Design und Evaluation von Hardware-Architekturen zur Powerline-basierten Kommunikation unter extremen Umweltbedingungen

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften
Dr.-Ing.
genehmigte Dissertation von

Tobias Stuckenberg, M. Sc.

geboren am 12.11.1990
in Göttingen, Deutschland

2022

1. Referent:	Prof. Dr.-Ing. Holger Blume
2. Referent:	Prof. Dr.-Ing. Dr. h.c. Gerhard Fettweis

Tag der Promotion: 28. Oktober 2022

Danksagung

Die vorliegende Dissertation entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Mikroelektronische Systeme (IMS) der Gottfried Wilhelm Leibniz Universität Hannover.

Zuerst möchte ich mich bei Herrn Prof. Dr.-Ing. Holger Blume für die fachliche und wissenschaftliche Unterstützung durch viele konstruktive und kritische Diskussionen während meiner Promotionszeit bedanken. Durch seine Anleitung und Betreuung konnte ich während meiner Zeit am Institut sowohl Fortschritte im Bereich von Forschung und Lehre beitragen, als auch diverse Soft Skills erlernen und weiterentwickeln. Weiterhin möchte ich mich bei Herrn Prof. Dr.-Ing. Dr. h.c. Gerhard Fettweis für die Übernahme des Korreferats, sowie für das gute und konstruktive Feedback zu meiner Arbeit bedanken. Bei Herrn Prof. Dr. Schneider bedanke ich mich für die Übernahme des Vorsitz der Prüfungskommission.

Bedanken möchte ich mich auch bei meinen Kollegen Fritz und Markus mit denen ich unzählige fachliche „5-Minuten-Diskussion“ geführt habe, die am Ende immer ein oder mehrere Stunden dauerten. Diese Diskussionen haben meine Arbeit stark bereichert und mir immer wieder gezeigt, wo meine Grenzen und Möglichkeiten liegen. Ein besonderer Dank geht an Nils für die langjährige Zusammenarbeit und Freundschaft seit unseren Abschlussarbeiten am Institut, die vielen fachlichen und nicht-fachlichen Diskussionen und deine fortwährende Unterstützung für mich und meine Familie. Auch allen anderen Kollegen, die mich während meiner Zeit am Institut begleitet haben und mit mir viele Fachdiskussionen geführt haben, möchte ich danke sagen. Eure Bereitschaft, mir zu helfen und meine Ergebnisse kritisch zu hinterfragen hat einen wesentlichen Teil zu dieser Arbeit beigetragen. Danke!

Abschließend möchte ich mich bei meiner Familie bedanken. Allen voran bei meiner Frau Lena, die mir in der Zeit der Promotion und Verteidigung stets zur Seite gestanden und Kraft gegeben hat. Sie gab mir den Rückhalt, um allen Aufgaben und Hindernissen zum Trotz diese Promotion erfolgreich abzuschließen. Vielen Dank, dass du immer für mich da bist! Auch bei meinen Kindern Finja und Erik möchte ich danken. Ihr bringt mich jeden Tag aufs neue zum Lächeln und zeigt mir, was die wirklich wichtigen Dinge im Leben sind. Zuletzt möchte ich mich noch bei meinen Eltern bedanken, die mich sowohl finanziell unterstützt haben als auch mit helfenden Händen an meiner Seite standen wann immer Aufgaben zu bewältigen waren.

Hannover, Dezember 2022 Tobias Stuckenberg

Kurzfassung

Moderne elektronische Geräte setzten vermehrt auf den Austausch von Informationen mit dem Benutzer oder einem Online-Dienst des Herstellers, wie zum Beispiel im *Smart Home*, der Staubsauger-Roboter oder die Waschmaschine. Die Verbindung erfolgt zumeist über drahtlose Kommunikation zum Beispiel auf dem 2,4 GHz-Funkkanal, welcher gerade in dicht besiedelten Städten teilweise schon überlastet ist.

