem Entwurf eines Übertragungssystems immer angestrebt werden.

Beispiel: Ein Audiosignal wird auf einer üblichen Compact Disc mit 16 Bit Genauigkeit gespeichert. Allein durch die Quantisierung bei der A/D-Wandlung liegt das SNR bei 98 dB [KAM92]. Wird dieses Signal digital weiterverarbeitet, so verschlechtert sich das SNR bei einer Wortbreitenanpassung auf 16 Bit um minimal drei dB, bei drei Anpassungen um sechs dB usw..

Wird ein digitales Filter mit 31 Koeffizienten genutzt, fällt das SNR schon auf 83 dB. Dieser Verlust kann jedoch deutlich reduziert werden, wenn das Filter intern mit einer höheren Genauigkeit rechnet und nicht nach jeder Multiplikation die Wortbreite wieder auf 16 Bit rundet.

Bei der Verringerung der Wortbreite sollte darauf geachtet werden, dass mathematisch korrekt gerundet wird. Dieser Vorgang ist in an einem Beispiel für Zahlen im Zweierkomplement dargestellt. Dabei wird an der Stelle des höchstwertigen Kapp-Bits eine Eins hinzuaddiert, bevor die Wortbreite reduziert wird. Dieser Vorgang entspricht dem Aufrunden aller Nachkommawerte die größer sind als 0,5.

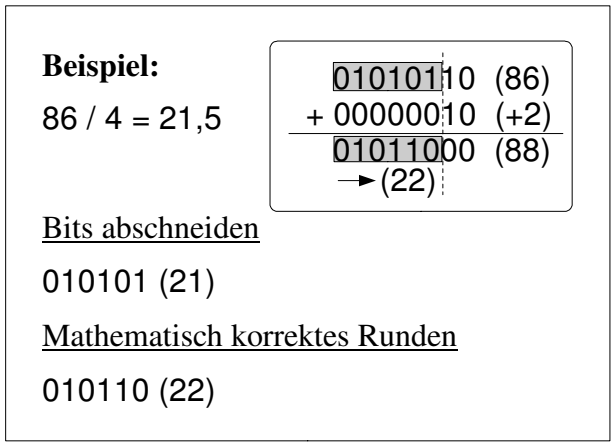


Abb. 4.11 Korrektes Runden von Binärzahlen

Die Rauschleistung der Quantisierung und Wortbreitenanpassung sollte kleiner sein als das bei dem Systementwurf kalkulierte maximale Rauschen des Übertragungskanals. Der Einfluss der Wortbreitenanpassung kann durch partielles Rechnen mit höherer Genauigkeit verkleinert werden.

4.4 Suboptimale A/D- und D/A-Wandlung

In den vorangegangenen Kapiteln wurden im Zusammenhang mit der Arbeitsweise der A/D- und D/A-Umsetzer auch die Störgrößen besprochen, die bei realen Bauelementen unvermeidlich auftreten. Doch auch unabhängig von den Bauteilen selbst kann es zu störenden Einflüssen bei der Wandlung kommen. Wichtig ist zunächst eine möglichst exakte Ein- bzw. Auskopplung des analogen Signals. Dieser Punkt ist jedoch schwer in diesem Kontext abzuhandeln, weil hier jedes Element in der analogen Kette zwischen Sender und Empfänger betrachtet werden muss. Dies ist nicht Schwerpunkt dieser Arbeit.

Ein weiterer Effekt tritt bei der Wandlung auf, der eine Störung des Nutzsignals verursacht. Ein unsauberes Taktsignal an den Wandlern führt unmittelbar zu einer Verfälschung des Signalverlaufs. Ein Beispiel für das ungenaue Auslösen der Abtastung ist eine Wandlung mit dem in Abb. 4.12 abgebildeten Taktsignal.

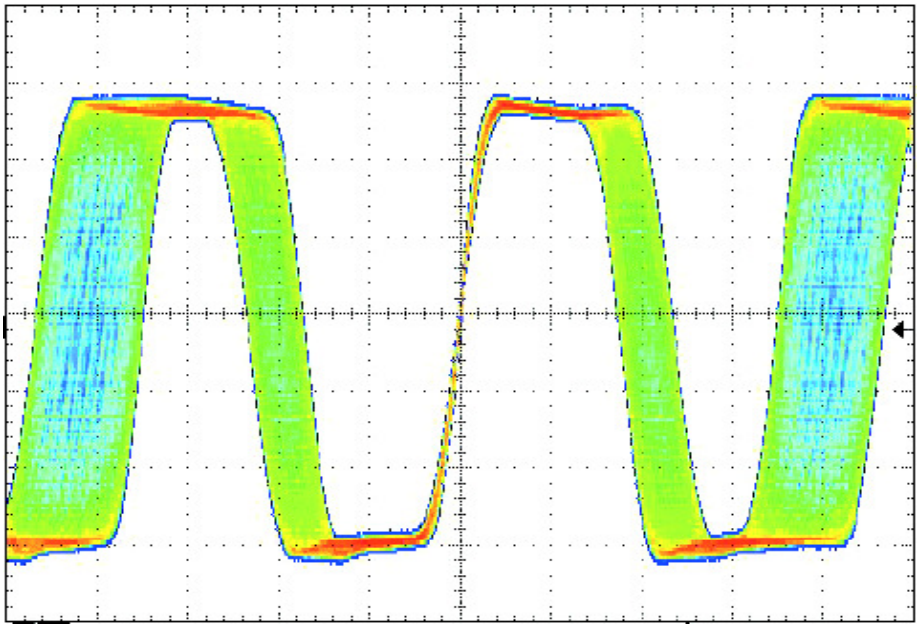


Abb. 4.12 Taktsignal mit starkem Clock-Jitter

Das Taktsignal wird hier mit einem Oszilloskop an einer Flanke getriggert (Bildmitte). Im Verhältnis zu diesem Zeitpunkt variieren die vorhergehende und die nachfolgende steigende Flanke sehr stark in ihrem zeitlichen Abstand. Dieser Effekt wird als *Clock-Jitter* bezeichnet und ist häufig direkt auf die takterzeugenden Oszillatoren zurückzuführen. Das obige Beispiel ist jedoch ein sehr extremer Fall, der hier nur zur Verdeutlichung gewählt wurde. So starken Clock-Jitter erzeugen in der Regel nur Bauteile, die extra dafür ausgelegt wurden. Sogenannte *Spread Spektrum* Oszillatoren nutzen den beschriebenen Mechanismus, um geringere bzw. spektral weniger stark konzentrierte elektromagnetische Emissionen zu verursachen. Für eine exakte Datenverarbeitung an Schnittstellen zwischen analogen und digitalen Signalen sollte Clock-Jitter unbedingt vermieden werden, weil dieser Fehler nicht oder nur sehr aufwendig zu korrigieren ist.

In einem Nachrichtenübertragungssystem können solche Taktungenauigkeiten sofort als zusätzliches Rauschen wahrgenommen werden, z.B. durch eine zusätzliche Unschärfe im Konstellationsdiagramm des Empfängers.

Noch deutlicher wirken sich Abtastfehler in Funkübertragungssystemen aus, bei denen die analogen Signale im Basisband als Real- und Imaginärteil verarbeitet werden. Hier arbeiten jeweils zwei Wandler parallel, was die Störeinflüsse addieren kann und einen Fehler der beiden Signalanteile relativ zueinander verursacht. In jedem Datenpfad kann es zu einer Verschiebung der Abtastzeitpunkte gegeneinander kommen, zu unterschiedlichem Clock-Jitter oder zu differierenden Amplituden. Diese Effekte sind sehr ähnlich zu der aus Mischern bekannten *IQ-Imbalance*, bei der

es auch zu einer Phasenverschiebung der beiden Signalanteile gegeneinander und/oder zu relativ zueinander verfälschten Amplituden kommt. Zum Teil können diese Einflüsse in digitalen Empfängern wieder herausgerechnet werden [STE05][TUB03], was den Verarbeitungsaufwand jedoch erheblich erhöht. Ein sorgfältiges Platinendesign und die inzwischen für die Nachrichtentechnik vielfältigen verfügbaren Dual-Wandler helfen, die IQ-Unausgewogenheit zu minimieren.

5 Komponenten eines OFDM-Systems

Die physikalische Schicht eines OFDM-Übertragungssystems setzt sich aus einer Vielzahl von Funktionsblöcken zusammen. Die in Abb. 5.1 dargestellte Struktur zeigt einen möglichen Aufbau, wie er in vielen modernen Standards zur kabellosen OFDM-Datenübertragung eingesetzt wird.

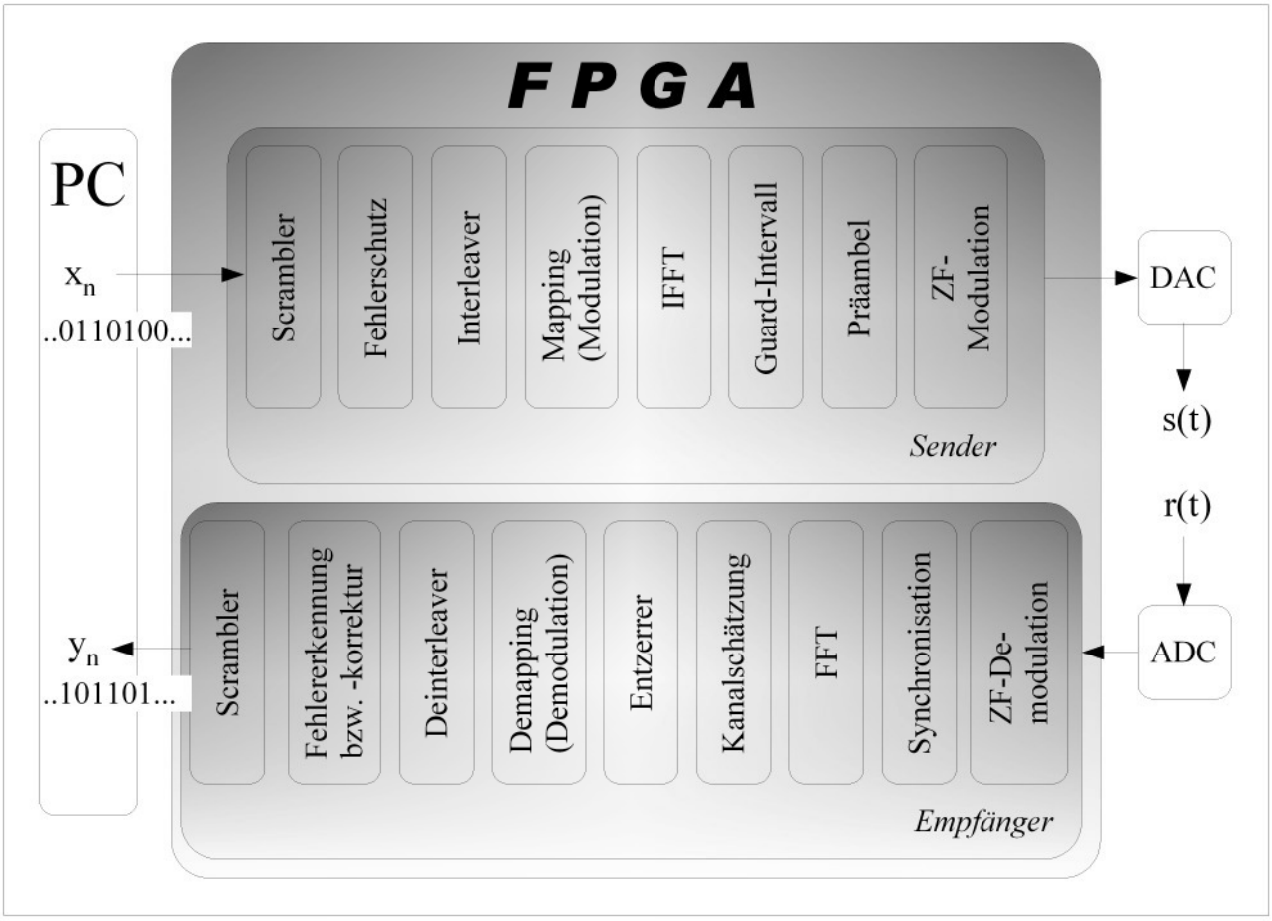


Abb. 5.1 Blockschaltbild einer OFDM-Übertragungsstrecke

In diesem Kapitel werden die einzelnen Module des im Blockschaltbild skizzierten Senders und Empfängers detailliert vorgestellt und untersucht. In dem besprochenen Testaufbau sind alle

Module von Abb. 5.1 außer den D/A- und A/D-Wandlern in einem FPGA enthalten. Der Entwurf der einzelnen Komponenten orientiert sich stark an den WLAN-Standards IEEE802.11a [IEE99] und HiperLAN/2 [HIP01]. Die wichtigsten Parameter der OFDM-Kette sind:

- Fehlerschutz durch einen Faltungscode der Rate ½ mit einer Einflusslänge von Sieben,
- Einsatz von Scrambler und Interleaver,
- vorgesehene Schemata beim Mapping sind BPSK, QPSK, 16-QAM und 64-QAM,
- FFT-Länge: 64,
- Guard-Intervalllänge: 16 Samples,
- Übertragungsbandbreite: 20 MHz,
- eine präambelbasierte Synchronisation und Kanalschätzung,
- Symboldauer von 4µs (3,2µs Nutzdaten plus 0,8µs Guard-Intervall),
- Subträgerabstand von 312,5 kHz.

Da alle Elemente der OFDM-Strecke als VHDL-Programm entwickelt werden, sind sie beliebig veränderbar und erneut in das FPGA ladbar. Ein solches System bietet die Möglichkeit, einzelne Parameter gezielt zu variieren und den Einfluss auf die OFDM-Kette zu untersuchen. Das Ziel ist dabei die Optimierung der unterschiedlichen Komponenten. Im Folgenden werden insbesondere die Parameter untersucht, die bei der FPGA-Implementierung festzulegen sind. Die Rechengenauigkeit ist z.B. ein solches Kriterium, welches wiederum Einfluss auf den FPGA-Ressourcenbedarf und die maximalen Taktraten hat. Solche Parameter sind in den WLAN-Standards nicht vorgegeben und spielen dennoch für den realen Entwurf des OFDM-Systems eine entscheidende Rolle.

In den anschließenden Unterkapiteln werden die implementierten Funktionseinheiten im Detail dargestellt. Die Arbeitsweise, der Einfluss auf die Performance des gesamten Übertragungssystems und Variationsmöglichkeiten werden aufgezeigt. Insbesondere wird dabei das Potenzial zur Optimierung bezüglich Platzbedarf und Geschwindigkeit erörtert. Ein entscheidendes Kriterium bei allen Variationen der FPGA-Module ist der Einfluss auf die Bitfehlerrate (BER) der gesamten OFDM-Strecke. Es geht hierbei um die Frage, wie ein Element beschleunigt oder verkleinert werden kann, ohne die BER signifikant zu verschlechtern. Es wird untersucht, ob die Flexibilität und/oder Funktionalität erhöht werden kann, ohne die Module deutlich langsamer oder größer zu machen.

Die nachfolgenden Unterkapitel behandeln die komplementären Funktionseinheiten von Sender und Empfänger jeweils gemeinsam. Dies vereinfacht die Diskussion von Designaspekten, bei denen die jeweiligen Baugruppen aufeinander abgestimmt sein müssen.

5.1 Scrambler

Die Scrambler sind von der Datenquelle bzw. -senke her gesehen die ersten Elemente der OFDM-Signalverarbeitungskette. Hier wird die eingehende Bitfolge durch ein festgelegtes Schieberegister transformiert. Ein Polynom definiert einen Pseudonoisegenerator, dessen Bitausgangsfolge mit dem Datenstrom über die XOR-Funktion verknüpft wird. Der Scrambler vermeidet so gleich bleibende Signalmuster und verwürfelt die Nutzdaten zu einer Pseudozufallsfolge. Das Verhältnis von Spitzen- zu Durchschnittswert (*Peak to Average Ratio*, PAR) wird dadurch in der Regel kleiner. Das PAR ist wiederum wichtig im Hinblick auf die Ansprüche an die Lineratiät der analogen Baugruppen.

Der Einfluss des Scramblers ist jedoch schlecht unmittelbar zu beobachten. Zum einen müssen die Nutzdaten noch korreliert sein, um von der Verwürfelung direkt zu profitieren. Ein zufälliges Bitmuster als Testsignal oder eine effiziente Quellencodierung reduziert den Einfluss des Scramblers auf das PAR. Zum anderen ist der Nutzen in der Regel abhängig von höheren Protokollschichten, die eine erneute Übertragung fehlerhafter Pakete auslösen (*Automatic Repeat Request*, ARQ). Ist ein OFDM-Symbol oder ein größerer Datenrahmen z. B. aufgrund eines ungünstigen *Peak-to-Average*-Verhältnisses verfälscht worden, wird eine Wiederholung derselben Daten durch den Scrambler anders verwürfelt, sodass mit hoher Wahrscheinlichkeit das PAR verändert und ggf. verbessert wird.

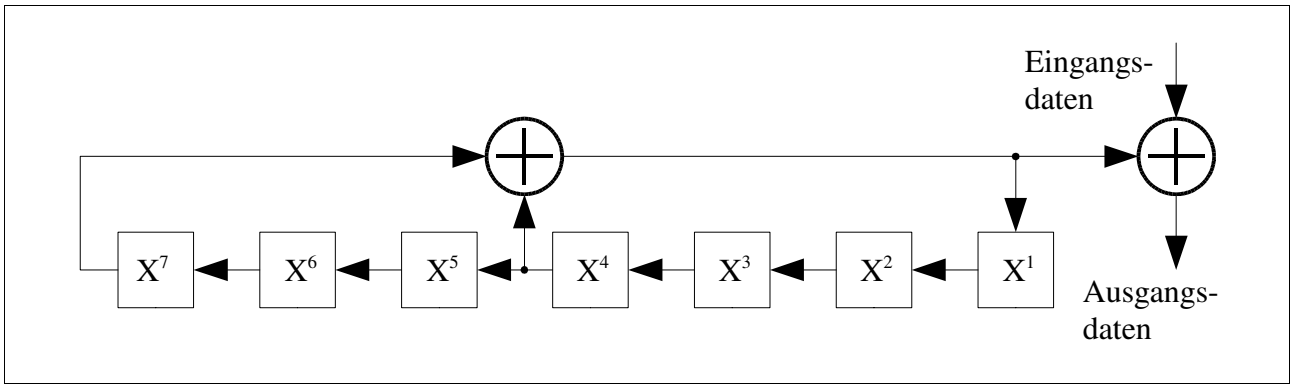


Abb. 5.2 Blockschaltbild eines Scramblers mit sieben Flipflop-Speichern

Dies macht deutlich, dass die Reinitialisierung des Scramblers nicht zu häufig oder nicht in jedem Fall auf die gleiche Weise ausgeführt werden sollte, damit dieser bei einer Paketwiederholung einen anderen Ausgangsstatus hat. Im Rahmen dieser Arbeit wird ausschließlich die Bitübertragungsschicht (engl. P*hysical Layer*) des OSI-Referenzmodells untersucht. Eine Performanceuntersuchung des Zusammenspiels von Scrambler und ARQ (Teil der Sicherungsschicht, engl. D*ata Link layer*) wird nicht gemacht, weil es nicht im direkten

Zusammenhang mit dem OFDM-System innerhalb des FPGAs steht. In [LI00] werden diese schichtübergreifenden Techniken genauer betrachtet.

Die Realisierung des Scramblers in einem FPGA lässt sich sehr effizient aus wenigen Flipflops und XOR-Gliedern aufbauen. Abb. 5.2 zeigt das Blockschaltbild eines HiperLAN/2-Scramblers. Der Descrambler ist mit dem Scrambler identisch. Die Umkehrfunktion, die einen verwürfelten Datenstrom wieder in die Ausgangsfolge transformiert, ist wieder eine XOR-Verknüpfung mit der gleichen Pseudonoisefolge, die durch einen identischen Generator erzeugt wird. Ein zweimaliges XOR (einmal im Sender und einmal im Empfänger) mit einer Null lässt das Ausgangsbit jeweils unverändert, während das zweimalige XOR mit einer Eins einer doppelten Invertierung entspricht und damit auch das Originalbit rekonstruiert. Abb. 5.3 zeigt ein Beispiel, bei dem einige Bits mit der Scramblerfolge transformiert werden. Der gesendete Bitstrom steht in der mittleren Zeile. Anschließend wird die Operation im Empfänger mit der gleichen Transformation wieder rekonstruiert. Längere Folgen von Nullen und Einsen können so mit hoher Wahrscheinlichkeit vermieden werden.

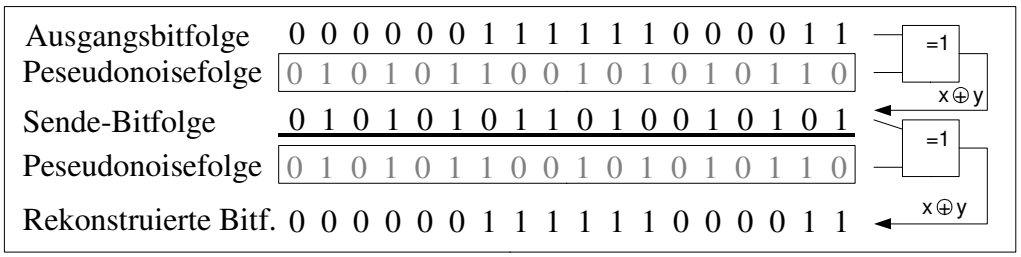


Abb. 5.3 Scrambler-Beispiel

Eine fehlerhafte Synchronisation von Scrambler und Descrambler ist jedoch fatal und führt zu vollständig unbrauchbaren Datenbits. Daher ist beim Systementwurf unbedingt darauf zu achten, dass keine Asynchronität auftreten kann. Eine regelmäßige Initialisierung von Scrambler und Descrambler begrenzt darüber hinaus die Länge der möglicherweise durch versetzte Bitfolgen zerstörten Datenblöcke. Um zu verhindern, dass wiederholte Datenpakete durch den Scrambler gleich verarbeitet werden, wird dieser z.B. nach dem HiperLAN/2-Standard nicht statisch initialisiert. Die Initialisierungsvorschrift ist festgelegt als $(X^7X^6X^5X^4X^3X^2X^1)=(111n^4n^3n^2n^1)$, wobei die Bits $n^1..n^4$ das *Frame Counter Field* aus dem Datenrahmen (dem *Broadcast Channel*, BCH) bilden, welche unverwürfelt übertragen werden [HIP01]. Diese Initialisierung erfolgt jeweils neu für die Datenblöcke der Sicherungsschicht, den *Protocol Data Units* (PDUs), die 9 und 54 Bytes lang sind [DOU02].

Tabelle 5.1 stellt die wichtigsten Merkmale der Schaltungsstruktur zusammen, die sich bei der Implementierung eines solchen Scramblers in einem FPGA ergeben. Im Vergleich zu den in den folgenden Kapiteln betrachteten Komponenten ist diese Verwürfelung sehr aufwandsgünstig und

macht nicht einmal 1% der FPGA-Auslastung des betrachteten OFDM-Gesamtsystems aus (vgl. auch Kap.6.4).

	Scrambler
Slices	20
Flipflops	16
LUT	34
Taktrate	198,57 MHz

Tabelle 5.1 Implementierungsdaten des Scramblers - dieses Modul ist für Sender und Empfänger vollkommen identisch.

Der Ressourcenverbrauch zur Integration einer solchen Verwürfelung im FPGA ist minimal. Die dabei notwendige genaue Synchronität der Bitfolgen ist in einem OFDM-System aufgrund der blockweisen Verarbeitung leicht einzuhalten. Allerdings ist das Verwürfeln nur bei deutlich korrelierten Datenfolgen und/oder bei der Nutzung von ARQ wirksam.

5.2 Fehlerschutz

In der heutigen Datenübertragung werden bei fast allen Systemen effiziente Algorithmen zur Fehlererkennung und zur Fehlerkorrektur (engl. *Forward Error Correction*, FEC) eingesetzt. Die Fehlerschutzmechanismen sind in der Regel unabhängig vom verwendeten Übertragungsverfahren. Eine Implementierung der FEC wird im Rahmen dieser Arbeit aus folgenden Gründen analysiert:

- Die betrachteten OFDM-Standards HiperLAN/2 und IEEE802.11a beschreiben die FEC als festen Bestandteil der physikalischen Schicht,
- die FPGA-Implementierung bietet eine Vielzahl von Parametern, zu deren Auswahl die folgenden Untersuchungen eine Hilfestellung bieten,
- der gesamte Block des vorgeschlagenen Fehlerschutzes ist auf ein OFDM-System abgestimmt,
- die Implementierung der Fehlerkorrektur im Empfänger beansprucht einen wesentlichen Teil der Ressourcen des Gesamtsystems,
- die Umsetzung der Faltungscode-FEC ist in einem FPGA effizient realisierbar.

Der Fehlerschutz besteht in der betrachteten Verarbeitung aus drei Komponenten: Dem Faltungscoder und -decoder als eigentlichem Mechanismus zur Fehlerkorrektur, der Punktierung zur

Anpassung der Coderate[1] R_C und dem Interleaver, der zur Dekorrelation von Büschelfehlern eingesetzt wird.

5.2.1 Fehlerschutz mit Faltungscodes

Zur Fehlerkorrektur nach [HIP01] wird ein Faltungscoder der Gedächtnislänge m=6 verwendet. Dabei werden die zu übertragenden Codebits aus dem aktuellen Informationsbit und einer Anzahl von m Vorgängerbits ermittelt. Als Einflusslänge (*Constraint Length*) werden alle Bits bezeichnet, aus denen die Codewörter generiert werden, also das Gedächtnis plus das aktuelle Bit (m+1). Die folgende Skizze zeigt die Umsetzung des Faltungscoders im Detail:

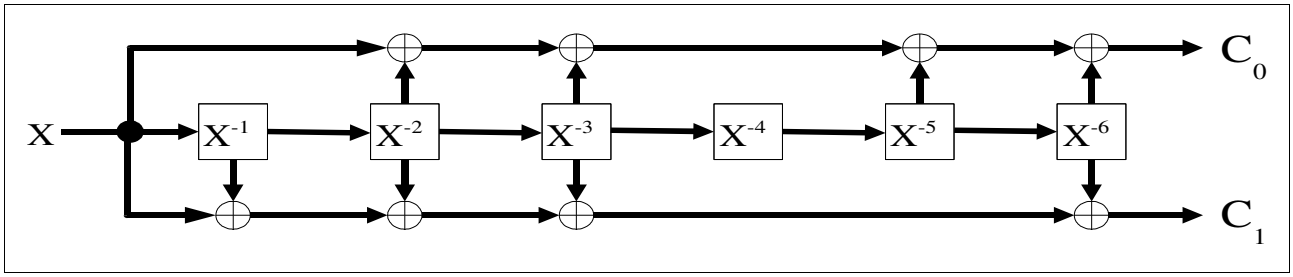


Abb. 5.4 Faltungscoder mit sechs Flipflops

Die Skizze macht deutlich, dass zur Implementierung des Coders nur sechs Flipflops und einige XOR-Gatter notwendig sind. Diese Schaltung kann im FPGA folgerichtig sehr leicht umgesetzt werden, verbraucht nur wenige Ressourcen und kann mit sehr hohen Taktraten arbeiten. Der dargestellte Faltungscoder generiert pro eingehendem Informationsbit zwei ausgehende Codebits. Daraus ergibt sich eine Coderate R_C von ½. Der Fehlerschutz kann verbessert werden, wenn mehr Redundanz hinzugefügt wird bzw. die Coderate verkleinert wird. Dadurch sinkt jedoch unmittelbar die Netto- bzw. Nutzdatenrate der Übertragung.

Eine Aufwandsbetrachtung für den Fehlerschutz mit Faltungscodes ist wegen der sehr einfachen Encoderstruktur fast nur vom Decoder geprägt. Dieser verbraucht in der Regel deutlich mehr Ressourcen (Verhältnis ca. 200:1, vgl. Tabelle 5.2) und ist aufgrund der aufwendigeren Logik auch sehr viel kritischer im Timing.

Die eigentliche Fehlerkorrektur wird im Empfänger durch eine Maximum-Likelihood Decodierung vorgenommen. Eine besonders anschauliche Art, diesen Algorithmus zu implementieren, hat Viterbi 1967 [VIT67] vorgestellt. Da die Programmierung im FPGA eine Umsetzung dieser von Viterbi vorgeschlagenen Struktur ist, wird sie im Folgenden als Viterbi-Decoder bezeichnet.

1 Die Coderate beschreibt das Verhältnis von Eingangs- zu Ausgangsbitrate des Coders.

Der Viterbi-Algorithmus kann in drei Stufen (siehe Abb. 5.5) unterteilt werden. Zunächst wird in der *Metrik Processing Unit* (MPU) aus den empfangenen Codewörtern eine Metrik berechnet, die eine Wahrscheinlichkeit dafür beschreibt, dass eine Null oder Eins gesendet wurde.

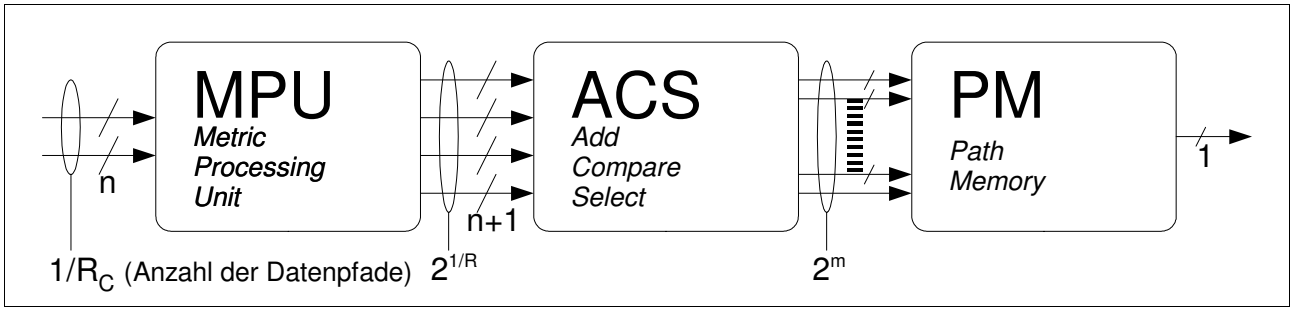


Abb. 5.5 Dreistufiges Blockschaltbild des Viterbi-Decoders

Die nachgeschaltete *Add-Compare-Select*-Einheit nutzt die Einflusslänge des Faltungscodes und ermittelt mit den Metriken aus der MPU eine Wahrscheinlichkeit für eine zusammenhängende Bitfolge. Da bei einem Faltungscode nur bestimmte Bitfolgen möglich sind, wird eine Empfangssequenz immer der ähnlichsten überhaupt möglichen Folge zugeordnet. Der Grad der Übereinstimmung ist dabei ein Maß für Wahrscheinlichkeit, dass die Bitmuster richtig empfangen wurden. In der ACS-Einheit werden für alle Bitsequenzen akkumulierte Metriken (Gesamtmetriken), die diese Übereinstimmung widerspiegeln, paarweise verglichen. Die jeweils größere bzw. wahrscheinlichere überlebt. Die ACS-Einheit gibt die Information der überlebenden Sequenz (engl. *survivor*) an den Pfadspeicher (PM, engl. *Path Memory*) weiter, der die Bitsequenzen zu den überlebenden Gesamtmetriken speichert. Nach einer festgelegten Verarbeitungstiefe, die der Länge des Pfadspeichers entspricht, können die Informationsbits mit der größten Wahrscheinlichkeit ausgegeben werden.

Die MPU hat $1/R_C$ Eingangswerte. Der implementierte Decoder verarbeitet *Soft Values*, also mehrere Bit breite (quantisierte) Codewörter. Diese enthalten neben der Information ob ein Codebit zu einer Null oder einer Eins entschieden wurde auch noch eine Gewichtung, mit welcher Wahrscheinlichkeit dieses Codewort korrekt ermittelt wurde. Die Gewichtung ist Teil der Demodulation (vgl. Kap. 5.3).

Die Eingangswerte des Decoders sind die letzte Schnittstelle in der OFDM-Kette, an der Zahlenwerte übergeben werden. Nach der Decodierung werden die entschiedenen Informationsbits ausgegeben. Die Darstellungsgenauigkeit (Wortbreite) der Soft Values, und damit die Genauigkeit, mit der die Fehlerkorrektur arbeiten kann, bestimmt, wie detailliert der Demodulator die Soft Values ermitteln muss. Wie der D/A-Wandler im Sender bestimmt der Decoder im Empfänger die Verarbeitungsgenauigkeit, die effektiv genutzt werden kann.

Die Quantisierung der Soft Values bestimmt die notwendige Rechengenauigkeit der vorgeschalteten Module und hat somit Einfluss auf den Entwurf der gesamten Empfängerstrecke.

Die Breite n_S der Soft Values hat Einfluss auf die Leistungsfähigkeit des Decoders. Wortbreiten von drei oder vier Bit sind jedoch in den meisten Fällen vollkommen ausreichend. Feiner quantisierte Werte ergeben kaum noch zusätzliche Gewinne. Abb. 5.6 zeigt den Einfluss der Wortbreite auf die BER-Performance.

Gegenüber der idealen Soft Decision gehen durch eine 3-Bit-Quantisierung ca. 0,25 dB verloren. Die Performance bei einer Auflösung von vier Bit fällt gegenüber einer exakten Rechnung kaum messbar ab.

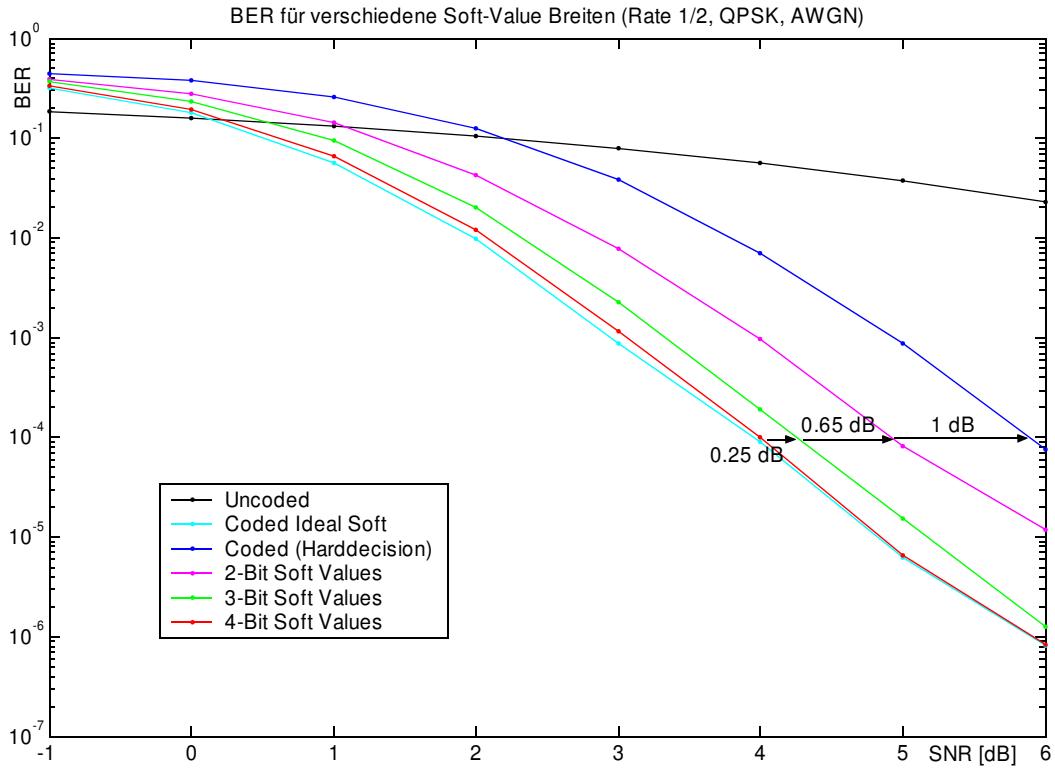


Abb. 5.6 Decoder-Performance in Abhängigkeit der Eingangswortbreite

Die quantisierten Soft Values c_0 und c_1 werden in der MPU ausgewertet. Es werden Metriken durch eine Addition bzw. Subtraktion aller zu einem Quellbit gehörenden eingehenden Soft Values gebildet. Für einen Halbratencoder gibt es vier mögliche Metriken:

$$M_1 = c_0 + c_1, \quad M_2 = -c_0 - c_1, \quad M_3 = c_0 - c_1, \quad M_4 = -c_0 + c_1 \qquad (5.1)$$

In Abhängigkeit von der Breite der Eingangswörter und der Soft-Value-Wortbreite n ergibt sich die folgende minimale Registerbreite M_{REG} innerhalb der MPU:

$$M_{REG} \geq \log_2\left(\frac{1}{R_C}\right) + n \qquad (5.2)$$

Es ist in jedem Fall sinnvoll, die von der MPU ermittelten Metriken in Registern zwischenzuspeichern, bevor sie an die nachgeschaltete ACS-Einheit weitergegeben werden. Durch ein solches Pipelining zwischen diesen beiden Modulen wird der zeitkritische Pfad verkürzt und der maximal mögliche Takt deutlich erhöht.

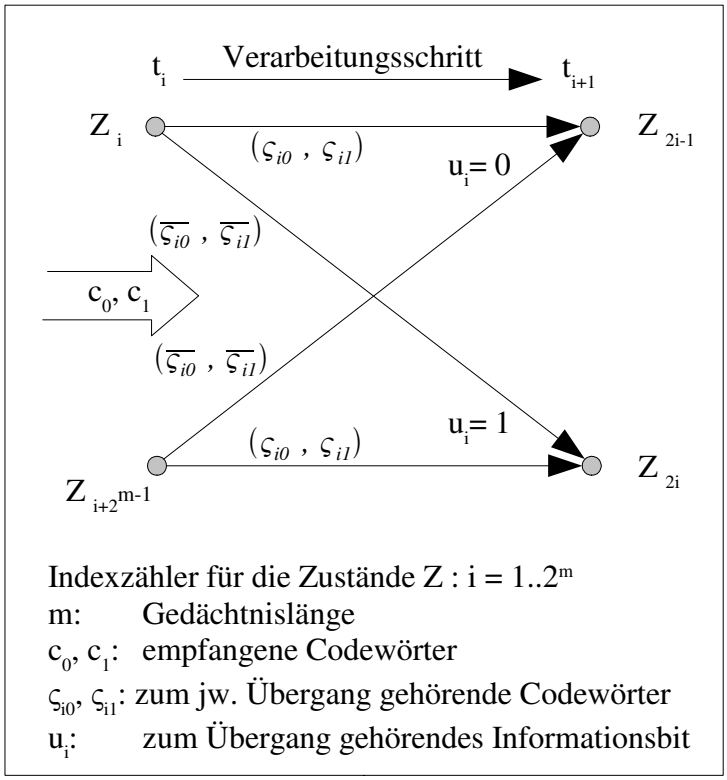


Abb. 5.7 Trellisstruktur (Butterfly)

Die Anzahl der MPU-Register ist abhängig von der Coderate. Für jede Kombination von Codewörtern wird eine Metrik errechnet. Die Gleichungen 5.1 zeigen ein Beispiel für zwei eingehende Soft Values. In diesem Fall ergeben sich vier unterschiedliche Metriken. Aufgrund der typischen Symmetrie, die in Abb. 5.7 dargestellt ist, wird in der Regel nur die Hälfte dieser Werte berechnet. Die anderen Werte sind jeweils Inverse, was bei der späteren Verarbeitung berücksichtigt wird. Die Zahl der benötigten Register ist in diesem Fall

$$A_{MPU} = 2^{\frac{1}{R_c}-1} \qquad (5.3)\ .$$

Die Zuordnung der Metriken zu den einzelnen Zuständen der ACS-Einheit ist durch die Schnittstelle zwischen den Elementen definiert. Für jeden der 2^m Zustände wird nur der Wert übergeben, der zu den Codewörtern des oberen Übergangs gehört. Der andere Wert ist aufgrund der Symmetrie der Trellisstruktur[2] das Inverse, was durch die Subtraktion des Übergabewertes berücksichtigt wird. Abb. 5.8 zeigt den Aufbau eines ACS-Elements, wie es für jeden der 64 Zustände in einem Viterbi-Decoder mit einer Gedächtnislänge von m=6 implementiert wird.

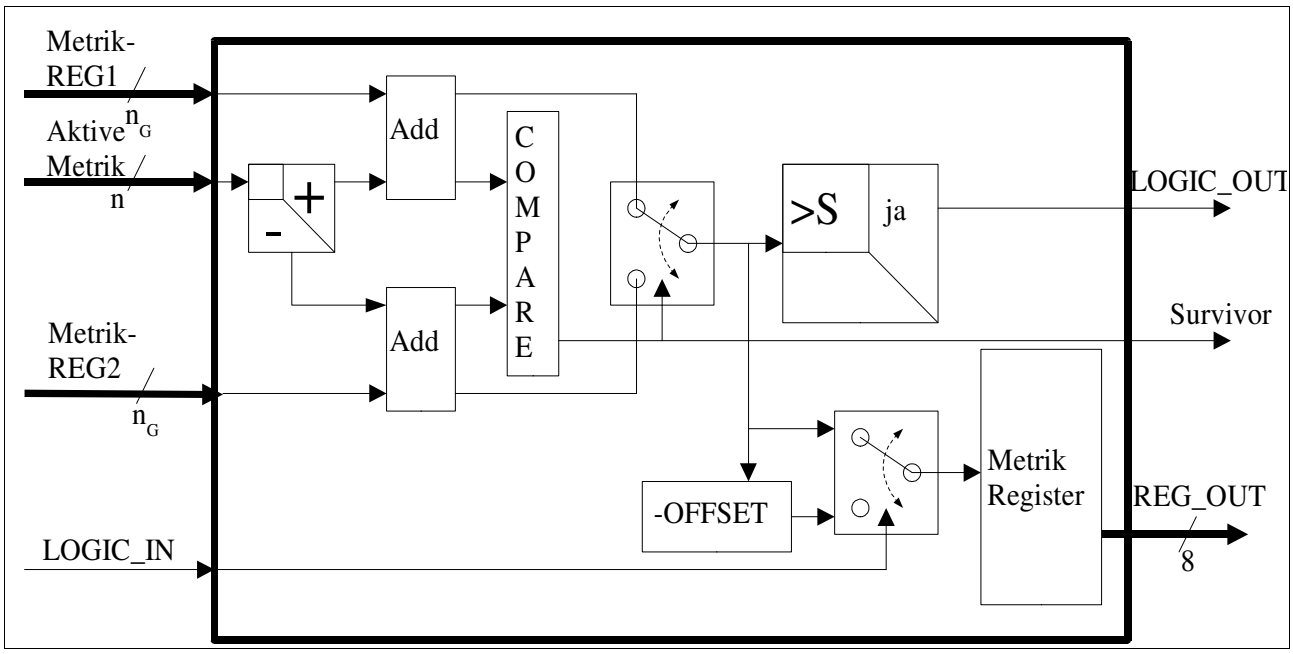


Abb. 5.8 Blockschaltbild der ACS-Einheit

2 Die Trellisstruktur bzw. das Trellisdiagramm ist eine Möglichkeit zur anschaulichen grafischen Darstellung solcher zustandsabhängigen Prozesse. Der Ausdruck Trellis kommt aus dem Englischen und steht für (Pflanzen-) Gitter/Spalier.

In diesem Block wird mit den in jedem Takt aktualisierten MPU-Metriken und den internen Summen gerechnet. Zur Unterscheidung werden diese im Folgenden entweder als aktuelle Metrik oder als Gesamtmetrik bezeichnet. In der ACS-Einheit werden durch das fortlaufende Aufaddieren der aktuellen Metriken die Gesamtmetriken für alle Zustände im Trellisdiagramm gebildet, welche die Wahrscheinlichkeit einer gesendeten Bitfolge repräsentieren. Dabei wird durch das Akkumulieren der Werte das Gedächtnis des Faltungscodes ausgenutzt. Bei fehlerfreien Codewörtern ist immer nur der Wechsel vom aktuell richtigen Zustand zu zwei möglichen Folgezuständen möglich. Bei einem korrekten Übergang wird die Gesamtmetrik größer, bei dem falschen Übergang wird die zugehörige Gesamtmetrik um denselben Betrag verkleinert.

Nach zwei Takten können aus einem korrekten Startpunkt vier Folgezustände erreicht werden, nach drei Takten acht usw.. Alle anderen Zustandswechsel werden zwar auch ausgewertet, wären aber nur durch fehlerhafte Übergänge, also verfälscht empfangene Codewörter, zu erreichen. Bei guter Übertragung haben sie somit eine geringere Gesamtmetrik bzw. Wahrscheinlichkeit. Nach m Takten ist von jedem Ausgangszustand jeder Folgezustand zu erreichen. Dies ist die Gedächtnislänge des Codes, die besagt, dass ein eingehendes Informationsbit auf eine nachfolgende Bitfolge dieser Länge Einfluss hat. Diese Größe ist wichtig für die Ermittlung der minimalen Registerbreite M_{ACS} innerhalb der ACS-Einheit. Doch zunächst soll die Arbeitsweise dieser Einheit erläutert werden.

Für jeden Zustand werden die aktuellen Metriken zu den beiden möglichen Vorgänger-Gesamtmetriken hinzugefügt (*Add*). Anschießend werden diese beiden Werte verglichen (*Compare*) und der Übergang mit der größeren Wahrscheinlichkeit wird ausgewählt (*Select*). Das bedeutet, dass als Ausgangssignale die Information des überlebenden Pfades und die aktuell ermittelte größere Gesamtmetrik weitergegeben werden. Bei der Realisierung dieser Einheit muss berücksichtigt werden, dass die Gesamtmetriken mit guter Übereinstimmung fortlaufend immer größer werden. Um den erlaubten Wertebereich nicht zu verlassen, ist es daher nötig, alle Gesamtmetriken wiederkehrend zu verkleinern. Dazu wird überwacht, ob eine der Metriken eine Schwelle überschreitet. Ist dies der Fall, wird die Reduktion um einen konstanten Faktor ausgeführt. Dies ist ohne Einschränkung der Leistungsfähigkeit möglich, weil nur der Abstand zwischen großen und kleinen Gesamtmetriken für die Funktion wichtig ist. Negative Überläufe werden durch Clipping abgefangen. Diese Zustände sind auf Grund des kleinen Zahlenwertes ohnehin so unwahrscheinlich, dass das Clipping keinen Einfluss auf den Decodiererfolg hat. Die Abb. 5.9 zeigt beispielhaft den Werteverlauf solcher Gesamtmetriken.

Die ACS-Operation bestimmt die maximale Geschwindigkeit des Viterbi-Decoders. Da die Kalkulation rekursiv ausgeführt wird, also die Gesamtmetrik des vorherigen Taktes als Eingangswert für die aktuelle Berechnung benötigt wird, kann diese Einheit nicht durch Pipelining beschleunigt werden. Der kritische Pfad ergibt sich hier aus der Durchlaufzeit für die beiden

Additionen und der Logik zur Auswahl des größeren Wertes. Außerdem muss entschieden werden, ob die Reduktion der Metriken nötig ist.

Die Logik zur Reduktion muss nicht unbedingt im aktuellen Takt ausgewertet werden. Es ist möglich, eine Schwelle für die Gesamtmetriken so zu definieren, dass die Reduktion erst im übernächsten Schritt ausgeführt werden muss. Damit hat diese Logik einen vollen Takt für die Auswertung zur Verfügung. Der Nachteil ist dann jedoch, dass die Reduktion nicht mehr zum frühestmöglichen Zeitpunkt ausgeführt wird. Somit kann die Gesamtmetrik größere Maximalwerte annehmen und dementsprechend muss auch die minimale Registerbreite wachsen. Breitere ACS-Register reduzieren wegen der größeren Schaltungslogik die mögliche Geschwindigkeit. Hier ist im Einzelfall abzuwägen, ob kleinere Register oder eine zeitversetzte Reduktion Vorteile bieten.

Bei dem in dieser Arbeit implementierten Referenzsystem wurde ein Viterbi-Decoder mit vier Bit breiten Soft Values implementiert, der mit einer Registerbreite von sieben Bit im ACS-Modul arbeitet. Die Reduktion wird bei einer Schwellwertüberschreitung erst einen Taktzyklus nach der Detektion der Überschreitung umgesetzt. Das folgende Beispiel zeigt die entsprechenden Zeilen aus dem VHDL-Quellcode, der hier jedoch etwas vereinfacht ist, wobei z.B. das Abfangen von Werteüberläufen nicht enthalten ist:

```
if DO_REDUCE_IN='1' then
  COMP1 := UPPER_METRIK_OLD(6)  & UPPER_METRIK_OLD  + METRIK - 32;
  COMP2 := LOWER_METRIK_OLD(6) & LOWER_METRIK_OLD - METRIK - 32;
else
  COMP1 := UPPER_METRIK_OLD(6)  & UPPER_METRIK_OLD  + METRIK;
  COMP2 := LOWER_METRIK_OLD(6) & LOWER_METRIK_OLD - METRIK;
end if;

if COMP1 >= COMP2 then
  SURVIVOR <= '0';
  GESAMT_METRIK <= COMP1;
else
  SURVIVOR <= '1';
  GESAMT_METRIK <= COMP2;
end if;
```

VHDL-Codebeispiel für Add-Compare-Select des Viterbi-Decoders

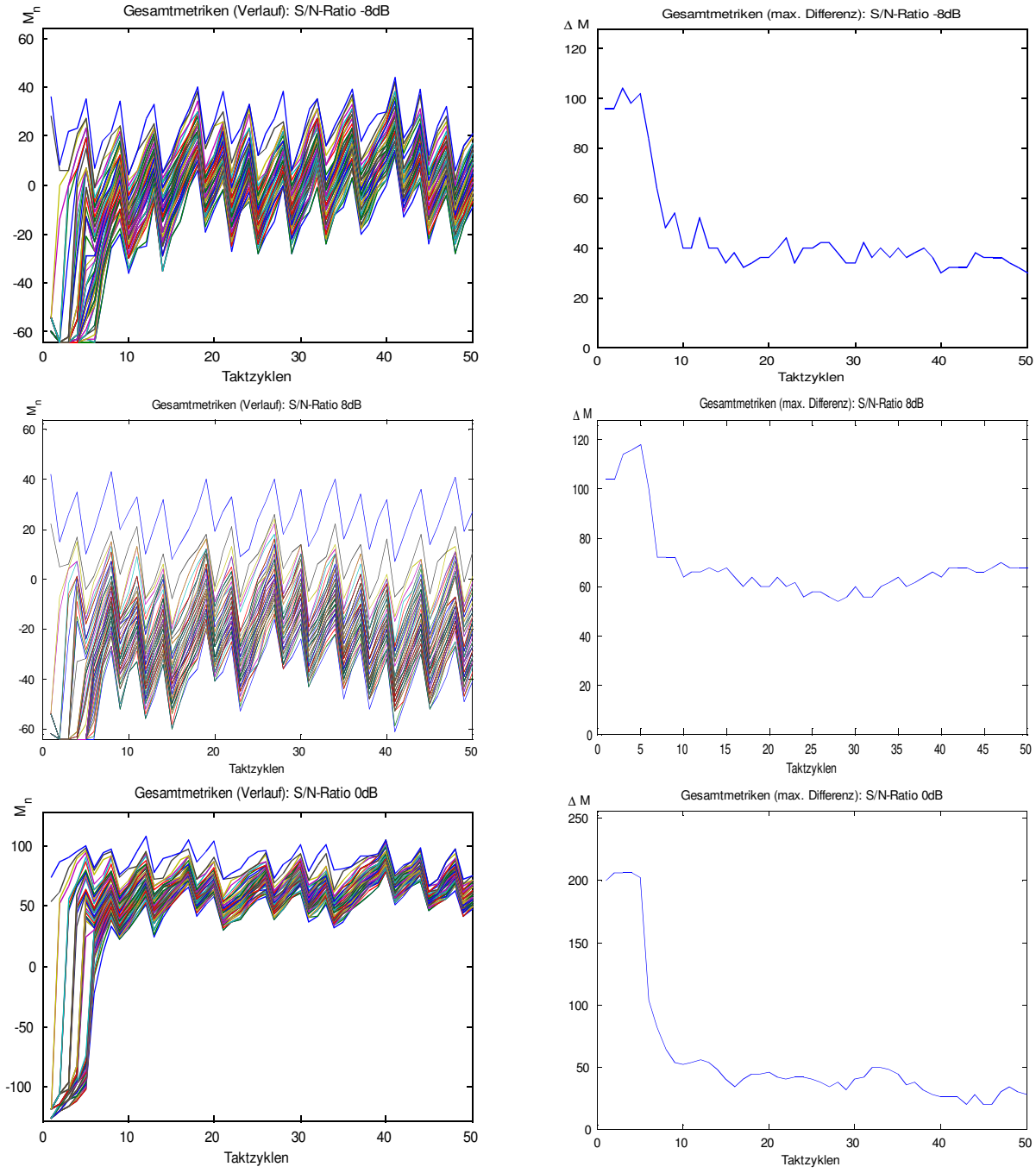


Abb. 5.9 Entwicklung der Viterbi-Metriken über der Zeit

Abb. 5.9 zeigt den Werteverlauf der Gesamtmetriken. In der grafischen Darstellung wird deutlich, dass eine Erhöhung der Wortbreiten vom Algorithmus nicht genutzt wird. Auf der linken Seite sind in den oberen beiden Grafiken die Metriken in einem für 4-Bit Soft Values optimalen Bereich zwischen 63 und -64 dargestellt (also 7-Bit interne Wortbreite). Ganz oben ist der Kanal stark verrauscht, in der Mitte ist hingegen ein Fall mit großem S/N-Verhältnis angenommen. Der

ausgewählte Pfad (Survivor) ist jeweils der obere, welcher sich im rauscharmen Fall sichtbar absetzt. Die Grafik unten links zeigt die Rechnung mit höherer Auflösung (8Bit), wobei deutlich nur die obere Hälfte genutzt wird. Abgesehen von der Anfangsphase, wo einige Werte auf -128 initialisiert sind, werden die kleinen Werte nicht erreicht. Hier zeigt sich auch keine Verbesserung der Bitfehlerrate.

Auf der rechten Seite ist jeweils für die betrachteten Fälle der Abstand zwischen größter und kleinster Gesamtmetrik dargestellt. Auch hier spiegelt sich das beschriebene Verhalten wieder. Im stark verrauschten Fall (oben rechts) sind die Unterschiede jedoch geringer, weil die kleinen Gesamtmetriken durch vereinzelte Fehler aufgewertet werden. In allen Fällen sind die Abstände zu Anfang wegen der Initialisierung noch sehr groß.

Die ausschlaggebende Größe für die Registerbreite innerhalb der ACS-Einheit ist der maximale Abstand, der sich zwischen größter und kleinster Gesamtmetrik ergeben kann. Sie errechnet sich aus der Einflusslänge multipliziert mit der größtmöglichen aktuellen Metrik, die wiederum von der Anzahl und der Wortbreite der Codebits abhängt.

Beim Beginn des Codierens ist der Faltungscoder in der Regel mit Nullen initialisiert. D.h., dass der Nullzustand (Z_0) der einzig mögliche ist. Dies wird bei der Initialisierung der Gesamtmetriken berücksichtigt, indem M_0 auf einen recht großen Wert (im implementierten Beispiel auf +32) und M_1 .. M_{63} auf den Minimalwert (-64) gesetzt werden. Auch im laufenden Betrieb kann durch eine Folge von sechs zusammenhängenden Nullen die Initialisierung in den Nullzustand erzwungen werden. Dies wird im Zusammenhang mit solchen Schieberegistern auch als Terminierung bezeichnet. Ist dem Empfänger der Zeitpunkt einer solchen Terminierung bekannt, kann der aktuelle Zustand mit absoluter Sicherheit bestimmt werden. Dies sollte dann immer zur Neuinitialisierung der Gesamtmetriken genutzt werden.

Die ACS-Einheit gibt die Informationen der überlebenden Pfade an den *Path Memory* weiter. Hier werden die zu den Zustandsübergängen gehörenden Informationsbits ermittelt und zwischengespeichert. Diese Form der Realisierung nennt sich Register Exchange Algorithmus[3] (REA). Dabei sind in jeder Zeile des Pfadspeichers alle Informationsbits zu dem gesamten Vorgängerpfad gespeichert. Die Länge dieses Speichers (Pfadlänge) ist wichtig für die Leistungsfähigkeit des Viterbi-Algorithmus und definiert die Zahl der Takte, nach der die decodierten Bits ausgegeben werden (vgl. Abb. 5.10).

3 Es gibt alternativ den *Trace Back* Algorithmus, welcher die vollzogenen Übergänge in der Trellisstruktur speichert. Die Informationsbits werden erst anschließend durch das Rückverfolgen der Pfade ermittelt. Dieser Algorithmus ist für eine Implementaion im FPGA ineffizient und wird daher hier nicht betrachtet.

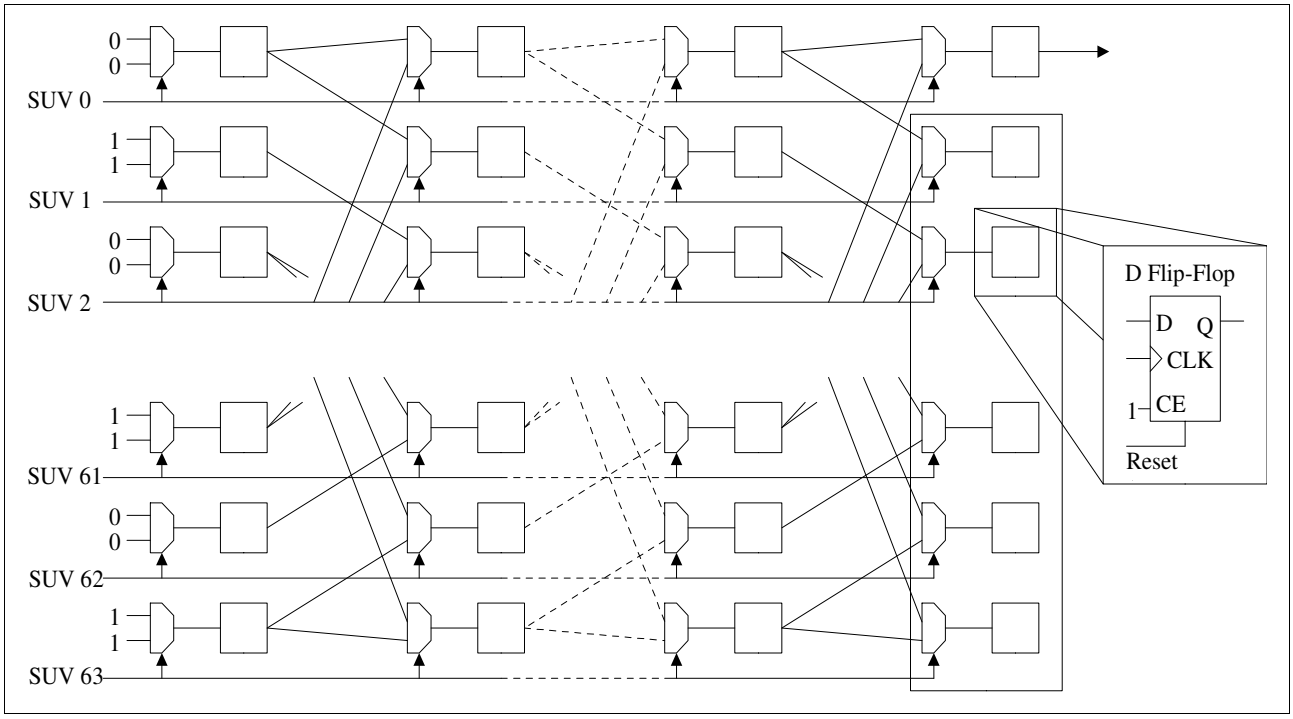


Abb. 5.10 Blockschaltbild des Pfadspeichers (Register-Exchange-Algorithmus)

Der Aufbau des Pfadspeichers ist in einem FPGA in effizienter Weise möglich. Da hier nur einzelne Bits vorgehalten werden, wird nur ein Flipflop pro Speicherelement benötigt. Außerdem ist jeweils ein Multiplexer nötig, der abhängig von dem überlebenden Pfad das Vorgängerbit auswählt.

Der Pfadspeicher kann bei der Implementierung bezüglich der Geschwindigkeit kaum weiter optimiert werden, ist aber sicherlich immer ausreichend schnell in Bezug auf die ACS-Schaltung. Der Ressourcenverbrauch hängt von der Zahl der Zustände im Trellis und der gewählten Pfadlänge ab. Außerdem gibt es zwei unterschiedliche Verfahren zur Auswahl des Informationsbits am Decoderausgang. Beim *Single-State* oder auch *Zero-State*-Verfahren wird jeweils das decodierte Bit aus der ersten Zeile des Pfadspeichers entnommen. Beim *Best-State*-Algorithmus wird jeweils das Bit ausgewählt, dessen Zustand gerade die maximale Metrik besitzt. Bei beiden Algorithmen wird angenommen, dass nach einer ausreichenden Pfadlänge alle Pfade zusammenlaufen. Im hinteren Teil des Pfadspeichers stehen dann spaltenweise identische Bits, die sich bei der Maximum Likelihood Decodierung durchgesetzt haben. Die Best-State-Auswahl bietet in den wenigen Fällen, in denen die Pfadlänge nicht für eine eindeutige Entscheidung ausreicht, den Vorteil, dass die Bitfolge ausgewählt wird, die sich mit größter Wahrscheinlichkeit durchsetzen würde. Single-State-Entscheider treffen in einem solchen Fall im Hinblick auf die Wahrscheinlichkeit eine zufällige Auswahl.

Auf der anderen Seite kann bei dem Best-State-Verfahren eine kürzere Pfadlänge gewählt werden, um die gleiche Leistungsfähigkeit der Fehlerkorrektur zu erreichen. Abb. 5.11 zeigt einen

Vergleich der beiden Verfahren und bietet einen Überblick zu den notwendigen Pfadlängen. Weiterführende Untersuchungen zu unterschiedlichen Coderaten sowie Single- und Best-State-Implementierungen finden sich in [TOE04].

Der *Best-State*-Algorithmus erfordert jedoch im Vergleich zum Single-State-Verfahren zusätzliche Schaltungslogik zur Ermittlung der maximalen Gesamtmetrik. Durch die Verkürzung des Pfadspeichers (bei gleicher Performance) verbraucht diese Implementierung dennoch insgesamt weniger Ressourcen. In dem aufgebauten Referenzsystem wurde ein Viterbi-Decoder mit einem Best-State-Entscheider und einer Pfadlänge von 33 gewählt.

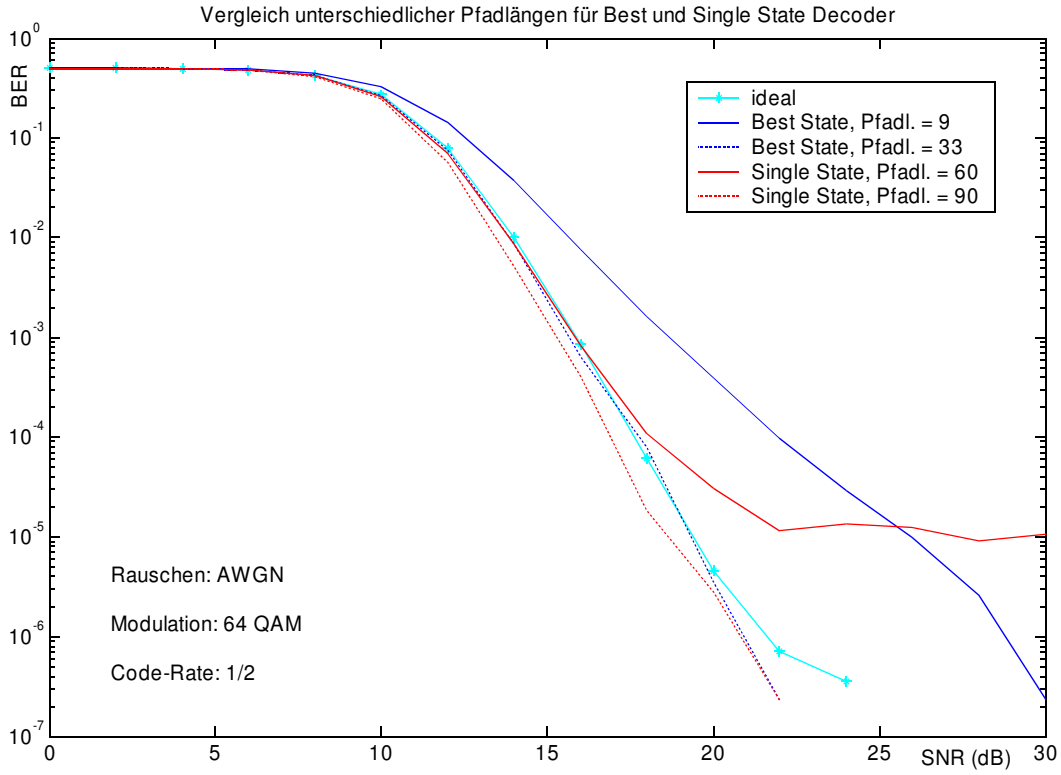


Abb. 5.11 Der Single-State-Algorithmus erfordert deutlich größere Pfadlängen als der Best-State-Decoder

Beim *Single-State*-Verfahren müssen die unmittelbaren Vorgänger der nicht ausgewerteten Zustände am Ende der Pfadmatrix nicht mehr gespeichert werden. Hier reduziert sich die Zahl der benötigten Speicherzellen minimal auf

$$R_{PM} = 2^m \cdot P_L - \sum_{i=0}^{m} 2^i \qquad (5.4)\,.$$

Der gesamte Viterbi-Decoder ist in Bezug auf alle Einzelkomponenten der OFDM-Kette ein sehr komplexes Modul. Eine optimale Auswahl der minimalen Registerbreiten und die Ausnutzung der Symmetrien führen zu einer schnellen Implementierung mit geringem Platzbedarf. Der wichtigste Faktor ist jedoch die Auswahl der Gedächtnislänge des Faltungscodes. Je größer sie ist, desto leistungsfähiger ist die Fehlerkorrektur. Allerdings steigt der Schaltungsaufwand exponentiell an.

Die Verlängerung der Gedächtnislänge um ein Element verdoppelt die Komplexität des Viterbi-Decoders.

Die betrachtete Implementierung des Viterbi-Decoders kann je nach Parametern einen Datendurchsatz von rund 90 MBit/s verarbeiten. Dies reicht für HiperLAN/2 und IEEE802.11a mit einem Maximum von 54 MBit/s vollkommen aus. Zukünftige Generationen werden jedoch deutlich höhere Datenraten erfordern, sodass Lösungen für noch schnellere Decoder notwendig werden.

Ein Ansatz ist die Radix-4-Struktur, bei der in jedem Takt aus doppelt so vielen Soft Values zwei Informationsbits decodiert werden. Somit ist auch bei geringeren Taktraten ein höherer Durchsatz möglich. Dabei werden aber die ACS-Elemente komplexer, weil dann jeweils vier Metriken verglichen werden müssen. Dies wiederum begrenzt den maximal den erlaubten Takt.

Der Radix-4-Viterbi-Decoder ist in [SUE93] beschrieben und hinsichtlich der FPGA-Implementierung betrachtet worden. Diese Struktur konkurriert jedoch mit dem Ansatz, zwei (oder mehr) Viterbi-Decoder parallel zu schalten und auch so die Datenrate zu erhöhen. Die Untersuchungen haben gezeigt, dass die Parallel-Lösung in fast allen Situationen der Radix-4-Struktur überlegen ist.

Ein Viterbi-Algorithmus kann mit angemessenem Schaltungsaufwand als 4-Bit Soft Value Version implementiert werden. Die Einsparungen von Ressourcen bei der 3-Bit-Version sind zwar durchaus vorhanden, fallen jedoch mit 1,5% (bezogen auf das Gesamtsystem) nicht so stark ins Gewicht. Auch die Geschwindigkeitszunahme ist nicht sehr groß, da eine gute Implementierung der 4-Bit-Ausführung schon über 90 MHz schnell ist, und damit recht dicht an den zu erwartenden Grenzen der Halbleitertechnologie liegt. Die 3-Bit-Version (ca. 94 MHz) wird also nur in wenigen Grenzfällen aufgrund von Implementierungslimits notwendig sein, ist aber auch von der Leistungsfähigkeit kaum unterlegen. Der zu erwartende Verlust liegt bei weniger als 0.2 dB [FRI94].

	Takt (MHz)	Slices	Flipflops	LUT	% Slices
Encoder	**180**	**20**	**12**	**33**	**0,1**
MPU	> 200	187	330	26	0.6
ACS	90	2602	607	4836	7,7
(ACS-Element)	*n/a*	*63*	*16*	*124*	*0,2*
PM	> 200	841	1609	1673	2,5
Best-State-Entscheider	> 200	794	950	940	2,3
Term.-Controller	> 200	21	12	40	0,1
Decoder gesamt	**90**	**4466**	**3508**	**7515**	**13,2**
Decoder (3Bit Soft Values)	94	*3,948*	3220	*6,430*	11,7

Tabelle 5.2 Implementierungsdaten des Faltungscoders und Viterbi-Decoders

Für zukünftige Decoder ist die Fehlerkorrektur bei gleicher Nettodatenrate noch weiter verbesserbar, wenn verkettete oder iterative Verfahren genutzt werden. Dazu kann z.B. der Viterbi-Decoder so erweitert werden, dass er wiederum mehrstufig quantisierte Soft-Werte ausgibt. Hier kann aus der Kenntnis der Gesamtmetriken bzw. dem Abstand zwischen den wahrscheinlichen Pfaden wiederum eine Gewichtung generiert werden, die in weiteren Decoderstufen nutzbar ist. Außerdem muss ein zusätzlicher Interleaver zwischen den einzelnen Stufen genutzt werden, um den Kanal quasi gedächtnislos zu machen. Solche SOVA-Decoder (*Soft Output Viterbi Algorithm*) sind z.B. in [HAG89] detailliert untersucht.

Jede weitere Stufe in einer Verkettung erhöht jedoch die Schaltungskomplexität jeweils um einen weiteren vollwertigen Decoder. Das ist eine enorme Steigerung des Platzbedarfs, die wohl erst für zukünftige FPGAs einen sinnvollen Einsatz erlaubt. Alternativ können auch Turbo-Codes genutzt werden, bei denen ein Decoder mehrfach durchlaufen wird. Turbo-Codes arbeiten rekursiv, d.h. der Decoderausgang wird rückgekoppelt. Bei gleichem Datendurchsatz des Empfangssystems muss eine solche Schaltungsstruktur mindestens doppelt so schnell getaktet werden können und das System hat aufgrund des weiteren Interleavers eine deutlich größere Latenzzeit. Wenn, wie in dem vorliegenden Referenzsystem, eine vollständige Signalverarbeitung in einem Chip implementiert werden soll, stößt man mit verketteten Codes oder *Turbo-Decodern* schnell an Kapazitätsgrenzen. Untersuchungen zur aufwandsreduzierten Turbodetektion finden sich in [HAA05].

5.2.2 Punktierung

Eine Punktierung kann eingesetzt werden, um die Coderate *R* zu vergrößern bzw. die durch das Codieren hinzugefügte Redundanz zu verkleinern. Insbesondere bei wechselnden Kanalsituationen kann so der Fehlerschutz flexibel an die Bedingungen angepasst werden.

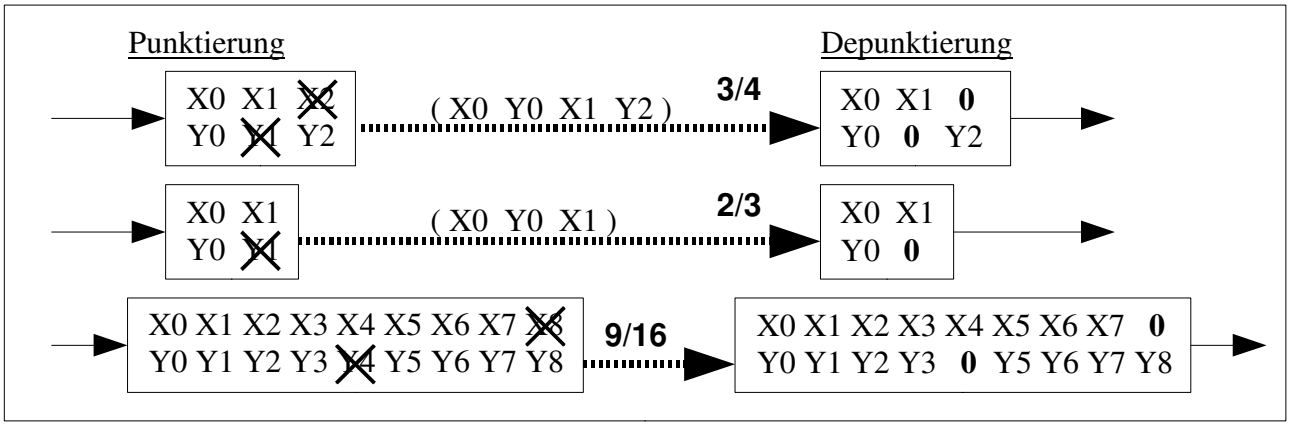


Abb. 5.12 Funktionsweise der Punktierung und Depunktierung sowie die resultierenden Coderaten bei einem vorgeschalteten Coder der Rate ½.

Abb. 5.12 zeigt schematisch die in den HiperLAN/2 und IEEE802.11a verwendeten Punktierungssequenzen für Halbratencoder zur Einstellung der Coderaten 3/4, 2/3 und 9/16. Die durchgestrichenen Codewörter werden verworfen und so der punktierte Datenstrom dementsprechend reduziert. Die nach der Punktierung resultierende Coderate R_{ges} ergibt sich aus der Rate des Coders R_{cod} und den Elementen der Punktierungsmatrix zu

$$R_{ges} = R_{cod} \frac{(Zahl\,der\,Matrixelemete)}{(Zahl\,der\,Matrixelemete - Zahl\,der\,Streichstellen)} \qquad (5.5).$$

Durch das Weglassen einzelner Codebits werden die Fehlerkorrektureigenschaften des Faltungscodes verschlechtert. In Kanalsituationen mit sehr guten Übertragungsbedingungen, bei denen es ohnehin fast nie zu Bitfehlern kommt, kann so jedoch der Datendurchsatz erhöht werden.

Abb. 5.13 zeigt, wie mit zunehmender Punktierung ein größeres SNR notwendig wird, um die gleiche Bitfehlerrate zu erzielen. Eine Auswertung dieses Simulationsergebnisses bei einer vorgegebenen BER von 10^{-3} zeigt, dass ausgehend von dem Rate-1/2-Code eine Erhöhung der Nutzdatenrate um 10% ein Plus von ca. 0,7dB beim SNR erfordert. Diese Faustformel ergibt z.B. für den uncodierten Fall, also einer Erhöhung der Nutzdatenrate um 100% gegenüber dem codierten Datenstrom, einen um 7dB höheren SNR-Wert.

Auch die weiteren Testpunkte dieser Formel entsprechen den Ergebnissen der Simulation (abgelesen bei der BER 10^{-3} und einem Referenz-SNR von 3dB für R=1/2): R=9/16 (+12,5%, 3,9dB), R=2/3 (+33,3%, 5,3dB), R=3/4 (+50%, 6,5dB) und R=1 (+100%, 10dB).

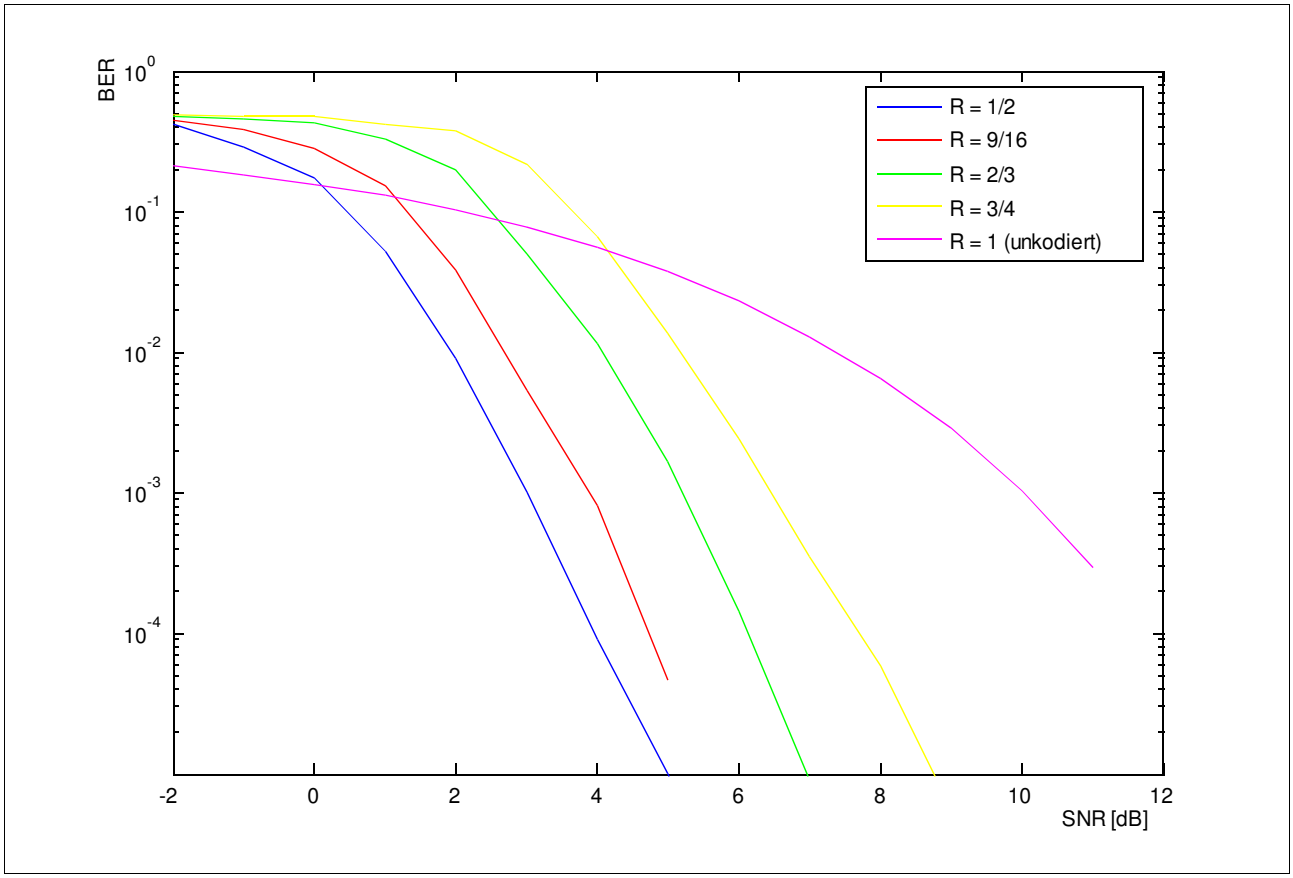


Abb. 5.13 Verschiedene Punktierungen im Vergleich (BPSK, Soft-Decodiert)

Die Implementierung der Punktierung im FPGA ist in einer sehr kleinen Schaltungsstruktur möglich. Die Punktierungstabellen sind i. d. R. klein und das Verwerfen von Codebits kann einfach durch die Steuerung eines ohnehin vorhandenen *Data-Valid*-Signals erfolgen, welches die entsprechenden Bits als ungültig kennzeichnet. Bei der Depunktierung, also dem inversen Prozess, wird an den Stellen, wo in der Punktierung ein Bit verworfen wurde, wieder ein Bit eingefügt. Im Regelfall wird eine Null oder, bei Soft-Value-Decodierung, ein Nullvektor eingefügt. Die Implementierungsdaten für die beiden Module sind in Tabelle 5.3 zusammengestellt.

Die Punktierungstabellen werden hier als Konstanten vorgegeben und sind somit als LUT-ROM in der Hardware aufgebaut. Ein LUT speichert dabei bis zu 16 Bit große Tabellen. Darüber hinaus wird ein Zähler benötigt, der die aktuelle Position in der gespeicherten Tabelle adressiert.

	Punktierung	Depunktierung
Slices	30	44
Flipflops	23	35
LUT	54	54
Taktrate	244,26 MHz	188,47 MHz

Tabelle 5.3 Implementierungsdaten für die Punktierung und Depunktierung

Die Werte in Tabelle 5.3 beschreiben Punktierungs- und Depunktierungsmodule, die einen Halbratencode flexibel nach R=2/3, 3/4, oder 9/16 umsetzen können. Bei dieser Implementierung liegen die verbrauchten Ressourcen bei ca. einem Promille des eingesetzten FPGAs. Die möglichen Taktraten liegen oberhalb fast aller anderen Verarbeitungseinheiten. Es gibt hier somit kaum Potenzial zur Optimierung der FPGA-Schaltung. Der leicht höhere Aufwand und kleinere Takt bei der Depunktierung resultiert aus der Verarbeitung von 4-Bit-Codewörtern (Soft Values).

Die Punktierung ist in einem OFDM-System nur optional, erhöht jedoch die Flexibilität bei den verwendbaren Coderaten. Bei einem ausschließlich ressourcenoptimierten System kann auf diese Einheit verzichtet werden.

Die Punktierung bietet die Möglichkeit, die Nutzdatenrate mit sehr geringem Hardwareaufwand an die jeweilige Übertragungssituation anzupassen. Auf dieses Plus an Flexibilität sollte nur in FPGA-Minimalsystemen verzichtet werden.

5.2.3 Interleaver

Interleaver werden zusammen mit Faltungscodes häufig eingesetzt, um die Fehlerkorrektur robust gegen Büschelfehler zu machen. Dies trägt der Eigenschaft des Viterbi-Decoders Rechnung, dass vereinzelt auftretende Fehler besser korrigiert werden können als Sequenzen, die deutliche Häufungspunkte von Fehlern aufweisen.

Bei einer mobilen Datenübertragung kommt es leicht zu Störungen, die sich auf mehrere benachbarte Datenbits auswirken. In Einträgersystemen tritt dies auf, wenn der Einfluss der Störung länger ist als die Symboldauer. In dem hier betrachteten OFDM-System werden sequentielle Bits dann gestört, wenn das beeinflusste Frequenzband breiter ist als der Subträgerabstand. Die Beobachtung der Kanaleinflüsse in Mehrwegeszenarien zeigt jedoch eine deutliche Korrelation bei den Störungen benachbarter Subträger. Darüber hinaus treten insbesondere bei höherwertigen Modulationsarten, bei denen mehrere Datenbits pro Modulationssymbol (und somit auf einem Subträger) abgebildet werden, blockweise Bitfehler verstärkt auf. Hier ist es wichtig, die einzelnen Bits zu dekorrelieren. Interleaver, die einzelne Bits umsortieren, müssen vor der Abbildung der Bits auf Modulationssymbole implementiert sein. Solche Bitinterleaver bieten in OFDM-Systemen mit höherwertiger Modulation Vorteile gegenüber Symbol- bzw. Subträgerinterleavern.

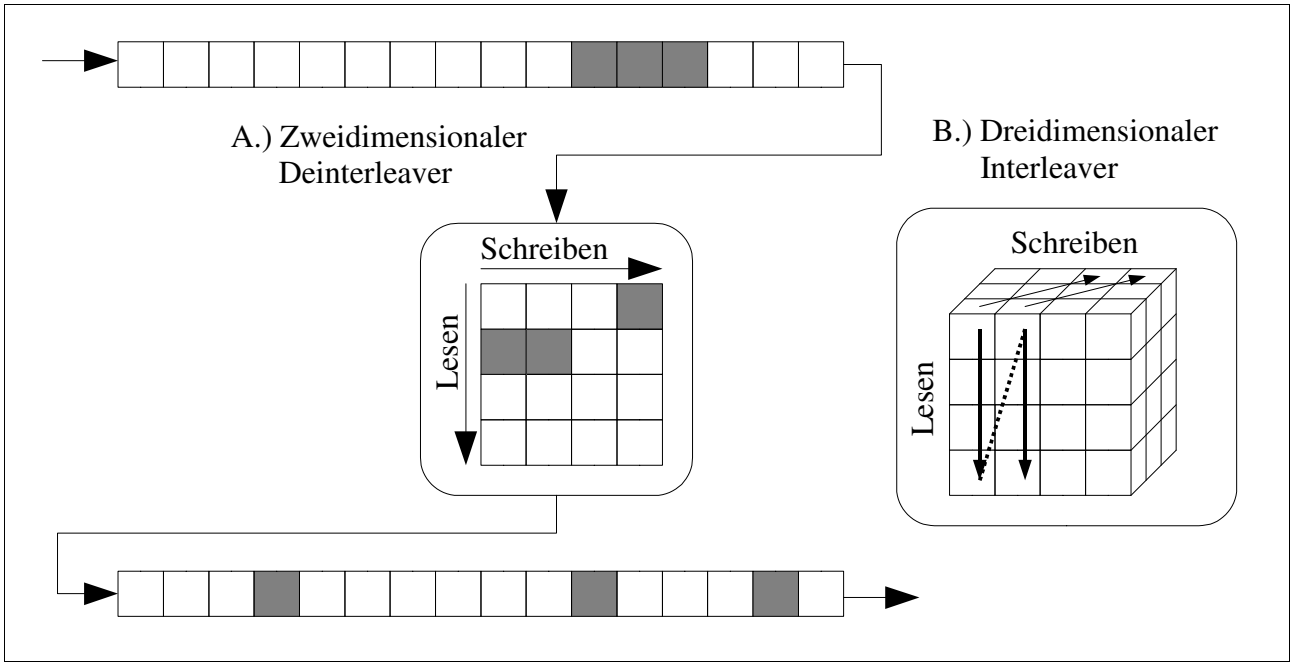


Abb. 5.14 Interleaver schreiben und lesen die Speichermatrix in unterschiedlichen Richtungen. Bei höherwertigen Modulationsarten kann ein dreidimensionaler Speicherblock genutzt werden (B.).

Der Interleaver verteilt die Datenbits so über die OFDM-Symbole, dass die zeitlich hintereinander eintreffenden (codierten) Daten nicht auf benachbarten Trägerfrequenzen und, bei

einem höherwertigen Modulationsschema, nicht auf dem gleichen Subträger oder der gleichen Bitposition übertragen werden. Dazu wird die Datensequenz im Sender umsortiert. Die bei der Übertragung möglicherweise auftretenden Büschelfehler werden dann im Deinterleaver, also beim Herstellen der ursprünglichen Reihenfolge, separiert. Abb. 5.14 zeigt ein Beispiel, wie drei zusammenhängende Fehler auseinander gezogen werden.

Im HiperLAN/2- und IEEE802.11a-Standard sind die Interleaver jeweils in Abhängigkeit vom Übertragungsmodus definiert. Je mehr Bits pro OFDM-Symbol übertragen werden, desto weiter sind die Eingangsbits separierbar. Dabei wird auch eine dreidimensionale Verwürfelung eingesetzt, wie sie in Abb. 5.14B dargestellt wird. Die Ausgangsreihenfolge der Bits wird in Abhängigkeit von der Modulationswertigkeit durch folgende zwei Gleichungen beschrieben [HIP01]:

$$i = \frac{N_{CBPS}}{16}\,(k \bmod 16) + \mathit{floor}(k/16) \quad , \quad k = 0,1,\ldots,N_{CBPS}-1 \tag{5.6}$$

$$\begin{aligned} j &= s \cdot \mathit{floor}\frac{i}{s} + \left\{ i + N_{CBPS} - \mathit{floor}\left(\frac{16 \cdot i}{N_{CBPS}}\right)\right\} \bmod (s) \\ &, \; i = 0,1,\ldots,N_{CBPS}-1 \; ; \; s = \max\left(\frac{N_{BPSC}}{2}, 1\right) \end{aligned} \tag{5.7}$$

Die in Gleichung 5.7 ermittelte Zahlenfolge gibt die Reihenfolge der Ausgabebits in Bezug auf die Eingangssequenz an. N_{CBPS} ist die Anzahl der codierten Bits pro Symbol, N_{BPSC} ist die Zahl der codierten Bits pro Subträger. Beide Werte ergeben sich direkt aus der Modulationswertigkeit und sind in Tabelle 5.4 aufgelistet.

	N_{CBPS}	N_{BPSC}	s
BPSK	48	1	1
QPSK	96	2	1
16-QAM	192	4	2
64-QAM	288	6	3

Tabelle 5.4 Parameter der Interleaver in Abhängigkeit von der Modulationswertigkeit

Die erste Umsortierung in Gleichung 5.6 verschiebt sequentiell eingehende Codebits auf nicht benachbarte Subträger. Die zweite Operation in Gleichung 5.7 verteilt die Bits der so erzeugten Folge auf Stellen mit unterschiedlichen Gewichtungen innerhalb der Konstellation. Dies trägt der

Tatsache Rechnung, dass diese Bits bei höherwertigen Modulationen mit unterschiedlichen Zuverlässigkeiten entschieden werden.