Eine Alternative bietet die drahtgebundene Kommunikation über die Stromversorgungsleitung, auch Powerline-Kommunikation genannt. Hierbei werden die Informationen auf freie Frequenzbänder oberhalb der Netzfrequenz moduliert und konkurrieren dabei nur mit den anderen Teilnehmern des eigenen Stromnetzes. Ein Problem bei dieser Art der Kommunikation ist der Übetragungskanal, der nicht für eine hochfrequente Übertragung ausgelegt ist und starke Störungen durch Lastwechsel oder Reflexionen an Impedanzsprüngen erzeugt. Um diese Störungen zu kompensieren verwenden moderne Powerline-Kommunikationsstandards robuste Kanalkodierverfahren, um Fehler in den übertragenen Informationen empfangsseitig korrigieren zu können.

Auf Grund dieser robusten Fehlerkorrketurverfahren, eignen sich diese Standards auch für die Kommunikation unter extremen Umweltbedingungen in der Tiefenbohrtechnik. Dabei befinden sich die elektronischen Komponenten entlang der letzten 100 m des Bohrstrangs mehrere Kilometer tief unter der Erde, wo Umgebungstemperaturen mehr als 150 °C, Drücke bis 207 MPa und mechanische Schocks auftreten. Diese extremen Umweltbedingungen haben sowohl Einfluss auf die elektronischen Komponenten, als auch auf den Übetragungskanal der Powerline-Kommunikation selbst.

In dieser Arbeit wird erstmalig eine Entwurfsraumexploration für hochtempteraturfeste Hardware-Plattformen eines Breitband-Powerline-Kommunikationssystems durchgeführt. Dabei wird ein Abtausch zwischen Durchsatz, Flexibiltät und Hardware-Ressourcen aufgezeigt, der verschiedene Pareto-optimale Punkte enthält. Diese Pareto-optimalen Punkte umfassen sowohl Prozessor-basiere Plattformen, als auch eine dedizierte Implementierung. Die dedizierte Implementierung wird anschließend in einer FPGA-basierten Emulationen bezüglich Durchsatz, Latenz und Skalierbarkeit des Netzwerkes evaluiert und optimale Konfigurationen aufgezeigt. Abschließend wird diese optimale Konfiguration für die Fertigung als Chip in einer Hochtemperaturtechnologie vorbereitet. Der gefertigte Chip wird auf einer hochtemperaturfesten Leiterplatte in einer Klimakammer unter extremen Umweltbedingungen verifiziert und evaluiert. Die Ergebnisse zeigen eine geringe Leistungsaufnahme und eine stabile Kommunikation mit geringen Paketverlustraten bis zu einer Sperrschichttemperatur von 220 °C. Die Messungen zeigen einen linearen Abtausch zwischen Spannungsversorgung und Stabilität der Kommunikation von 7 mW/°C.

Damit erweitert diese Arbeit den aktuellen Stand der Forschung um den ersten Breitband-Powerline-Kommunikation-Chip, der unter extremen Umweltbedingungen evaluiert und charakterisiert wurde.

Schlagwörter — ASIC, Chip-Design, Entwurfsraumexploration, extreme Umweltbedingungen, Hochtemperaturtechnologie, Powerline-Kommunikation, Tiefenbohrtechnik

Abstract

Modern electronic devices increasingly rely on the exchange of information with the user or an online service provided in e.g. the smart home or by devices such as vacuum cleaner robots or washing machines. The connection of this communication is usually wireless and most often the 2,4 GHz radio channel, which is already overcrowded in densely populated cities.

An alternative is wired communication via the power supply line, also known as powerline communication. In this case, the information is modulated on free frequency bands above the network frequency and only competes with the other subscribers of the own power network. One problem with this type of communication is the transmission channel, which is not designed for high-frequency transmission and generates strong interference due to load changes or reflections from impedance jumps. To compensate for this interference, modern powerline communication standards use robust channel coding techniques to correct errors in the transmitted information at the receiving end.

Due to these robust error correction methods, the standards are also suitable for communication under extreme environmental conditions in deep drilling technology. Here, components are located along the last 100 m of the drill string, several kilometers underground, where ambient temperatures exceed 150 °C, pressures of up to 207 MPa and mechanical shocks occur. These extreme environmental conditions have an influence on the electronic components themselves, as well as on the transmission channel of the powerline communication.

For the first time, this work presents a design space exploration of high temperature-resistant hardware-platforms for powerline communication. Thereby a trade-off between throughput, flexibility and hardware resources is presented, which contains several Pareto-optimal points. These Pareto-optimal points include both processor-based platforms, as well as a dedicated implementation. The dedicated implementation is evaluated regarding throughput, latency, and network scalability using an FPGA-based emulations to find an optimal configuration. Finally, this optimal configuration is prepared for fabrication as a chip in a high temperature technology.

The fabricated chip is verified and evaluated on a high temperature PCB under extreme environmental conditions inside a climate chamber. The results show low power consumption and stable communication with low packet loss rates up to a junction temperature of 220 °C. The measurements show a linear trade-off between power supply and stability of communication of 7 mW/°C.

This work extends the current state of research with the first broadband powerline communication chip evaluated and characterized under extreme environmental conditions.

key words — ASIC, chip design, deep drilling, design space exploration, harsh environment, high-temperature, powerline communication

Inhaltsverzeichnis

Abkürzungsverzeichnis

ACS	Add-Compare-Select
ADC	Analog-to-Digital-Converter
AFE	Analog Frontend
AGC	Adaptive Gain Control
AHB	Advanced High-performance Bus
ALU	Arithmetic Logic Unit
APB	Advanced Peripheral Bus
ARMT	Azimuthal Resistivity Measurement Tool
ARP	Address Resolution Protocol
ARQ	Automatic Repeat Request
ASIC	Application-Specific Integrated Circuit
ASIP	Application-Specific Instrucion-Set Processor
AWGN	Additive White Guassian Noise
AXI4	Advanced eXtensible Interface 4
BPSK	Binary Phase-Shift Keying
BM	Branch Metric
BRAM	Block RAM
CBC	Cypher Block Chaining
CFO	Carrier Frequency Offset
CIFS	Contention Interframe Spcae
CRC	Cyclic Redundancy Check
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CW	Contention Window

DA	Destination Address
DAB	Digital Audio Broadcasting
DAC	Digital-to-Analog-Converter
DBPSK	Differential Binary Phase-Shift Keying
DES	Data Encryption Standard
DIF	Differentiation In Frequency
DIT	Differentiation In Time
DL	Data Link
DMA	Direct Memory Access
DQPSK	Differential Quadrature Phase-Shift Keying
DDC	Digital Down Converter
DSL	Digital Subscriber Line
DSP	Digital Signal Processor
DVB-T	Digital Video Broadcasting – Terrestrial
EFG	End of Frame Gap
EIFS	Extended Interframe Spcae
FC	Frame Control
FCC	Federal Communications Commission
FCCS	Frame Control Check Sequence
FCS	Frame Check Sequence
FEC	Forward Error Correction
FIPS	Federal Information Processing Standard
FFT	Fast Fourier Transformation
FPGA	Field Programmable Gate Array
FU	Functional Unit
GCC	GNU Compiler Collection
GI	Guard Interval
GMD	Generalized Minimun Distance
GPP	General Purpose Processor
HWPE	Hardware Processing Engine
ICI	Inter-Carrier Interference
ICMP	Internet Control Message Protocol
ICV	Integrity Check Value
IF	Instruction Fetch
IFFT	Inverse Fast Fourier Transformation
IoT	Internet of Things

IP	Internet Protocol
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
ISI	Inter-Symbol Interference
IV	Initialization Vector
LFSR	Linear Feedback Shift Register
LLC	Logical Link Control
LLR	Log-Likelihood Ratio
LSB	Least Significant Bit
LTE	Long Term Evolution
LUT	Look-Up Table
LWD	Logging While Drilling
lwIP	Leightweight Internet Protocol
MAC	Medium Access Control
MAP	Maximum Aposteriori Probability
MCC	Motherboard Control Chip
MF	Matched Filter
MIMO	Multiple Input Multiple Output
ML	Maximum Likelihood
MME	MAC Management Entry
MMI	MAC Management Information
MPDU	MAC Protocol Data Unit
MSB	Most Significant Bit
MSDU	MAC Serice Data Unit
MTU	Maximum Transmission Unit
MWD	Measurement While Drilling
OFDM	Orthogonal Frequency Division Multiplexing
OUI	Organizationally Unique Identifier
PCB	Printed Circuit Board
PCI	Peripheral Component Interconnect
PHY	Physical
PLC	Powerline Communication
PM	Path Metric
ppm	parts per million
PSK	Phase Shift-Keying
PXI	PCI eXtensions for Instrumentation