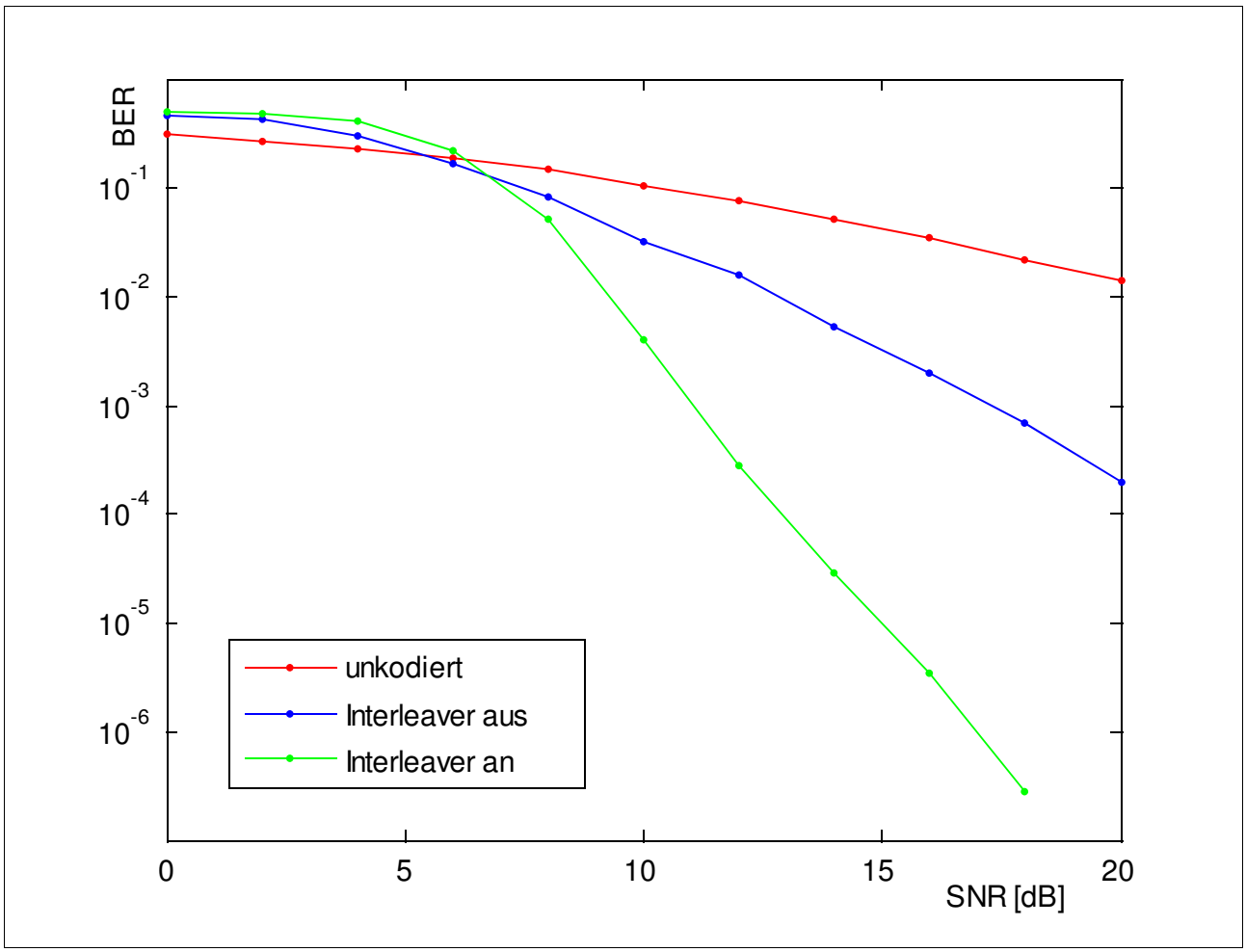


Abb. 5.15 Einfluss des Bit-Interleavers auf die Bitfehlerwahrscheinlichkeit bei einem OFDM-Signal mit einer 16-QAM über einen WSSUS-Kanal

Der Schaltungsaufwand hängt von der Größe der Interleavermatrix ab. Für beliebige Verwürfelungen muss ein Speicher integriert werden, der die Daten eines OFDM-Symbols vollständig vorhält. Um einen kontinuierlichen Datenfluss zu ermöglichen, ist es sinnvoll, diese Speichermatrix doppelt anzulegen. Dabei wird jeweils ein Block geschrieben, während der andere in der entsprechenden veränderten Reihenfolge ausgelesen wird. Eine solche Trennung verhindert außerdem, dass unerlaubte Lese- und Schreibzugriffe auf die gleiche Speicherzelle ausgeführt werden können. Des Weiteren hängt der Ressourcenverbrauch von der Anzahl der unterstützten Interleavermodi ab, weil für jeden Modus die Adressierung der Speicher unterschiedlich erfolgt.

Die Implementierung der Interleaver wird in der vorliegenden Arbeit einstellbar für alle Modi des HiperLAN/2-Standards realisiert. Tabelle 5.5 zeigt die verbrauchten Ressourcen und die maximal mögliche Taktrate. Insgesamt benötigen diese Elemente mit jeweils weniger als 1% der logischen Zellen nur einen kleinen Teil der Schaltungslogik des Gesamtsystems.

Durch die Verwendung von Dual-Port-Speicherblöcken wird sichergestellt, dass die nach der Codierung ausgegebenen zwei Soft-Werte parallel verarbeitet werden. Eine Lösung, bei dem die beiden Codewörter nacheinander mit nur einem Speicherport verarbeitet werden, erfordert einen im Vergleich zum Coderausgang doppelten Takt. Im betrachteten Experimentalsystem würde dies einen Interleavertakt von 120 MHz bedeuten und somit in einen kritischen Bereich fallen. Die Lösung mit dem doppelten Speicherport erlaubt hingegen mit einem Takt von 60 MHz und ist somit kein limitierendes Element im Gesamtsystem.

	Interleaver	***Deinterleaver***[4]
Flipflops	84	146
LUTs	432	619
RAMs	2	3
Taktrate	118,3 MHz	128,6 MHz

Tabelle 5.5 Implementierungsdaten des Interleavers und Deinterleavers

Die vollständige Trennung von Schreib- und Lesespeicher führt zu einer Verzögerung im Datenfluss um ein OFDM-Symbol, was bei zeitkritischer Datenübertragung berücksichtigt werden muss.

Die Komplexität des Deinterleavers ist unmittelbar an die des Interleavers gekoppelt, weil dieser abhängig vom aktiven Modus und der verwendeten Interleavermatrix die inverse Vertauschung der Daten vollzieht. Bei dem benötigten Speicher ist jedoch zu bedenken, dass im Deinterleaver Soft Values statt einzelner Codebits vorgehalten werden. Bei 4-Bit-Soft-Values muss der Speicher also viermal so groß sein.

4 Die Nutzung von drei Speicherblöcken ist hier implementiert worden, weil der Deinterleaver im 64-QAM Fall ein Permutationszyklus von drei aufweist. Auf diese Weise können die Speicher abwechselnd beschrieben werden. Eine Implementierung mit nur zwei Speicherblöcken ist ebenfalls möglich, erfordert jedoch das Schreiben aller Daten in beide Speicherblöcke. In den Modi BPSK, QPSK und 16-QAM ist der dritte Speicherblock ungenutzt.

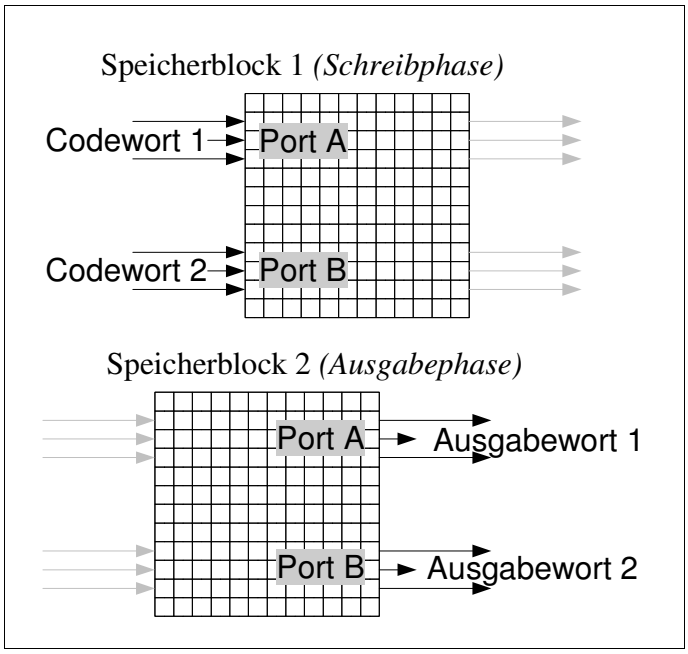


Abb. 5.16 Interleaver mit wechselndem Schreib- und Lesespeicher

Der Interleaver ist für ein OFDM-Übertragungssystem nicht zwingend erforderlich (wenngleich aus BER-Performancesicht sehr sinnvoll). Wenn die Verzögerung um jeweils ein Symbol in Sender und Empfänger zu lang ist, kann auch ein kleinerer Datenblock verwürfelt werden. Dies reduziert jedoch auch die Breite des maximal korrigierbaren Büschelfehlers.

Der Interleaver verbessert die Fehlerkorrektureigenschaften bei Büschelfehlern. Der Schaltungsaufwand ist gering. Als Nachteil sind die Verzögerung um ein OFDM-Symbol und die Nutzung von mehreren Speicherblöcken anzuführen.

5.3 Digitale Modulation

Die digitale Modulation erzeugt aus einer eingehenden Bitfolge komplexe Zahlen. Im Allgemeinen folgt dieser Schritt nach den Verarbeitungsschritten zum Fehlerschutz. Die eingehende Bitfolge besteht somit aus Codewörtern, die gegebenenfalls punktiert und verwürfelt sind. Je nach Modulationswertigkeit werden unterschiedlich viele Bits auf einen komplexen Wert abgebildet. Im HiperLAN/2-Standard sind vier unterschiedliche Modi erlaubt: BPSK (*Binary Phase Shift Keying*), QPSK (*Quadrature Phase Shift Keying*), 16-QAM (Quadratur Amplituden Modulation) und 64-QAM. Abb. 5.17 zeigt, welche Bitfolgen in den einzelnen Modi auf welche Werte in der komplexen Ebene abgebildet werden, wobei eine Gray-Codierung angenommen wird.

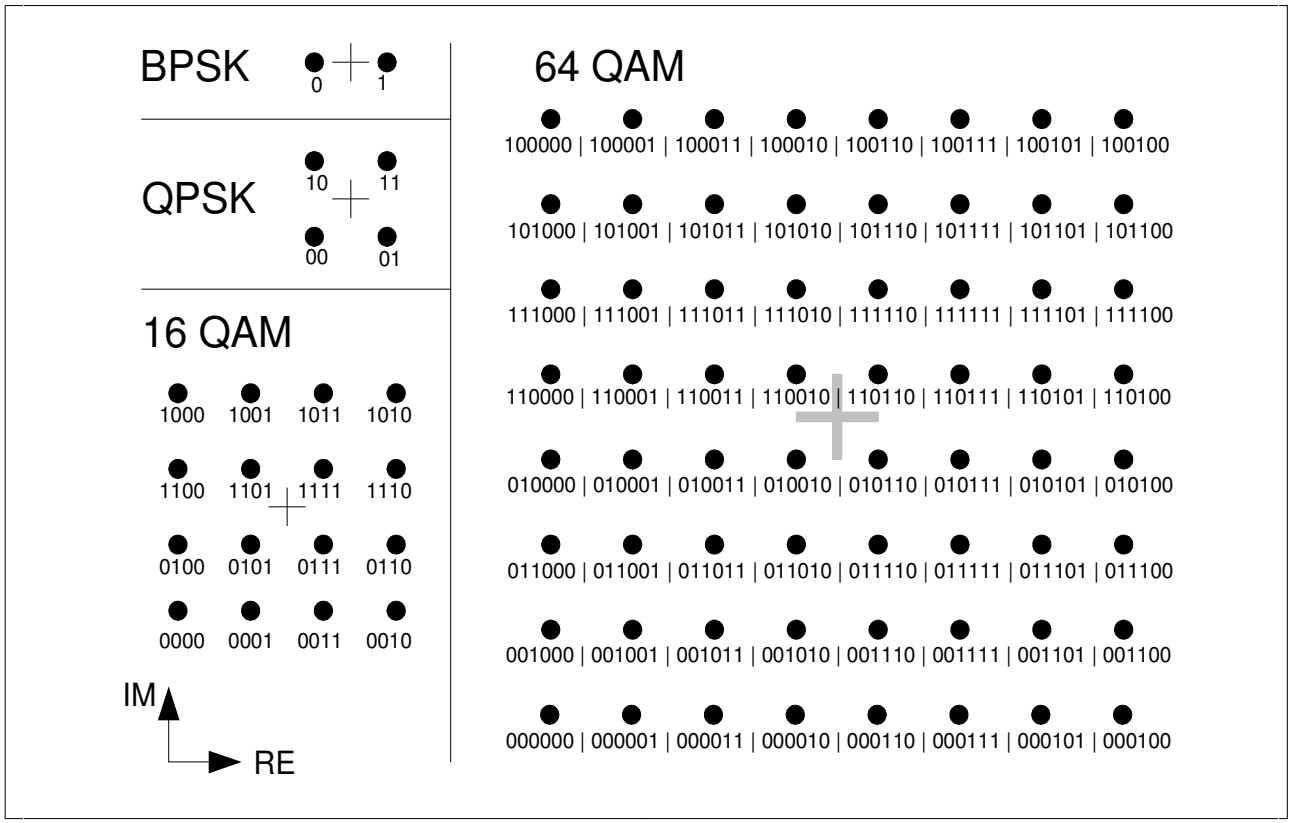


Abb. 5.17 Mögliche Konstellationsdiagramme der verschiedenen Modi des OFDM-Systems

Generell wird bei der Auswahl der Modulationswertigkeit abgewogen, möglichst viele Bits pro Abbildungswert zu übertragen (hohe Datenrate) oder durch große Abstände der Konstellationspunkte eine Entscheidung mit größerer Sicherheit im Empfänger zu erlauben (unempfindlich gegenüber Störungen). Daher bieten viele Standards mehrere Konstellationen an, um die Verarbeitung an die Übertragungsbedingungen anzupassen. Ist die Übertragungsqualität sehr gut, z.B. bei einer direkten Sichtverbindung (LOS), wird ein höherwertiges Modulationsverfahren

eingesetzt, um eine hohe Datenrate zu erzielen. Bei ungünstigen Parametern wird hingegen BPSK oder QPSK genutzt, um stark verrauschte Empfangswerte noch sicher entscheiden zu können.

Die Modulation kann im FPGA optimal implementiert werden, wenn die Ausgangsbitbreite m_M der Zahl der maximal möglichen Konstellationspunkte M gemäß $m_M=log_2(M)$ angepasst wird. Eine höhere Genauigkeit ist nicht notwendig, weil die erlaubten komplexen Werte von vornherein definiert sind. Eine Skalierung der Leistung ist an dieser Stelle in der OFDM Kette wenig sinnvoll, weil dafür der Zahlenbereich vergrößert werden müsste. Eine beliebige Leistungsanpassung kann mit geringem Aufwand in der letzten Multiplikationsstufe der IFFT erfolgen oder, bei Faktoren aus Zweierpotenzen, durch einen entsprechenden Bitshift am IFFT-Ausgang.

Die Abbildung der Bits, auch *Mapping* genannt, kann bei einer QAM jeweils für den Real- und Imaginärteil unabhängig erfolgen. Dadurch wird die Logikstruktur im FPGA recht klein und für beide Teile des komplexen Wertes kann dieselbe Entscheidungsschaltung angewendet werden.

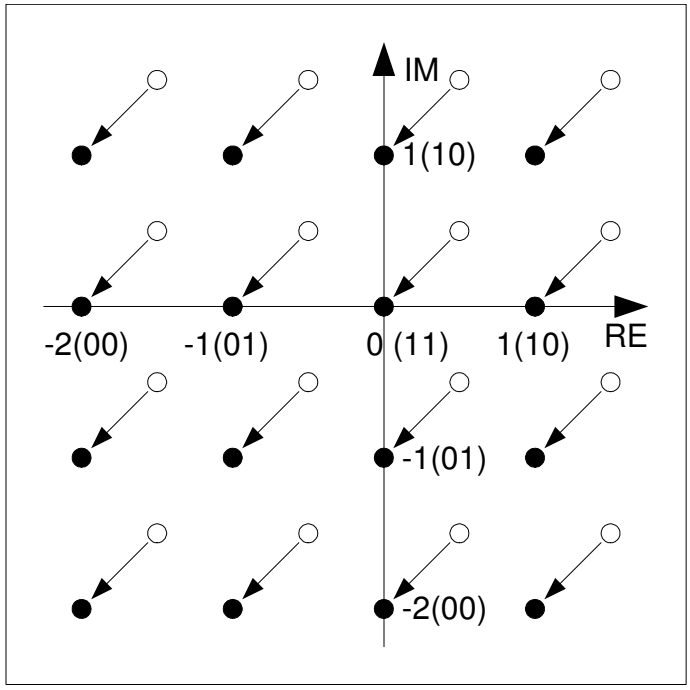


Abb. 5.18 Verschiebung der 16-QAM Konstellation durch die binäre Codierung mit den minimal notwendigen Bits

Das Mapping in der komplexen Ebene kann für eine FPGA-Schaltung auch unsymmetrisch implementiert werden, um mit einem Bit weniger Auflösung auszukommen. Wie in Abb. 5.18 dargestellt, wird die gesamte Konstellation dann um einen konstanten Wert (0.5+0.5j) verschoben. Diese Verschiebung bewirkt in der nachgeschalteten IFFT einen additiven Anteil im ersten Ausgangswert, dem Gleichanteil. Da die Werte nach der IFFT ohnehin fein quantisiert sind, kann dieser Offset leicht korrigiert werden. Für die 64-QAM als höchstwertige Modulation genügen somit jeweils drei Bit für den Real- und Imaginärteil.

Wenn die Darstellung der Konstellationspunkte mit den minimal notwendigen Bits genutzt wird, ergeben sich die Ausgangswörter direkt als Logikfunktion mit gleicher Ein- und Ausgangsbitbreite aus den Eingangsbits. Die Modulation kann in diesem Fall im FPGA auf eine einheitliche Logik reduziert werden, die der IFFT mitteilt, ob ein, zwei, vier oder sechs Bits als Eingangswert interpretiert werden sollen.

Dies wird deutlich, wenn in Abb. 5.17 die Bits der 64-QAM-Konstellation betrachtet werden. In diesem Fall werden alle sechs Bit weitergegeben. Bei 16-QAM sind es (von links) die Bits eins und zwei, sowie vier und fünf. Bei QPSK sind es die Bits eins und vier, bei BPSK nur Bit vier für den Realteil. Die Logik für die vorderen und die hinteren drei Bits ist identisch.

Üblicherweise werden die komplexen Zahlenwerte bei der digitalen Modulation, wie auch in den Abbildungen gezeigt, Gray-codiert. Dies hat den Vorteil, dass die benachbarten Konstellationspunkte jeweils nur in einem Bit verändert sind. Eine Fehlentscheidung im Empfänger hin zum direkten Nachbarwert verursacht dann auch nur einen einzelnen Bitfehler.

Da der gleiche Block den Real- und Imaginärteil codieren kann, ist der Schaltungsaufwand sehr gering. Bei einer 64-QAM genügt eine Logik, die drei Eingangsbits in 3-Bit-Gray-Code umsetzt und nacheinander den Real- und den Imaginärteil erzeugt. Dies entspricht einer Tabelle mit acht Einträgen.

Es ist außerdem sinnvoll, alle Konstellationen auf Werte der gleichen Ausgangsbitbreite abzubilden, um wechselnde Wortbreiten am IFFT-Eingang zu vermeiden. Dazu werden diejenigen Bits, die bei dem aktuellen Modulationsmodus verändert werden, als *Most Significant Bit* (MSB) des Ausgangswertes interpretiert.

Neben dem Mapping auf komplexe Zahlenwerte übernimmt das Modulationsmodul die Aufgabe, das OFDM-Symbol nach einem vorgegebenen Muster zusammenzustellen. In der Regel sind nicht alle Subträger genutzt, sodass nicht in jedem Takt komplexe Ausgangswerte gebildet werden. Bei HiperLAN/2-Systemen ist ein 64 Subträger breites OFDM-Symbol wie folgt strukturiert:

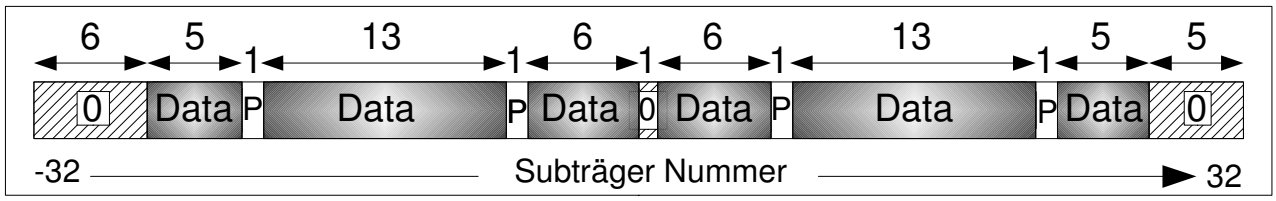


Abb. 5.19 HiperLAN/2-Symbol mit 48 Nutzdatenträgern, 4 Piloten und 12 ungenutzten Trägern

Das OFDM-Symbol besteht aus Null- und Datenträgern sowie Pilotsymbolen. Die Nullträger reduzieren die Frequenzanteile an den Rändern des genutzten Frequenzbandes und unterdrücken den Gleichanteil im Ausgangssignal. Die Pilotsymbole dienen dem Empfänger als Referenz und werden zum Nachführen der Phasenlage bei der Entzerrung genutzt.

Bei der Auswahl der Pilotsymbole ist zu beachten, dass diese möglichst einen der verfügbaren Konstellationspunkte in der komplexen Ebene nutzen. Im HiperLAN/2-Standard ist dies jedoch nicht immer gegeben. Hier sind die Pilotsymbole rein reell. Diese Punkte sind jedoch in einem Abbildungsfeld für QPSK, 16- und 64-QAM nicht darstellbar. Bei diesen Modulationen liegen die

Piloten zwischen den Konstellationspunkten, wie in Abb. 5.20 gezeigt. In diesen Fällen muss der Ausgangswertebereich nur für die Piloten um ein Bit verbreitert werden.

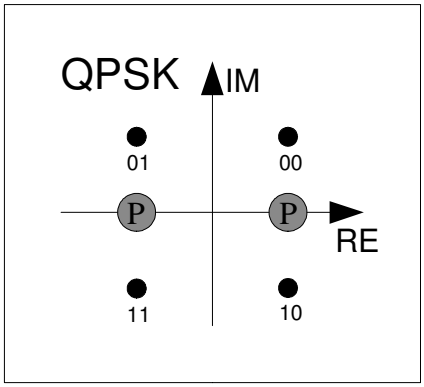


Abb. 5.20 QPSK mit reellen Piloten

Rein reelle Pilotsymbole vereinfachen in der Regel die Auswertung und das Nachführen der Phasenlage im Empfänger. Es ist abzuwägen, ob die Vergrößerung des Ausgangswertebereichs um mindestens ein Bit die Aufwandsersparnis in der Entzerrung aufwiegt. Tendenziell sollten dem Modulationsmodus angepasste Piloten, die mit den vorhandenen Konstellationspunkten auskommen und einen konstanten Phasenoffset haben, einen geringeren FPGA-Verarbeitungsaufwand bedeuten.

Bei der Demodulation werden die entzerrten komplexen Empfangswerte wieder Bitfolgen zugeordnet. Dabei gilt der Konstellationspunkt mit dem geringsten Abstand zum Empfangssymbol als maximal wahrscheinlich. Diese direkte Zuordnung wird als *Hard Decision* bezeichnet. Falls nur diese Entscheidung ausgeführt wird, genügen Eingangswerte, welche wie die komplexen Werte am Ausgang der Modulation nur jeweils drei Bit für den Real- und Imaginärteil haben (jeweils drei Bit wegen der 64-QAM als höchstwertiges Schema). Jegliche feinere Quantisierung wird bei der Hard Decision ohnehin verworfen.

Ist dem Demodulator jedoch ein Decoder nachgeschaltet, der gewichtete Informationen (Soft Values) verarbeiten kann, muss eine *Soft Decision* ausgeführt werden. Dabei bleiben die ermittelten Codebits die gleichen wie bei der Hard Decision. Zu jedem Bit wird darüber hinaus jedoch noch ein Soft Value λ errechnet, der die Zuverlässigkeit des entschiedenen Bits $\hat{c}_{k,j}$ gewichtet [OHM05]. Die in dieser Arbeit umgesetzte Implementierung für eine spätere bitweise Decodierung erfolgt nach

$$\lambda(\hat{c}_{k,j}) = (-1)^{\hat{c}_{k,j}} \cdot |\hat{H}_k|^2 \cdot \left(|\tilde{R}_k - \hat{S}_k|^2 - |\tilde{R}_k - \overline{\hat{S}_{k,j}}|^2 \right) \qquad (5.8).$$

Der Soft Value wird aus der Differenz der Abstandsquadrate des Empfangssymbols $\tilde{R}_k$ zu dem entschiedenen wahrscheinlichsten Konstellationspunkt $\hat{S}_k$ und zu einem konstruierten $\overline{\hat{S}_{k,j}}$ Referenzpunkt bestimmt. Die Herleitung der Gleichung 5.8 ist in Anhang A dargestellt. Dieser Wert ist pro Bit entweder nur vom Real- oder nur vom Imaginärteil abhängig, sodass nur der Abstand entlang einer dieser Achsen betrachtet werden muss.

Der Demodulator errechnet zu jedem Bit einen Soft Value. Bei der 64-QAM sind es sechs Werte pro Konstellationspunkt. Ist z.B. ein Eingangswert hart der Bitfolge 110011 zugeordnet worden, wird für jedes der sechs Bits ein wahrscheinliches (bzw. nächstgelegenes) Fehlersymbol gebildet,

indem das entsprechende Bit invertiert wird. Für das erste Bit ist das 010011, für das zweite Bit 100011 usw. Aus den Abständen vom Empfangswert zum entschiedenen und zum konstruierten fehlerhaften Konstellationspunkt wird dann der jeweilige Ausgangswert ermittelt.

Für die Soft Value Berechnung ist im Demodulator eine verfeinerte Quantisierung der Eingangswerte notwendig. Die Breite dieser Wörter hängt von der gewünschten Genauigkeit der gewichteten Ausgangswerte und von der Zuordnung der Zuverlässigkeitsinformation, der Log-Likelihood Ratio (LLR) ab. Abb. 5.21 zeigt ein Beispiel für die lineare Verknüpfung:

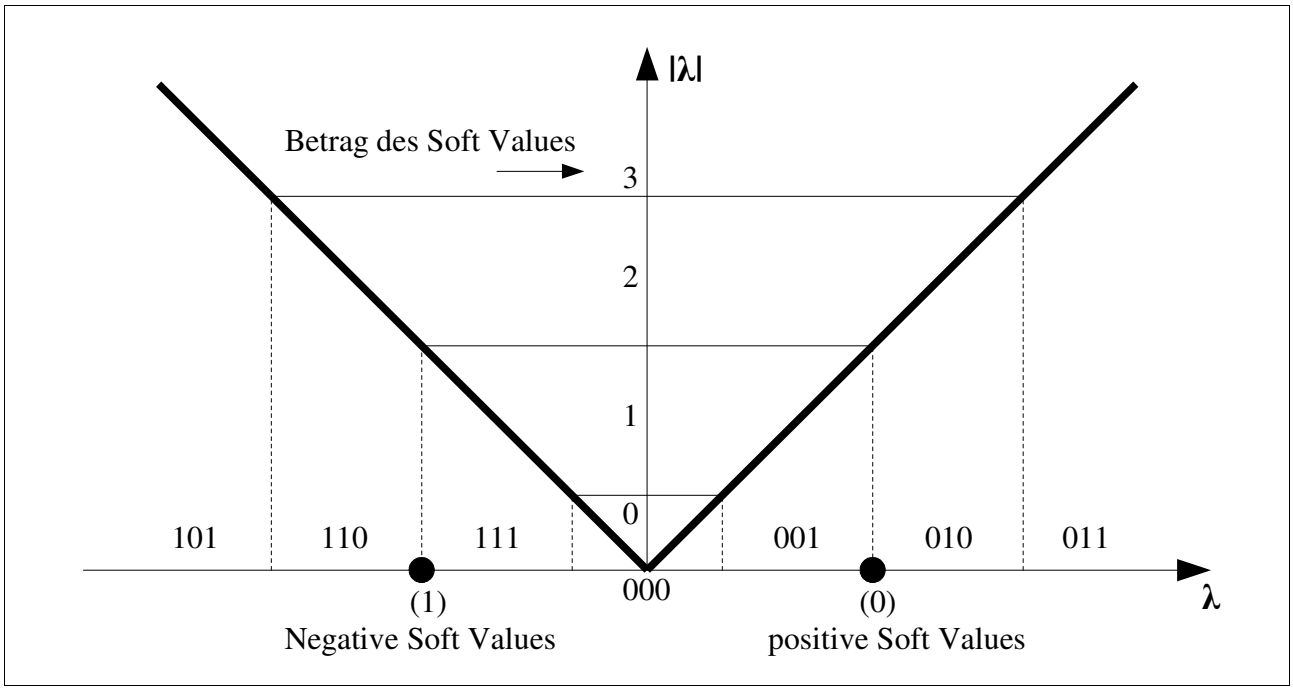


Abb. 5.21 Quantisierungsstufen der Demodulator Eingangswerte r, die Log-Likelihood Ratio und die Zuordnung von 3-Bit Soft Values (Vorzeichen plus zwei LLR-Bits)

Die Zahl der Quantisierungsstufen der Soft-Information legt fest, wie fein die Eingangswerte des Demodulators aufgelöst sein müssen. Für BPSK und QPSK liegen die Soft Values auf einer eindeutigen Geraden. Bei höheren Modulationswertigkeiten wie 16- oder 64-QAM gilt diese Formel zwar auch, aber sie kann für die einzelnen Bits in unterschiedlichen Wertebereichen liegen (abhängig vom Abstand zum nächsten konstruierten Fehlersymbol).

Die Breite der Soft Values ergibt sich aus der Zahl der Konstellationspunkte (die Information der Hard Decision) und der Anzahl der Quantisierungsstufen um die einzelnen Konstellationspunkte herum. Tabelle 5.6 zeigt die minimalen Eingangswortbreiten für verschiedene Modulationsmodi.

	Hard Decision	2 Bit Soft Value	3 Bit Soft Value	4 Bit Soft Value
BPSK	1	2 (1)[5]	3 (2)	4 (3)
QPSK	2	3 (1)	4 (2)	5 (3)
16-QAM	4	5 (1)	6 (2)	7 (3)
64-QAM	6	7 (1)	8 (2)	9 (3)

Tabelle 5.6 Minimale Eingangsbitbreiten des Demappers in Abhängigkeit der Soft Values

Bei den Werten in Tabelle 5.6 werden die Schwellen für die Soft Values als äquidistant vorausgesetzt. Andere Abbildungsvorschriften wären sicherlich denkbar, versprechen aber kaum Gewinne und erhöhen unmittelbar die Anforderungen an die Verarbeitungsgenauigkeit.

Die Berechnung der LLR zu den einzelnen Bits führt bei höherwertigen Modulationsschemata (insbesondere ab 64-QAM) zu variierenden Soft Values, ohne dass die Quantisierung zwischen den Konstellationspunkten berücksichtigt werden muss. Allein die Abstände zu den nächstmöglichen Fehlersymbolen führen zu unterschiedlich sicheren Entscheidungen.

Beispiel: Ein Empfangswert wird bei einer 64-QAM der Bitfolge 110011 zugeordnet. Die konstruierten Fehlersymbole für die einzelnen Bits zeigt Tabelle 5.7. Die zugehörigen Abstände für LLR-Berechnung sind in Abb. 5.22 dargestellt.

Aktives Bit	*Fehler-symbol*	*Abstand*
1	010011	1
2	100011	3
3	111011	1
4	110111	3
5	110001	1
6	110010	1

Tabelle 5.7 Bitsicherheit bei 64-QAM

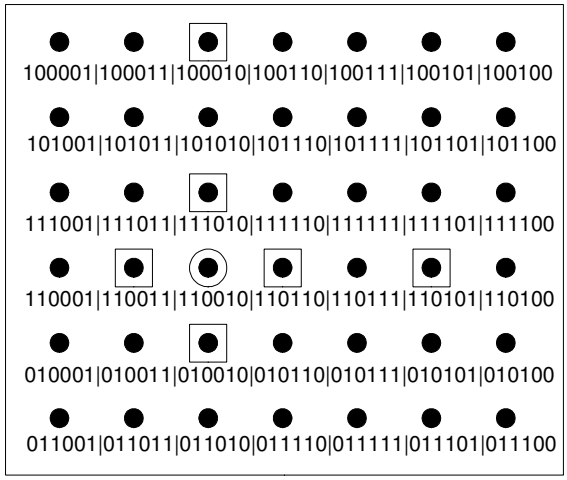


Abb. 5.22 Konstruierte Fehlersymbole bei 64-QAM (hier nur ein Ausschnitt)

Die Soft Values für die Bits zwei und vier werden aufgrund der Konstellation immer größer sein als die der anderen vier Bits. Die höhere Sicherheit der Entscheidung ist darauf zurückzuführen, dass die Konstellationspunkte mit dem entsprechenden inversen Bit weit von dem Empfangswert entfernt liegen. Es wäre somit eine

5 Die Zahl in der Klammer gibt die Bits pro Konstellationspunkt an, also die notwendigen Quantisierungsstufen zur Darstellung des Abstands von Empfangs- und Vergleichssymbol.

größere Störleistung notwendig, um den Originalwert in einen dementsprechend fehlerhaften Bereich zu schieben.

Die Soft Values werden bei ihrer Ermittlung mit Informationen aus der Kanalschätzung skaliert. Dazu wird jedem Subträger ein Wert (*Channel State Information*, CSI oder *Reliability Information*, RI) zugeordnet, der proportional zum Quadrat des Übertragungsfaktors ist. Die Multiplikation mit diesem Kanalkennwert ist in Gleichung 5.8 als $|\hat{H}_k|^2$ enthalten. Sollte ein Subträger bei frequenzselektiver Übertragung stark gedämpft sein, ist für ihn das Signal- zu Rauschverhältnis weit unterhalb des Durchschnitts. Einzelne Subträger können durch Fading fast vollständig ausgelöscht werden. Der CSI-Faktor kann in diesem Fall den Soft Value reduzieren und damit die Zuverlässigkeitsinformation den Kanalparametern anpassen. Somit gehen die Bits des stark gedämpften Trägers durch die niedrige Gewichtung weniger stark in die Decodierung ein und haben daher nur geringen Einfluss auf das Decodierergebnis haben.

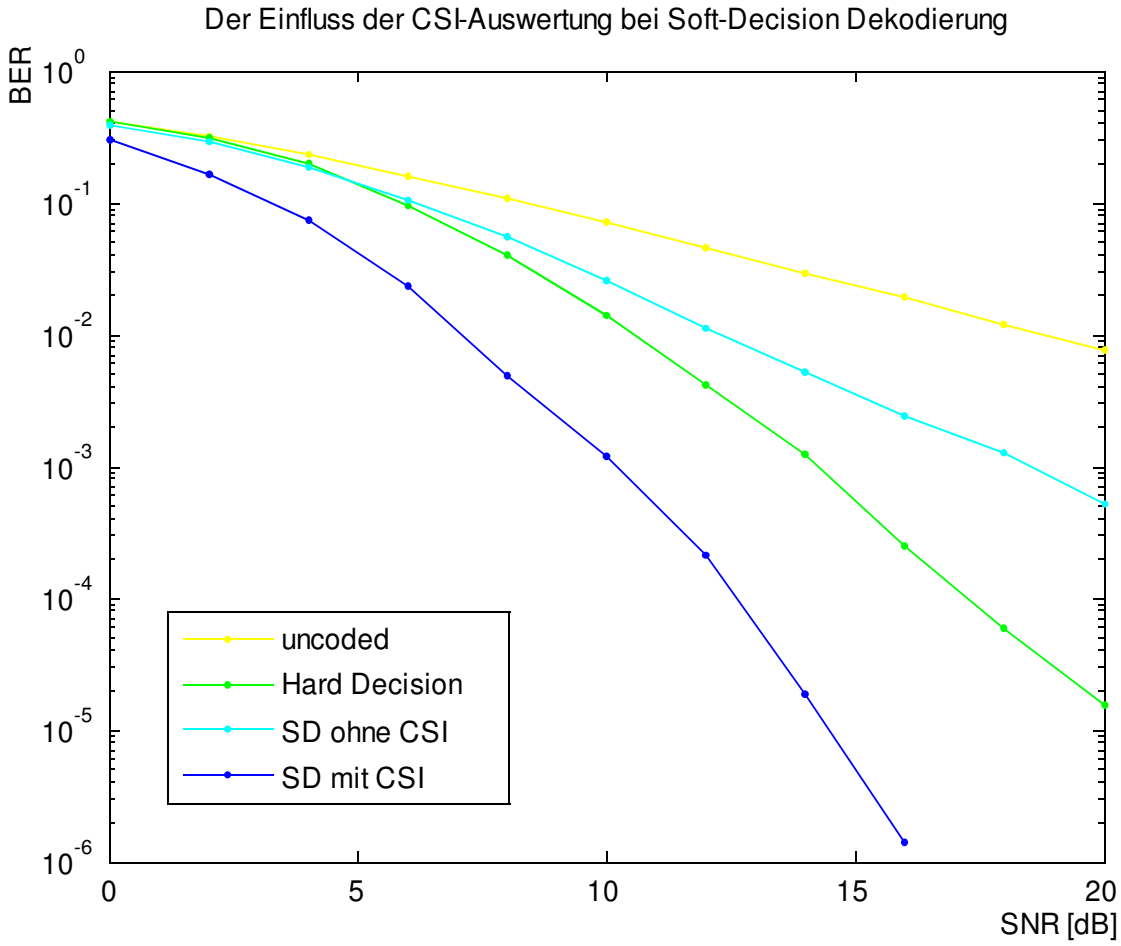


Abb. 5.23 Vergleich der BER bei einer QPSK modulierten Übertragung in einem WSSUS-Kanal in Abhängigkeit der unterschiedlichen Zuverlässigkeitsinformationen der Demodulation

Ohne die CSI-Skalierung werden stark gedämpfte Subträger, die durch die Entzerrung sehr ungenau korrigiert werden, falscherweise als zuverlässig markiert. Fehlt diese Gewichtung, arbeitet das Soft-Value-System in Mehrwegekanälen mit starken Fading Einbrüchen schlechter als eine Implementierung ohne Wahrscheinlichkeitsinformation (Hard Decision).

Die mit dem CSI-Faktor skalierten Soft Values führen erwartungsgemäß zu den besten Ergebnissen bei der Fehlerkorrektur. Abb. 5.23 zeigt den Verlauf der BER von einer in dieser Form verarbeiteten Übertragung. Zum Vergleich sind die Hard-Decision-Decodierung, eine nicht CSI skalierte Soft Decision (um ggf. Hardwareaufwand zu sparen) und der uncodierte Fall eingezeichnet.

Eine vollständige Implementierung der Modulation und Demodulation in einer FPGA-Verarbeitung ist recht unkritisch. Die Größe der Elemente bestimmt sich in erster Linie aus der Anzahl der unterstützten Modulationsformen. In dem in dieser Arbeit referenzierten OFDM-Demonstrator sind BPSK, QPSK, 16- und 64-QAM integriert. Bei dem Demodulator spielt auch die Auflösung der Soft Values eine Rolle, wobei eine Wortbreite von mehr als den hier realisierten vier Bit (siehe Kap. 5.2.1) nur in wenigen Fällen sinnvoll erscheint. Tabelle 5.8 zeigt den Ressourcenverbrauch dieser Module und die möglichen Taktraten der Datenbits. In Bezug auf das Gesamtsystem verbraucht dieser Verarbeitungsschritt weniger als fünf Prozent der Gesamtkapazität und ist auch vom Datendurchsatz kein begrenzender Schaltungsteil.

	Modulator	***Demodulator***
Slices	132	1472
Flipflops	48	464
LUTs	247	2596
Multiplizierer	0	12
RAMs	1	0
Taktrate	118,3 MHz	128,6 MHz

Tabelle 5.8 Implementierungsdaten des Modulators und Demodulators

5.4 IFFT/FFT

Die Diskrete Fourier-Transformation (DFT) kann in einer FPGA-Hardware recheneffizient als *Fast Fourier-Transformation* (FFT) implementiert werden. Dabei wird durch Ausnutzung der Symmetrien und Periodizität die Operation auf eine minimale Anzahl an Rechenschritten reduziert. Darüber hinaus führt die optimierte Struktur dazu, dass in einigen Stufen nur Multiplikationen mit -1, j und -j ausgeführt werden müssen. Diese sind in einer Hardwarerealisierung fast ohne Aufwand möglich. In Abb. 5.24 ist die Radix2^2-Struktur für eine 16-Punkt-FFT skizziert.

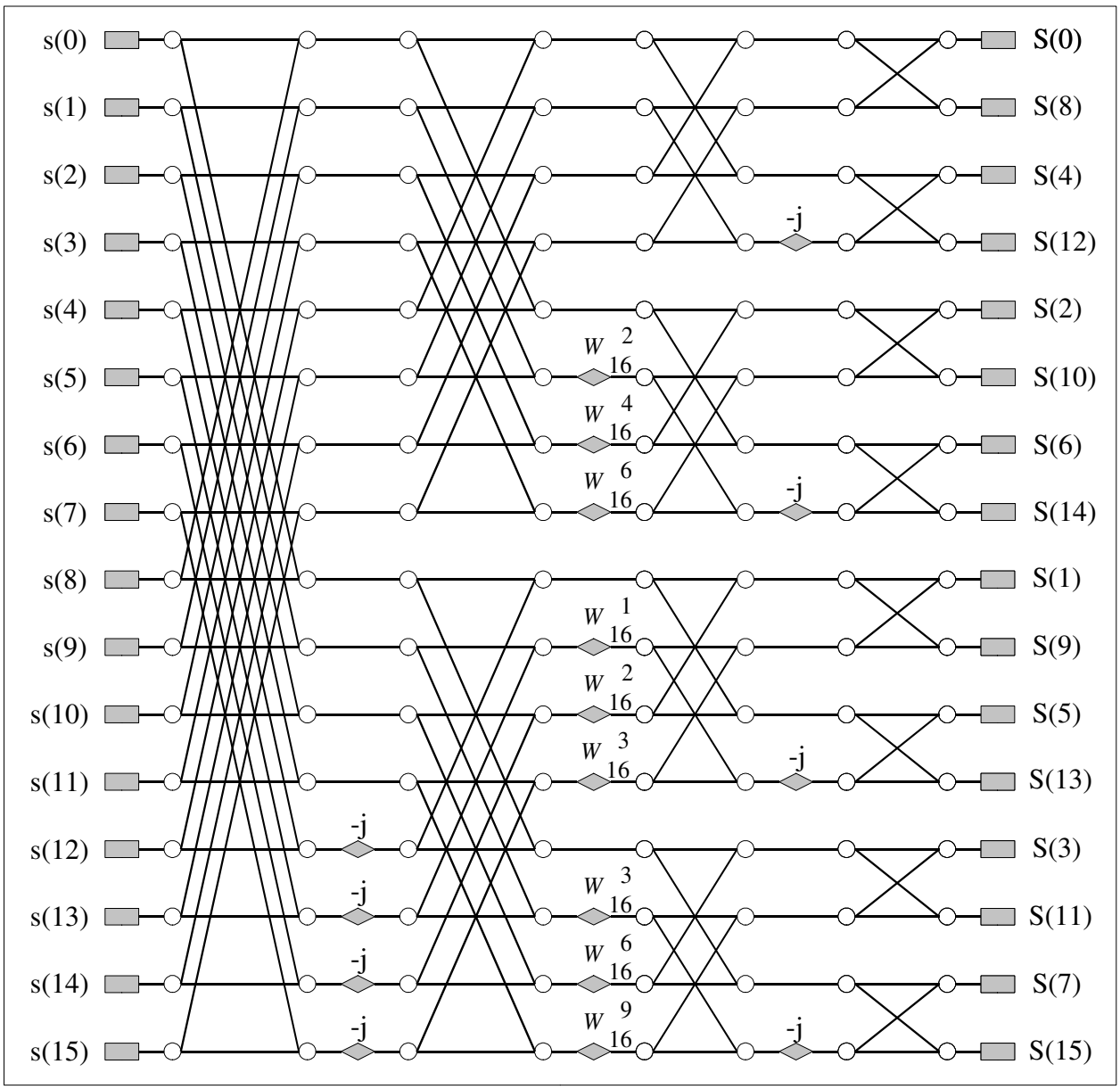


Abb. 5.24: Radix-2^2-Struktur für N=16

Die hier verwendeten ungeraden Konstanten (W_N^{kn}) werden als Twiddle-Faktoren bezeichnet [WOL84]. Sie sind vorab bestimmbar und können in der Hardware in einem Konstantenspeicher abgelegt werden. Die Umstellung der DFT-Gleichung auf die Radix2^2-Struktur [HE96][YEH03] und die Bestimmung der Twiddle-Faktoren sind in Anhang B ausgeführt. FFT und *Inverse FFT* (IFFT) arbeiten mit dem identischen Algorithmus. Um eine IFFT zu berechnen, müssen lediglich am Ein- und Ausgang jeweils der Real- und Imaginärteil der Operanden bzw. Ergebniswerte vertauscht werden.