QPSK	Quadrature Phase-Shift Keying
RIFS	Response Interframe Spcae
ROBO	Robust OFDM
RSS	Rotary Steering System
RTT	Round Trip Time
SA	Source Address
SCO	Sampling Clock Offset
SIMD	Single Instruction Multiple Data
SISO	Single Input Single Output
SNR	Signal-to-Noise Ratio
SoC	System-on-Chip
SOI	Silicon-on-Insulator
SRAM	Static Random-Access Memory
TCL	Tool Command Language
TCP	Transmission Control Protocol
TIE	Tensilica Instruction Set Extension
UDP	User Datagram Protocol
uIP	Micro Internet Protocol
uIPm	Micro Internet Protocol Modified
VHDCI	Very-High-Density Cable Interconnect
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VLIW	Very Long Instruction Word
WLAN	Wireless Local Area Network

Formelverzeichnis

$\lfloor \cdot \rfloor$	Abrunden
$\lceil \cdot \rceil$	Aufrunden
$\lvert \cdot \rvert$	Betrag
$(\cdot)^*$	komplexe Konjugation
$\ln(\cdot)$	Natürlicher Logarithmus
$\max(\cdot)$	Maximum
$\min(\cdot)$	Minimum
$(\cdot)^T$	Transponiert
$\text{sign}(\cdot)$	Vorzeichen

KAPITEL 1

Einleitung

Weltweit ist der Energiebedarf in den letzten 25 Jahren um mehr als 70 % gestiegen und wird Statistiken zufolge in den nächsten Jahren kontinuierlich weiter steigen [Hoh20]. Dabei sind fossile Brennstoffe wie Erdöl und Erdgas mit 54,1 % noch immer Hauptbestandteil der Energieerzeugung [Bre20]. Zusätzlich entsteht durch die geplante Abschaltung der Kohlekraftwerke, allein in Deutschland, ein Defizit von ca. 20 TWh an Wärmeenergie. Diese fehlende Wärme kann unter anderem durch Geothermiebohrungen und das anschließende fördern von erhitztem Wasser aufgefangen werden [Oei19].

Um diesen steigenden Nachfragen gerecht werden zu können und neue Rohstoffvorkommen zu erschließen, wurde die Technologie der Bohrstrangsysteme in den vergangenen Jahren stark verbessert. Es wurden unter anderen Systeme zum präzisen Richtungsbohren, Nehmen und Analysieren von Gesteinsproben oder Erstellen einer 3D-Umgebungskarte mittels Ultraschall entwickelt [Teo21]. Diese Techniken ermöglichen den Zugang zu bisher nicht erreichbaren Rohstoffquellen, gleichzeitig entsteht aber ein immer komplexeres Bohrstrangsystem mit vielen analogen und digitalen Elektronikkomponenten. Diese müssen mit Strom versorgt werden und erzeugen große Mengen von Mess- und Analysedaten.

Die elektronischen Komponenten befinden sich entlang der unteren 100 m des Bohrtrangs, der sich mehrere Kilometer tief in der Erde befindet. Eine Kommunikation der elektronischen Komponenten mit der Basisstation an der Erdoberfläche ist nur mit Hilfe der Bohrflüssigkeit möglich. Diese wird durch das Innere des Bohrgestänges gepumpt, kühlt den Bohrkopf während des Bohrvorganges und spült die Gesteinsreste zurück an die Oberfläche. Bei letzterem Vorgang können mit Hilfe von Druckpulsen Schwingungen an der Flüssigkeitsoberfläche erzeugt werden, die dann mit Hilfe von optischen Messverfahren in eine binäre Nachricht dekodiert werden. Aktuelle Modulationsverfahren erreichen mit dieser Methode eine Datenrate von 10 bit/s [Ber19].

Auf Grund dieser geringen Übertragungsdatenraten ist es nicht möglich, eine zentrale Steuereinheit über Tage zu betreiben, die alle Komponenten unter Tage steuert. Die gesamte Datenvorverarbeitung, Ansteuerung von Motoren oder Verarbeitung von Sensordaten muss unter Tage

geschehen. Dies führt zu zwei wesentlichen Anforderungen an die Elektronik: Daten müssen lokal verarbeitet und übertragen werden und die Komponenten müssen Temperaturen über 150 °C, mechanischen Schocks und Drücken von bis zu 207 MPa standhalten [Ahm14; Kir20].