Bei einer FFT von festgelegter Länge ist die geforderte Rechengenauigkeit der wichtigste Faktor für Ressourcenverbrauch und Geschwindigkeit bei der Implementierung in einem FPGA. Bei einer N-Punkt-FFT werden $log_4\ N-1$ komplexe Multiplikationen und $N-1$ komplexe Speichereinheiten benötigt. Die Butterfly-Struktur jeder Stufe wird nur einmal implementiert, weil die Daten in diesen Schaltungsteilen nacheinander verarbeitet werden können. Für das Beispiel aus Abb. 5.24 wird jeweils ein 16-, 8-, 4- und 2-Knoten-Butterfly benötigt.

Die interne Wortbreite wird bei den Operationen von der gewünschten Genauigkeit am FFT-Ausgang und den tolerierbaren Rundungsfehlern bestimmt. Die Summe der Rundungsfehler sollte jedoch nicht größer sein als das Quantisierungsrauschen des D/A-Wandlers der Sendeeinheit.

Nach jeder Addition oder Multiplikation wird das Resultat wieder auf die interne Wortbreite n reduziert. Ausgenommen sind hier die Multiplikationen mit -1, j und -j, die den Wertebereich nicht verändern. Jede Rundung führt zu einem Rundungsfehler von maximal $2^{-(n+1)}$. In jeder Quantisierungsstufe wird ein Fehler dieser Größe addiert, sodass der Gesamtfehler unmittelbar von der Zahl der Stufen abhängt. Dieser Einfluss der Wortbreitenanpassung wurde in Kap. 4.3 detailliert besprochen. Da jedoch der Ausgangswert nur eine definierte Wortbreite m hat, kann zwar intern beliebig genau gerechnet werden, am Ende aber nur eine Genauigkeit von 2^{-m} aufgelöst werden.

FFT-Länge	***Stufen***	***Rundungen***	***Zusätzliche interne Genauigkeit***	***Registerbreiten für einen 12 Bit Ausgabewert***
4	2	2	+1	13
8	3	4	+2	14
16	4	5	+3	15
32	5	7	+3	15
64	6	8	+3	15
256	8	11	+4	16
1024	10	14	+4	16

Tabelle 5.9 Interne Rechengenauigkeit der IFFT/FFT in Abhängigkeit der FFT-Länge

Es sollte intern mit $b \geq log_2(Anzahl\ Rundungen)$ zusätzlichen Bits gerechnet werden. Die internen IFFT/FFT Rechenschritte erfordern somit idealerweise eine Registerbreite von $n=m+b$. Eine höhere Genauigkeit bringt keinen Vorteil, ein kleinerer Wert führt zu zusätzlichem Quantisierungsrauschen. Tabelle 5.9 listet die Quantisierungsparameter für verschiedene FFT Längen auf.

Die Rundung der Zahlen sollte bei den Multiplikationen immer exakt durchgeführt werden. Ein direkter Bitshift wäre immer die Auswahl der nächstkleineren ganzen Zahl, also kein arithmetisches Runden.

Beim exakten Runden, wie in Kap. 4.3 beschrieben, wird eine zusätzliche Addition (plus 0.5) benötigt. Dies ist jedoch ausschließlich nach der Multiplikation mit den Twiddle-Faktoren notwendig.

Rechenbeispiel: 7 * -7 * 7 * 7 = -2401
Zahlenbereich auf vier Bit anpassen: -2401 : 8^3 = -4.68945
Ganzzahliges Runden ergibt -5

4 Bit interne Registerbreite	**6 Bit interne Registerbreite**
0111 * 1001 = 11001111 (-49)	0111 * 1001 = 11001111 (-49)
(7) (-7) +100 (+4)	(7) (-7) +1 (+1)
11010011 (-45)	11010000 (-48)
1010 * 0111 = 11010110 (-42)	101000 * 0111 = 1101011000 (-168)
(-6) (7) +100 (+4)	(-24) (7) +100 (+4)
11011010 (-38)	1101011100 (-164)
1011 * 0111 = 11011101 (-35)	101011 * 0111 = 1101101101 (-147)
(-5) (7) +100 (+4)	(-21) (7) +10000 (+16)
11100001 (-31)	1101111101 (-131)
Ergebnis: 1100 (-4)	Ergebnis: 1011 (-5)

Beispiel 5.1 Drei Multiplikationen mit unterschiedlicher Rechengenauigkeit. Die Ein- und Ausgangswortbreite beträgt jeweils vier Bit. Eine interne Wortbreite von vier oder fünf Bit reicht nicht aus, um den exakten Ausgangswert zu ermitteln. Ab sechs Bit Rechengenauigkeit wird selbst nach drei Rundungen der Ausgangswert nicht mehr verfälscht. Die Additionen vor dem Ausschneiden der Zahlen sind für das korrekte Runden notwendig.

Bei den Additionen von zwei Zahlen führt der Bitshift immer zum Abrunden. Hier sind jedoch nur die Nachkommastellen Null und 0.5 möglich. Ob die 0.5 auf- oder abgerundet wird, spielt für die Fehlerbetrachtung keine Rolle. Es ist häufig so definiert, dass die 0.5 aufgerundet werden soll. Hier würde es jedoch den Schaltungsaufwand erhöhen. Das Quantisierungsrauschen bleibt von der Größe her identisch.

Neben den einzelnen Rechenschritten sind die Zahl und die Anordnung der Register für den gesamten Ressourcenverbrauch der IFFT/FFT wichtig. Die Daten werden wortweise seriell in die

Einheit geschrieben. Die Struktur in Abb. 5.24 macht deutlich, dass die Summanden für die Additionsknoten nicht direkt nacheinander eingehen. Je nach Anzahl *N* der FFT-Punkte und Stufen der FFT werden die Daten in Ringspeichern vorgehalten.

In der Eingangsstufe besitzt dieser Ringspeicher die Tiefe *N/2*. In der zweiten Stufe wird ein Speicher der Tiefe *N/4*, in der dritten Stufe mit *N/8* Wörtern usw. benötigt. Die Gesamtzahl der gespeicherten Wörter beträgt

$$R_{FFT} = N - 1 \qquad (5.9).$$

Moderne FPGA Architekturen (z.B. Xilinx Virtex-II) können diese Ringspeicher aufwandsgünstig mit LUTs realisieren, die als Schieberegister (siehe auch Kapitel 3.3) genutzt werden können.

Abb. 5.25 zeigt den Einfluss der IFFT-/FFT-Rechengenauigkeit auf die Bitfehlerrate bei einem OFDM-System mit 12-Bit-D/A-Wandlern. Zum Vergleich dazu zeigt Abb. 5.26 den jeweiligen Ressourcenverbrauch und die maximal möglichen Taktraten.

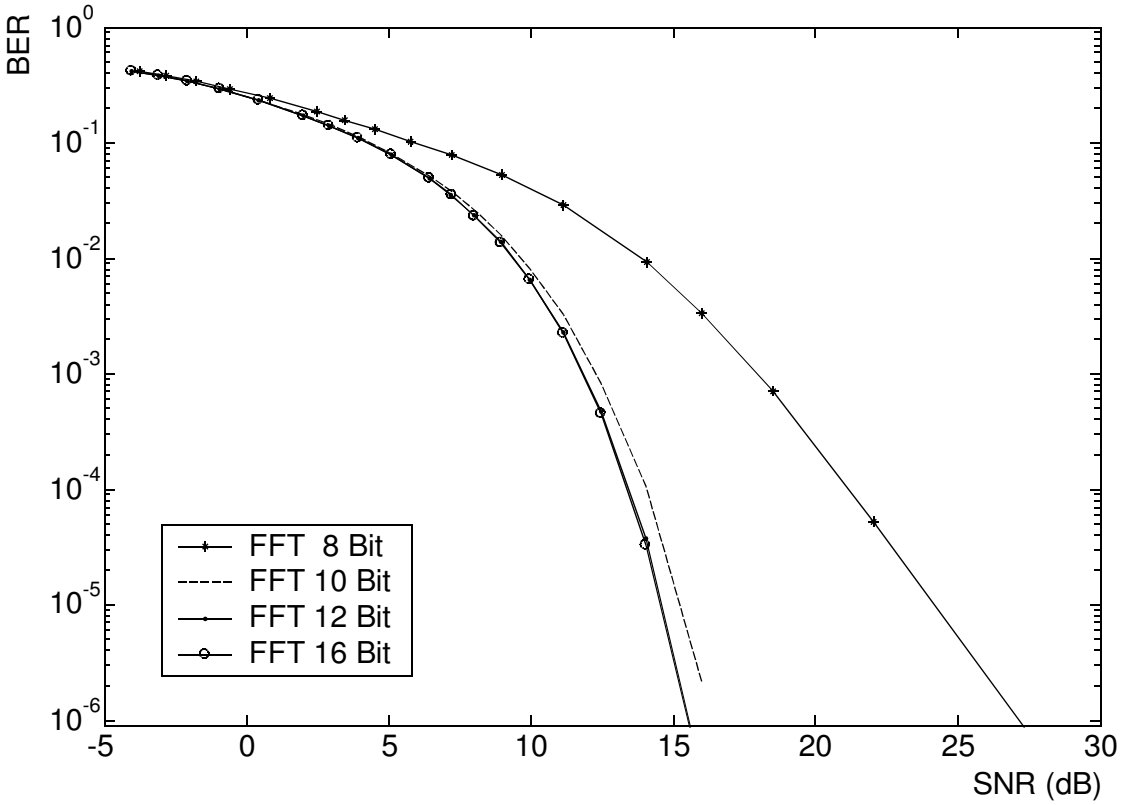


Abb. 5.25 BER Performance des Gesamtsystems abhängig von der IFFT- und FFT-Rechengenauigkeit (QPSK, uncodiert). Die Kurven der 12- und 16-Bit-FFTs liegen in dieser Darstellung übereinander.

Die BER-Messungen und die Simulationsergebnisse zeigen, dass bei der implementierten IFFT/FFT der Einfluss einer intern höheren Rechengenauigkeit sehr gering ist. Optimale Ergebnisse bekommt man ab einer internen Rechengenauigkeit von 15 Bit. Die Abb. 5.25 eingezeichnete Kurve mit 16-Bit Wortbreiten bringt keine zusätzliche Verbesserung, wenn der nachgeschaltete D/A-Wandler nur die signifikantesten zwölf Bits verarbeiten kann. Aber auch eine FFT mit einheitlich 12-Bit Genauigkeit verschlechtert die BER-Performance nur minimal, sodass dieser Unterschied in der Grafik kaum ablesbar ist. Erst Rechengenauigkeiten von zehn oder acht Bit erhöhen die Fehlerrate spürbar.

Die Beschränkung auf 12-Bit-Ausgabewerte führt in der Regel nur zu einem geringen Verlust gegenüber der 15-Bit-Verarbeitung. Die Verarbeitungsgeschwindigkeit kann dadurch um 10% erhöht werden und 15% der Chipressourcen werden eingespart. Diese Möglichkeit der Verkleinerung ist dann sinnvoll, wenn die Ressourcen des verfügbaren FPGAs knapp sind.

Der Einfluss der Rechengenauigkeit in der FFT ist bei höherwertigen Modulationswertigkeiten größer, was bei OFDM-Systemen, bei denen z.B. eine 256-QAM erlaubt ist, beachtet werden sollte. Wird der Fehlerschutz der Übertragungsstrecke aktiviert, wird das System jedoch insgesamt robuster, also auch gegenüber möglichen Quantisierungsfehlern innerhalb dieses Verarbeitungsschrittes. Weitere Untersuchungen zum Einfluss der FFT- Rechengenauigkeit sind in [STE05] dargestellt.

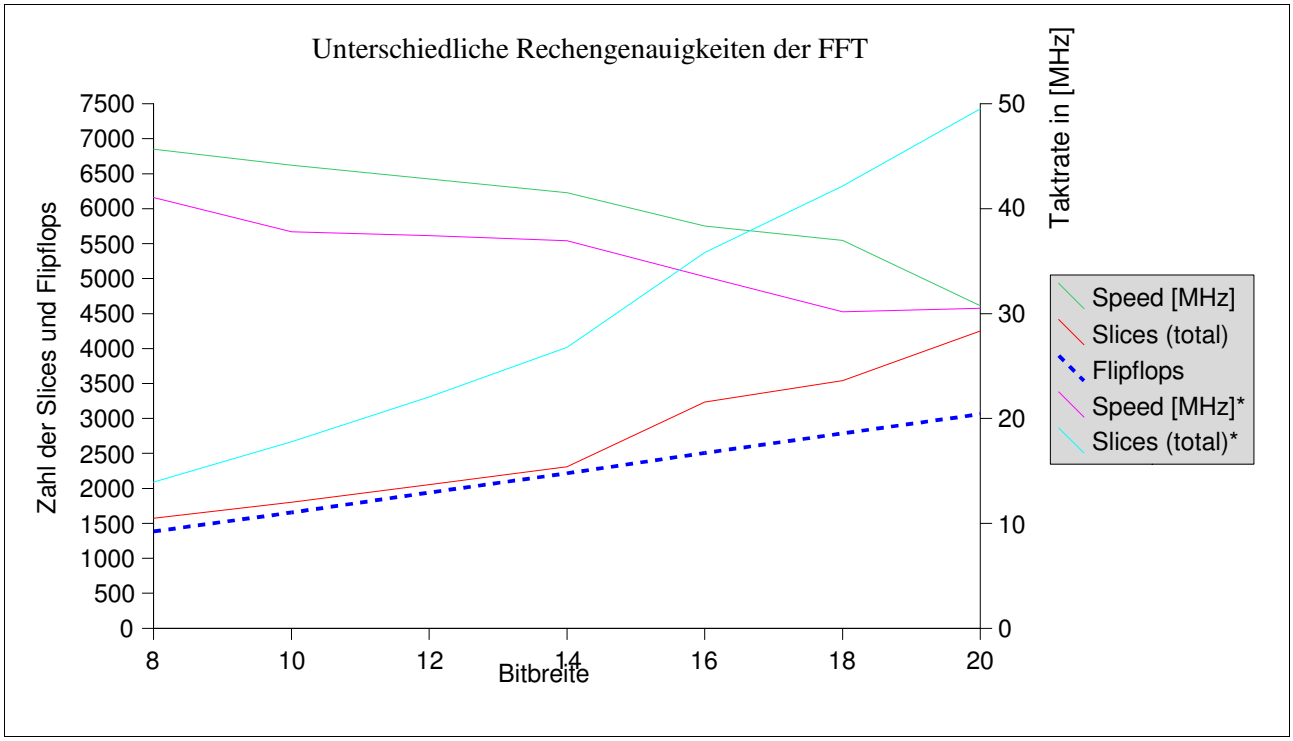


*Abb. 5.26 Ressourcen und Geschwindigkeit verschiedener FFT Implementierungen * FFT-Schaltung ohne die Multiplizierer des Virtex-II FPGA*

Ein zusätzliches Kriterium sind die im FPGA zur Verfügung stehenden Multiplizierer, die im Virtex-II ohnehin Werte von 18 Bit Breite verarbeiten können. Fehlen der FPGA-Technologie diese Elemente, müssen sie aus Standardzellen aufgebaut werden. In diesem Fall spart eine geringere interne Wortbreite deutlich mehr Ressourcen.In dem in dieser Arbeit aufgebauten OFDM-Demonstrator wurde eine interne Rechengenauigkeit von 16 Bit gewählt. Diese Schaltung erfordert 3233 Slices im FPGA, was einem Verbrauch von 8,3% der Ressourcen des gewählten Bausteins entspricht. Da in diesem System Sender und Empfänger in einem Chip realisiert sind, ergibt der Aufwand für die 64'er FFT insgesamt eine Belegung von 16,8%. In Systemen mit größeren FFT-Längen wird dieser Teil deutlich komplexer.

In Tabelle 5.10 sind die Implementierungsdaten noch einmal zusammengestellt. Die Schaltung ist so aufgebaut, dass sie kontinuierlich arbeitet. Es wird pro Taktzyklus ein Eingangswert verarbeitet und ein Ausgangswert weitergegeben.

FFTs sind in der digitalen Signalverarbeitung inzwischen sehr verbreitet. So wird z.B. von der Firma Xilinx auch ein IP-Core angeboten, der eine stark auf die Architektur der Virtex FPGAs optimierte FFT nutzt [XIL03]. In diesem Core wird intern mit deutlich höheren Taktraten gerechnet (verglichen mit dem Systemtakt des Datenstromes) und es erfolgt eine blockweise Verarbeitung, bei der zunächst ganze FFT-Blocklängen zwischengespeichert werden. Diese IP-Cores sind im Allgemeinen gut einsetzbar. Sie werden im Rahmen dieser Arbeit nicht verwendet, weil die kontinuierliche Verarbeitung besser zu dem Datenfluss des OFDM-Systems passt und weil die Verwendung eines IP-Cores eine Herstellerabhängigkeit erzeugt, die der selbst entwickelte VHDL-Code vermeidet. Die in Tabelle 5.10 mit aufgelisteten vier Block-RAMs werden bei dieser Implementierung genutzt, um die Daten für das Guard-Intervall einzufügen und um die Ausgangswerte in die gewünschte Reihenfolge zu sortieren, da am Ausgang der Radix-2^2 Struktur die Ausgangswerte nicht in fortlaufender Reihenfolge anliegen (vgl. Abb. 5.24).

	IFFT (gleich wie FFT)
Slices	3233
Flipflops	2507
LUTs	3593
Multiplizierer	8
RAMs	4
Taktrate	38,35 MHz

Tabelle 5.10 Implementierungsdaten der Fast-Fourier-Transformation

Bei einem OFDM-System nach dem HiperLAN/2-Standard mit einer 64'er FFT-Länge bildet die IFFT den deutlich größten Schaltungsteil des Senders in der physikalischen Schicht. Im komplexeren Empfänger ist es immer noch einer der vier größten Verarbeitungsblöcke.

5.5 Synchronisation

Eine Synchronisation ist im Empfänger unbedingt notwendig, um den Beginn von empfangenen Datenpaketen zu erkennen. Korrelationsalgorithmen können diesen Zeitpunkt schätzen und damit den Datenblock für die weitere Verarbeitung ausschneiden. Neben einer zeitlichen Zuordnung ist es wichtig, die Lage im Spektrum sehr genau zu rekonstruieren. Durch (leichte) Unterschiede bei der Takterzeugung der Mischer im Sender und Empfänger kommt es häufig zu einer Abweichung der beiden Trägerfrequenzen (*Carrier Frequency Offset,* CFO). Eine Frequenzsynchronisation korrigiert etwaige Frequenzabweichungen zwischen den Lokaloszillatoren der Sender- und Empfängerseite.

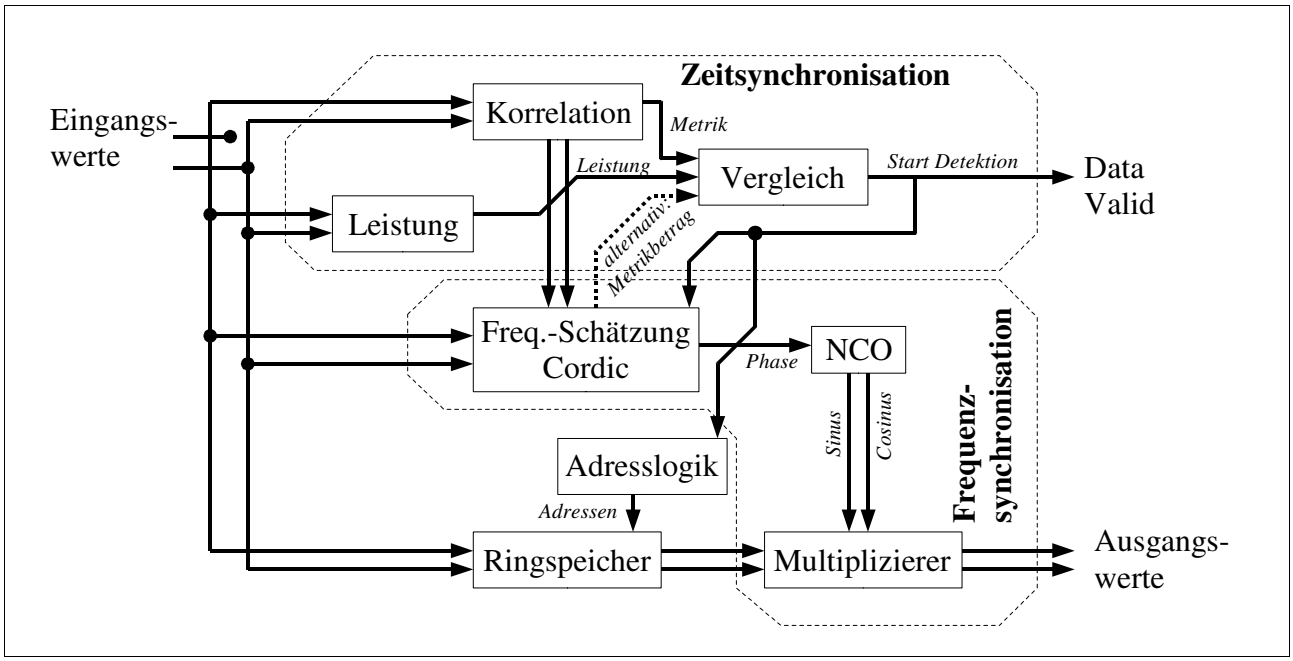


Abb. 5.27 Blockschaltbild der Zeit- und Frequenzsynchronisation

Die Synchronisationseinheit ist die erste Verarbeitungsstufe der digitalen Basisbanddaten innerhalb eines OFDM-Empfängers. Hier werden aus einem kontinuierlichen Datenstrom die Nutzdatenblöcke ausgeschnitten (Zeitsynchronisation). Die Kennzeichnung der gültigen Werte wird über ein *Data-Valid*-Ausgangssignal an die folgenden Verarbeitungsstufen weitergeben. Parallel dazu wird die Frequenzlage des Empfangssignals geschätzt und korrigiert (Frequenzsynchronisation). Eine Verzögerung der Daten (*Delay*) ist in diesem Modul unvermeidbar, wenn, wie in dem implementierten Fall, die ausgewerteten Präambelinformationen Teil des gültigen Datenblocks sind. Die hier genutzten Präambeldaten sind die gleichen, die auch in der

Kanalschätzung verwendet werden[6]. Die Verzögerung liegt in der Größenordnung der Präambeldauer, die bei dem aufgebauten Referenzsystem zwei OFDM-Symbole beträgt.

5.5.1 Zeitsynchronisation

Die Präambel am Beginn eines Datenpakets muss durch die Verarbeitung am Empfängereingang zuverlässig detektiert werden. Der Synchronisationsalgorithmus basiert jedoch nicht darauf, dass die Präambel durch direktes Vergleichen erkannt werden muss. Dies wäre nach der Signalverfälschung durch den Kanal sehr unzuverlässig. Vielmehr wird eine Präambel eingesetzt, die aus einer sich wiederholenden Sequenz besteht. Ein Algorithmus, der auf einer Korrelation dieser periodischen Sequenzen basiert, bietet den Vorteil, dass er Referenzdaten nutzt, die in der gleichen Weise vom Kanal beeinflusst werden [LÜK92][MÜL00]. Für die Gleichheit der Kanaleinflüsse müssen jedoch zwei Kriterien eingehalten werden: Die Kanalimpulsantwort muss kurz sein im Vergleich zur Sequenzlänge und die Zeitvarianz muss klein sein in Bezug auf die Präambellänge. Für die maximale Länge der Kanalimpulsantwort ist bei dem OFDM-Systementwurf schon eine Annahme getroffen worden (nicht länger als die Ausdehnung des Guard-Intervalls). Für die Zeitsynchronisation sollte diese Länge noch etwas größer veranschlagt werden, weil die fehlerhafte oder verpasste Erkennung eines Datenblocks direkt zum Datenverlust führt.

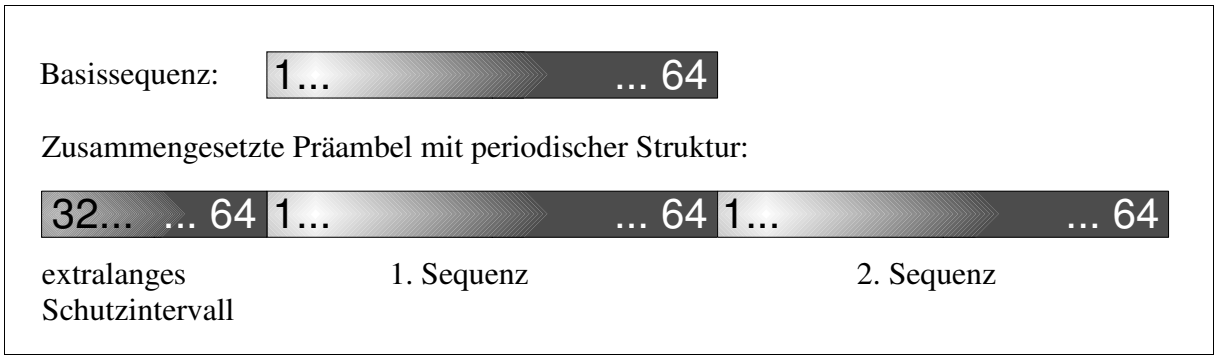


Abb. 5.28 Aufbau einer Präambel zur zeitlichen Synchronisation

In Abb. 5.28 wird die Struktur einer möglichen Präambel aufgezeigt. Zur Erkennung dieser Sequenz werden die Daten in einem festgelegten Abstand (Periodenlänge 64, passend zur IFFT) miteinander verglichen. Eine Korrelationsrechnung ermittelt fortlaufend einen Metrikwert, der mit sehr hoher Wahrscheinlichkeit nur dann ein Maximum erreicht, wenn gerade die Präambel verarbeitet wird. In dem Beispiel aus Abb. 5.28 ist die Sequenz 64 Werte lang. Die

6 Es kann auch eine eigene Präambel zur Synchronisation eingesetzt werden, die dann vollständig verworfen wird. Darauf wurde hier verzichtet, um den Overhead zu reduzieren. Dies entspricht jedoch nicht den Standards HiperLAN/2 und IEEE802.11a.

Korrelationsmetrik wird mit folgender Gleichung beschrieben, wobei N = *64* und $\bar{r}$ die zu r konjugiert komplexe Zahl ist:

$$M_S(nT) = \sum_{n-N}^{n} r(nT) \cdot \bar{r}(n(T - T_s)) \qquad (5.10)$$

Diese Metrik wird in dieser Verarbeitungsstufe mit der empfangenen Leistung, also der Autokorrelation verglichen.

$$S_S(nT) = \sum_{n-N}^{n} r(nT) \cdot \bar{r}(nT) \qquad (5.11)$$

Auf diese Weise wird verhindert, dass absolute Unterschiede bei der empfangenen Leistung, die auch die Metrik skalieren, den Detektionszeitpunkt der Präambel beeinflussen.

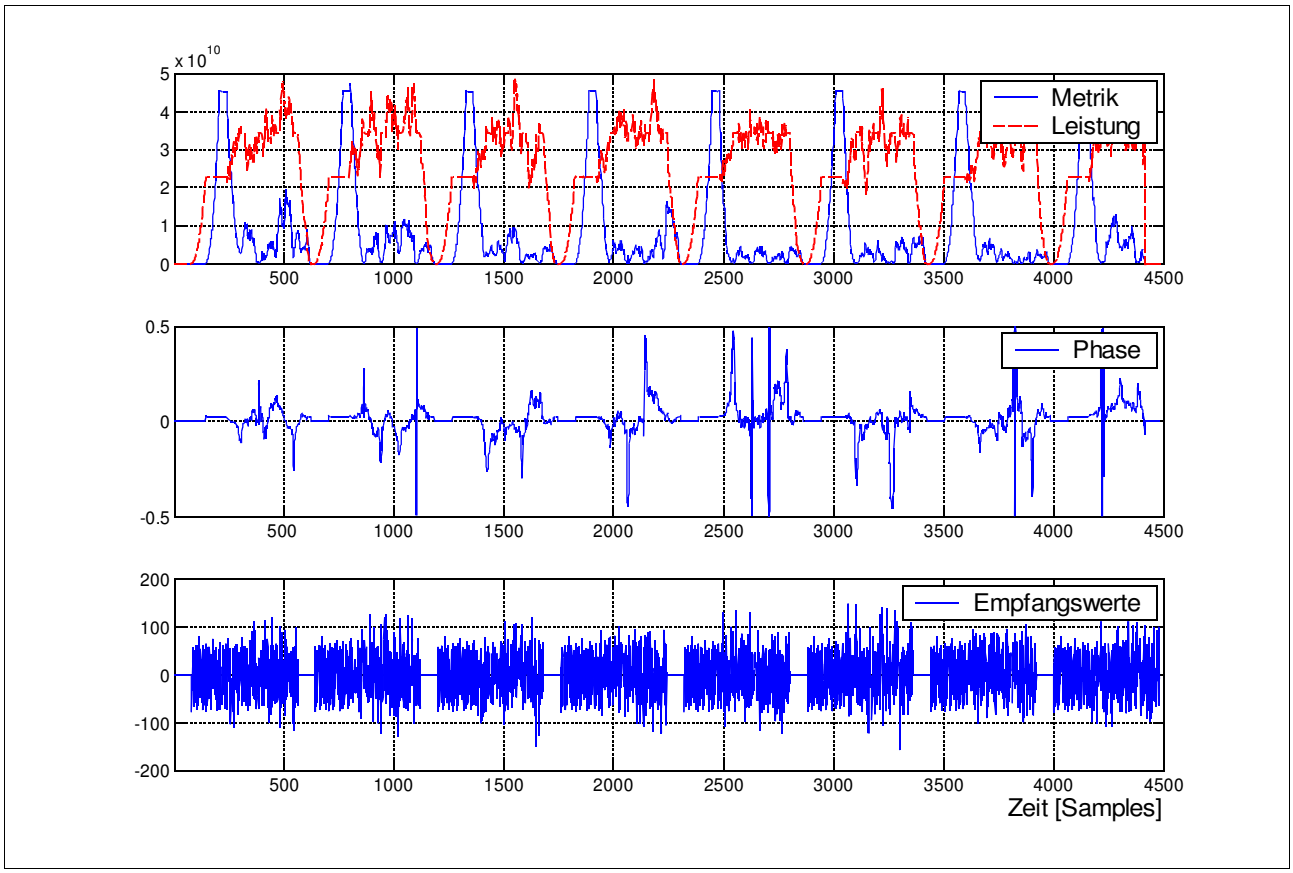


Abb. 5.29 Rechengrößen der Synchronisation: Metrik und Leistung, Frequenzschätzung und Eingangswerte

Abb. 5.29 zeigt die Rechengrößen bei einem unverrauschten Empfangssignal, die in dem Synchronisationsmodul verarbeitet werden. Im oberen Bildausschnitt sind die Korrelationsmetrik und die Referenzleistung aufgetragen. Im mittleren Teil ist das Ergebnis der Frequenzschätzung (vgl. Kap. 5.5.3) und unten ist der Realteil des Eingangssignals dargestellt.

Das bei dem im Rahmen dieser Arbeit aufgebauten OFDM-Demonstrator eingesetzte Entscheidungskriterium für die Wiedererkennung einer Präambel wird dann aktiv, wenn der Betrag des Metrikwertes aus Gleichung 5.10 größer ist als die halbe Leistung aus Gleichung 5.11:

$$\left| M_S(nT) \right| = \left| \sum_{n-N}^{n} r(nT) \cdot \overline{r}(n(T-T_s)) \right| > \frac{1}{2} S_S(nT) = \frac{1}{2} \sum_{n-N}^{n} r(nT) \cdot \overline{r}(nT) \qquad (5.12)$$

Außerdem muss der Metrikwert noch eine (relativ kleine) absolute Schwelle überschreiten, damit bei sehr kleinen Rauschsignalen und somit sehr kleiner Leistung die Metrik nicht zufällig diese Bedingung erfüllt. In Abb. 5.29 (obere Grafik) kann diese Schwelle anschaulich bei ca. $2*10^{10}$ festgesetzt werden. In dem hier dargestellten unverrauschten Fall wird deutlich, dass die Metrik nur am Blockanfang, also in der Präambelphase, deutlich ansteigt, wobei die Leistung während des gesamten OFDM-Symbols auf einem hohen Level bleibt. Die Einbrüche von Metrik und Leistung entstehen hier durch kurze Sendepausen zwischen den Datenblöcken. Die Leistungskurve ist in dieser Grafik mit dem Faktor 0.5 skaliert, damit die Detektionsbedingung aus Gleichung 5.12 direkt abgelesen werden kann (hier wäre das erste Synchronisationsevent ungefähr bei dem Zeitindex 170).

Abhängig vom verwendeten Entscheidungskriterium muss noch der Abstand zwischen Synchronisationsevent und dem Anfang des Datenblocks bestimmt werden. Da der Datenblock inklusive Präambel weiterverarbeitet wird, ist dieser Index negativ. Der untere Bildausschnitt von Abb. 5.29 zeigt deutlich, wann wieder Daten empfangen werden. Beim Zeitindex 60 wird der Anfang der Präambel empfangen, die zum Zeitindex 170 detektiert wird.

Im FPGA werden die jeweiligen Empfangsdaten ständig in einem Ringspeicher vorgehalten, sodass immer das vollständige Datensymbol weitergegeben werden kann, obwohl die Detektion erst 110 Takte nach dem Blockanfang erfolgt. Der Abstand zwischen Synchronisationsevent und dem Anfang des Datenblocks beträgt in diesem Fall -110.

Für die Implementierung der Synchronisation können mehrere Punkte berücksichtigt werden, die den Schaltungsaufbau im FPGA deutlich vereinfachen.

- Um Divisionen zu vermeiden, werden gleich lange Datenblöcke für die Metrik und für die Leistung aufsummiert (in diesem Fall Blöcke der Länge 64). In diesem Fall können die Werte

direkt verglichen werden, ohne eine Mittelung durch die Division mit der Blocklänge auszuführen.

- Die Summe der 64 Werte wird nicht in jedem Takt vollständig errechnet. Es wird nur der jeweils neue Wert hinzuaddiert und der am weitesten zurückliegende Wert abgezogen. Die zurückliegenden Werte werden einem zusätzlichen Speicher entnommen.
- Da die Metrik komplexwertig ist, können Metrik- und Leistungswert erneut quadriert werden, um die Betragsermittlung des komplexen Wertes zu vermeiden. Eine einfache Multiplikation ersetzt damit einen deutlich aufwendigeren iterativen Schaltungsblock zur Betrag- und Phasenzerlegung (CORDIC, siehe Anhang D).

Bei der Zeitsynchronisation wird anhand des Vergleichs von Metrik und Leistung ein Zeitpunkt ermittelt, der es ermöglicht, die OFDM-Symbole passend auszuschneiden. Weil die eigentlichen Daten für die weitere Verarbeitung dabei unverändert bleiben, muss die Rechengenauigkeit in dieser Einheit nur daran bemessen werden, ob die Detektion der Blockanfänge sicher gewährleistet ist.

Die bei der Leistung und Metrik durchgeführte doppelte Quadrierung weitet jedoch den möglichen Wertebereich sehr stark auf. Diese Abbildung von $f(x) \sim x^4$ bildet einen dann maximalen Eingangswert von 2^{15} auf einen Ergebnisraum von 2^{60} ab. Somit müsste die Auswertung der Synchronisation auf 60 Bit breiten Zahlen erfolgen. Das ist wenig effizient und verbraucht zu viele Ressourcen. Eine Anpassung des Wertebereichs ist in diesem Fall ein Parameter, der in Abhängigkeit der analogen Empfangseinheit bestimmt werden muss. Kleine Wortbreiten verhindern gegebenenfalls zwar, dass die Zahlenwerte insgesamt sehr groß werden, löschen dabei jedoch die Information der kleinen Eingangswerte vollkommen aus. In Gleichung 5.13 wird die Größenordnung bestimmt, in der die Rechenwerte für Leistung und Metrik liegen, wenn nach jeder Quadrierung das Zwischenergebnis wieder auf eine 16 Bit Wortbreite reduziert wird.

$$f(x) \sim \left(x^2 \frac{1}{2^{15}} \right)^2 \frac{1}{2^{15}} \sim \frac{x^4}{2^{45}} \qquad (5.13)$$

Diese Gleichung macht deutlich, dass durch die Anpassung des Wertebereichs nach jeder Quadrierung alle Werte verloren gehen, deren vierte Potenz kleiner ist als 2^{45}. Bei einer optimalen Aussteuerung der Eingangswerte ist das unkritisch. Wenn die Empfangsleistung jedoch schwankt, z.B. durch Fading-Kanäle und eine ungenaue *Automatic Gain Control (AGC)*, setzt die Synchronisation schnell aus.

Zur Kompensation dieses Effekts kann die Wortbreite der Metrik- und Leistungswerte erhöht werden, was jedoch unmittelbar den Schaltungsaufwand erhöht. Eine weitere Möglichkeit ist, die

Eingangswerte durch größenabhängige Bitshiftoperationen so zu verändern, dass immer die MSBs besetzt sind. Da dies für die Leistungs- und Metrikberechnung in gleicher Weise geschieht, wird das Verhältnis der beiden Größen nicht verändert. Die Detektion benötigt lediglich die Information, ob der Metrikwert doppelt so groß ist wie die Leistung.

Ein anderer Ansatz ist, auf die zweite Quadrierung zu verzichten und doch den Betrag der Korrelationsmetrik zu bestimmen. Dann ist die Wertebereichsanpassung weit weniger kritisch, weil bei Gleichung 5.13 nur noch der Term innerhalb der Klammer ausgewertet wird. Es muss jedoch ein CORDIC-Schaltungsblock integriert werden.

An dieser Stelle besteht die Möglichkeit, einen Schaltungsteil im FPGA doppelt zu verwenden, der zu unterschiedlichen Zeiten benötigt wird. Ein CORDIC-Block ist bei der im folgenden Kapitel betrachteten Frequenzsynchronisation unvermeidlich. Die Bestimmung des Phasenoffsets geschieht jedoch nur einmal pro Präambel bzw. Datenblock. Erst nachdem die Zeitsynchronisation den Anfang eines Datenblocks detektiert hat, wird (nach einer definierten Zeit, z.B. 30 Takte) einmalig der Phasenoffset ermittelt. Es ist somit sichergestellt, dass Zeit- und Frequenzsynchronisation niemals zum gleichen Zeitpunkt aktiv sind. Damit ist es möglich, diesen CORDIC-Block (der prinzipbedingt immer Betrag und Phase bestimmt) sowohl zur Betragsbestimmung der Metrik als auch zur Ermittlung des Frequenzoffsets zu nutzen.

Während bei vielen anderen OFDM-Modulen die FPGA-Implementierung ohne nennenswerte Performanceeinbußen vereinfacht werden kann, ist die Synchronisation sehr stark an den verwendeten Algorithmus gekoppelt. Hier kann nur schwer eine alleinige optimale Umsetzung ermittelt werden. Insbesondere muss hier berücksichtigt werden, ob das analoge Frontend eine exakte AGC sicherstellen kann. Darüber hinaus muss festgelegt werden, bis zu welchem Signal- zu Rauschverhältnis die Synchronisation sicher arbeiten soll. Die beschriebenen Regeln zur Implementierung sollen jedoch Anhaltspunkte geben, um den Synchronisationsblock für FPGAs optimiert entwerfen zu können.

Die Implementierungsdaten der im Rahmen dieser Arbeit untersuchten Varianten sind in Tabelle 5.11 aufgeführt. Im ersten Fall ist eine Version mit einer Rechengenauigkeit von 16-Bit implementiert worden, die ohne die Verwendung eines CORDIC-Blocks auskommt. Im zweiten Fall wird die CORDIC-Schaltung aus der Frequenzsynchronisation mitbenutzt und die Rechengenauigkeit auf 12-Bit reduziert, da hier eine Stufe der Wortbreitenanpassung entfällt. Dies ist die später im Gesamtsystem eingesetzte Variante.

	16-Bit-Synchronisation	12-Bit-Synchronisation (klein)
Slices	1341	968
Flipflops	541	510
LUTs	2504	1772
BRAMs	8	8
Multiplizierer	13[7]	10
Taktrate	33,04 MHz	45,42

Tabelle 5.11 Implementierungsdaten der Zeitsynchronisation im FPGA

Der benötigte Arbeitstakt von 20 MHz wird von beiden Varianten übertroffen. Die Geschwindigkeit von der 12-Bit-Version wird von dem CORDIC-Block limitiert, welcher aber auch schneller implementiert werden kann (vgl. Anhang D). Insgesamt zählt dieser Schaltungsteil der Synchronisation mit 8,6% der verbrauchten Slices zu den größeren Modulen im OFDM-Empfänger.

Die hier besprochene Zeitsynchronisation ist in der Lage, die am Empfänger eintreffenden Daten zuverlässig im erlaubten Toleranzbereich auszuschneiden. Hierbei wird die zeitliche Toleranz in der Regel durch das Guard-Intervall bestimmt. In dem in dieser Arbeit betrachteten System können somit das 64 Bit lange OFDM-Symbol aus dem inklusive Guard-Intervall 80 Bit langen Sendesymbol ausgeschnitten werden. Wobei je nach Übertragungssituation ein Ausschnitt hinteren Bereich des Empfangssymbols am wenigsten Störungen durch Nachbarsymbole aufweist. Der mögliche zeitliche Versatz des ausgeschnittenen Symbols führt zu einer Drehung der Konstellation. Diese Phasenverschiebung ist unvermeidlich, kann jedoch durch die später noch beschriebene Kanalschätzung wieder korrigiert werden.

5.5.2 Einfluss von Rauschen auf die Zeitsynchronisation

Bis hierher wurde das Prinzip der Zeitsynchronisation unter der Annahme einer rauschfreien Übertragung betrachtet. In diesem Fall ist es offensichtlich einfach, einen Datenblock exakt auszuschneiden, weil die Rechengrößen aufgrund der unverfälschten Präambel immer exakt identisch sein werden. Kommen jedoch Rauschen und Leistungseinbrüche im genutzten Frequenzband hinzu, kann die Synchronisation nicht mehr so exakt arbeiten. Der Korrelationsalgorithmus funktioniert aber auch noch im verrauschten Fall, solange das Rauschen oder der Leistungseinbruch nicht zu stark werden. Und bei einem ungenügenden SNR ist auch der nachfolgende Empfänger ohnehin nicht mehr in der Lage, die Nutzdaten zu rekonstruieren.

7 Die drei zusätzlichen Multiplizierer sind auf das Quadrieren von Leistung und Metrik zurückzuführen, welches in den 12-Bit-Versionen vermieden wird.

Der in Kap. 5.5.1 vorgeschlagene Algorithmus wird hier noch einmal hinsichtlich der Robustheit gegenüber realen Übertragungsbedingungen betrachtet. Das Verhalten der Korrelationsauswertung ist in Abb. 5.30 für drei Fälle betrachtet: rein additives Rauschen (*Additive White Gaussian Noise*, AWGN), Fading-Einbrüche durch Mehrwegeempfang und die Kombination dieser beiden Einflüsse.

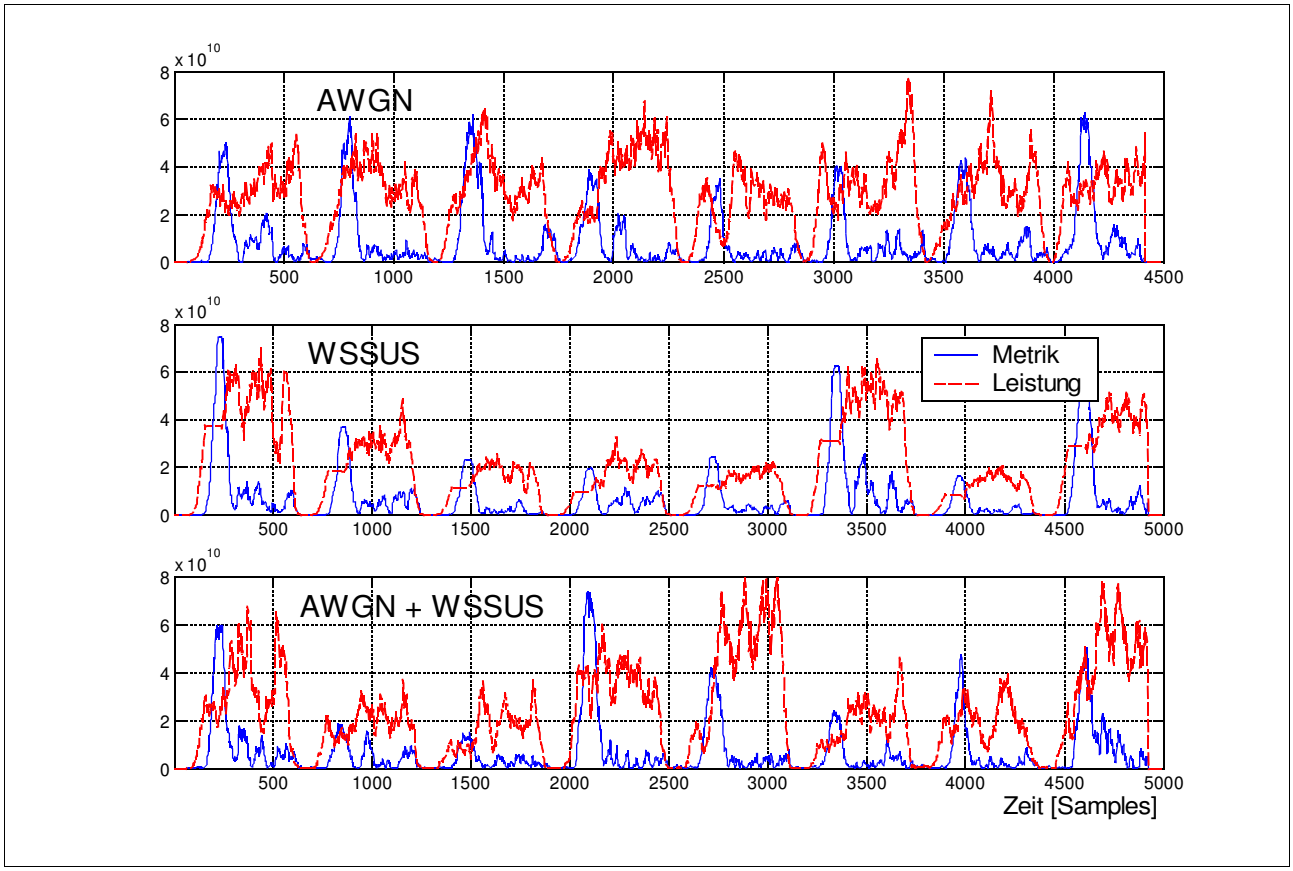


Abb. 5.30 Einfluss von Rauschen und Mehrwegeausbreitung auf die Synchronisation

Additives Rauschen hat einen deutlichen Einfluss auf die Genauigkeit der Synchronisation. Dabei ist nicht nur die Tatsache von Belang, dass die Rechengrößen wie Korrelationsmetrik und Leistung einen unsauberen Signalverlauf annehmen. Die Abb. 5.30 zeigt auch, dass AWGN allgemein die Leistung des Empfangssignals anhebt, während die Korrelationsmetrik in der Größenordnung des rauscharmen Falls bleibt. Etwas verliert auch die Metrik, weil die Korrelationssequenz durch den Rauscheinfluss nicht mehr zu 100% exakt ist. Insgesamt führt das Rauschen zu einer Verschiebung des Detektionszeitpunktes nach hinten. Die Abb. 5.31 und 5.32 zeigen diesen Sachverhalt als Histogramm der ermittelten Indizes in Abhängigkeit des additven Rauschens, wobei sich die Detektionszeitpunkte gegenüber kleinem SNR (vorne in der dreidimensionalen Grafik) mit Zunahme der Rauschleistung immer mehr nach hinten (rechts in der

Abb.) verschieben. Die in Abb. 5.32 gezeigte Kurve beruht auf einem modifizierten, im Folgenden noch näher betrachteten Detektionsverfahren, welches diesen Effekt weitgehend kompensiert.

Die Grafiken zeigen, dass die Zahl der Detektionen bei einem schlechten SNR deutlich abnimmt. Ab einem SNR-Wert von 0dB fällt die Zuverlässigkeit der Präambelerkennung bei diesen Verfahren schon sehr stark ab. Wenn die Rauschleistung deutlich über der Leistung des Nutzsignals liegt (was für eine sinnvolle Datenübertragung kaum geeignet wäre), muss der Synchronisationsalgorithmus daran zusätzlich angepasst werden.

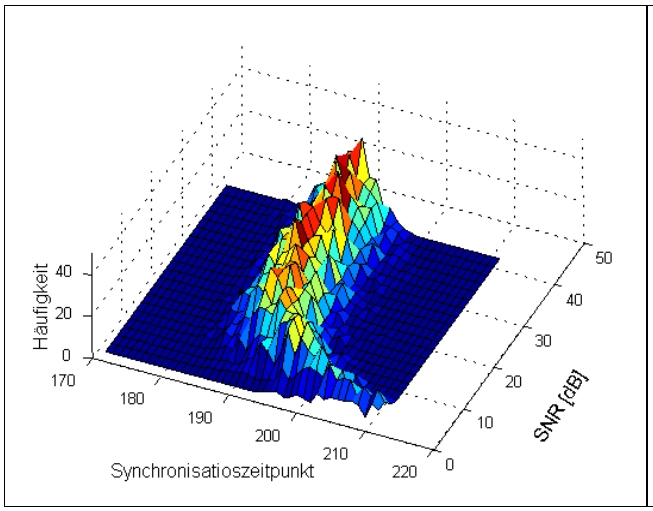


Abb. 5.31 Synchronisationsdetektion ohne Offest-Ausgleich

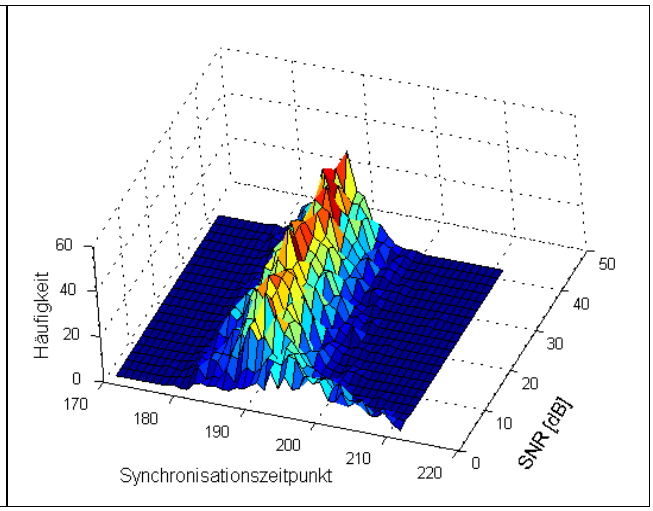


Abb. 5.32 Synchronisationsdetektion mit Offest-Ausgleich

Der Unterschied der beiden in den Abbildungen betrachteten Verfahren liegt in der Berechnung der Leistung. Bei dem herkömmlichen Verfahren wird zu Beginn der Rechnung die Leistung der ersten 64 Eingangswerte aufsummiert. Anschließend wird das Fenster der Auswertung um einen Zeitschritt nach hinten versetzt. Im FPGA wird dabei nicht erneut die Summe aus 64 Leistungswerten gebildet, sondern nur der (eine) neue Wert hinzuaddiert und der herausgefallene Wert subtrahiert. Dies ist schaltungstechnisch sehr viel aufwandsgünstiger.

$$\begin{aligned} S(n+1) &= S(n) + |r(n+1)|^2 && , n = 1..64 \\ S(n+1) &= S(n) + |r(n+1)|^2 - |r(n-64)|^2 && , n = 65..\infty \end{aligned} \tag{5.14}$$

Die ersten 64 Takte sind eine Initialisierungsphase, bei der die Leistung des Signals kontinuierlich größer wird (weil noch keine Werte aus dem Fenster herausfallen). Die Größe des Wertes in dieser Startphase ist jedoch sehr stark vom Rauschen bzw. von der Rauschleistung abhängig.

Anschaulich bedeutet dies: Wenn die Synchronisation in einer Empfangspause aufstartet, wäre der Initialisierungswert im rauschfreien Fall Null, im verrauschten Fall jedoch die aufsummierte Leistung der Rauschanteile. Abb. 5.33 zeigt den Verlauf der beiden Leistungsgrößen bei starkem Rauschen im Vergleich.

Die definitionsgemäß errechnete Signalleistung (grün) ist neben dem modifizierten Leistungssignal (blau) dargestellt, welches ohne die Summe in der Initialisierungsphase berechnet wird, wo also *S(0)...S(64)=0* gesetzt werden. Hier bleibt das Signal in Empfangspausen auch zu Zeitpunkten größer *n=64* offset-frei, da der addierte und der subtrahierte Wert (also ausschließlich Rauschanteile) im Mittel gleich groß sind.

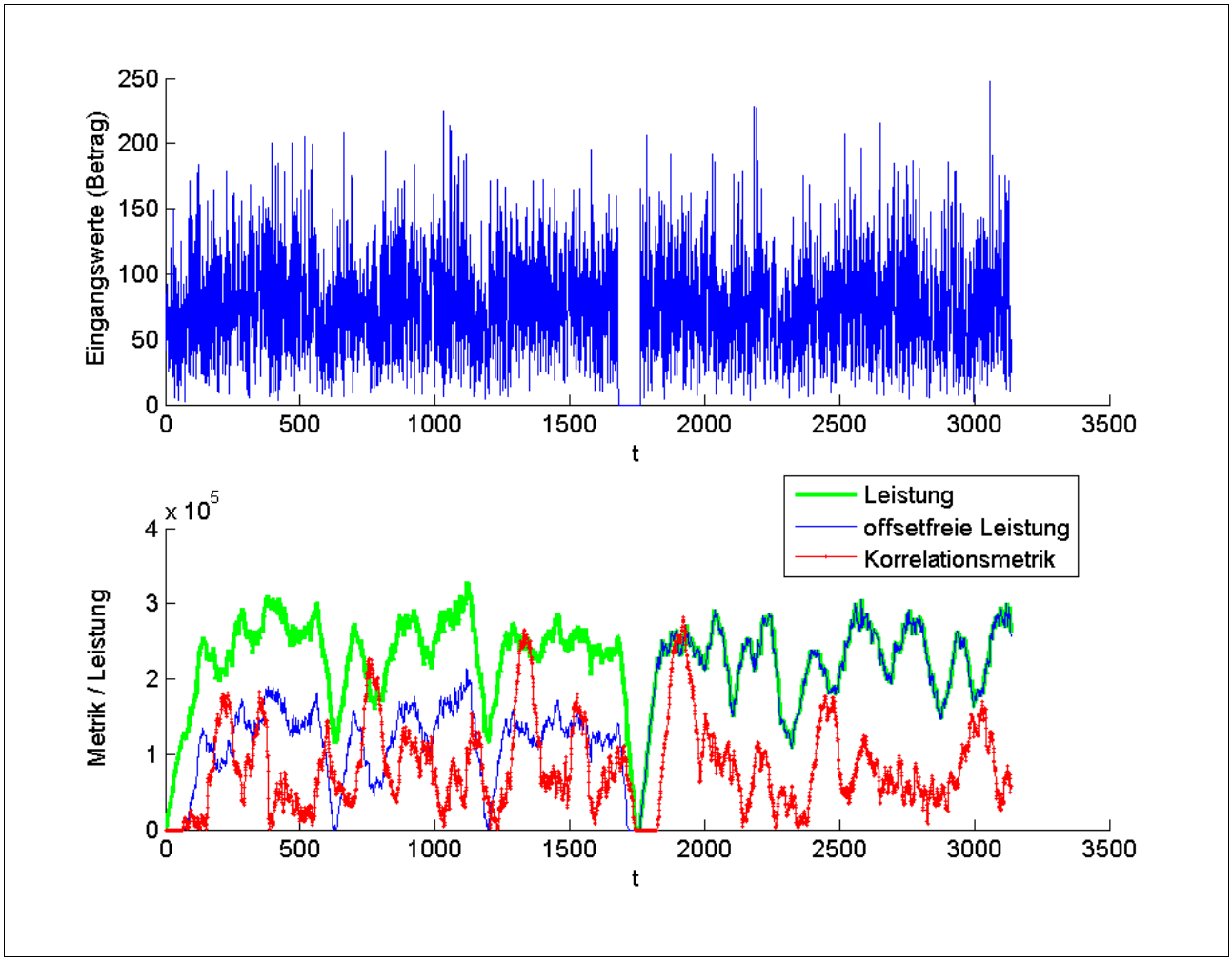


Abb. 5.33 Vergleich der normalen und der offestfreien Leistung

Bis auf den Offset verlaufen die beiden verglichenen Leistungssignale identisch. Da die Korrelationsmetrik nicht durch AWGN angehoben wird (hier werden die Rauschanteile herausgemittelt), wird die Zeitsynchronisation nach Gleichung (4.8) durch den Offset ungenauer.

Der höhere Level der offsetbehafteten Vergleichsleistung führt zu einer späteren Detektion, oder, wenn das Rauschen zu groß ist, schneller zum Verlust von Datenblöcken.

In Abb. 5.33 ist ab dem Zeitpunkt *n=1700* eine völlig rauschfreie Empfangspause dargestellt. Dieser theoretische Fall macht deutlich, dass dann beide Leistungssignale identisch werden. Eine solche Phase führt zwischenzeitlich zu einem *S(n)=0*. Dieses Verhalten des offset-kompensierenden Verfahrens legt nahe, zwischen den einzelnen Synchronisationsblöcken eine Reinitialisierung mit *S=0* durchzuführen, um auf längerer Sicht einem Verlust der Offsetfreiheit vorzubeugen.

Schaltungstechnisch bedeutet die Implementierung des vorgeschlagenen offsetfreien Vergleichssignals keinen Mehraufwand. Sie bringt jedoch eine zuverlässigere und stabilere Erkennung des Synchronisationszeitpunktes. Insbesondere die zeitliche Verschiebung bei zunehmendem Rauschen wird vermieden.

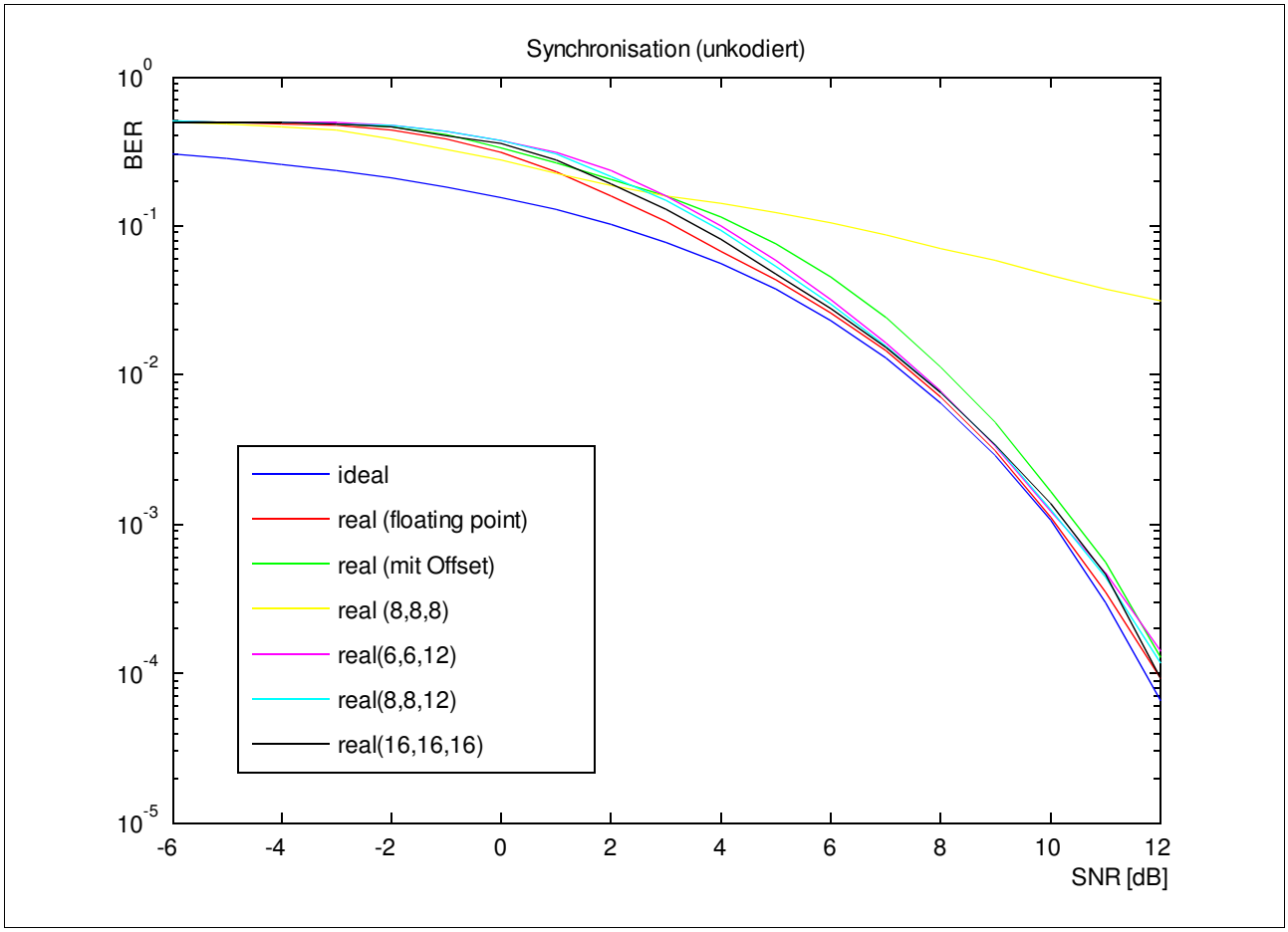


Abb. 5.34 BER im uncodierten Fall in Abhängigkeit der Implementierungs-Parameter

Dennoch ist eine exakte Zeitsynchronisation in Kanälen mit additivem Rauschen sehr anspruchsvoll. Der Ausschnitt des Datenblocks kann immerhin noch die Toleranz des eingefügten

Guard-Intervalls nutzen, wobei diese Werte unter Umständen schon leicht verfälscht sein können. In Abb. 5.34 ist der Einfluss der Synchronisation auf die Bitfehlerrate in Abhängigkeit des SNR abzulesen. Hier ist ein uncodiertes QPSK-moduliertes Signal bei einem AWGN-Kanal untersucht worden.

Als Referenz ist eine Kurve eingezeichnet, bei der der Beginn der Datenblöcke bekannt ist und die Symbole ideal ausgeschnitten werden (blau). Darüber hinaus sind zwei Kurven mit *Floating-Point*-Genauigkeit (64 Bit) simuliert worden, wobei die rote Kurve mit und die grüne Kurve ohne Offset-Ausgleich (vgl. Abb. 5.31 und 5.32) erzeugt wurden. Die übrigen Kurven nutzen den besseren offsetfreien Algorithmus, wobei in den Klammern die Verarbeitungswortbreiten der drei Rechenebenen angegeben sind: Eingangs-, Verarbeitungs- und Ausgangswortbreite. Es wird deutlich, dass acht Bit in der Entscheiderstufe nicht ausreichen, um bei gutem SNR sicher in den fehlerfreien Betrieb zu gelangen. Die Fälle, in denen die letzte Stufe den Metrik-/Power-Vergleich mit mindestens 12 Bit ausführen kann, arbeiten sehr zuverlässig. Die Vorteile der sehr hohen Genauigkeiten sind fast nur in den Übergangsbereichen mit einer BER um 10% zu unterscheiden.

Bei sechs Bit Eingangsgenauigkeit (6,6,12) ist kaum noch ein Qualitätsverlust zu erkennen, wobei hier die Annahme einer perfekten Aussteuerung am Empfängereingang getroffen wird. Die in dem aufgebauten Referenzsystem implementierte Schaltung mit acht Bit Eingangswortbreite und Zwölf-Bit-Entscheiderstufe ist dagegen deutlich toleranter gegenüber einer ungenauen AGC.

5.5.3 Frequenzsynchronisation

Die Frequenzsynchronisation im Empfängereingang ist eine Feinsynchronisation bzw. eine Justierung der Subträger auf die vorgesehenen Frequenzen. Der Mischprozess (eine Multiplikation) mit einem Versatz der Trägerfrequenz (CFO) führt nach der Fourier-Transformation zu einer Verschiebung des OFDM-Spektrums. In der Frequenzsynchronisation wird ermittelt, ob sich aufgrund unterschiedlicher Mischerfrequenzen im Sender und Empfänger die Subträger gegenüber ihrer Ausgangslage leicht verschoben haben. Leicht deshalb, weil sich prinzipiell in diesem Verarbeitungsschritt nur Frequenzverschiebungen ermitteln lassen, die kleiner sind als maximal der halbe Subträgerabstand. In der Regel arbeiten aktuelle Oszillatoren jedoch so genau, dass der Offset auch nicht größer wird.

Die Möglichkeit der Frequenzschätzung beruht darauf, dass sich die Präambel nach 64 Werten, also einer Symbollänge, wiederholt. Wird jeweils der aktuelle Wert mit dem Wert vor 64 Takten verglichen, so gäbe es im idealen Fall keinen Phasenunterschied. Ist jedoch im Empfänger ein konstanter Phasenoffset entstanden, kann dieser während der Präambelverarbeitung an einer Phasendrehung der sich wiederholenden Sequenz abgelesen werden.

Dass hier die Phasen von zwei Werten verglichen werden, die eine volle Symbollänge auseinander liegen, führt dazu, dass der Phasenoffset nicht größer sein darf als ±π/64; denn sollte er größer sein, würde sich der Vektor bei 64 Schritten mehr als um ±180° drehen. Und da die Zahl der Drehungen bei dieser Verarbeitung nicht nachvollziehbar ist, kann der Winkel dann nicht mehr eindeutig aufgelöst werden.

Der Phasenunterschied kann generell (identisch) an jedem Wertepärchen im Abstand einer Symbollänge innerhalb der Präambel abgelesen werden. Um die Frequenzschätzung jedoch robust gegenüber zufällig verteiltem Rauschen zu machen, werden mehrere Phasenunterschiede gemessen und gemittelt.

Eine genaue Betrachtung der Korrelationsmetrik aus Gleichung 5.14 zeigt, dass ihre Phase einer Mittelung über 64 Wertepärchen entspricht.

Die innere Multiplikation r(t) mit dem konjugiert komplexen r(t-64) ergibt als Winkel die Phasendifferenz der beiden Werte und die Aufsummierung über 64 komplexe Vektoren entspricht einer Mittelung der Phase.

Die resultierende Phase der Korrelationsmetrik ist in Abb. 5.29 dargestellt. In diesem ungestörten Fall wird deutlich, dass der Phasenwert ein Plateau bildet, solange sich der Verarbeitungsblock vollständig innerhalb der Präambel befindet.

Die Frequenzsynchronisation ermittelt an einer Stelle innerhalb dieses Plateaus den Phasenwert der Korrelationsmetrik und gibt diesen an einen *Numerically Controlled Oscillator* (NCO) weiter. Dieser gibt die entsprechenden Sinus- und Cosinuswerte aus, mit denen die Empfangswerte multipliziert werden. Diese Multiplikation im Zeitbereich entspricht einer Verschiebung im Frequenzbereich und schiebt das Spektrum des OFDM-Signals so, dass die Subträger wieder an der vordefinierten Frequenz auszuwerten sind.

Etwas allgemeiner formuliert ist die maximale Frequenzkorrektur dabei von dem Abstand der Wertepärchen innerhalb der Präambel abhängig, was ggf. als Entwurfsparameter variiert werden kann. Wie schon erwähnt, kann in der komplexen Ebene nur der Winkelbereich π bis $-\pi$ aufgelöst werden. Die maximal erlaubte Phasendrehung resultiert daraus als

$$f_{Offset} \leqslant \left| \frac{\pi}{\Delta T} \right| \qquad (5.15).$$

Je kürzer der Korrelationsblock, desto kleiner bleibt der Betrag der Metrik, aber desto größer ist der zulässige Frequenzshift. Bei kurzen Blocklängen werden jedoch die Metriken und die Phasenschätzung über eine geringere Stichprobenzahl gemittelt und sind somit empfindlicher gegenüber Rauschen.

In Abb. 5.35 ist der ermittelte Winkel (der in der Simulation vorgegebene Wert liegt bei 0.025 Rad/ π) der Frequenzsynchronisation gegenüber dem Signal- zu Rauschverhältnis aufgetragen. Ähnlich wie bei der Zeitsynchronisation ist bei einem hohen Rauschanteil kaum ein exakter Wert zu ermitteln. Bei stärkerer Dominanz des Nutzsignals wird der Phasenoffset zunehmend sicher erkannt.

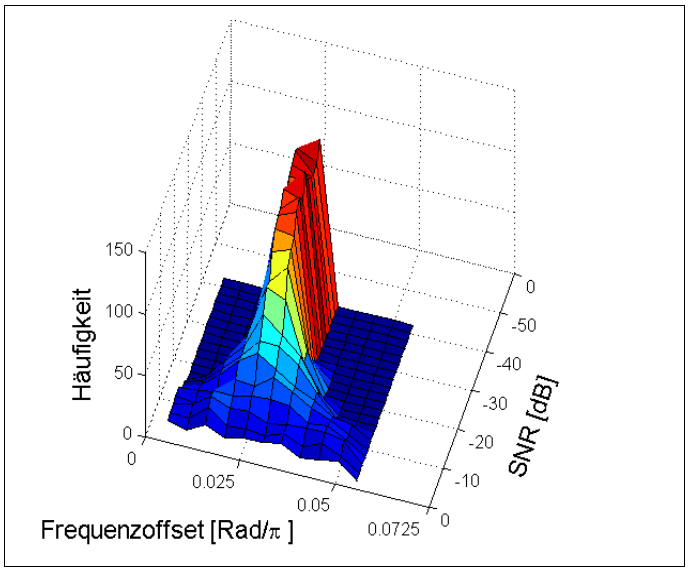


Abb. 5.35 Frequenzsynchronisation und

Der während dieser Präambelverarbeitung der ermittelte Frequenzoffset wird direkt durch eine Phasendrehung korrigiert. Die Komplexität dieser Schaltung ergibt sich aus einem CORDIC-Block, der aus der vorhandenen komplexen Metrik die Phase ermittelt, einer gespeicherten Tabelle für Sinus- und Cosinuswerte und einer komplexen Multiplikation. Für die Ausgabe von Sinus- und Cosinuswerten kann auch ein Makro (*IP-Core*) eingesetzt werden, welches automatisch nur mit Hilfe des vorgegebenen Phasenoffsets die Funktionswerte generiert. Diese Standardschaltungen, die aus einem Zähler und einer Wertetabelle konstruiert sind, nutzen die Symmetrien der Sinus- und Cosinusfunktion, um Konstantenspeicher auf ein Minimum zu reduzieren. Die IP-Cores sind unter den Namen *Numerically Controlled Oscillator* (NCO) oder *Direct Digital Synthesizer* (DDS) in den Entwicklungsumgebungen zu finden [XIL01].

Die Phasenkorrektur erfolgt im FPGA mit vier Multiplikationen und jeweils einer Addition bzw. Subtraktion, wie in Gleichung 5.16 dargestellt.

$$(a+ib)e^{i\phi}=(a*\cos(\phi)-b*\sin(\phi))+i(b*\cos(\phi)+a*\sin(\phi)) \qquad (5.16)$$

Insgesamt ist dieser Schaltungsteil recht klein in Bezug auf die gesamte Verarbeitungskette. Er ist insbesondere von der Art der Implementierung des CORDIC-Blocks (vgl. Anhang D) abhängig. Für den hier referenzierten OFDM-Demonstrator sind die Werte der FPGA-Schaltung in Tabelle 5.12 angegeben.

Der maximale Schaltungstakt ist hier durch den CORDIC-Schaltungsteil bestimmt, welcher wiederum so umgesetzt wird, dass er den notwendigen Gesamttakt von 20 MHz sicherstellt, ansonsten jedoch so klein wie möglich aufgebaut ist. Bei Bedarf kann hier der Durchsatz erhöht werden.

In Tabelle 5.12 ist zusätzlich eine Implementierung ohne Hardware-Multiplizierer angegeben. An dieser Stelle werden die Multiplikationen nicht für hohe Taktraten benötigt, sodass Multiplizierer für zeitkritische Module gespart werden können. Allerdings erhöht sich dann die Zahl der benötigten Standardzellen.

	Ressourcen (in % bezogen auf das FPGA)	Ohne HW-Multiplizierer
Slices	608 (1,8%)	1168 (3,5%)
Flipflops	246 (0,4%)	246 (0,4%)
Look-Up-Table	1045 (1,6%)	2169 (3,2 %)
Block-RAM	1 (0,7%)	1 (0,7%)
Multiplizierer	13 (2,8%)	0
Taktrate	32,13 MHz	32,13 MHz

Tabelle 5.12 Implementierungsdaten der Frequenzsynchronisation

Die Frequenzsynchronisation ist mit der Korrektur des Frequenzoffsets, was einer Verschiebung des OFDM-Symbols im Frequenzbereich entspricht, ein Mechanismus, der das Empfangsspektrum vor der Verarbeitung exakt justiert. Im folgenden Kapitel der Kanalschätzung und Entzerrung wird auf die Funktion der Pilotsymbole eingegangen, die dort zum Nachführen der Phasenlage (*Phase Tracking,* vgl. Kapitel 5.6.2) genutzt werden. Dieser Verarbeitungsschritt korrigiert den verbleibenden Phasen-Restfehler; dies geschieht nicht mehr subträgerweise sondern nur noch mit einem konstanten Korrekturfaktor für alle Empfangswerte pro OFDM-Symbol. Zu diesem Zweck genügt eine geringe Zahl an Piloten (im HiperLAN/2-System vier) um den Offset zuverlässig zu bestimmen. Im Unterschied zu der hier betrachteten Frequenzsynchronisation ist das Phase Tracking aus den Pilotsymbolen jedoch aktueller, also jeweils aus dem direkten Vorgängersymbol. Bei der oben gezeigten Schätzung wird die Korrektur (analog zur Zeitsynchronisation) erst mit der Verarbeitung einer neuen Präambel aktualisiert und kann somit jeweils schon mehrere OFDM-Symbole alt sein.

Die Exaktheit der Frequenzsynchronisation hängt dabei von der Verfälschung durch additives Rauschen und der zeitlichen Konstanz des CFO ab. Der Einfluss auf die Fehlerrate bei der Demodulation ist darüber hinaus sehr stark vom Modulationsschema abhängig. Bei einem 64-QAM Signal führt eine Trägerungenauigkeit schneller zu einer Fehlentscheidung als bei einer BPSK-Zuordnung.

5.6 Kanalschätzung und Entzerrung

Die Kanalschätzung basiert auf dem Prinzip, dass mehrere aufeinander folgende Symbole durch einen zeitlich nur langsam veränderlichen Kanal sehr ähnlich verzerrt werden. Unter dieser Annahme ist es möglich, den Kanaleinfluss auf bekannte Symbole zu Beginn eines Datenblocks zu messen und alle nachfolgenden Symbole entsprechend zu korrigieren. Diese Entzerrung nach Betrag und Phase wird für alle Subträger einzeln ausgeführt. Ändert sich der Kanal jedoch gegenüber der letzten Kanalschätzung, arbeitet dieser Mechanismus nicht mehr korrekt.

Es ist daher wichtig, den Wiederholungszyklus der Kanalschätzung an die Zeitvarianz des Kanals anzupassen oder zusätzlichen Aufwand zu spendieren, um eine Nachführung der Entzerrung zu ermöglichen. Im HiperLAN/2-Standard werden hierzu die in jedem Symbol vorhandenen Pilot-Subträger ausgewertet. Sie werden neben der Entzerrung mit der Kanalschätzung genutzt, um die exakte Phasenlange des gesamten OFDM-Symbols, also den nach der CFO-Korrektur verbleibenden Restfehler, nachzuregeln. Dieses *Phase Tracking* kann alle Subträger nur um den gleichen konstanten Wert korrigieren. Ein Verdrehen der Konstellation innerhalb eines längeren Datenblocks kann damit wirksam verhindert werden.

5.6.1 Präambelbasierte Entzerrung

Die Kanalschätzung ermittelt die Kanalübertragungsfunktion durch den Vergleich einer empfangenen bekannten Datenfolge mit den Originaldaten, die im Empfänger vorliegen müssen. Um alle Subträger entzerren zu können, muss die Übertragungsfunktion spektral und zeitlich dicht genug abgetastet werden. In dem hier referenzierten System wird zu diesem Zweck mindestens ein vollständiges OFDM-Symbol für diesen Abgleich übertragen. Da in dieser Zeit keine Nutzdaten übertragen werden können, erfordert die Kanalschätzung einen Overhead beim Datentransfer. Um diesen Overhead so klein wie möglich zu halten, können hier jedoch als Referenzsymbole die Daten aus derselben Präambel verwendet werden, die für die Synchronisation benötigt wird.

Die Präambel aus Abb. 5.28 enthält zwei vollständige OFDM-Symbole. Damit ist es möglich, zwei Kanalschätzungen pro Datenblock zu ermitteln. In der hier vorgeschlagenen Implementierung werden beide Schätzungen gemittelt, um den Einfluss der schnell veränderlichen und zufälligen Rauschanteile auf die Entzerrung etwas zu dämpfen. Diese Entzerrung kann nur die langsamen Veränderungen der Kanalimpulsantwort korrigieren, weil sie pro Datenblock (*Burst*) nur einmal aktualisiert wird. Dennoch wird der reale Funkkanal immer auch zeitvariant sein. Die Anteile des zeitvarianten Kanaleinflusses müssen jedoch innerhalb des Datenblockintervalls ausreichend klein sein.

Eine direkte Schätzung mit nur einem Präambelsymbol ist aber ebenfalls möglich. Ein Vergleich der Schätzungen basierend auf der ersten, der zweiten oder der gemittelten Präambel zeigt geringe

Unterschiede (bei gemittelter Schätzung 3 dB weniger Rauschen und 1,2 dB Gewinn bei der BER [TOE07]). In sehr ressourcenkritischen Implementierungen kann die Mittelung, die die Zwischenspeicherung eines OFDM-Symbols erfordert, eingespart werden. Die Addition der beiden Werte und Bitshift um eins nach rechts (Division durch zwei) sind jedoch vernachlässigbar klein.

Die mathematische Operation zur Korrektur der Empfangsdaten (im Frequenzbereich) geschieht in folgenden Schritten: Die empfangene Präambel ist die durch die Kanalimpulsantwort veränderte Originalpräambel $\widetilde{P}=H\,P$. Diese Gleichung kann nach H umgestellt werden und ergibt die Kanalschätzung $\hat{H}$ (hier gemittelt) als

$$\widehat{H}=\frac{1}{2}\left(\frac{\widetilde{P_1}}{P_1}+\frac{\widetilde{P_2}}{P_2}\right) \qquad (5.17)\ .$$

Diese Gleichung wird in einem Multiträgersystem subträgerweise umgesetzt, so dass pro genutzter Trägerfrequenz ein Schätzwert ermittelt wird. Die Entzerrung erfolgt nach der Vorschrift

$$\widetilde{R}=\frac{R}{\widehat{H}}=S+\frac{N}{\widehat{H}} \qquad (5.18)\ .$$

Die Implementierung dieser Rechnung in Hardware erfordert folgende Komponenten:

- Speicher für die Original-Präambel,
- komplexe Dividierer und Multiplizierer zur Entzerrung,
- Speicher für die geschätzten Koeffizienten.

In Hardware werden noch einige Zusammenhänge berücksichtigt, welche die Umsetzung in Logikschaltungen vereinfachen. Zunächst wird nicht $\widehat{H}$ sondern direkt $1/\widehat{H}$ berechnet. Der Aufwand dafür bleibt zunächst identisch. Die Korrektur der Daten kann auf diese Weise jedoch als Multiplikation ausgeführt werden, was in einem FPGA weniger kritisch ist. Zwei Divisionen werden so zu einer Division und einer Multiplikation umgestellt.

Sehr aufwendig sind die beschriebenen Rechnungen auch dadurch, dass alle Zahlen komplexwertig sind. Eine Multiplikation von zwei komplexen Zahlen in kartesischer Darstellung erfordert vier reelle Multiplikationen und zwei Additionen. Für eine Division müssen zusätzlich noch zwei Werte quadriert und addiert sowie zwei reelle Divisionen ausgeführt werden.

$$Z_1 Z_2 = (a+ib)(c+id) = (ac-cd)+i(ad+bc) = |Z_1||Z_2|e^{\phi_1+\phi_2} \quad \text{(5.19) komplexe Multiplikation}$$

$$\frac{Z_1}{Z_2} = \frac{a+ib}{c+id} = \frac{ac+bd}{c^2+d^2} + i\frac{bc-ad}{c^2+d^2} = \frac{|Z_1|}{|Z_2|}e^{\phi_1-\phi_2} \quad \text{(5.20) komplexe Division}$$

Dieser Aufwand ist reduzierbar, wenn die Werte vorab in Polarkoordinaten umgerechnet werden. Dadurch reduziert sich die komplexe Multiplikation (Division) auf eine einfache Multiplikation (Division) der Beträge und eine Addition (Subtraktion) der Phase. Hinzu kommen jedoch der Aufwand für die Umrechnung in Polarkoordinaten und anschließend die Transformation zurück in die kartesische Darstellung, die als CORDIC-Algorithmus implementiert werden muss.

Tatsächlich werden aber nur die Kanalschätzwerte in Polarkoordinaten umgerechnet. Die Nutzdaten werden dann direkt in einer weiteren CORDIC-Stufe gedreht und anschließend wird der Betrag durch die Multiplikation von Real- und Imaginärteil korrigiert. So wird die Division komplexer Zahlen in kartesischen Koordinaten vermieden und das Zurücktransformieren in Real- und Imaginärteil entfällt.

Abb. 5.36 Zeigt ein Blockschaltbild der vollständigen Entzerrerschaltung inklusive der Phasen-Nachführung mit Hilfe der Piloten.

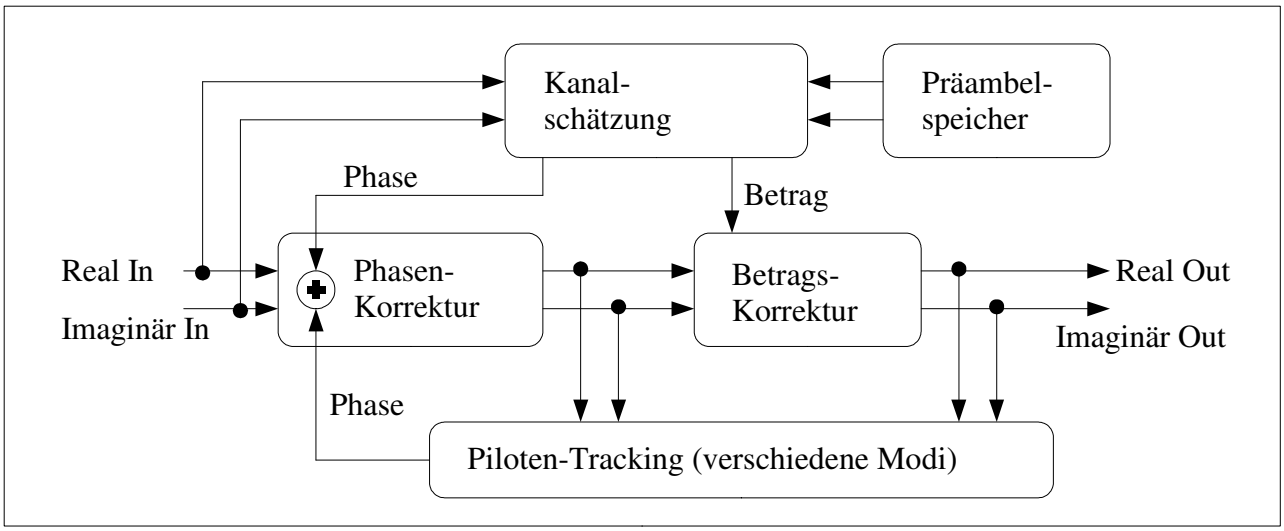


Abb. 5.36 Blockschaltbild der Kanalschätzung und Entzerrung

Die grafischen Darstellungen in und 5.38 zeigen, wie die Entzerrung eine Unterscheidung der Konstellationspunkte vorbereitet. Je nach Kanalsituation, wie bei dieser Simulation mit angenommener Mehrwegesituation (WSSUS-Kanal), wird die Demodulation durch die Entzerrung überhaupt erst möglich.

Im Allgemeinen kann das Rauschen die Kanalschätzung noch durch eine zusätzliche Filterung in Frequenzrichtung verbessert werden. Dabei wird die Annahme genutzt, dass die Impulsantwort endlich und somit die Übertragungsfunktion in Frequenzrichtung nicht beliebig schnell schwanken kann. Hierzu kann idealerweise ein Wiener-Filter genutzt werden, wie es in [KAI98] beschrieben wird. Auf eine Implementierung dieser Rauschreduktion ist im Rahmen dieser Arbeit wegen der zu erwartenden hohen Hardwarekomplexität verzichtet worden.