Für die lokale Verarbeitung werden unter anderem elektronische Komponenten eingesetzt, die außerhalb ihrer spezifizierten Temperaturbereiche betrieben werden. Deren Spannungsversorgung erfolgt durch einen Generator, der durch die Bohrspühlung im Bohrstrang angetrieben wird. Eine Kommunikation der elektronischen Komponenten ist meistens nicht vorgesehen und Komponenten prozessieren und speichern Daten lokal, bis diese manuell über Tage ausgelesen werden [Goo19]. Um zukünftig autark Entscheidungen über einen Bohrverlauf treffen zu können, müssen unter anderem Daten mehrerer Sensoren lokal prozessiert und fusioniert werden. Dazu werden leistungsstarke Prozessoren und eine Kommunikation der Komponenten untereinander benötigt, die Sensordaten austauscht und Entscheidungen kommuniziert.

Eine mögliche Idee für die Vernetzung ist die Verwendung eines Powerline-Kommunikationssystems (engl.: *Powerline Communication*, PLC). Hierbei wird die bereits existierende Spannungsversorgungsleitung zur Datenübertragung mitverwendet, indem Daten auf nicht belegten Frequenzbändern im unteren MHz-Bereich übertragen werden. Dazu verwenden Breitband-PLCs meist das Orthogonale Frequenzmultiplexverfahren (engl.: *Orthogonal Frequency Division Multiplexing*, OFDM), das robust gegen Mehrwegeausbreitung und Schmalbandstörungen ist. Dieses Verfahren wird heutzutage in vielen Paket-basierten Kommunikationsstandards verwendet, wie zum Beispiel in DSL, WLAN, LTE, DVB-T oder DAB [Chi12].

Die extremen Umweltbedingungen in der Tiefbohrtechnik haben dabei sowohl Einfluss auf die kommunizierende Hardware, als auch auf die Qualität des Übertragungskanals einer Powerline-Kommunikation. Ein real gemessener digitaler Übertragungskanal entlang einer Spannungsversorgungsleitung eines Bohrstranges in mehreren Kilometern Tiefe wurde bis heute noch nicht veröffentlicht und kann aktuell nur im Labor simuliert werden. Zusätzlich gibt es weitere Randbedingungen, wie eine geringe Leistungsaufnahme des Kommunikationssystems, da das Energiebudget durch den Generator limitiert ist oder eine geringe Chipfläche, da der Bauraum für elektronische Komponenten in entlang eines Bohrstrangs begrenzt ist.

1.1. Zielsetzung

Das Ziel dieser Arbeit ist die Erforschung eines PLC-Systems für den Einsatz unter extremen Umweltbedingungen in der Tiefbohrtechnik. Es soll anhand von Hochtemperaturmessungen in einem Klimaschrank gezeigt werden, wie sich die Umweltbedingungen auf die Kommunikationseigenschaften der entworfenen Hardware auswirken und welche architekturtechnischen Änderungen eine verbesserte Systemleistung liefern kann. Hierfür soll eine Applikationsspezifische Integrierte Schaltung (engl.: *Application Specific Integrated Circuit*, ASIC) entworfen und gefertigt werden, der als Versuchsträger für diese Arbeit dient.

Das erste Forschungsziel ist das Design, die Implementierung und die Exploration des Hardware-Entwurfsraumes eines Powerline-Kommunikationssystems für die Verwendung unter extremen Umweltbedingungen. Dazu sollen in dieser Arbeit die aktuellen Standards für Powerline-Kommunikationssysteme untersucht und hinsichtlich einer Eignung für den Einsatz in einem geothermalen Tiefbohrsystem bewertet werden. Ein geeigneter Standard soll anschließend auf verschiedenen Hardware-Plattformen implementiert und evaluiert werden. Hierzu werden nach dem

Stand der Forschung geeignete Hardware-Plattformen verwendet und durch eigene Implementierungen ergänzt. Anschließend wird gezeigt, wie eine Exploration des Entwurfsraumes durchgeführt werden kann und die Implementierungen hinsichtlich Fläche, Verlustleistung, Durchsatz und Latenz bewertet. Die effizientesten Implementierungsvarianten werden anschließend durch Variation der Architekturparameter optimiert und durch dedizierte Hardware-Erweiterungen beschleunigt. Das Ziel dieser Exploration ist eine parametrierbare und skalierbare Hardware-Architektur zu identifizieren, die hinsichtlich einer oder mehrerer Architekturparameter Pareto-optimal für die Verwendung unter extremen Umweltbedingungen geeignet ist.