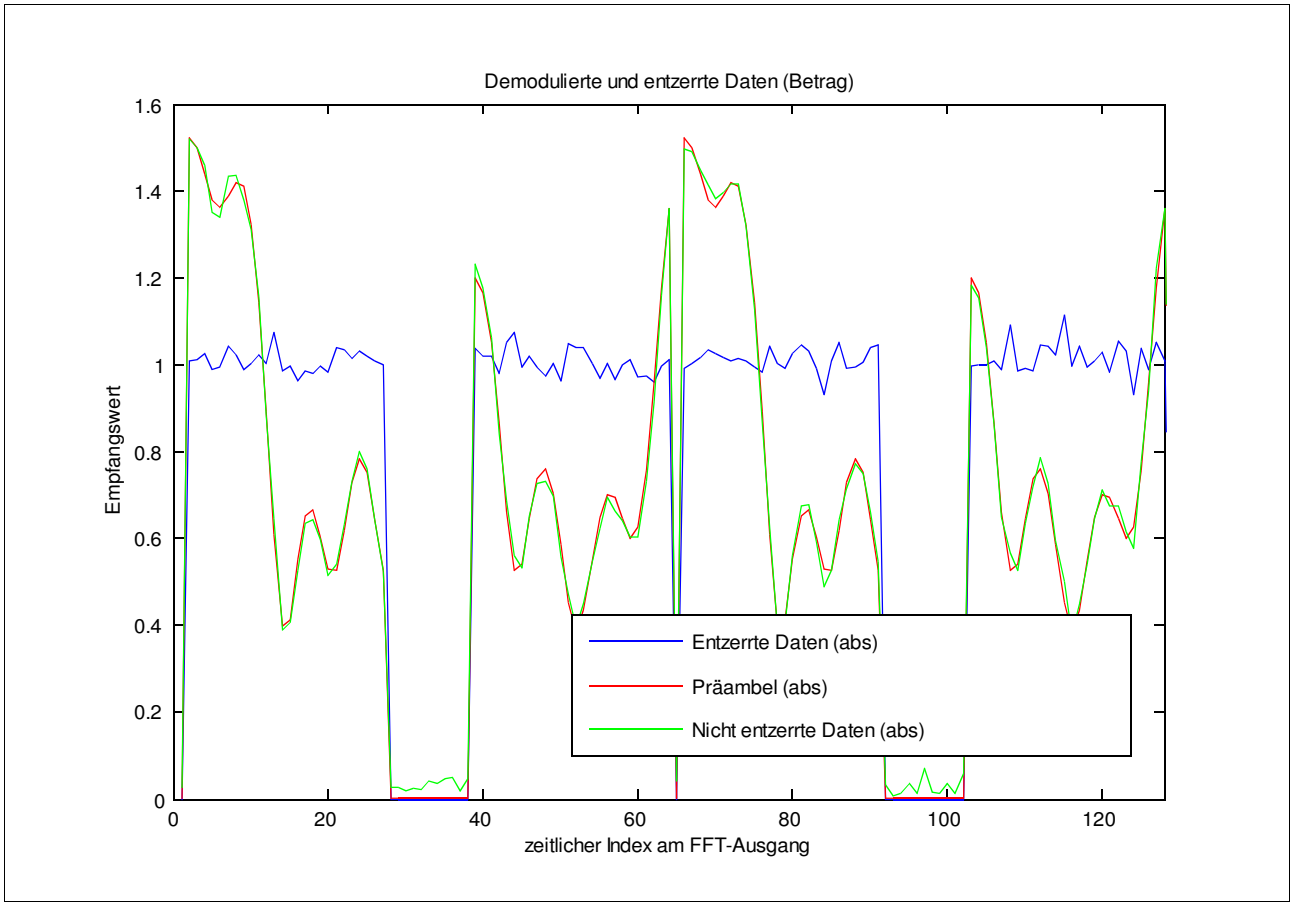


Abb. 5.37 Die OFDM-Empfangsdaten nach der FFT: Beträge vor und nach der Entzerrung sowie die Präambeldaten

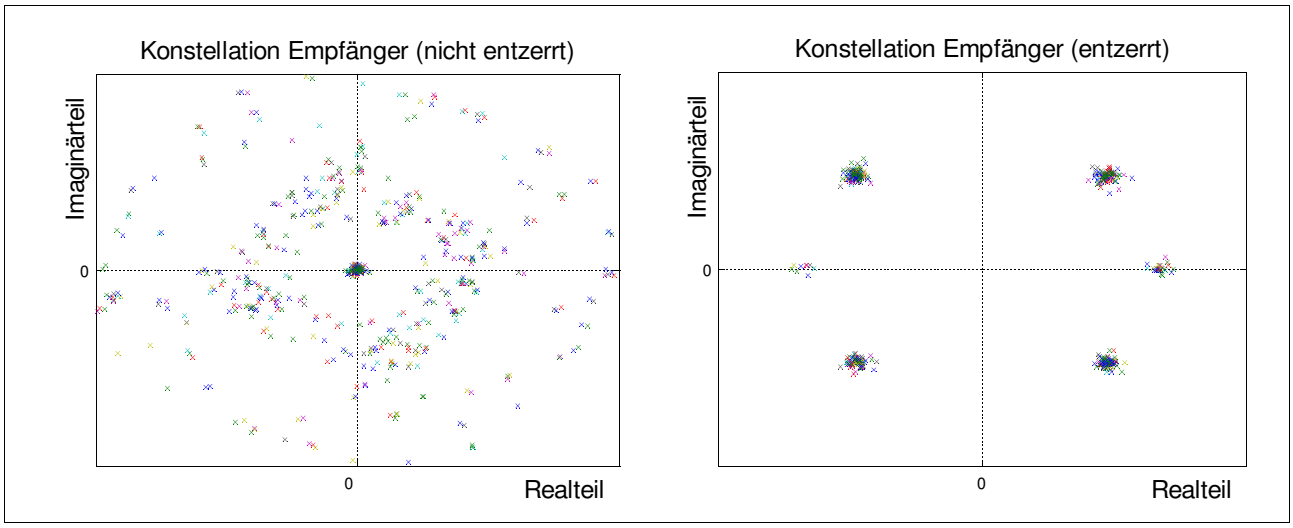


Abb. 5.38 Konstellationsdiagramm vor und nach der Entzerrung (WSSUS-Kanal)

5.6.2 Pilotenbasierte Phasennachführung (Phase Tracking)

Die bislang betrachtete Korrektur der Phasenlage der Konstellationspunkte wie in Abb. 5.38 erfolgt einmal pro Datenburst. Abhängig von dessen Länge, ist die Aktualisierung dieser Entzerrung nicht schnell genug, um ein Abdriften der Modulationssymbole effizient zu verhindern. Eine leicht ungenaue Phasenkorrektur oder ein Offset des Abtasttakts (*Sampling Frequency Offset*, SFO) führen zu dem Wunsch, eine Phasennachführung für jedes einzelne OFDM-Symbol zu installieren. Dieses Phase Tracking wird mithilfe der Piloten realisiert, die laut HiperLAN/2- bzw. IEEE802.11a-Standard in jedem OFMD-Symbol enthalten sind. Zum Nachführen des Winkels kann die Phasendrehung der bekannten Pilotsymbole vermessen werden. Da die vier rein reellen (BPSK-modulierten) Piloten gemeinsam der Ermittlung eines Phasenwertes dienen, können alle Werte zur Bestimmung des Winkels aufaddiert (bzw. bei negativen Piloten entsprechend subtrahiert) werden. Anschließend wird diese Summe an den CORDIC-Block der Kanalschätzung gegeben. Dieses CORDIC-Element kann hier wiederverwendet werden, weil es für den ersten Teil der Kanalschätzung nur während der Präambelverarbeitung benötigt wird. Die Piloten sind jedoch in den Nutzdatensymbolen enthalten, die im Anschluss an die Präambel übertragen werden und sich in der Verarbeitung zeitlich nicht überschneiden.

Die Auswertung der Piloten kann in verschiedenen Modi ausgeführt werden. Es besteht die Möglichkeit, leistungsstarke Pilotträger stärker zu gewichten als leistungsschwache. Abhängig vom Modus ist daher, ob die Piloten vor oder nach der Betragsentzerrung (vgl. Abb. 5.36) ausgewertet werden.

Piloten Modi:

00: Pilot Tracking deaktiviert,

01: Pilotenauswertung nach der Phasenkorrektur (*Equal Gain Combining*),

02: Pilotenauswertung nach Phasen- und Betragskorrektur,

03: Pilotenauswertung nach der Phasenkorrektur gewichtet mit der Trägerleistung aus der Kanalschätzung (*Maximum Ratio Combining*).

In dem hier referenzierten OFDM-Demonstrator wird das Maximum Ratio Combining als Standardeinstellung verwendet. Auf die anderen Modi kann auf Wunsch umgeschaltet werden. Für den Schaltungsaufwand im FPGA spielen diese Wahlmöglichkeiten so gut wie keine Rolle.

Wichtig bei diesem Verarbeitungsschritt ist jedoch, dass die Zahl der Piloten (hier vier von 64 Subträgern) nicht ausreicht, um eine echte Schätzung der Kanalübertragungsfunktion durchzuführen. Analog zur Bandbegrenzung bei der A/D-Wandlung gilt auch für eine Abtastung im Frequenzbereich die Einschränkung, dass nur zeitbegrenzte (langsam veränderliche) Signale eindeutig rekonstruiert werden können. [FLI91] bezeichnet dies als Abtasttheorem für zeitbegrenzte Signale. Bei Annahme einer Impulsantwort der Länge des Guard-Intervalls erfüllen die Piloten das Abtasttheorem im Frequenzbereich. Nach Nyquist müsste bei 25% Guard-Intervall jeder vierte Subträger ein Pilot sein, um den Kanal korrekt schätzen zu können.

Generell unempfindlicher gegen solche Phasenfehler ist eine differentielle Modulation, welche aber in den Standards HiperLAN/2 und IEEE802.11a nicht vorgesehen ist. Untersuchungen zu OFDM-Systemen, die auf dieser Technik basieren, finden sich in [MAY99].

5.6.3 Implementierungsdaten der Kanalschätzung und Entzerrung

Die Implementierungsdaten sind hier für die gesamte Kanalschätzungs- und Entzerrerschaltung angegeben, weil einige Komponenten für beide Mechanismen genutzt werden. In diesem Fall ist es schwer, den Schaltungsaufwand entsprechend dem einem oder anderen Modul zuzuordnen. Der zweite Dividierer, der in Tabelle 5.13 berücksichtigt ist, dient der Berechnung der Channel State Information (CSI), welche in Kapitel 5.3 eingeführt wurde. Dieser Wert bildet prinzipiell den Kehrwert der Empfangsleistung einzelner Subträger und wird in der Demodulation als Zuverlässigkeitsinformation verarbeitet.

	pro CORDIC (zwei benötigt)	Dividierer (zwei benötigt)	Modul gesamt
Slices	503	1065	4015 (11,9 %)*
Flipflops	135	1858	4855 (7,2 %)*
Look-Up-Table	937	642	4868 (7,2 %)*
Block-RAM	-	-	4 (2,8 %)*
Multiplizierer	-	-	9 (6,3 %)*
Taktrate	32,1 MHz	197 MHz	31,6 MHz

Tabelle 5.13 Implementierungsdaten der Kanalschätzung & Entzerrung (*In Prozent bezogen aufs FPGA)

Die maximale Taktrate dieses Schaltungsblocks ist durch die Parametrisierung des CORDIC-Blocks bestimmt, der so klein wie möglich gehalten wird (vgl. Anhang D). Insgesamt konnte die Schaltung durch die Reihenfolge der Berechnung und durch die Doppelnutzung der CORDIC-Schaltung verkleinert werden. Eine hohe Rechengenauigkeit kann an dieser Stelle der Verarbeitungskette durchaus von Nutzen sein. Wenn stark gedämpfte Empfangswerte, die im günstigsten Fall wenig verrauscht sind, durch den Entzerrer wieder entsprechend groß skaliert werden, bleibt ein fein quantisierter Eingangswert sehr exakt. Daher ist in dem hier referenzierten OFDM-Demonstrator dieses Modul mit einer Verarbeitungswortbreite von 16 Bit implementiert. Etwas zusätzlicher Schaltungsaufwand wird durch die Flexibilität erzeugt, dass alle in Kapitel 5.6.2 beschriebenen Modi unterstützt werden.

Die bei dem hier referenzierten OFDM-Demonstrator eingesetzten Wandler wurden schon im Kapitel 4.1 ausführlich beschrieben. Ihre Auflösung mit 14- (DAC) bzw. 12-Bit-Wortbreite (ADC) hat sich als sehr leistungsfähig erwiesen. Die hohen zulässigen Wandlertakte dieser Bauteile sind zudem geeignet, ein Signal direkt auf einer Zwischenfrequenz auszugeben, was im folgenden Kapitel beschrieben ist.

5.7 Digitale ZF-Modulation

Das digitale Mischen auf eine Zwischenfrequenz ist in dieser OFDM-Verarbeitung optional. Es bietet den Vorteil, dass alle Störungen durch ungleichmäßiges Wandeln oder Mischen der In- bzw. Quadraturphase von vornherein vermieden werden können und als externe Beschaltung ein einziger D/A-Wandler (bzw. A/D-Wandler) ausreicht.

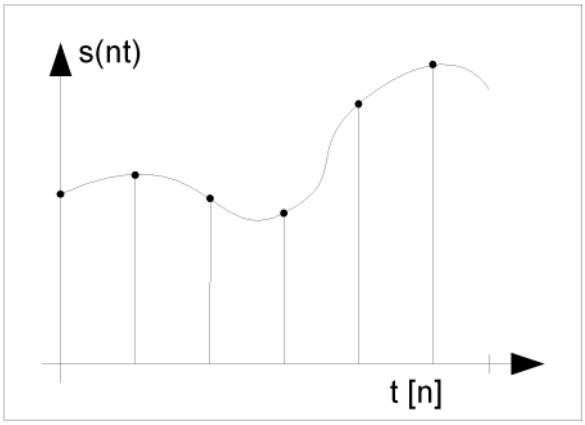
Der Mischprozess ist jedoch schaltungstechnisch relativ aufwendig, weil hier jeweils eine digitale Filterung notwendig wird. Das Berechnen solcher FIR-Filter erfordert eine hohe Rechengenauigkeit, damit es nicht zu starkem Quantisierungsrauschen bei der Wortbreitenan-

passung kommt. Außerdem „verbreitert“ die Faltung mit den Filterkoeffizienten das OFDM-Symbol, sodass das Risiko von Intersymbolinterferenzen steigt (was jedoch bei analogen Filtern ebenso der Fall ist). Hierbei sollte eine Verbreiterung über das Guard-Intervall hinaus vermieden werden.

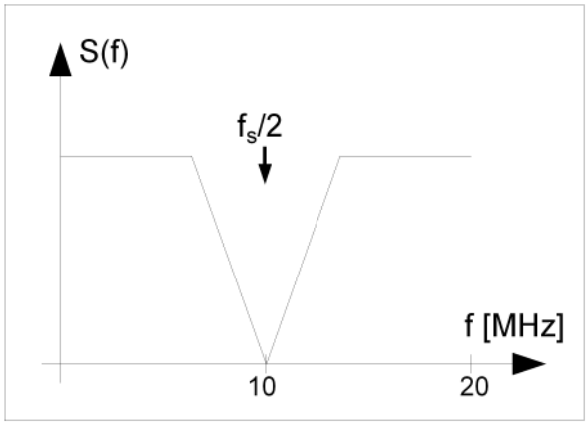
In Abb. 5.39 sind die einzelnen Schritte des digitalen Mischens skizziert. Zunächst wird das Signal durch das Einfügen von Nullen auf einen höheren Abtasttakt umgesetzt. In dem gezeigten Beispiel handelt es sich um Vierfach-*Upsampling*, wobei jeweils zwischen zwei Eingangswerten drei Nullen ergänzt werden. Der Abtasttakt muss dadurch viermal schneller sein. Dieser Vorgang erzeugt im Frequenzbereich zusätzliche Wiederholspektren [FLI93], die durch ein anschließendes FIR-Tiefpassfilter (Interpolationsfilter) entfernt werden müssen. Schließlich wird mit einer (stark vereinfachten) komplexen Multiplikation das Nutzspektrum in den gewünschten Bereich verschoben. Ausgegeben an einen nachfolgenden D/A-Wandler wird nur der Realteil des errechneten Signals, sodass man ein zur Frequenz Null symmetrisches Spektrum bekommt.

Im Rahmen dieser Arbeit wird das Basisbandsignal auf eine Mittenfrequenz von 20 MHz gemischt. Dabei werden FIR-Tiefpässe mit 16-Bit-Rechengenauigkeit und 32 Koeffizienten eingesetzt. Die 32 Koeffizienten werden bei dem hochgetasteten Signal angewendet, sodass die Faltung das OFDM-Symbol um 32*1/80MHz = 0,4µs verbreitert. Dies ist bezogen auf das Guard-Intervall von 0,8µs gerade noch akzeptabel, weil jeweils ein solches Filter im Sender und Empfänger genutzt wird.

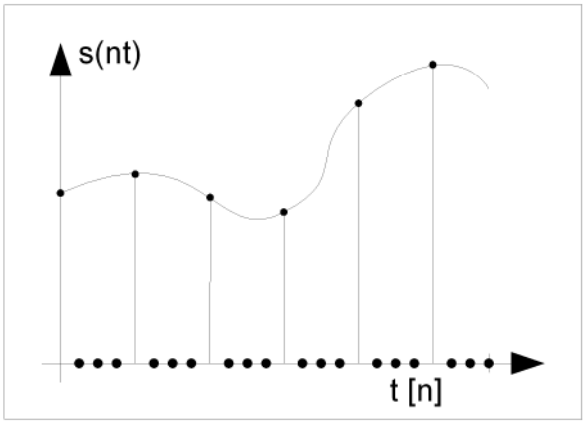
Wichtig bei der Implementierung der Abtastratenerhöhung sind einige Vereinfachungen, die den Schaltungsaufwand maßgeblich reduzieren. Das Tiefpassfilter wird als Polyphasenfilter implementiert [ZOE96]. Dies bedeutet unter anderem, dass die Faltung nur mit den Zahlenwerten ungleich Null gerechnet wird. Da die zuvor eingefügten Nullen keinen Einfluss auf das Ergebnis haben, müssen pro Ausgabewert nur acht Multiplikationen berechnet werden.



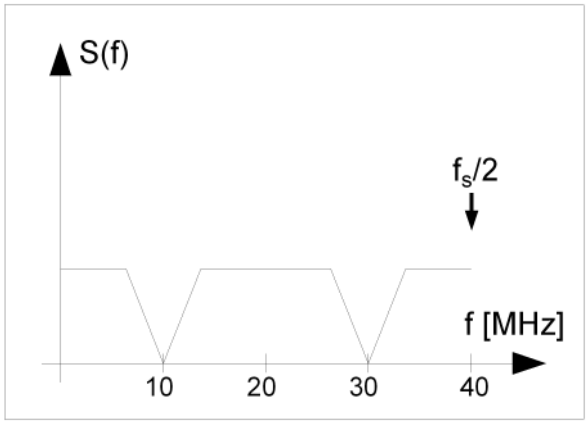
I. Signal mit einer Abtastfrequenz 20 MHz



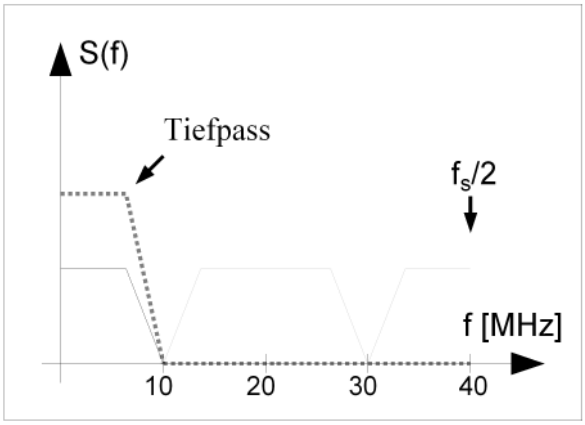
II. Das zu I. gehörige Spektrum



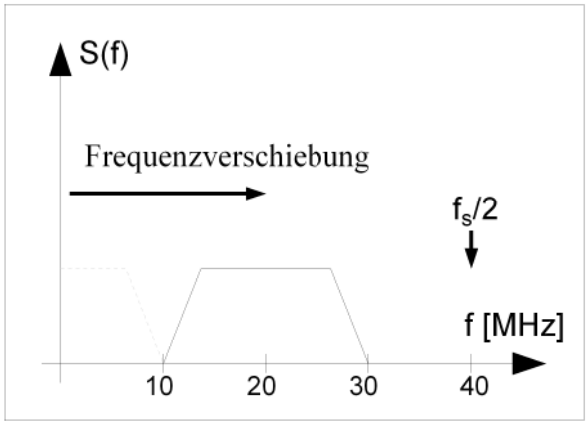
III. Erhöhung der Abtastfrequenz auf 80 MHz durch Einfügen von Nullen



IV. Das zu III. gehörige Spektrum, jedoch nur bis zur Nyquistgrenze $f_s/2$ dargestellt



V. Ein Tiefpassfilter entfernt unerwünschte Wiederholspektren



VI. Modulation auf 20 MHz Mittenfrequenz - eine Verschiebung im Frequenzbereich

Abb. 5.39 Die Operation der digitalen ZF-Modulation auf einen 20-MHz-Träger mit Vierfach-Upsampling

Die zweite Vereinfachung ergibt sich bei der Verschiebung um 20 MHz, also auf ein Viertel des neuen Abtasttaktes. Der komplexe Faktor ergibt durch das Einsetzen von der Verschiebungsfrequenz $f=f_s/4$ und der Abtastzeitpunkte $t = n/f_s$

$$e^{(j*2\pi f\ t)} = e^{(j*2\pi \frac{f_s}{4}\frac{n}{f_s})} = \left\{ \begin{array}{l} 1\ für\ n=0,4,8\ \ ... \\ j\ für\ n=1,5,9\ \ ... \\ -1\ für\ n=2,6,10 ... \\ -j\ für\ n=3,7,11 ... \end{array} \right\} \qquad (5.21)$$

eine sich wiederholende Folge von 1, j, -1 und -j. Die Multiplikation mit diesen Werten kann dann durch ein Invertieren und/oder ein Vertauschen von Real- und Imaginärteil ersetzt werden und ist somit sehr aufwandsgünstig.

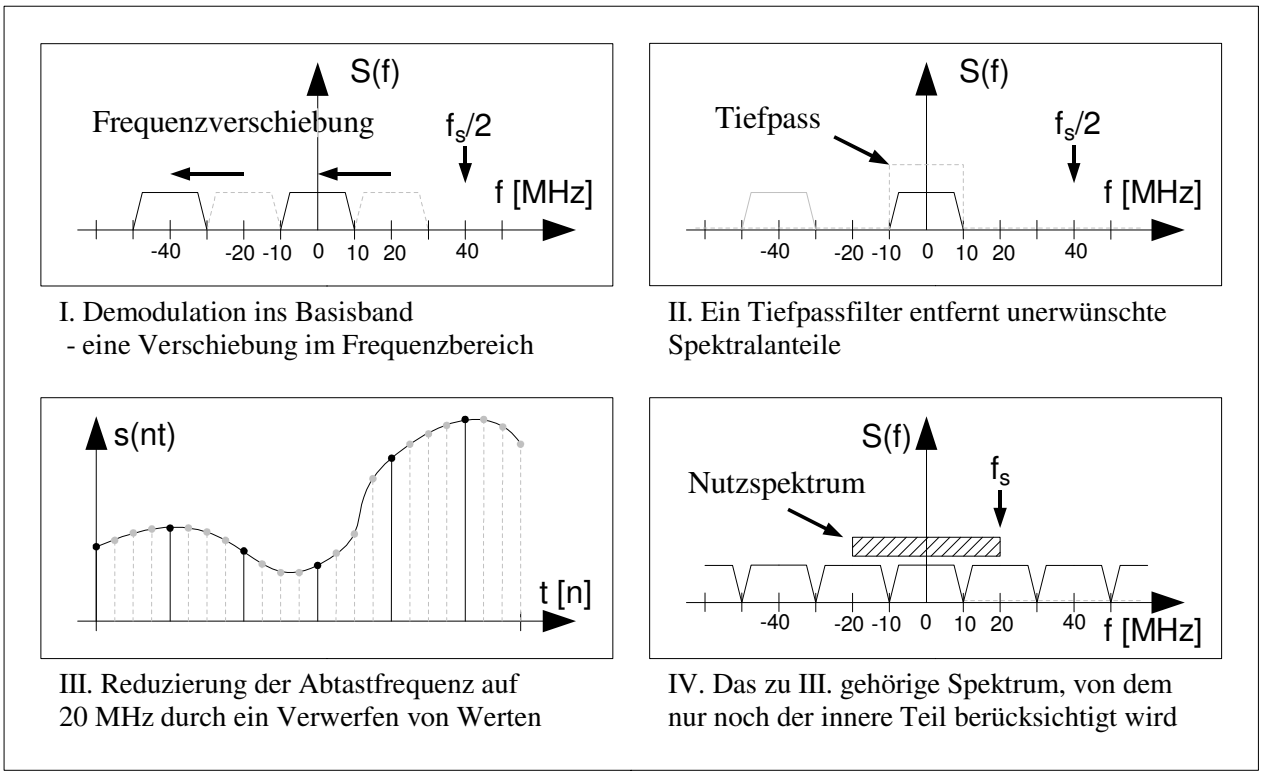


Abb. 5.40 Die Operation der digitalen ZF-Demodulation ins Basisband mit Vierfach-Downsampling

Um die Berechnung mit der hohen Auflösung schnell genug ausführen zu können (der Ausgangstakt beträgt ja jetzt 80 MHz), werden vier Filterstufen parallel berechnet und moduliert, was zunächst noch vollständig mit dem Eingangstakt von 20 MHz passiert. Bei dieser Polyphasenfilterung benutzt jedes einzelne Filter nur eine Teilmenge der Koeffizienten. Erst direkt am Ausgang des Schaltnetzwerkes werden die vier Datenströme dann gemultiplext und so mit der

erhöhten Abtastrate an den Wandler gegeben. In diesem Schaltungsteil muss also, um die Geschwindigkeitsanforderungen erfüllen zu können, parallel verarbeitet werden, was zu einer deutlichen Steigerung der Schaltungskomplexität führt.

Der Mischer im Empfänger, der ein ZF-Signal wieder in das Basisband mischt, arbeitet nach den in Abb. 5.40 gezeigten Schritten. Das Empfangssignal liegt analog zum Senderausgang bei einer 20 MHz Zwischenfrequenz. Dieses wird zunächst erneut durch eine Multiplikation mit 1,-i,-1 und 1 (wieder jeweils durch Tauschen bzw. Invertieren von Real- und Imaginärteil) zurück in das Basisband geschoben.

Anschließend muss das Signal für eine Abtastratenreduzierung hinreichend bandbegrenzt werden, damit es nicht zu Überlappungen im Spektrum (engl. *Aliasing*) kommen kann. Der zugelassene Bereich liegt innerhalb der Nyquistfrequenz des reduzierten Taktes. Im oben gezeigten Beispiel ist das der Abschnitt ±10MHz. Die übrigen Frequenzen (außerhalb des Bandes ±10MHz) müssen vor dem *Downsampling* durch ein Tiefpassfilter ausreichend gedämpft werden [FLI93]. Schließlich werden drei von vier Werten verworfen, sodass die neue Abtastrate einem Viertel der Ausgangsfrequenz entspricht. Zu bedenken ist, dass dieses Spektrum nicht symmetrisch ist und somit Real- und Imaginärteil weiterverarbeitet werden müssen.

Eine Vereinfachung der Rechnung ergibt sich daraus, dass nur jeder vierte Wert nach dem Tiefpassfilter weiterverwertet wird. Die anderen drei Werte, die ohnehin verworfen werden, müssen nicht berechnet werden. Auch hier wird wieder ein Polyphasenentwurf für die Filterung verwendet, sodass wie bei der Aufwärtstastung mit dem reduzierten Takt gearbeitet werden kann.

Auch wenn die digitale ZF-Modulation wie oben beschrieben mit geringst möglicher Komplexität implementiert wird, belegt sie im Verhältnis zu einem OFDM-Gesamtsystem einen recht großen Teil der FPGA-Ressourcen (vgl. Kap. 6.4). Die Rechengenauigkeit sollte mindestens so hoch sein wie die Auflösung an den D/A- bzw. A/D-Wandlern. Aufgrund der Wortbreitenanpassung empfiehlt sich sogar, je nach Ordnung der digitalen Filter, eine interne Erhöhung der Wortbreite (vgl. Kap.). Ob eine digitale ZF-Modulation im FPGA durchgeführt wird, hängt unter Umständen auch von dem verfügbaren analogen Frontend ab. Häufig werden diese Module in einem Chip integriert angeboten, wie z.B. der Max2728 [MAX03], der einen IQ-Eingang hat und somit ohne externen ZF-Mischer auskommt.

	ZF-Modulation (Sender)	ZF-Demodulation (Empfänger)
Slices	1359	1907
Flipflops	1227	1554
LUT	2251	2534
Multiplizierer	64	64
Taktrate[8]	43,1 MHz	43,5 MHz

Tabelle 5.14 Implementierungsdaten der Kanalschätzung und Entzerrung

Die in Tabelle 5.14 aufgeführten Implementierungsdaten beziehen sich auf eine digitale Modulation bei einer Zwischenfrequenz von 20 MHz mit vierfacher Abtastratenerhöhung bzw. -reduktion. Die Tiefpassfilter haben 32 Koeffizienten und die Wortbreite für alle Rechnungen beträgt 16 Bit.

5.8 Auflösung der D/A- und A/D-Wandler

Die Wandler bestimmen die Genauigkeit bei der Umsetzung der digitalen in analoge Signale und umgekehrt. Sie beeinflussen maßgeblich die sinnvolle Rechengenauigkeit in der digitalen Verarbeitungskette. Eine Auflösung von 16 Bit in der FFT bringt keinen Vorteil, wenn der nachgeschaltete D/A-Wandler nur 10 Bit davon auflösen kann. Es ist also wichtig, zunächst Wandler auszuwählen und dann die Wortbreiten der OFDM-Kette daran anzupassen.

Ideale D/A- und A/D-Wandler werden durch zwei Kenngrößen charakterisiert: Die Quantisierungsgenauigkeit und der maximale Takt zur Wandlung. Generell gilt die Regel, je höher die Genauigkeit, desto geringer der erlaubte Takt.

Der minimal zulässige Sampletakt F_S des Wandlers wird bei einem OFDM-System durch die genutzte Bandbreite B bzw. die Grenzfrequenz F_g bestimmt: $F_S \geq B = 2F_g$. Soll auf eine digitale Zwischenfrequenz gemischt werden, muss der Oversamplingfaktor n_f berücksichtigt werden $F_S \geq B \cdot n_f = 2F_g \cdot n_f$.

Von den Wandlern, die diese Geschwindigkeitsanforderungen erfüllen, versprechen die mit hoher Wortbreite die geringsten Quantisierungsfehler, also die beste Übertragungsqualität. Es ist jedoch nicht in jedem Fall notwendig, sehr genaue Wandler zu verwenden. Ab einer gewissen Stufe ist das Quantisierungsrauschen gegenüber den Kanaleinflüssen zu vernachlässigen.

8 Dies ist der interne Verarbeitungstakt, mit dem jeweils vier Stufen parallel arbeiten. Der gemultiplexte Ausgabetakt ist viermal höher.

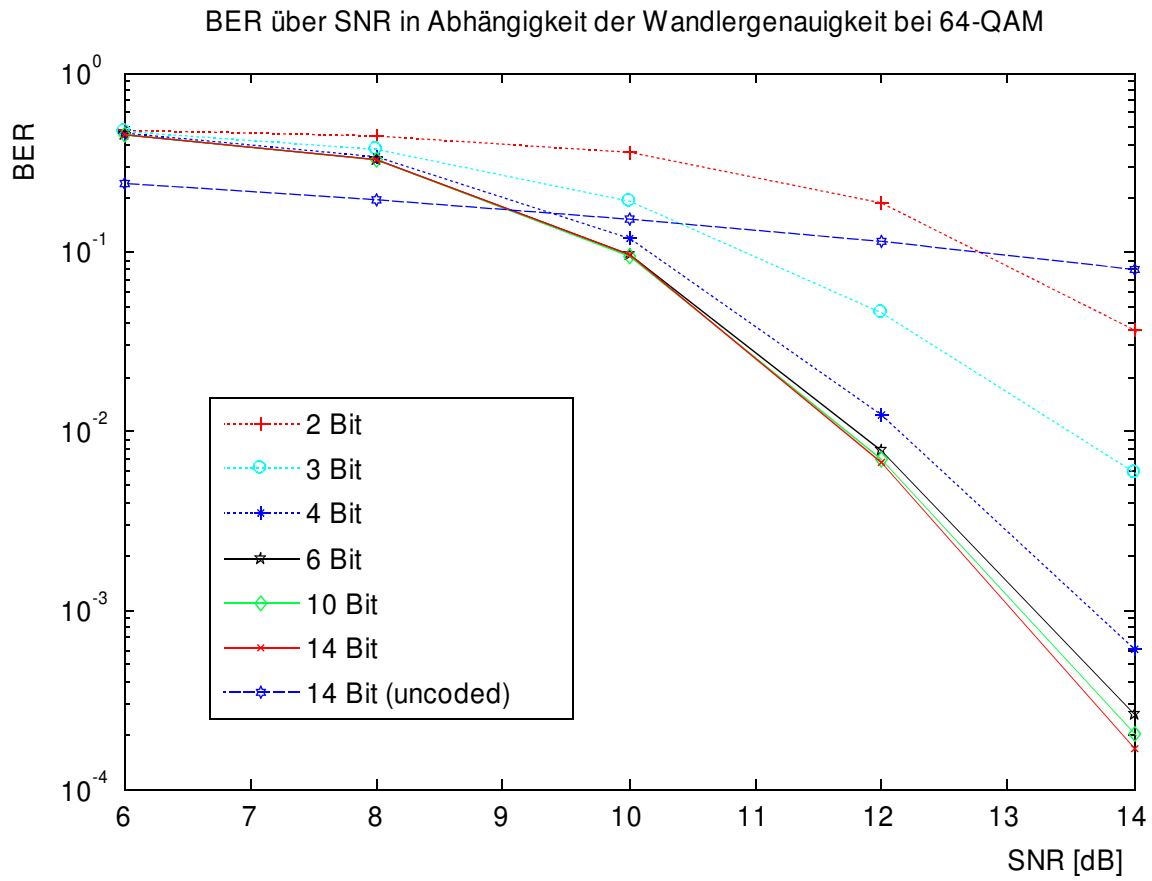


Abb. 5.41 BER-Kurven für unterschiedliche Wandlergenauigkeiten. Der D/A- und A/D-Wandler arbeiten bei dieser Messung mit der gleichen Wortbreite.

Die Abb. 5.41 zeigt den Einfluss von Quantisierungseffekten der Wandler am Beispiel eines 64-QAM modulierten Signals. Die BER über dem SNR ist das Maß für die Übertragungsqualität.

Die Genauigkeitsanforderungen bei der Umsetzung sind relativ stark daran gekoppelt, wie wenig das Signal durch den Übertragungskanal verfälscht wird. Erst bei exzellenten Kanälen kann eine hohe Modulationswertigkeit genutzt werden, die wiederum eine genaue digitale Verarbeitung erfordert. Wichtig ist aber auch, den genutzten Aussteuerungsbereich der Wandler zu betrachten. Wenn in einem Empfänger z.B. auf eine exakte AGC zur Pegelanpassung verzichtet wird, kann es sein, dass der Wandler nur in einem kleinen Teil seines Eingangsbereichs angesteuert wird. In diesem Fall kann das Signal digital angepasst (verstärkt) werden. Hier kann der Quantisierungsfehler leicht verhältnismäßig groß werden, denn es darf dann nur die Auflösung des Wandlers betrachtet werden, die in dem Bereich des eingehenden (minimalen) Signalpegels zur Verfügung steht.

Extrem schnelle Wandler (bei mehr als 100 MHz) bezüglich des Aufbaus der Hardware sehr anspruchsvoll. Es muss sichergestellt werden, dass die angeschlossene Peripherie (z.B. ein FPGA)

die Daten auch mit dieser Geschwindigkeit ausgeben bzw. einlesen kann. Beim Platinenlayout ist zu bedenken, dass Laufzeiteffekte und Signalreflexionen auftreten können. Insbesondere bei Taktleitungen sind einheitliche Längen und saubere Signalformen sehr wichtig, um Störungen wie Clock-Jitter oder IQ-Imbalance zu vermeiden. Darüber hinaus ist die Leistungsaufnahme schneller A/D-Wandler ist sehr hoch. Das Versorgungsnetzwerk muss entsprechend dimensioniert und entkoppelt werden, damit es nicht zu Störungen durch instabile Versorgungsspannungen kommt.

Für OFDM-Systeme sind zusätzlich die realen Eigenschaften der Wandler wie z.B. die Linearität wichtig (vgl. Kap. 4.1) wichtig. Aber generell sind die Auflösung und die Taktung systembestimmende Parameter, die für ein Übertragungssystem in der frühen Planungsphase definiert werden sollten und aus denen sich unmittelbar viele Anforderungen der digitalen Signalverarbeitung im FPGA ableiten lassen.

6 Experimentalhardware und Implementierungsdaten des OFDM-Demonstrators

Zur Untersuchung von OFDM-Systemen auf einer FPGA-basierten Hardwareplattform ist im Rahmen dieser Arbeit eine PCI-Platine entwickelt worden. Dieser Experimentalaufbau erlaubt Messungen bei realer Datenfunkübertragung, kann aber auch zur Simulation von OFDM-Systemen genutzt werden.

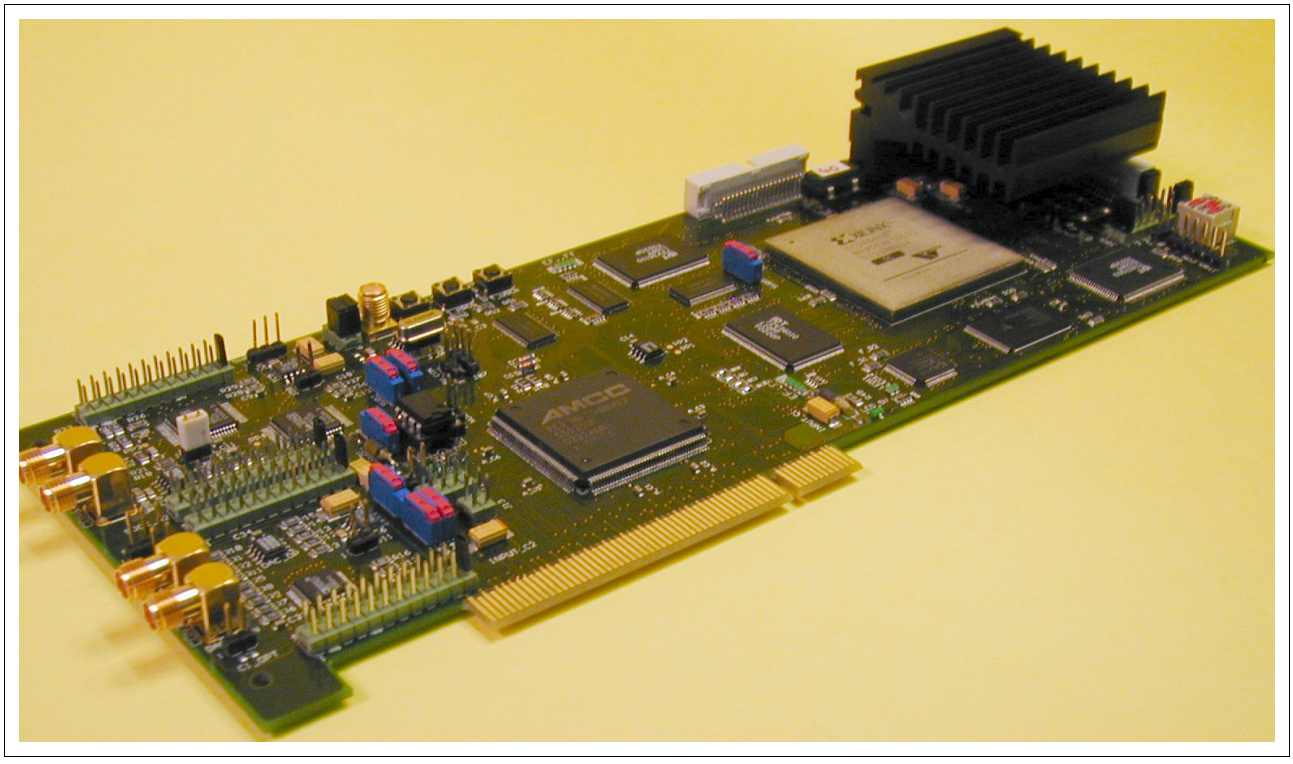


Abb. 6.1 Die FPGA-Experimentalplatine: PCI-Board mit FPGA

Im folgenden Abschnitt werden der Aufbau und die Möglichkeiten dieser Hardware erläutert. Anschließend wird im zweiten Teil gezeigt, dass die PCI-Einsteckkarte Kanalsimulationen, die im PC berechnet werden, sehr stark beschleunigen kann. Auf diese Weise kann die FPGA-Hardware zum Test und zur Optimierung der OFDM-Komponenten genutzt werden. Diese Tests dienen der

Untersuchung und Verifikation der Algorithmen, die in Kapitel 5 beschrieben wurden. Abschließend werden die Daten der Implementierung aller eingesetzten OFDM-Module zusammengefasst dargestellt und hinsichtlich ihrer Komplexität in Bezug auf das Gesamtsystem eingeordnet.

6.1 Die Experimentalhardware

Die in Abb. 6.1 gezeigte Einsteckkarte passt in den in PCs üblichen PCI-Sockel (*Peripheral Component Interconnect*). Eine Ankopplung an den Computer über den PCI-Bus bietet den Vorteil einer Anbindung mit hoher Datenrate, welche in diesem Fall knapp oberhalb von 1 Gbit/s liegt. Außerdem kann die Hardware direkt in PCs eingesetzt werden, ohne eine externe Spannungsversorgung oder ein eigenes Gehäuse zu erfordern.

Die Anbindung an den PCI-Bus erfolgt über einen Interfacechip, den AMCC S5935 [AMC99]. Dieser übernimmt die Ansteuerung der für den PCI-Bus vorgeschriebenen Signale und stellt der Platine eine leistungsfähige Schnittstelle bereit.

Das PC-Betriebssystem ist in der Lage, über einen Treiber auf die Experimentalhardware zuzugreifen. Dabei stehen zwei Modi der Datenübertragung zur Verfügung: Datentransfers über *First-In-First-Out*-Speicher (FIFOs) und *Pass-Thru*-Zugriffe, die unmittelbar auf den Peripheriebus zugreifen.

FIFO-Transfers bieten den Vorteil, dass Schreib- und Lesezugriffe entkoppelt werden. Es ist jederzeit möglich, Daten zum FPGA zu schicken. Diese werden dann im Eingangs-FIFO so lange zwischengespeichert, bis das FPGA die Werte übernimmt. In der anderen Richtung kann das FPGA Werte jederzeit schreiben, ohne dass der PCI-Bus verfügbar sein muss. Blockiert wäre das System erst dann, wenn die FIFOs voll sind (in diesem Beispiel nach 131.072 Wörtern). Die FIFO-Datentransfers können zur Beschleunigung auch als *Bus-Master*-Zugriffe weitgehend CPU-unabhängig durchgeführt werden. Dabei werden ganze Speicherbereiche von speziellen PCI-Buscontrollern automatisch transferiert. Dieses ist zur Übertragung mit hohen Datenraten, wie es bei dem implementierten OFDM-Modem vorgesehen ist, der wichtigste Modus.

Pass-Thru-Zugriffe können jederzeit unabhängig von den anderen Transfers ausgeführt werden. Konfigurations- und Steuerinformationen können auf diese Weise effizient direkt vom PC an das FPGA übergeben werden. Auch Kontroll- und Monitorfunktionen innerhalb des OFDM-Modems können so unmittelbar ausgelesen werden.

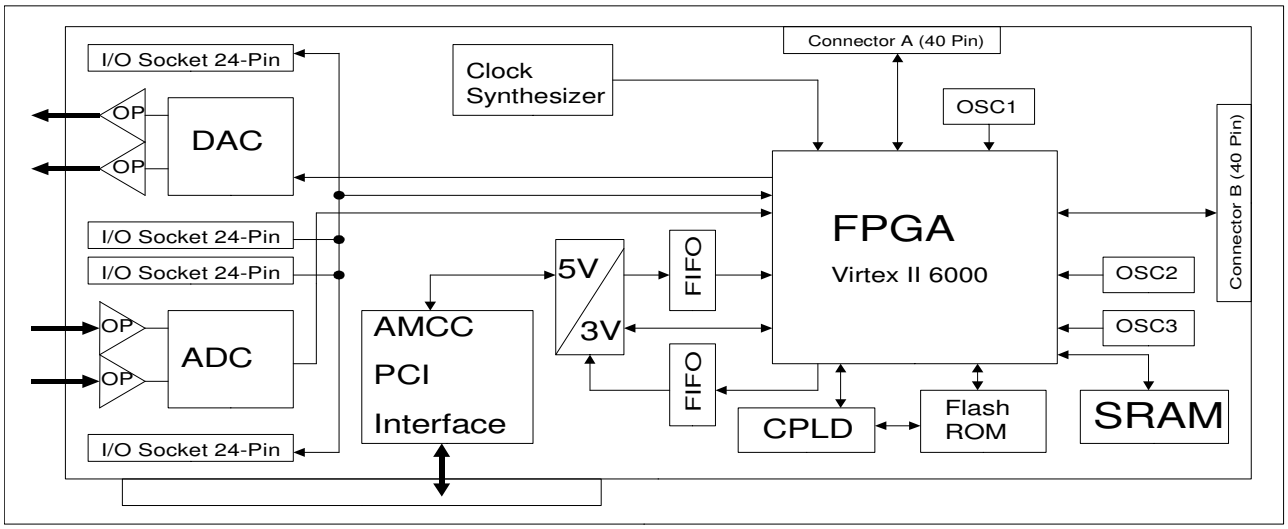


Abb. 6.2 Blockschaltbild der PCI-FPGA-Platine

Neben der Anbindung an den PCI-Bus und dem komplexen FPGA verfügt die Steckkarte noch über zusätzliche Komponenten, die den Einsatz als Modem oder universelle Entwicklungsplattform unterstützen [TOE04a]. Die folgende Auflistung stellt die wichtigsten Elemente zusammen:

- Jeweils zwei schnelle und hochauflösende D/A- und A/D-Wandler (vgl. Kap. 4.1),
- Operationsverstärker-Treiberstufen für die analogen Signale,
- Erweiterungssockel zum Aufstecken von Zusatzplatinen (z.B.: für Mehrantennenanwendungen),
- Schnittstellen für Flachbandkabel,
- eigenständige Oszillatoren zur Takterzeugung,
- Spannungsregler zur Erzeugung der benötigten Versorgungsspannungen (1,8V; 3,3V und 5V),
- 24 frei programmierbare/schaltbare LEDs für Überwachungsfunktionen,
- Flash-Speicher und Controller zum Laden des FPGA Programms (PC programmierbar),
- 8-MBit SRAM-Speicher.

Wichtig für den Einsatz als OFDM-Funkmodem sind die analogen Schnittstellen. Sie bieten die Möglichkeit, das in der digitalen Verarbeitungskette prozessierte Sendesignal direkt auszugeben bzw. ein analoges Empfangssignal einzuspeisen. Die Wandler sind so ausgelegt, dass sowohl IQ-Basisbandsignale als auch digital modulierte Zwischenfrequenzsignale genutzt werden können.

Die Einsteckkarte hat eine Vielzahl von zusätzlichen Schnittstellen, die direkt mit dem FPGA verbunden sind. Damit ist es möglich, weitere Baugruppen als Aufsteckkarte oder über Kabelverbindungen anzubinden. Beispiele dafür sind eine im Institut für Nachrichtentechnik an der

Technischen Universität Hamburg-Harburg entwickelte Mehrantennenplattform und eine USB2.0-Interfaceplatine.

Verschiedene Oszillatoren zur Takterzeugung sowie eine große Zahl an frei programmierbaren LEDs erweitern die flexible Nutzbarkeit. Ein acht Megabit großer SRAM-Baustein steht für speicherintensive Anwendungen zur Verfügung. Damit das FPGA nach dem Einschalten eigenständig gebootet werden kann, ist ein nichtflüchtiger Flash-Speicher integriert. Dieser kann sowohl über die Programmierschnittstelle der Platine als auch über den PCI-Bus gelesen und geschrieben werden.

6.2 Bedienoberfläche des aufgebauten Demonstrator-Systems

Zur Steuerung des OFDM-Demonstrators wurde eine Software entwickelt, deren Schwerpunkt auf den Monitoring-Möglichkeiten des aktuellen (FPGA-internen) Zustands des OFDM-Modems liegt. Zur Untersuchung und Verbesserung der einzelnen Verarbeitungsschritte können hier Statusinformationen der Funktionseinheiten sowie verschiedene Parameter des Übertragungskanals beobachtet werden. Eine beispielhafte Oberfläche dieser Steuerungssoftware ist in Abb. 6.3 dargestellt. Die Software ist so aufgebaut, dass in einem Basisrahmen verschiedene Funktionen des OFDM-Demonstrators ausgewählt werden können. So können die jeweils gerade gewünschten Überwachungsfunktionen nebeneinander dargestellt werden. In der hier vorliegenden Konfiguration gibt es sechs benachbart angeordnete Rahmen, in den solche *Add-In's*[1] gestartet werden können. Jeder Rahmen kann zusätzlich per „Registerkarte“ noch einmal bis zu vier Funktionen beinhalten [OHL04].

Als Testsignal für hohe Datenübertragungsraten wird häufig ein Videosignal eingesetzt. Dieses Sende- oder Empfangssignal ist in dem Rahmen oben rechts zu erkennen. Darüber hinaus sind hier ein Konstellationsdiagramm der Empfangsdaten und die aktuellen Werte der Kanalschätzung (Betrag und Phase) dargestellt. Weitere mögliche Funktionen sind Steuerungen der OFDM-Parameter (FFT-Länge, Art der Synchronisation usw.), die Darstellung der Daten des A/D-Wandlers oder eine vollständige BER-Messung in Echtzeit. Außerdem können die Soft Values der Demodulation als Histogramm angezeigt werden. Ein weiteres Plug-In gibt die Daten der Synchronisation (Leistung und Metrik) grafisch wieder.

1 Mit dem englischen Begriff *Add-In* wird hier ein Softwaremodul bezeichnet, welches durch Ziehen mit dem Mauszeiger aus einer Liste der verfügbaren Funktionen auf die Rahmenoberfläche gestartet wird.

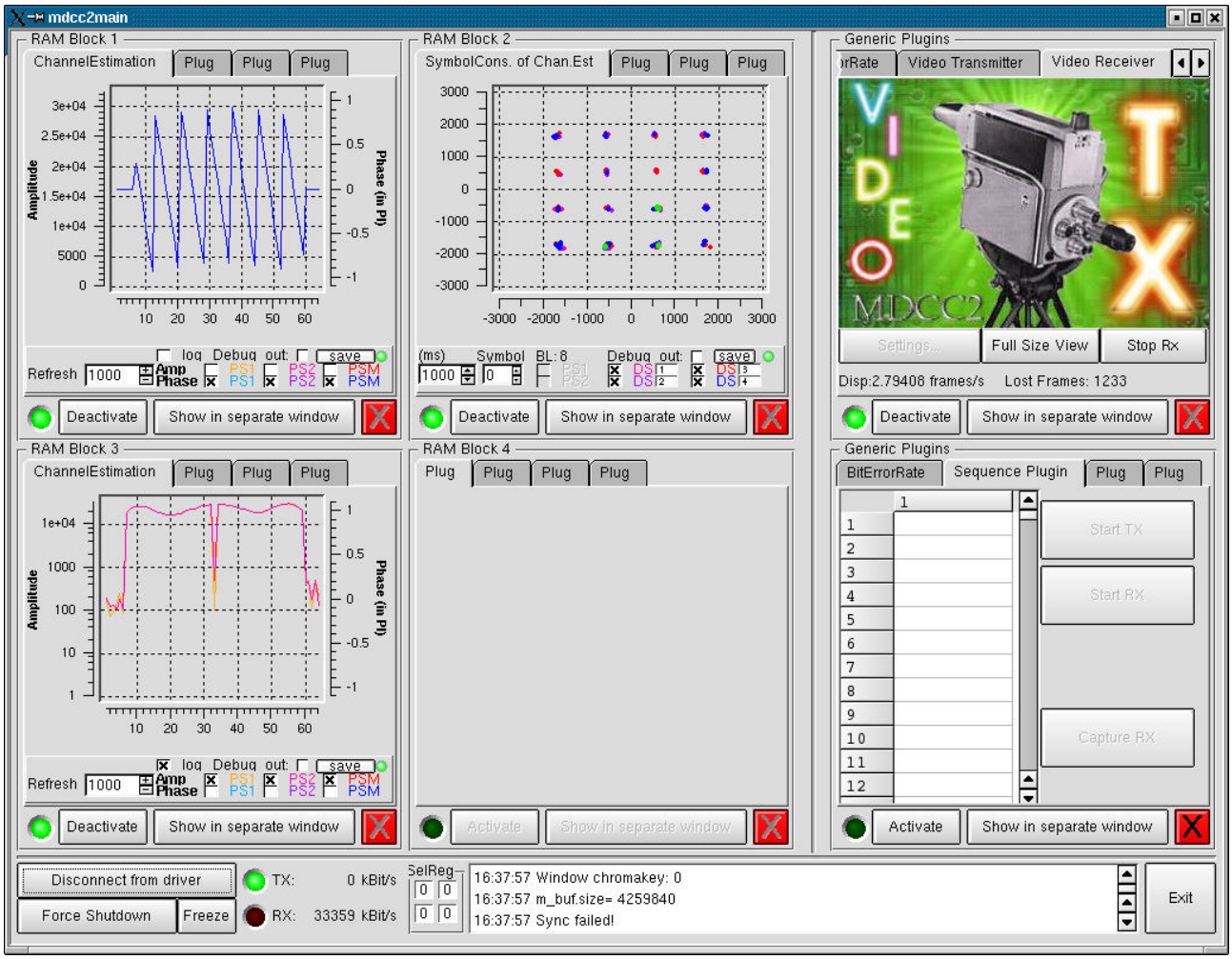


Abb. 6.3 Bedienoberfläche der OFDM-Demonstratorsoftware

Die Trennung zwischen Nutzdatenstrom und den Überwachungsdaten orientiert sich an den beiden Modi FIFO-Transfer und Pass-Thru-Mode der PCI-Schnittstelle. Während die Nutzdaten immer mit hoher Geschwindigkeit über die FIFO-Speicher mit dem OFDM-System ausgetauscht werden, greift die Überwachungssoftware bei Bedarf im Pass-Thru-Verfahren auf das FPGA zu.

Alle Steuerungs- und Überwachungskomponenten sind im FPGA als Speicherbereich hinterlegt, damit direkt darauf gelesen oder geschrieben werden kann. So ist z.B. bei der Kanalschätzung immer die Schätzung der Phasen- und Amplitudenwerte für die spätere Entzerrung zwischenzuspeichern. Das Bevorraten solcher Basisdaten ist im vorliegenden Demonstrator so implementiert, dass diese Wörter über einen zweiten Port unabhängig von der laufenden Datenverarbeitung für die Darstellung im Überwachungsfenster ausgelesen werden können. Für diese Methode werden insbesondere die Dual-Port-Speicher des Virtex-II FPGAs genutzt. Die Steuerungslogik muss aber dennoch sicherstellen, dass nur zusammenhängende Datenblöcke (z.B. OFDM-Symbole) an die Monitoring-Software übergeben werden.

Generell ist es auf diese Weise möglich, beliebige Daten zur Untersuchung eines Datenübertragungssystems zu beobachten. Je nach Kapazität des verwendeten FPGAs kann sogar eine Vorverarbeitung der Informationen innerhalb des Übertragungssystems integriert werden. Ein universeller Zugriff auf Analysedaten, der jederzeit durch eine Umprogrammierung verändert oder erweitert werden kann, stellt einen wesentlichen Vorteil dieser FPGA-basierten Experimentalhardware da.

6.3 Hardwaretest mit simulierten Kanalmodellen

Neben der Integration von Monitorfunktionen in die OFDM-Demonstrator-Umgebung selbst bietet das vorgestellte Experimentalsystem auch die Möglichkeit, herkömmliche Softwaresimulationen mit den Vorteilen der sehr schnellen FPGA-Verarbeitung zu verbinden.

Bei der Entwicklung von Nachrichtenübertragungssystemen stehen häufig umfangreiche Bibliotheken von Algorithmen für Softwaresimulationen zur Verfügung. Diese bieten hilfreiche Anhaltspunkte, wie leistungsfähig diese Algorithmen bei bestimmten Einflüssen des Kanals sind und welche Übertragungsqualität sie bieten. Solche Simulationen vernachlässigen jedoch in der Regel die nur begrenzte Rechengenauigkeit realer Systeme. Außerdem sind sie sehr zeitaufwendig. Dennoch bieten sie wichtige Vergleichswerte, die als Referenz für die in Hardware implementierten Schaltungen dienen können.

Die Datenverarbeitung im FPGA erreicht fast nie die Genauigkeit von Computersimulationen mit Wortbreiten von 32 oder 64 Bit in Fließkommaarithmetik. Aufgrund der begrenzten Ressourcen und der Geschwindigkeitsoptimierung ist es vielmehr wichtig, möglichst nur mit der minimal nötigen Genauigkeit zu rechnen. Eine solche Datenverarbeitung wird immer etwas schlechter sein als die Simulation mit (fast) unendlicher Rechengenauigkeit. Ziel ist es jedoch, die FPGA-Verarbeitung so nah wie möglich an diese optimalen Werte heranzubringen.

Um beispielsweise die Leistungsfähigkeit der realen OFDM-Algorithmen mit reduzierter Genauigkeit vergleichen zu können, kann eine gemischte Simulation genutzt werden. Dabei wird z.B. die Sender-Datenverarbeitung vollständig im FPGA ausgeführt. Anschließend werden die Daten jedoch direkt wieder in den PC zurück gelesen. Die Einflüsse des Übertragungskanals werden dann im Rechner mit den gewünschten Modellen simuliert und können so die Datenübertragung vom Sender zum Empfänger mit definierten Kanalparametern nachbilden. Damit ist es möglich, die gleichen Kanalmodelle wie bei der reinen Softwaresimulation zu benutzen. Abschließend kann die Empfangsdatenverarbeitung wieder im FPGA erfolgen. Dieses Beispiel zeigt, dass beliebige Zwischenschritte einer software- und hardwarebasierten OFDM-Datenverarbeitung mithilfe eines um eine FPGA-Plattform erweiterten PCs möglich sind.

Eine rechnerbasierte Kanalmodellierung bietet den Vorteil, dass genau bekannt ist, welche Störeinflüsse die Datenübertragung beeinträchtigen. Eine Auswertung der Bitfehlerwahrscheinlichkeit in Abhängigkeit der Rauschleistung ist damit sehr exakt möglich. Außerdem kann mit unterschiedlichen Kanalparametern für Delay- oder Doppler-Spread die Robustheit des Systems für unterschiedliche Umgebungssituationen abgeschätzt werden.

Der größte Nachteil aller rechnergestützten Simulationen ist der große Zeitaufwand im Vergleich zu einer echtzeitigen Implementierung auf spezieller Hardware. Der hier vorgestellte Demonstrator bietet die Möglichkeit, beliebige Verarbeitungsschritte in die schnelle FPGA-Schaltung auszulagern. Wenn alle Komponenten auf der PCI-Karte ausgeführt werden, ist der Schritt von einer Simulation zum Experimental-OFDM-System vollzogen.

6.4 Platzbedarf der einzelnen OFDM-Komponenten innerhalb des FPGAs

Der Verbrauch an Schaltungsressourcen ist für den Aufbau eines solchen Nachrichtenübertragungssystems von großer Bedeutung. Das Ziel ist es, alle notwendigen Komponenten einer OFDM-Verarbeitung[2] in einem Chip vorzuhalten.

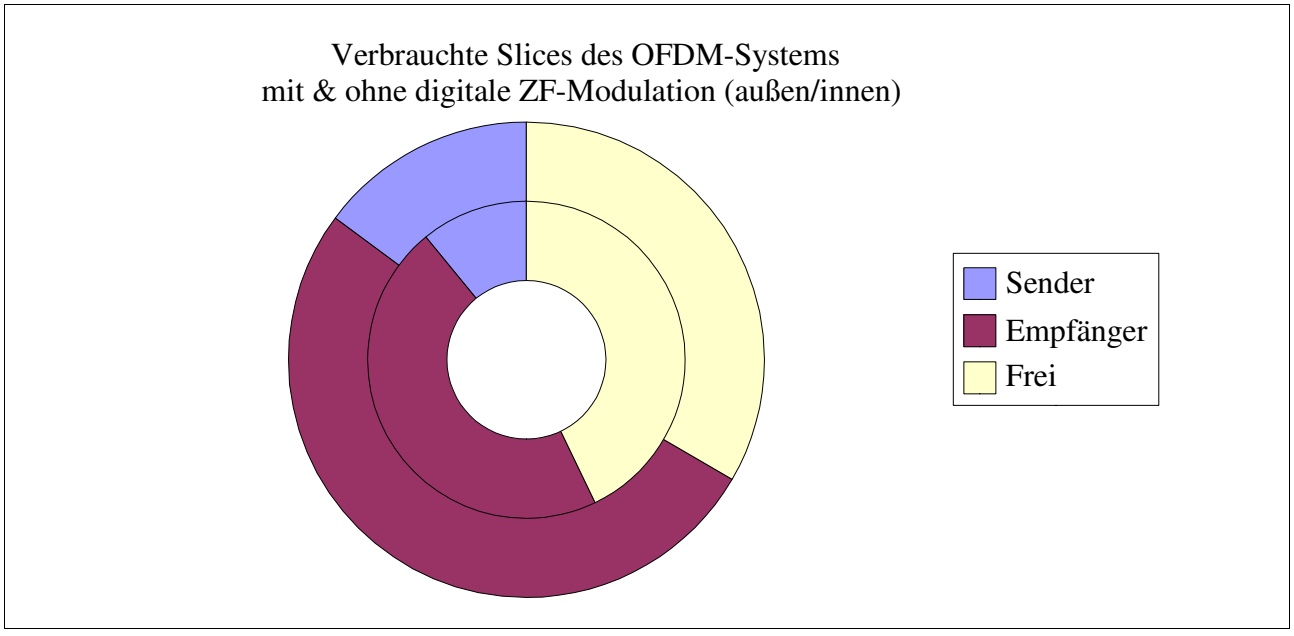


Abb. 6.4 Genutzte Slices in Bezug auf die insgesamt verfügbaren Einheiten des FPGAs (VirtexII-6000)

Der aufgebaute Demonstrator erfüllt dieses Ziel und beherbergt neben dem vollständigen Sender und Empfänger noch die in Kapitel 6.2 angesprochenen Steuerungs- und Überwachungsfunktionen.

2 Gemeint ist die Verarbeitung der physikalischen Schicht außer den analogen HF-Komponenten.

Die Auslastung des eingesetzten FPGAs ist in Abb. 6.4 dargestellt. Als Referenzgröße dient die Zahl der FPGA-Grundelemente, der Slices. Es ist gelungen, eine OFDM-Verarbeitung bei einer Auslastung von ca. 2/3 der Ressourcen in einem Chip aufzubauen. Wie am äußeren Ring der Darstellung zu erkennen ist, wird das Design bei integrierter digitaler ZF-Mischung etwas komplexer.

Im Folgenden werden die Module getrennt nach Sender und Empfänger einzeln aufgeschlüsselt. Hier werden die Komplexitätsunterschiede der einzelnen Funktionen sehr gut deutlich.

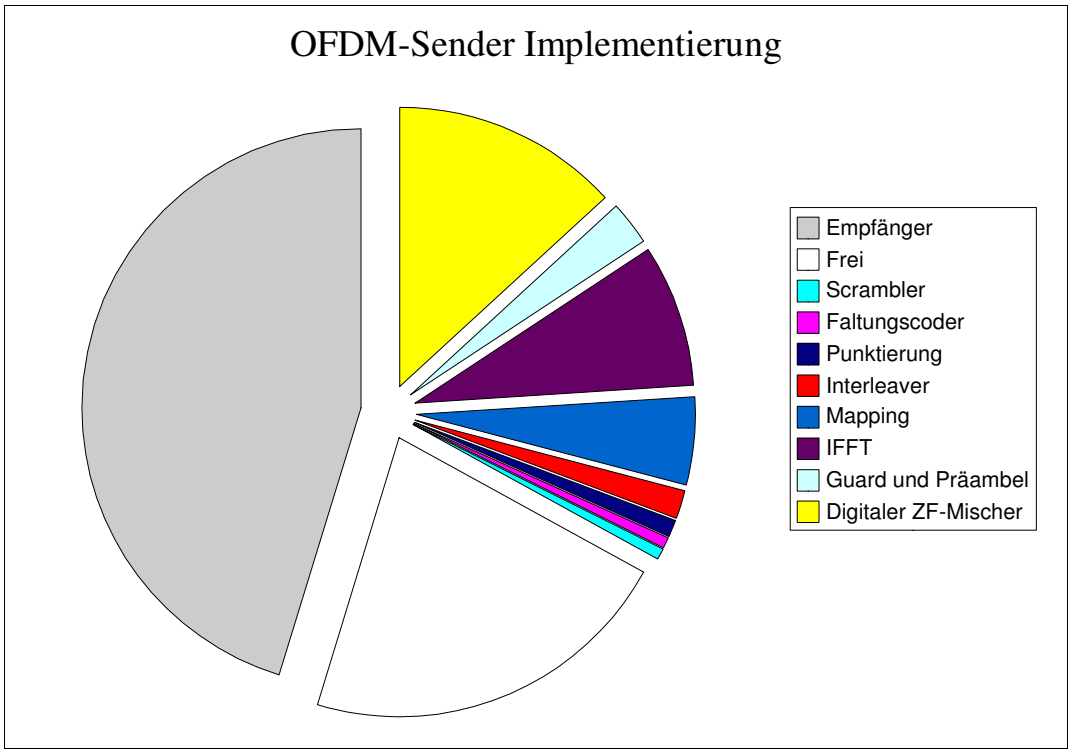


Abb. 6.5 Ressourcenverteilung des OFDM-Senders

Abb. 6.5 macht deutlich, dass außer bei der digitalen ZF-Mischung kaum Schaltungsaufwand gespart werden kann. Die 64-Punkt-FFT ist für ein HiperLAN/2-System vorgegeben. 64 Subträger sind für eine Breitbandübertragung schon im unteren Bereich, wenn der Vorteil einer größeren Symboldauer noch signifikant bleiben soll. Größere FFT-Längen würden die Komplexität weiter erhöhen. Die anderen Komponenten bieten so gut wie kein Sparpotenzial für die FPGA-Ressourcen.

Der bei weitem größte Teil der Schaltungslogik fällt im Sendezweig auf die IFFT, auch wenn er mit weniger als 10% der Chipauslastung absolut betrachtet noch nicht ressourcenkritisch ist. Der zweitgrößte Teil ist die digitale Zwischenfrequenzmischung. Diese beiden Elemente sind die einzigen, die mit fein quantisierten Zahlenwerten rechnen müssen. Alle anderen Module basieren auf bitweisen Operationen bzw. auf einfachen Abbildungen von wenigen Bits. Diese simpleren

Operationen können durch die feingranulare Technik des FPGAs sehr effizient umgesetzt werden. Insbesondere die Funktionen Scrambler, Faltungscoder und Punktierung können durch einfachste Schaltungen mit wenigen Flipflops aufgebaut werden.

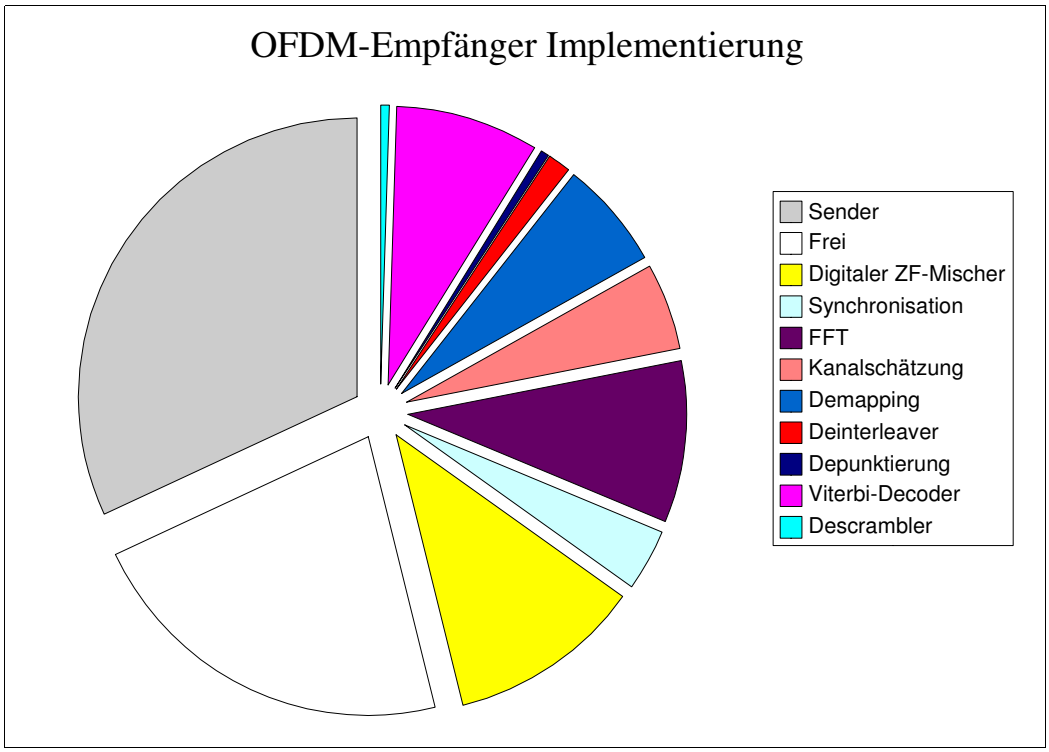


Abb. 6.6 Ressourcenverteilung des OFDM-Empfängers

Der Fehlerschutz mit Hilfe eines Faltungscodes erfordert im Empfänger einen komplexen Viterbi-Decoder. Hier besteht der größte Unterschied zwischen den komplementären Modulen im Sender und Empfänger. Obwohl der Encoder sehr klein ist, hat der Decoder den größten Ressourcenbedarf der gesamten OFDM-Verarbeitung, wie in Abb. 6.6 zu erkennen ist. Dies ist umso bemerkenswerter, weil der Viterbi-Decoder in FPGAs sehr effizient implementiert werden kann (vgl. Kap.5.2.1).

Auch im Empfänger sind wiederum alle Elemente groß, bei denen mit breiten Zahlenwerten gerechnet werden muss. Die Kanalschätzung/Entzerrer, die FFT, die digitale ZF-Mischung und die Synchronisation zählen zu diesen Verarbeitungsschritten. Da aber bis zum Decoder die Daten als Soft Values, also mindestens als Wort von vier Bit Breite, verarbeitet werden, erfordern alle Module ein entsprechend größeres Schaltnetzwerk. Einzig der Scrambler stimmt in der Größe exakt mit dem komplementären Element im Sender überein.

Da der Empfänger insgesamt sehr viel größer ausfällt, bietet er auch mehr Einsparpotenzial. Wenn ein geringerer Fehlerschutz ausreicht oder Techniken genutzt werden, die ohne eine leistungsfähige Entzerrung auskommen können, lassen sich noch kleinere Empfängerstrukturen aufbauen. Allerdings geht dann in der Regel auch ein Teil der Leistungsfähigkeit verloren. Der hier aufgebaute Empfänger wurde so dimensioniert, dass er alle Modi eines HiperLAN/2- bzw. IEEE802.11a-Übertragungssystems mit hoher Performance unterstützt.

Die Abb. 6.4 bis Abb. 6.6 zeigen die FPGA-Auslastung des im Rahmen dieser Arbeit aufgebauten OFDM-Modems. Folgende Hinweise gilt es bei der Einordnung der gezeigten Ergebnisse zu berücksichtigen:

- Die Implementierungsdaten können je nach Einstellung der Software, nach Auslastung des Chips und nach Art der Implementierung, wie in Kapitel 5 beschrieben, etwas variieren. Die hier aufgeführten Werte sind direkt aus dem aufgebauten Experimentalsystem ermittelt.
- Durch das Auswerten der im FPGA aufgebauten Schaltungsteile spiegeln die Implementierungsdaten nicht nur die Komplexität des reinen Algorithmus wieder, sondern auch die notwendige Steuer- und Kontrollbeschaltung. Da diese zusätzliche Logik in realen Systemen immer notwendig ist, wird die Auswertung der Gesamtschaltung hier gegenüber der theoretischen Komplexität bevorzugt.
- Bei Ressourcen, die im FPGA nur begrenzt zur Verfügung stehen (wie z.B. Block RAMs oder Multiplizierer), ist es sehr wichtig, dass der Programmierer entscheidet, welchen Funktionen diese Schaltungsblöcke zugeteilt werden sollen. Hier sind in der Regel die zeitkritischen Aufgaben zu bevorzugen. Aber auch diese manuelle Zuteilung hat Einfluss auf die Implementierungsdaten.
- Der Schaltungsaufwand ist auch abhängig von den zusätzlich implementierten Überwachungs- und Steuerungsfunktionen, wie sie in Kapitel 6.3 dargestellt wurden. Da diese nur einen geringen Teil der Gesamtkomplexität ausmachen und davon ausgegangen wird, dass diese Komfortfunktionen normalerweise gewünscht sein werden, sind die dafür verbrauchten Ressourcen mit einbezogen.

6.5 Maximale Taktraten der OFDM-Schaltungsteile

Der maximal mögliche Datendurchsatz eines OFDM-Systems wird maßgeblich von der Verarbeitungsgeschwindigkeit der einzelnen Schaltungsteile bestimmt. Um diese Geschwindigkeit ermitteln zu können, wird zu den einzelnen Systemkomponenten jeweils die maximal erlaubte Taktrate ermittelt. In Abb. 6.7 und 6.8 sind die maximalen Taktraten jeweils für die einzelnen Module des Senders und des Empfänger dargestellt.

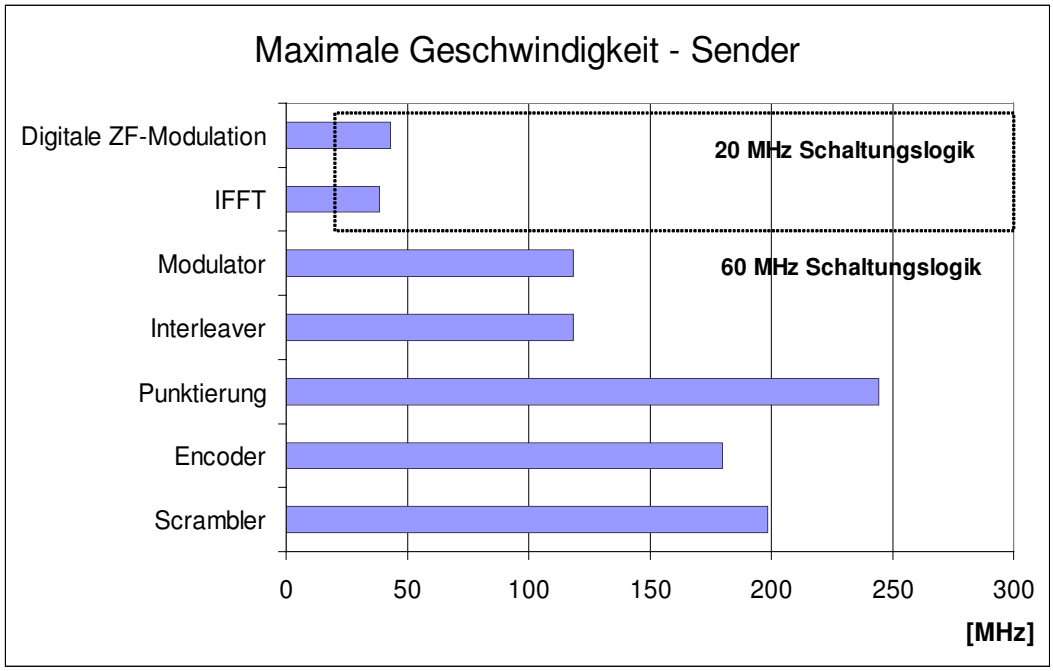


Abb. 6.7 Maximale Taktraten der OFDM-Sender-Baugruppen

Wichtig bei der Betrachtung dieser Taktraten ist jedoch die Tatsache, dass die hier ermittelte Frequenz keine unmittelbare Aussage über den möglichen Datendurchsatz erlaubt. Datendurchsatz und Taktrate korrespondieren nur dann, wenn pro Taktzyklus genau ein Informationsbit verarbeitet wird. Dies ist z.B. in dem Encoder und Decoder der Fall, wo 60 MHz Schaltungstakt auch einer Datenrate von 60 Mbit/s entsprechen. Insbesondere nach dem Bitmapping ist dieser Zusammenhang nur noch schwer zu erkennen, da hier der Bitstrom in Zahlenwerte „verpackt“ wird. Hier muss dann zur Ermittlung des Datendurchsatzes die Modulationswertigkeit mit einbezogen werden. Deutlich wird dieser Sachverhalt auch daran, dass ab dem Modulator der Systemtakt 20 MHz beträgt. Dies entspricht der konstanten Abtastrate eines HiperLAN/2-Systems, bei dem je nach verwendetem Modus jedoch ein Datendurchsatz zwischen 6 und 54 Mbit/s eingestellt werden kann.

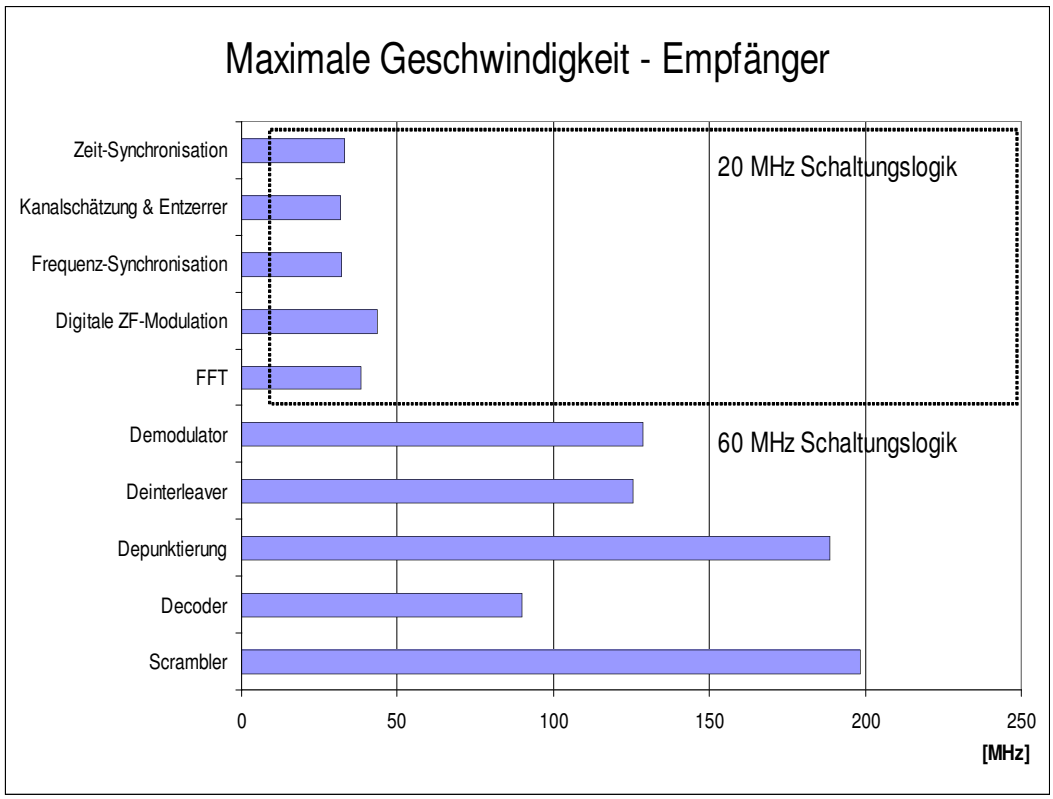


Abb. 6.8 Maximale Taktraten der OFDM-Empfänger-Baugruppen

Die Taktraten sind dennoch für den Systementwurf sehr interessant, da sie zum einen Hinweise auf die Komplexität der Schaltung (insbesondere des Logiknetzwerkes) geben und zum anderen immer eine notwendige Eingangsgröße darstellen, wenn der Datendurchsatz berechnet werden soll. Auch wenn je nach Art der Verarbeitung dazu noch weitere Parameter notwendig sind.

Eine Einordnung der Verarbeitungsgeschwindigkeit der einzelnen Komponenten ist in ihrer detaillierten Betrachtung in Kapital 5 erfolgt. In Abb. 6.7 und 6.8 ist zu erkennen, dass die jeweiligen Einzelfunktionen noch Potenzial für eine Erhöhung des Datendurchsatzes bieten. Der jeweils erforderliche Schaltungstakt in dem aufgebauten OFDM-System (entweder 20 oder 60 MHz) in in den Darstellungen kenntlich gemacht. Am ehesten ausgereizt ist in diesem Fall die Fehlerkorrektur mit Hilfe des Viterbi-Decoders.

Die Implementierung in einem FPGA bietet darüber hinaus die Möglichkeit, solche Engpässe durch den Aufbau von parallel arbeitenden Schaltungsteilen zu umgehen. So kann bei der blockweisen OFDM-Datenverarbeitung auch eine FFT oder die Fehlerkorrektur doppelt installiert werden, was direkt zu einer Verdoppelung der Datenrate in diesen Modulen führen würde. Für den hier zugrunde liegenden HiperLAN/2-Standard waren solche Maßnahmen auf der aufgebauten Experimentalhardware aber nicht notwendig.

Insgesamt bieten heutige programmierbare Logikbausteine die nötigen Leistungs- und Geschwindigkeitsreserven, um die digitale Signalverarbeitung von Kommunikationssystemen mit hoher Bandbreite zu prozessieren.

7 Zusammenfassung

Die Umsetzung der Signalverarbeitung für Nachrichtenübertragungssysteme auf moderner FPGA-Hardware verspricht ein hohes Maß an Leistungsfähigkeit, Erweiterbarkeit und Flexibilität.

Die gute Parallelisierbarkeit der Algorithmen für die OFDM-Übertragungstechnik kann in diesen programmierbaren Logikbausteinen sehr effizient genutzt werden. Die vorliegende Arbeit zeigt Wege auf, wie die einzelnen Verarbeitungseinheiten für die Implementierung im FPGA optimiert werden können. Mit genauer Kenntnis der Hardwarestrukturen ist es möglich, den Schaltungsaufwand zu minimieren und Signalverarbeitung mit kleinen und kostengünstigen Halbleiterbausteinen zu realisieren. Außerdem hilft eine hardwaregünstige Programmierung, leistungsfähigere und aufwendigere Algorithmen zu integrieren oder die Verarbeitungsgeschwindigkeit zu erhöhen.

In der vorliegenden Arbeit sind alle Komponenten der physikalischen Schicht einer OFDM-Datenverarbeitung in VHDL programmiert und auf einer FPGA-Hardware umgesetzt worden. Die Designkriterien der einzelnen Module wie Verarbeitungsgenauigkeit, Ressourcenverbrauch und Rechengeschwindigkeit werden aufgezeigt. Das Ziel ist, die gesamte Kette der OFDM-Verarbeitungsschritte ideal abzustimmen und so eine hardwareeffiziente Umsetzung zu finden. Beispielsweise macht es wenig Sinn, die Algorithmen im OFDM-Sender mit hoher Rechengenauigkeit zu programmieren, wenn der D/A-Wandler am Ende der Verarbeitungskette diese Genauigkeit nicht umsetzen kann.

Jedes Modul soll so kompakt wie möglich implementiert werden, solange dies keinen nennenswerten Verlust bezüglich der Gesamtperformance zur Folge hat. Solche aufwandsoptimierten Elemente führen bei der FPGA-Implementierung zu den besten Ergebnissen: geringer Verbrauch von Chipkapazitäten und maximal hoher Schaltungstakt.

Ein weiterer Fokus liegt auf Designparametern, bei denen zwischen platzsparender Implementierung und größerer Leistungsfähigkeit abgewogen werden muss. So kann z.B. eine aufwendige Fehlerkorrektur zu einer robusteren Übertragung führen. Die Anforderungen an die Hardware steigen dabei aber deutlich an. Für eine sinnvolle Abwägung solcher Parameter werden verschiedene Erfolg versprechende Varianten untersucht. Die Ergebnisse aus diesen Simulationen und Messungen zeigen die zu erwartenden Gewinne in Relation zum Aufwand. Mit solchen Daten

ist es möglich, die Größe der Verarbeitungseinheiten den freien Kapazitäten der Hardware anzupassen und so die bestmögliche Performance bei der Datenübertragung zu erzielen.

Große Vorteile programmierbarer Schaltungsstrukturen sind die Flexibilität der Designparameter und die Möglichkeit, die VHDL-Programmierung auf unterschiedlichen FPGA- und CPLD-Plattformen einzusetzen. Moderne Halbleiterbausteine bieten immer noch ein hohes Potenzial für Weiterentwicklungen. Insbesondere kann die vorgestellte OFDM-Signalverarbeitung von der kontinuierlichen Verbesserung der FPGA-Technologie profitieren.

Die Untersuchungen zeigen, dass ein vollständiger OFDM-Transceiver in einem Chip implementiert werden kann. Sender und Empfänger können unabhängig voneinander im selben FPGA arbeiten. Die Größe des Designs hängt dabei wesentlich von vier Faktoren ab: Länge der OFDM-Symbole, Verarbeitungsgenauigkeit, Leistungsfähigkeit der Fehlerkorrektur und erforderliche Verarbeitungsgeschwindigkeit. Außerdem beeinflusst die Zahl der möglichen Verarbeitungsmodi, wie z.B. verschiedenen Modulationswertigkeiten oder unterschiedliche Coderaten, die Komplexität des Designs.

Der implementierte Empfänger verbraucht dabei mehr als $^2/_3$ der Ressourcen des Tranceivers. Dieser Anteil wird bei aufwendigeren Designs im Verhältnis zum Sender noch weiter erhöht. Beim Aufbau von Funkübertragungsstrecken mit unsymmetrischem Datenaufkommen kann eine Anpassung der jeweiligen Sende- und Empfangsmodule die Anforderungen an die Hardware weiter reduzieren. Hier kann der Übertragungsrichtung mit geringerer Datenrate (bei Netzwerkzugriffen in der Regel der Uplink) ein weniger komplexer Empfänger bereitgestellt werden.

Der aufgebaute OFDM-Demonstrator sowie die in dieser Arbeit betrachteten Algorithmen zeigen, dass FPGAs neben DSPs und ASICs eine leistungsstarke und zukunftsweisende Alternative zur digitalen Signalverarbeitung darstellen.

Anhang A: Soft Value Approximation

Die Demodulation mit einer Soft-Value-Werteübergabe an den Viterbi-Decoder verspricht wie in Kapitel 5.3 beschrieben einen deutlich höheren Codiergewinn. Dabei gibt Gleichung 5.8 eine Möglichkeit an, wie die Wahrscheinlichkeiten der im Empfänger entschiedenen Symoble bit-basiert zu ermitteln. Diese Werte werden im Viterbi-Decoder zum Bilden von Metriken genutzt und dienen zur Auswahl von Bitsequenzen nach einer *Maximum Likelihood Sequence Estimation* (MLSE). Eine Maximum-A-Posteriori-Schätzung (MAP), bei der noch berücksichtigt wird, mit welcher Wahrscheinlichkeit ein Symbol überhaupt gesendet wurde (Vorwissen), maximiert die Rückschlusswahrscheinlichkeit. Bei dem gegebenen System kann jedoch in guter Näherung angenommen werden, dass mit fast identischer Wahrscheinlichkeit eine Null oder eine Eins gesendet wurde. Laut [GAL06] kann damit die wahrscheinlichste Bitfolge $\hat{S}$, unter der Bedingung, dass R empfangen wurde, als Summe ermittelt werden zu

$$\hat{S}_k = arg\,max[P\{S_k|R_k\}] = arg\,max\sum_k \ln(p(R_k|S_k)) := arg\,max\sum_k \lambda(S_k) \qquad (7.1)$$

Der Index *k* markiert dabei als fortlaufender Index die Sendefolge. Diese Gleichung impliziert schon die Vereinfachung, dass das additive Rauschen statistisch unabhängig ist und deshalb ohne Einfluss auf die Wahrscheinlichkeit bleibt. Die einzelnen λ geben dabei Einzelwahrscheinlichkeiten für jedes im Empfänger entschiedene Bit an. Wird bei der Bestimmung von λ der Kanalzustand H berücksichtigt (stark gedämpfte Subträger sollen die Dekodierung geringer beeinflussen als leistungsstarke) und ein statistisch unabhängiges gaußverteiltes Rauschen angenommen, ergibt sich die Metrik für ein mögliches Sendesymbol S_K als

$$\lambda(S_k) = \ln(p(R_k|S_k)) = \ln\left(\frac{1}{2\pi\sigma_{n_k}^2}\cdot e^{\frac{-|R_k-H_kS_k|^2}{2\sigma_{n_k}^2}}\right) = -\ln(2\pi\sigma_{n_k}^2) - \frac{|R_k-H_kS_k|^2}{2\sigma_{n_k}^2} \qquad (7.2)$$

Für eine Maximum-Bestimmung genügt es bei der Annahme konstanter Rauschleistung (Weißes Rauschen) den Term

$$\lambda_1(S_k) = |R_k-H_kS_k|^2 = |H_k|^2\,|\tilde{R}_k-S_k|^2 \qquad (7.3)$$

zu betrachten. $\tilde{R}$ ist dabei das entzerrte Empfangssymbol. Diese Metrik eignet sich zur symbolbasierten Decodierung.

Bei höherwertigen Modulationsverfahren ist es notwendig, im Decoder für jedes einzelne Bit eine Metrik zu ermitteln. Der in dem betrachteten OFDM-System eingesetzte Interleaver erzeugt weitestgehend die für den Decoder wichtige statistische Unabhängigkeit der betrachteten Bits.

So kann analog zu Gleichung 7.1 die Bestimmung der wahrscheinlichsten Empfangsbits angegeben werden als

$$\hat{b}_{(k,\,j)} \;=\; arg\,max[P\{R|b\}] \;=\; arg\,max\sum_{k,\,j}\ln\left(p(R_k|b_{(k,\,j)})\right) \;:=\; arg\,max\sum_{k,\,j}\lambda(b_{(k,\,j)}) \qquad (7.4)$$

j bezeichnet dabei die j-te (Bit-)Stelle des betrachteten Symbols S_K .

$$\lambda(b_{k,\,j}) = \ln\left(p(R_k|b_{k,\,j})\right) = \ln\left(\sum_{S_k|b_{k,\,j}} p(R_k|S_k)\right) \;=\; \ln\left(\sum_{S_k|b_{k,\,j}} \frac{1}{2\pi\sigma^2_{n_k}}\cdot e^{\frac{-|R_k-H_kS_k|^2}{2\sigma^2_{n_k}}}\right) \qquad (7.5)$$

Die hier beschriebene Summe enthält die euklidischen Abstände zu allen Konstellationspunkten. Dabei liefert das Sendesymbol mit dem geringsten Abstand zum Empfangssymbol den signifikantesten Beitrag der beschriebenen Metrik. Daher kann in guter Näherung diese Metrik auf den Beitrag des dichtesten Sendesymbols beschränkt werden [GAL06]

$$\lambda_2(b_{k,\,j}) \approx -min_{S_k|b_{k,\,j}}\lambda(S_k) \qquad (7.6)$$

Die Funktion 7.5 besitzt einen kompliziert zu berechnenden Exponentialanteil. Alternativ kann zur Bestimmung der Maxima der Wahrscheinlichkeit die logarithmische Likelihood-Funktion (*Log-Likelihood*) genutzt werden. Diese Funktion hat ihre Maxima an identischen Stellen.

Um jedoch eine Metrik zu bestimmen, die direkt zur Soft-Value-Viterbi-Decodierung nutzbar ist, wird häufig das Verhältnis der Log-Likelihood-Werte der beiden möglichen Empfangsbits bestimmt als

$$LLR(R_k|b_{k,\,j}) = \ln\left(\frac{p(R_k|b_{k,\,j}=+1)}{p(R_k|b_{k,\,j}=-1)}\right) \qquad (7.7)$$

Hier bestimmt das Vorzeichen die Bitentscheidung, während der Betrag die Wahrscheinlichkeitsinformation wiedergibt. Die realisierte Bestimmung dieses LLR folgt wird analog zu den Gleichungen 7.2 und 7.3 für jedes hart entschiedene Bit $\hat{b}_{k,j}$ berechnet als

$$\lambda(\hat{b}_{k,j}) = (-1)^{\hat{b}_{k,j}} \cdot |\hat{H}_k|^2 \cdot \left(|\tilde{R}_k - \hat{S}_k|^2 - |\tilde{R}_k - \overline{\hat{S}_{k,j}}|^2 \right) \qquad (7.8)$$

Da bei höherwertigen Modulationen der Empfangswert für eine Bitkombination steht, wird eine Metrik bzw. das LLR immer für das aktuell betrachtete Bit bestimmt. Während $\hat{S}_k$ das hart entschiedene Symbol bezeichnet, ist $\overline{\hat{S}_{k,j}}$ das zugehörige Hilfssymbol, welches an der aktuell betrachteten Bitposition invertiert wird.

Die in Gleichung 7.8 beschriebene Metrik kann sehr gut in der FPGA-basierten Verarbeitung realisiert werden und liefert in Kombination mit einem Viterbi-Decoder eine leistungsfähige Fehlerkorrektur.

Anhang B: Herleitung der FFT Radix-2^2 Struktur

Die Diskrete Fourier-Transformation kann für eine besonders aufwandsgünstige Implementierung in Hardware als FFT realisiert werden. Die Radix-2^2-Struktur minimiert dabei die Zahl der in Hardware auszuführenden vollwertigen Multiplikationen so weit wie möglich. Multiplikationen mit -1, j und -j werden dabei nicht mitgezählt, weil sie bei der Implementierung einer Bittransformation entsprechen und so, verglichen mit einer vollwertigen Multipliziererschaltung, nur minimalen Aufwand bedeuten. Ausgehend von der Dirskreten Fourier-Transformation

$$S(n) = \sum_{k=0}^{N-1} s_n(k) \mathrm{e}^{\frac{-j2\pi nk}{N}} = \sum_{k=0}^{N-1} s_{n,k} \mathrm{W}_N^{kn} \quad , \quad n=0,\ldots,N-1 \qquad (7.9)$$

wird W_N als komplexer Twiddle-Faktor $\mathrm{W}_N := \mathrm{e}^{\frac{-j2\pi}{N}}$

eingeführt. Die Realisierung als FFT nutzt die konjugiert-komplexe Symmetrie dieses Faktors

$$\mathrm{W}_N^{(N-n)} = \mathrm{W}_N^{-kn} = \overline{\mathrm{W}_N^{kn}} \qquad (7.10)$$

und die Periodizität in n und k

$$\mathrm{W}_N^{kn} = \mathrm{W}_N^{k(n+N)} = \mathrm{W}_N^{(k+N)n} \qquad (7.11)$$

Durch ein Zerlegen der Koeffizienten

$$k := \frac{N}{2} \cdot \kappa_1 + \frac{N}{4} \cdot \kappa_2 + \kappa \qquad mit\ \kappa_1, \kappa_2 \in \{0,1\}\ und\ \kappa \in \{0,\ldots,\frac{N}{4}-1\} \qquad (7.12)$$

$$n := \nu_1 + 2 \cdot \nu_2 + 4 \cdot \nu \qquad mit\ \nu_1, \nu_2 \in \{0,1\}\ und\ \nu \in \{0,\ldots,\frac{N}{4}-1\} \qquad (7.13)$$

kann Gleichung 7.9 geschrieben werden als

$$S(n) = S(\nu_1, \nu_2, \nu) = \sum_{\kappa=0}^{\frac{N}{4}-1} \sum_{\kappa_2=0}^{1} \sum_{\kappa_1=0}^{1} s\left(\frac{N}{2}\kappa_1 + \frac{N}{4}\kappa_2 + \kappa\right) W_N^{\left(\frac{N}{2}\kappa_1 + \frac{N}{4}\kappa_2 + \kappa\right)(\nu_1 + 2\nu_2 + 4\nu)} \tag{7.14}$$

Die innere Summe $\tilde{S}$ lässt sich durch Einsetzen von $\kappa_1 \in 0,1$ umformen zu

$$\tilde{S}(\kappa, \kappa_2 \nu_1, \nu_2, \nu) := \sum_{\kappa_1=0}^{1} s\left(\frac{N}{2}\kappa_1 + \frac{N}{4}\kappa_2 + \kappa\right) W_N^{\left(\frac{N}{2}\kappa_1 + \frac{N}{4}\kappa_2 + \kappa\right)(\nu_1 + 2\nu_2 + 4\nu)}$$
$$= s\left(\frac{N}{4}\kappa_2 + \kappa\right) W_N^{\left(\frac{N}{4}\kappa_2 + \kappa\right)(\nu_1 + 2\nu_2 + 4\nu)} + s\left(\frac{N}{2} + \frac{N}{4}\kappa_2 + \kappa\right) W_N^{\left(\frac{N}{2} + \frac{N}{4}\kappa_2 + \kappa\right)(\nu_1 + 2\nu_2 + 4\nu)}$$

Durch Substitution der beiden Twiddle-Faktoren durch Ω_1 und Ω_2 mit

$$\Omega_1(\kappa, \kappa_2 \nu_1, \nu_2, \nu) := W_N^{\left(\frac{N}{4}\kappa_2 + \kappa\right)(\nu_1 + 2\nu_2 + 4\nu)} = W_N^{\nu_1\left(\frac{N}{4}\kappa_2 + \kappa\right)} \cdot W_N^{(2\nu_2 + 4\nu)\left(\frac{N}{4}\kappa_2 + \kappa\right)}$$

$$\Omega_2(\kappa, \kappa_2 \nu_1, \nu_2, \nu) := W_N^{\left(\frac{N}{2} + \frac{N}{4}\kappa_2 + \kappa\right)(\nu_1 + 2\nu_2 + 4\nu)} = W_N^{\frac{N}{2}(\nu_1 + 2\nu_2 + 4\nu)} \cdot W_N^{\left(\frac{N}{4}\kappa_2 + \kappa\right)(\nu_1 + 2\nu_2 + 4\nu)}$$
$$= (-1)^{\nu_1} \cdot W_N^{\nu_1\left(\frac{N}{4}\kappa_2 + \kappa\right)} \cdot W_N^{(2\nu_2 + 4\nu)\left(\frac{N}{4}\kappa_2 + \kappa\right)} = (-1)^{\nu_1} \cdot \Omega_1(\kappa, \kappa_2 \nu_1, \nu_2, \nu)$$

lässt sich $\tilde{S}$ umformen zu

$$\tilde{S}(\kappa, \kappa_2 \nu_1, \nu_2, \nu) = \left\{ s\left(\frac{N}{4}\kappa_2 + \kappa\right) + (-1)^{\nu_1} s\left(\frac{N}{2} + \frac{N}{4}\kappa_2 + \kappa\right) \right\} \Omega_1(\kappa, \kappa_2 \nu_1, \nu_2, \nu)$$
$$:= T(\kappa_2, \kappa, \nu) \cdot \Omega_1(\kappa, \kappa_2 \nu_1, \nu_2, \nu)$$

und damit die Ausgangsgleichung 7.9 schreiben als

$$S(n) = \sum_{\kappa=0}^{\frac{N}{4}-1} \sum_{\kappa_2=0}^{1} T(\kappa_2, \kappa, \nu) \cdot \Omega_1(\kappa, \kappa_2 \nu_1, \nu_2, \nu) \cdot$$

Mit

$$\Omega_1(\kappa,\kappa_2\nu_{1,}\nu_{2,}\nu) := W_N^{(\frac{N}{4}\kappa_2+\kappa)(\nu_1+2\nu_2+4\nu)} = W_N^{(N\kappa_2\nu)} W_N^{\frac{N}{4}\kappa_2(\nu_1+2\nu_2)} W_N^{\kappa(\nu_1+2\nu_2)} W_N^{4\kappa\nu}$$
$$=(-j)^{\kappa_2(\nu_1+2\nu_2)} W_N^{\kappa(\nu_1+2\nu_2)} W_{\frac{N}{4}}^{\kappa\nu}$$

folgt

$$S(n) = \sum_{\kappa=0}^{\frac{N}{4}-1} \sum_{\kappa_2=0}^{1} T(\kappa_{2,}\kappa,\nu)\cdot(-j)^{\kappa_2(\nu_1+2\nu_2)} W_N^{\kappa(\nu_1+2\nu_2)} W_{\frac{N}{4}}^{\kappa\nu} \cdot$$

Durch Auflösen der inneren Summe wird *M* definiert zu

$$M(\kappa,\nu_{1,}\nu_2) := \left\{ s(\kappa)+(-1)^{\nu_1} s\left(\kappa+\frac{N}{2}\right)\right\} + (-j)^{\nu_1+2\nu_2}\left\{ s\left(\frac{N}{4}+\kappa\right)+(-1)^{\nu_1} s\left(\frac{3N}{4}+\kappa\right)\right\} \quad (7.15)$$

und mit Gleichung 7.13 folgt

$$S(n) = \sum_{\kappa=0}^{\frac{N}{4}-1} \left\{ M(\kappa,\nu_{1,}\nu_2)\cdot W_N^{\kappa(\nu_1+2\nu_2)}\right\} W_{\frac{N}{4}}^{\kappa\nu} \quad (7.16)$$

$$\tilde{a}(\tilde{\kappa}) := \left\{ M(\kappa,\nu_{1,}\nu_2)\cdot W_N^{\kappa(\nu_1+2\nu_2)}\right\} \quad (7.17)$$

Die Darstellung der Gleichung in 7.16 zeigt, dass eine N-Punkt-FFT als Verkettung von N/4-Punkt-FFTs aufgebaut werden kann, wobei auf den inneren Term der Summe $\tilde{a}(\tilde{\kappa})$ jeweils wieder eine N/4-Punkt-FFT angewandt wird. Für *N* gilt jedoch die Einschränkung, dass es eine Zweierpotenz sein muss, wobei der Exponent ganzzahlig und größer als zwei sein muss. Abb. 7.1 zeigt die FFT-Struktur für N=8.

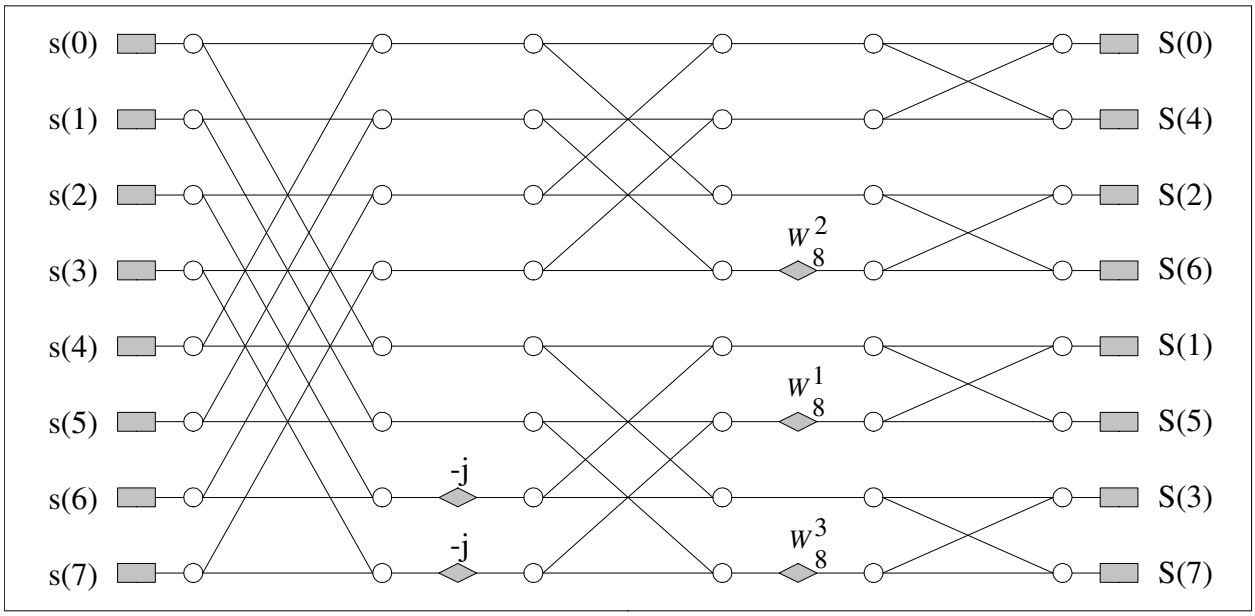


Abb. 7.1 Radix -2^2-Struktur für N=8

Der Term aus $M(\kappa, \nu_1, \nu_2)$ aus Gleichung 7.15 beinhaltet dabei nur Trivial-Multiplikationen. Dies entspricht in den Abb. 7.1 und 7.2 den ersten beiden Butterfly-Stufen, die ohne komplexe Multiplikationen mit Twiddle-Faktoren auskommen. Abb. 7.2 zeigt die 16-Punkt-FFT mit vier 4-Punkt-FFTs. Die Auflösung dieser Blöcke ergibt dann die vollständige Radix2^2-Struktur für N=16, die in Kapitel 5.4 in Abb. 5.24 gezeigt ist.

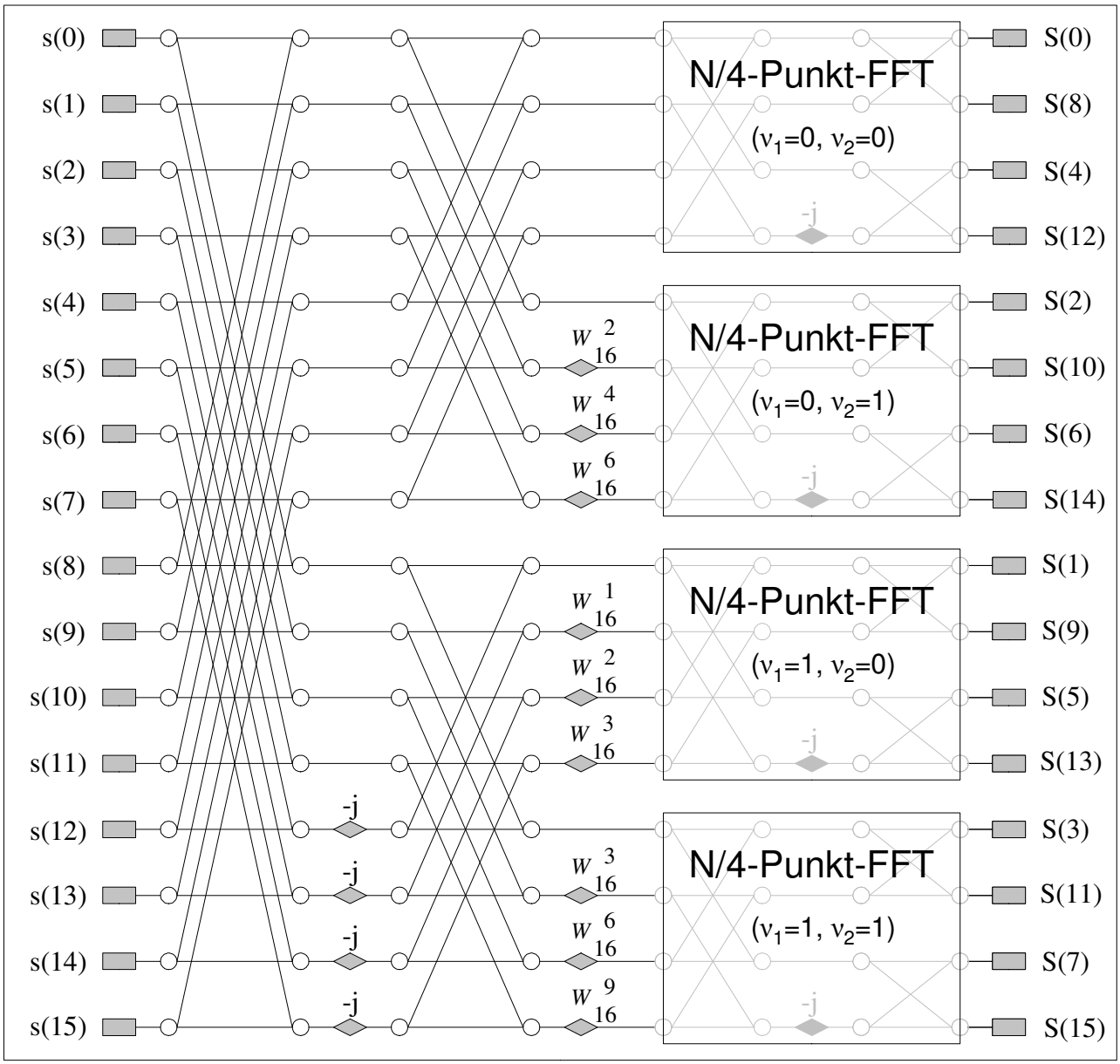


Abb. 7.2: Radix-2^2-Struktur für N=16

Eine in Radix2^2-Struktur kommt somit mit sehr wenigen Multiplikationsschaltungen aus. Bei der *N=8* und *N=16* Implementierung gibt es jeweils nur eine Spalte mit Twiddle-Faktor Operationen, bei *N=32* und *N=64* ergeben sich dann zwei dieser Spalten. Da die Struktur sequentiell abgearbeitet werden kann, ergibt sich eine minimale Anzahl an Operationen mit $log_4\ N-1$ Multiplikationen, $4log_4N$ Addierern und *N-1* Zwischenspeichern.

Ein Aufwandsvergleich zu anderen FFT-Strukturen ist beispielsweise in [HE96] aufgelistet.

Anhang C: Vergleich der physikalischen Schicht von HiperLAN/2 und IEEE802.11a

	HiperLAN/2	**IEEE802.11a**
Sendefrequenz	5,4 GHz (lizenzfrei)	5,4 GHz (lizenzfrei)
FFT-Länge	64 (3,2 µs)	64 (3,2 µs)
Guard-Intervall-Länge	16 (0,8 µs)	16 (0,8 µs)
Genutzte Subträger	52 (davon vier für Piloten)	52 (davon vier für Piloten)
Coderaten R	$\frac{1}{2}, \frac{9}{16}, \frac{3}{4}$	$\frac{1}{2}, \frac{2}{3}, \frac{3}{4}$
Modulationsschemata	BPSK QPSK 16-QAM 64-QAM	BPSK QPSK 16-QAM 64-QAM
Datenraten	6 Mbit/s *(BPSK R=1/2)* 9 Mbit/s *(BPSK R=3/4)* 12 Mbit/s *(QPSK R=1/2)* 18 Mbit/s *(QPSK R=3/4)* 27 Mbit/s *(16-QAM R=9/16)* 36 Mbit/s *(16-QAM R=3/4)* 54 Mbit/s *(64-QAM R=3/4)*	6 Mbit/s *(BPSK R=1/2)* 9 Mbit/s *(BPSK R=3/4)* 12 Mbit/s *(QPSK R=1/2)* 18 Mbit/s *(QPSK R=3/4)* 24 Mbit/s *(16-QAM R=1/2)* 36 Mbit/s *(16-QAM R=3/4)* 48 Mbit/s *(16-QAM R=1/2)* 54 Mbit/s *(64-QAM R=3/4)*
Fehlerschutz (Einflusslänge)	Faltungscode (*m+1=7*)	Faltungscode (*m+1=7*)
Scramblerpolynom	$x^7 + x^4 + 1$	$x^7 + x^4 + 1$
Sendleistung	maximal 1,0 Watt	maximal 1,0 Watt
Geplante Reichweite in Gebäuden / Freifeld	30 m / 100-150 m	30 m / 100-150 m
Zugriffverfahren	TDMA	CSMA-CS

Tabelle C-1 Die physikalische Schicht der WLAN Standards HiperLAN/2 und IEEE802.11a stimmt in vielen Parametern überein.

Anhang D: Der CORDIC-Algorithmus zu Betrags- und Phasenbestimmung

Bei der digitalen Signalverarbeitung in dem in dieser Arbeit beschriebenen System wird zum Reduzieren der Schaltungskomplexität zum Teil bei der Berechnung zwischen kartesischen und Polar-Koordinaten gewechselt. Zur Transformation in Betrag und Phase wird der CORDIC[1]-Algorithmus verwendet. Dieser wurde 1959 von Jack E. Volder erstmals vorgestellt. Mit ihm ist es möglich, trigonometrische Funktionen und komplexe Multiplikationen bzw. Divisionen (wie z. B. auch das Drehen von Vektoren) durch in digitalen Schaltungsstrukturen einfach realisierbare Additionen und Schiebeoperationen zu bilden.

Dieser Algorithmus ist bei der Umsetzung der genannten Operationen in FPGA-Schaltungen nicht nur wegen seiner Effizienz interessant. Er bietet darüber hinaus die Möglichkeit, die Verarbeitungsgeschwindigkeit gegenüber Schaltungskomplexität zu skalieren. So sind die CORDIC-Schaltungen in dem referenzierten OFDM-Demonstrator immer an die umgebende Schaltung adaptiert. Wird an einer Stelle in der Verarbeitungskette, die z.B. mit 20 MHz Schaltungstakt arbeitet, eine Transformation in Polar-Koordinaten erforderlich, so wird das entsprechende CORDIC-Modul so eingestellt, dass die Verarbeitungsgeschwindigkeit nur minimal schneller ist. Denn je kleiner der zulässige Takt gewählt wird, desto mehr Iterationen können pro Verarbeitungszyklus ausgeführt werden. Dies führt wiederum dazu, dass der Ressourcenverbrauch der Schaltungsstruktur minimiert wird.

Die Basisfunktion des CORDIC-Algorithums ist die Drehung eines komplexen Vektors $z = x + jy$ um den Winkel Φ.

$$\begin{aligned} \hat{x} &= x\cos(\Phi) - y\sin(\Phi) \quad bzw. \quad \hat{x} = \cos(\Phi)\cdot[x - y\tan(\Phi)] \\ \hat{y} &= y\cos(\Phi) + x\sin(\Phi) \quad bzw. \quad \hat{y} = \cos(\Phi)\cdot[x + x\tan(\Phi)] \end{aligned} \tag{7.18}$$

Die zweite Darstellung des gedrehten Vektors $\hat{z}$ wird bei der Verarbeitung nun durch eine iterative Rotation angenähert, wobei der Winkel so beschränkt wird, dass $\tan(\phi) = \pm 2^{-k}$ ist bzw. aus mehreren dieser Elemente iterativ zusammengesetzt wird. 2^{-k} entspricht dabei einer einfachen Bit-Shift-Operation. Darüber hinaus kann der Term $\cos(\phi)$ durch eine Konstante ersetzt werden, sodass sich die Gleichung 7.18 umformen lässt zu der Iterationsvorschrift

$$\begin{aligned} x_{k+1} &= K_k\,[x_k - y_k \cdot d_k \cdot 2^{-k}] \qquad mit \quad K_k = \cos(\tan^{-1}\cdot 2^{-k}) = \frac{1}{\sqrt{1+2^{-2k}}} \\ y_{k+1} &= K_k\,[y_k + x_k \cdot d_k \cdot 2^{-k}] \qquad d_k = \pm 1 \end{aligned} \tag{7.19}$$

1 COordinate Rotation DIgital Computer

Diese Vorschrift erfordert neben der Konstante K_k nur noch Additionen und Bit-Shift-Operationen. Die in Hardware umgesetzte Schaltung bildet zunächst die Iterationen aus Gleichung 7.19 ohne die Konstante K_k nach.

Diese Konstante wird im Anschluss an das Iterationsverfahren als Gesamtfaktor $1/A_n$ eingerechnet. Die Implementierung in Hardware erfolgt gemäß der Iterationsvorschrift

$$
\begin{aligned}
x_{k+1} &= x_k - y_k \cdot d_k \cdot 2^{-k} \\
y_{k+1} &= y_k + x_k \cdot d_k \cdot 2^{-k} \\
\delta_{k+1} &= \delta_k - d_k \cdot \tan^{-1}(2^{-k}) \qquad mit \;\; \delta_0 = \phi \; und \; d_k = -1 \; für \; Z_i < 0, sonst + 1 \\
A_n &= \prod_{k=0}^{n-1} \sqrt{1 + 2^{-2k}} \qquad als \; Produkt \; der \; K_k
\end{aligned}
\tag{7.20}
$$

Dabei gibt *n* die Zahl der Iterationen an, was wiederum von der gewünschten Rechengenauigkeit abhängt. In jedem Schritt muss außerdem der noch zu drehende Restwinkel δ_k bestimmt werden. Wichtig ist hierbei, dass die Faktoren $\tan^{-1}(2^{-k})$ vorab berechnet werden können und dann aus einem Konstantenspeicher entnommen werden. In Tabelle 7.1 sind die Implementierungsdaten von CORDIC-Schaltungen für verschiedene Wortbreiten zusammengefasst. Auch der Faktor $1/A_n$ wird in der Regel nicht als „echte“ Division implementiert, sondern aus einer Kombination aus Bit-Shift-Operationen konstruiert. Dabei wird der Faktor aus den Teilbrüchen $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \cdots$ zusammengesetzt, die sich jeweils aus dem Ausgangswert mit einem Bit-Shift um 1, 2, 3 usw. ergeben.

	16 Bit	**12 Bit**	**8 Bit**	**12 Bit (schnell)**
Iterationen / Stufe	8	6	4	2
Slices	974	349	328	334
Flipflops	114	103	63	236
Taktzyklus [MHz]	33,04	45.92	71.17	125,35

Tabelle 7.1 Leistungsdaten verschiedener CORDIC-Implementierungen

Die auf diese Weise implementierten CORDIC-Blöcke erlauben Winkeloperation auf komplexen Zahlen mit sehr hohem Takt der FPGA-Schaltung. Dabei kann die Geschwindigkeit über die Zahl der Zwischenspeicherungen der Iterationsergebnisse beeinflusst werden. Die langsamste, aber auch kleinste Realisierung ist die, wo alle Rechenschritte ausgeführt werden, bevor sie in einem Register abgelegt werden.

Die schnellste Implementierung erfordert hingegen ein Register nach jedem Iterationsschritt. In der Regel erweisen sich Zwischenschritte zwei bis acht Iterationen pro Register als optimal (abhängig von dem Schaltungskontext). In Tabelle 7.1 ist zur Veranschaulichung der 12-Bit-CORDIC auf zwei verschiedene Weisen implementiert. Das Einfügen von Registern zur Steigerung des maximal erlaubten Schaltungstakts ist im Übrigen nichts anderes als das in Kapitel 3.6 eingeführte Pipelining.

Abkürzungen

ABEL	Advanced Boolean Equation Language
ABNT	Arbeitsbereich Nachrichtentechnik der Technischen Universität Hamburg-Harburg
ACS	Add, Compare and Select
AGC	Automatic Gain Control
ARQ	Automatic Repeat Request
ASIC	Application-Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BCH	Broadcast Channel
BER	Bit Error Rate
BGA	Ball Grid Array
BPSK	Binary Phase Shift Keying
CDMA	Code Division Multiple Access
CFO	Carrier Frequency Offset
CLB	Configurable Logic Block
CPLD	Complex Programmable Logic Device
CSI	Channel State Information
DAB	Digital Audio Broadcast
DCM	Digital Clock Manager
DDS	Direct Digital Synthesizer
DFT	Diskrete Fourier-Transformation
DLL	Data Link Layer
DPRAM	Dual Port Random Access Memory
DSL	Digital Subscriber Line
DSP	Digital Signal Processor
DVB-T	Digital Video Broadcast -Terrestrial
FDM	Frequency Division Multiplexing
FDMA	Frequency Division Multiple Access
FEC	Forward Error Correction
FFT	Fast Fourier-Transformation
FIFO	First In First Out (Memory)

FPGA	Field Programmable Gate Array
FPLA	Field Programmable Logic Array
GAL	Generic Array Logic
GSM	Global System for Mobile Communications
HiperLAN	High Performance Radio Local Area Network
HDL	Hardware Description Language
(I)DFT	(Inverse) Diskrete Fourier-Transformation
(I)FFT	(Inverse) Fast Fourier-Transformation
I/O	Input/Output
IC	Integrated Circuit
ICI	Inter Channel Interference (Inter Carrier Interference)
IFFT	Inverse Fast Fourier-Transformation
IP-Core	Intellectual Property Core
ISI	Inter Symbol Interference
LC	Logic Cell
LLR	Log-Likelihood Ratio
LOS	Line of Sight
LSB	Least Significant Bit
LTI	Linear Time Invariant
LUT	Look-Up-Table
LVDS	Low-Voltage Differential Signaling
LVTTL	Low-Voltage Transistor-Transistor Logic
MAC	Medium Access Control
MPU	Metric Processing Unit
MSB	Most Significant Bit
MUX	Multiplexer
NCO	Numerically Controlled Oscillator
NLOS	No Line of Sight
OFDM	Orthogonal Frequency Division Multiplexing
PAL	Programmable Array Logic
PAR	Peak-to-Average Ratio
PALASM	PAL Assembler

PC	Personal Computer
PCI	Peripheral Component Interconnect
PDU	Protocol Data Unit
PLD	Programmable Logic Device
PM	Path Memory
PROM	Programmable Read Only Memory
QAM	Quadratur Amplituden Modulation
QPSK	Quadrature Phase Shift Keying
RAM	Random Access Memory
REA	Register Exchange Algorithm
RI	Reliability Information
ROM	Read Only Memory
SDFR	Spurious Free Dynamic Range
SFO	Sampling Frequency Offset
SoC	System on a Chip
SOVA	Soft Output Viterbi Algorithm
TDMA	Time Division Multiple Access
UMTS	Universal Mobile Telecommunications System
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
WLAN	Wireless Local Area Network

Formelzeichen

A_{MPU}	Zahl der Register in der MPU
α	Dämpfungsfaktor
B	Bandbreite
B_K	Kohärenzbandbreite
c	Lichtgeschwindigkeit
C_n	Codewort n, $n \in \mathbb{N}$
δ	Dirac Impuls
f	Frequenz
f_0	(Mitten-) Trägerfrequenz
f_C	Trägerfrequenz
f_D	Dopplerfrequenz
F_g	Grenzfrequenz bandbegrenzter Signale
F_S	Sampletakt der Wandler
h	Kanalimpulsantwort
H	Übertragungsfunktion
$\widehat{H}$	Geschätzte Übertragungsfunktion
l	Wortbreit Bits bei der Quantisierung
ΔL	Längendifferenz
λ	Softwert der Demodulation
m	Ausgangswortbreite der FFT/IFFT
m_M	Interne Wortbreite des Modulators (Mapping)
M	Anzahl der Modulationssymbole
M_n	Metrik n, $n \in \mathbb{N}$
M_{REG}	Registerbreite der MPU
n	Eingangswortbreite der FFT/IFFT
N	Rauschen
N_P	Anzahl der Ausbreitungspfade
N_{BPSC}	Zahl der codierten Bits pro Subträger
N_{CBPS}	Zahl der codierten Bits pro Symbol
n_f	Oversampling-Faktor
n_S	Breite der Soft Values
υ	Index der Ausbreitungspfade
P_L	Pfadlänge bei der Viterbi-Decodierung
Φ	Phasenverschiebung
r	Empfangssignal im Zeitbereich
R	Empfangssignal im Frequenzbereich
$\widetilde{R}$	Entzerrtes Empfangssignal
R_C	Coderate
R_{FFT}	Anzahl der gespeicherten Wörter in der FFT
R_{PM}	Anzahl der Speicherzellen im Pfadspeicher
s	Sendesignal im Zeitbereich
S	Sendesignal im Frequenzbereich
S_S	Sendeleistung

t	Zeit
T	Periodendauer der diskreten Verarbeitung, Kehrwert des Takts
T_G	Länge des Guard-Intervalls
T_K	Kohärenzzeit
T_S	Nutzsymboldauer
TW^n	Twiddle-Faktor, $n \in \mathbb{N}$
τ_{max}	(Maximaler) Laufzeitunterschied
u_n	Informationswort bzw. Informationsbit
v	Geschwindigkeit
W_n	Länge des Ausbreitungspfades n, $n \in \mathbb{N}$
x_n	Eingangsdaten-Bitstrom vor der Verarbeitung
y_n	Ausgangsdaten-Bitstrom nach der Verarbeitung

Literaturverzeichnis

[AMC99] AMCC: S9535 PCI Produkt Data Book, www.amcc.com, 1999.

[AUE01] A. Auer, D. Rudolf: FPGA Feldprogrammierbare Gate Arrays, Hüthig, 1995.

[BRÜ02] K. Brüninghaus: Ein Beitrag zur Demodulation, Synchronisation und Kanalschätzung in OFDM-basierten Funkübertragungssystemen, Dissertation an der Technischen Universität Hamburg-Harburg, 2002.

[COE01] E. Coersmeier, Y. Xu, L. Schwoerer: High Precision Analog Front End Architecture for Wireless Local Area Network, pp. 29.1-29.4, Hamburg, 6th International OFDM-Workshop, 2001.

[COO65] J.W. Cooley, J. W. Tukey: An algorithm for the machine calculation of complex Fourier series, pp. 297-301, Math. Comput. 19, 1965.

[COS01] E. Costa, C. Yin, H. Haas: Comparison of MC-CDMA and Coded-OFDM-FDMA downlink performance, pp. 25.1-25.4, Hamburg, 6th International OFDM-Workshop, 2001.

[DEM91] M. J. Demler: High-Speed Analog-To-Digital Conversion, Academic Press, Inc., 1991.

[DOP42] C. A. Doppler: Über das farbige Licht der Doppelsterne und einiger anderer Gestirne des Himmels, Prag, 1842.

[DOU02] A. Doufexi, S. Armour, M. Butler, A. Nix, D. Bull, J. McGeehan, P. Karlsson: A comparison of the HIPERLAN/2 and IEEE 802.11a wireless LANstandards, pp. 172-180, Communications Magazine, IEEE, Vol. 40, Iss. 05, 2002.

[FIS96] R.F.H Fischer, J. B. Huber: A new loading algorithm for discrete multitone transmission, IEEE GLOBECOM, pp. 724-728, 1996.

[FLI91] N. Fliege: Systemtheorie, B. G. Teubner, 1991.

[FLI93] N. Fliege: Multiraten - Signalverarbeitung, Teubner, 1993.

[FOS96] G. J. Foschini: Layered space-time architecture for wireless communications in a fading environment when using multi-element antennas, pp. 41–59, , Bell Labs Technical Journal, 1996.

[FRI94] B. Friedrichs: Kanalcodierung, Springer Verlag, 1994.

[GAL06] D. Galda: Ein Beitrag zum Vielfachzugriff, zur Synchronisation und Kanalschätzung in einem OFDM-basierten Mobilfunksystem, Dissertation an der Technischen Universität Hamburg-Harburg, 2006.

[GRÜ00] R. Grünheid: Vielfachzugriffsverfahren für die Multiträger-Übertragungstechnik, Dissertation an der Technischen Universität Hamburg-Harburg, 2000.

[GRÜ01] R. Grünheid: A Blockwise loading algorithm for the adaptive modulation technique in OFDM systems, IEEE Vehicular Technoligy Conference, (Atlantic City, USA), 2001.

[HAA05] P. Haase: Iterative Detektionsalgorithmen in differentiell modulierten OFDM-Übertragungssystemen, Dissertation an der Technischen Universität Hamburg-Harburg, 2005.

[HAG89] J. Hagenauer, P. Hoecher: A Viterbi algorithm with soft-decision outputs and its applications, pp. 47.11-47.17, Dallas, Proc. IEEE GLOBECOM, 1989.

[HE96] S. He, M. Torkelson: A New Approach to Pipeline FFT Processor, pp. 766-770, Honolulu, Hawaii, Proceedings of IPPS, 1996.

[HEI82] E. Heilmayr: AD-DA-Wandler - Bausteine der Datenerfassung, Markt & Technik, 1982.

[HIP01] ETSI TS 101 475: HiperLAN2 Physical Layer, Ver 1.3.1, 2001.

[IEE99] IEEE: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, High Speed Physical Layer in the 5 GHz Band, 1999.

[KAI98] S.Kaiser: Multi-Carrier CDMA Mobile Radio Systems - Analysis and Optimization of Detection, Decoding and Channel Estimation, Fortschritt-Berichte VDI, Reihe 10, Nr. 531, VDI-Verlag, 1998.

[KAM92] K. D. Kammeyer: Nachrichtenübertragung, B.G. Teubner, 1992.

[KRI96] M. Krim, M. Viberg: Two Decades of Array Signal Processing Research, pp. , , IEEE Signal Processing Magazine, 1995.

[KUE90] C. Kühnel: AD/DA-Praxis, Franzis-Verlag, 1990.

[LAM04] M. Lampe: Adaptive Techniques for Modulation and Channel Coding in OFDM Communication Systems, Dissertation an der Technischen Universität Hamburg-Harburg, 2004.

[LAM99a] M. Lampe: Combining multilevel coding and adaptive modulation in OFDM systems, Proc. of the 4th International OFDM Workshop, (Hamburg, Germany), 1999.

[LAM99b] M. Lampe, H. Rohling: Reducing Out-Of-Bandwidth Emissions due to Nonlinearites in OFDM Systems, pp. 2255-2259,(Houston, USA), May 1999.

[LEH94] G. Lehmann, B. Wunder, M. Selz: Schaltungsdesign mit VHDL, Franzis, 1994.

[LI00] H. Li: Automatic Repeat Request (ARQ) Mechanism in HIPERLAN/2, Vehicular Technology Conference (VTC), Tokyo, Japan, 2000.

[LÜK92] H. D. Lüke: Korrelationssignale, Springer, 1992.

[MAX03] Maxim Integrated Products: 2.4GHz/5GHz, Single-Band and Dual-Band,802.11g/a RF Transceiver ICs, Maxim, 12/2003.

[MAY98] T. May, H. Rohling: Reducing the Peak-to-Average Power Ratio in OFDM Radio Transmission Systems, Proc. of Vehicular Technologie Conference, (Ottawa, Canada), 1998.

[MAY99] T. May: Differentielle Modulation und Kanalkodierung in breitbandigen OFDM-Funkübertragungsystemen, Dissertation an der Technischen Universität Hamburg-Harburg, 1999.

[MÜL00] S. Müller-Weinfurtner: OFDM for Wireless Communicantions, Dissertation Erlangen, 2000.

[MÜL98] S. H. Müller, J. B. Huber: OFDM with reduced peak-to-mean power ratio by optimum combination of partial transmit sequences, Electronics Letters, Vol. 33, pp. 368-369, Feb 1997.

[OHL04] L. Ohliger, N. Tönder, M. Reinert, H. Rohling: Configuration and Monitoring Concept for a FPGA based OFDM Modem, pp. 294-298, Hamburg, 9th International OFDM-Workshop, 2004.

[OHM05] R. Ohm, H. D. Lüke: Signalübertragung, Springer, 2005.

[PAE99] M. Pätzhold: Mobilfunkkanäle, Vieweg, 1999.

[REI03] J. Reichardt, B. Schwarz: VHDL-Synthese. Entwurf digitaler Schaltungen und Systeme, Oldenbourg, 2003.

[STE03] M. Stege: Analyse zur Leistungsfähigkeit von Mehrantennensystemen für die mobile Datenkommunikation, VDI Verlag, .

[STE05] D. Steinke: Kanalunabhängige Störeinflüsse bei der OFDM-Signalverarbeitung, Diplomarbeit an der Technischen Universität Hamburg-Harburg, 2005.

[STR92] H. Stroppe: Physik, Fachbuchverl. Leipzig, 1992.

[SUE93] H. Süncksen: Simulation und Implementation eines parallelen Viterbi-Algorithmus auf einem FPGA, Studienarbeit an der Technischen Universität Hamburg-Harburg, 1993.

[SYN99] Synopsys Inc.: FPGA Compiler II / FPGA Express - VHDL Referenz Manual, Synopsys Documentation, 1999.

[TOE04] N. Tönder, M. Reinert, H. Rohling: Optimal Channel Code Parameter Selection for Hardware Efficient OFDM Systems, Proc. IEEE VTC'04 Fall, Los Angeles, 2004.

[TOE04a] N. Tönder, M. Reinert, L. Ohliger, H. Rohling: Flexible OFDM Demonstrator with a Single FPGA Implementation, pp. 192-196, Hamburg, 9th International OFDM-Workshop, 2004.

[TOE07] N. Tönder: Kanalschätzung, Demodulation und Kanalcodierung in einem FPGA-basierten OFDM-Funkübertragungssystem, Dissertation an der Technischen Universität Hamburg-Harburg, 2007.

[TUB03] J. Tubbax: Compensation of IQ-Imbalance in OFDM systems, pp. , IEEE Int Conference of Communication (ICC), 2003.

[VIT67] A. J. Viterbi: Error Bounds for Convolutional Codes and Asymptotically Optimum Decoding Algorithm, IEEE-IT, Vol. 13, pp. 260-269, 1967.

[WAN98] M. Wannemacher: Das FPGA Kochbuch, Int. Thomson Publ., 1998.

[WEI71] S.B. Weinstein, P. M. Ebert: Data Transmission by Frequency-Division Multiplexing using Diskrete Fourier Transform, IEEE Transactions on Communications, Vol. 19, pp. 628 - 634, 1971.

[WOL84] E. H. Wold: Pipeline and Parallel-Pipeline FFT Processors for VLSI Implementations, pp. 414-425, University of California, USA, IEEE Transactions on Computers, Vol. c-33, No. 5, May 1984.

[XIL00] Xilinx: Virtex-II 1.5V Field-Programmable Gate Arrays Advance Produkt Specification, www.xilinx.com, 2000.

[XIL01] Xilinx: Direct Digital Synthesizer (DDS) V4.0, www.xilinx.com, 2001.

[XIL03] DS260(v2.0): IP-Core datasheet: Fast Fourier Transformation v. 2.0, Xilinx, 2003.

[XIL06] Xilnix Documentation: HDL Coding Practices to Accelerate Design Performance, Xilinx, 2006.

[YEH03] W. C. Yeh, C.W. Jen: High-Speed and Low-Power Split-Radix FFT, pp. 864-874, IEEE Transactions on Signal Processing, Vol. 51, No. 3, March 2003.

[ZAN83] H. Zander: Digital-Analog-Wandler in der Praxis, Markt & Technik, 1983.

[ZAN85] H. Zander: Datenwandler, Vogel-Buchverlag, 1985.

[ZOE96] U. Zölzer: Digitale Audiosignalverarbeitung, B. G. Teubner, 1996.

Stichwortverzeichnis

Lebenslauf

Persönliche Daten

Name: Marc Reinert
geboren: 15.07.72 in Leer

Schule

1978 – 1982	Grundschule in Jheringsfehn
1982 – 1984	Orientierungsstufe Moormerland
1984 – 1992	Gymnasium in Leer

Studium

10/1993 - 08/2000	Studium der Elektrotechnik Vertiefungsrichtung Nachrichtentechnik an der Technischen Universität Hamburg-Harburg, Studienmodell Digitale Signalverarbeitung

Tätigkeiten

15.09.2000 - 30.06.2005	Wissenschaftlicher Mitarbeiter im Arbeitsbereich Nachrichtentechnik an der Technischen Universität Hamburg-Harburg.
Seit 01.08.2005	Entwicklungsingenieur bei Airbus Deutschland (Buxtehude) Schwerpunkt: Geräteentwicklung Kabinenelektronik