Das zweite Forschungsziel ist es, Korrelationen zwischen den extremen Umweltbedingungen und den Kommunikationseigenschaften der Powerline-Kommunikation zu finden. Hierzu wird die performanteste Plattform aus der Hardware-Exploration des ersten Forschungsziels gewählt und als Chip in einer Hochtemperatur-ASIC-Technologie gefertigt. Der produzierte Chip wird anschließend in eine temperaturfeste Leiterplatte (engl. *Printed Circuit Board*, PCB) integriert und in einer Klimakammer unter extremen Umweltbedingungen charakterisiert. Dabei sollen die Grenzen der Powerline-Kommunikation bezogen auf Temperatur, Spannungsversorgung und Stabilität der Kommunikation gefunden werden. Die Ergebnisse der Evaluation können für die zukünftige Entwicklung von Kommunikationssystemen und anderen Elektronikbauteilen genutzt werden, um Komponenten für den Betrieb unter extremen Umweltbedingungen stabiler, sicherer und robuster zu entwerfen.

1.2. Aufbau dieser Arbeit

Diese Arbeit gliedert sich wie folgt: Kapitel 2 gibt eine Übersicht über den aktuellen Stand der Forschung zur Hochtemperaturelektronik, den extremen Umweltbedingungen in der Tiefbohrtechnik, der Powerline-Kommunikation und der ASIC-Technologien für Hochtemperaturanwendungen. Kapitel 3 präsentiert das Powerline-Modem nach dem Kommunikationsstandard und zeigt den Aufbau von dessen MAC- und PHY-Schicht. Außerdem wird nach dem aktuellen Stand der Forschung die im Standard nicht definierte Empfangsseite des Powerline-Systems vorgestellt und in einer Referenzimplementierung algorithmisch evaluiert. Ein besonderes Augenmerk liegt dabei auf dem OFDM-basierten Synchronisationssystem. In Kapitel 4 wird die Entwurfsraumexploration der Hardware-Architekturen für die PHY- und MAC-Schicht getrennt durchgeführt. Anschließend werden die performantesten Architekturen als gesamtes Powerline-Modem aufgebaut und in Echtzeit auf FPGA-basierten Plattformen gegen proprietäre Hardware verifiziert und evaluiert. Die Ergebnisse dieser Kommunikationssystem-Evaluation dienen als Grundlage für die spätere ASIC-Integration. Kapitel 5 zeigt die Integration höherer OSI-Schichten zur TCP-/IP-basierten Kommunikation über die Powerline unter extremen Umweltbedingungen. Im abschließenden Kapitel 6 wird die Adaption des FPGA-Desings für die Fertigung als ASIC gezeigt und auf Basis der finalen Netzliste verifiziert und evaluiert. Anschließend wird ein hochtemperaturfestes PCB aufgebaut und im Klimaschrank vermessen und. Die Diskussion der Ergebnisse der Temperatur, Leistungs- und Performancemessungen und die daraus abgeleiteten Erkenntnisse zur Erweiterung des aktuellen Standes der Forschung schließend dieses Kapitel ab. Eine Zusammenfassung dieser Arbeit folgt in Kapitel 7.

KAPITEL 2

Aktueller Stand von Forschung und Technik

Die Definition von extremen Umweltbedingungen ist in der Literatur nicht eindeutig und kann je nach Einsatzgebiet und Beschaffenheit der elektronischen Komponente unterschiedlich ausgelegt werden. Für aktuell erhältliche Konsumerelektronik können 150 °C bereits als extrem angesehen werden, da dies deutlich oberhalb ihres spezifizierten Temperaturbereichs ist, wohingegen temperaturfeste Komponenten auf einer Keramiksubstrat-Leiterplatte Temperaturen oberhalb von 200 °C standhalten können [McC19]. Neben der Temperatur können hohe Strahlung, mechanische Schocks, elektrisch-magnetische Induktionen, hoher Druck oder hohe Luftfeuchtigkeit als extreme Umweltbedingungen gelten. Allgemein sind extreme Umweltbedingungen für elektronische Komponenten immer dann vorhanden, wenn diese außerhalb standardisierter Betriebsbereiche betrieben werden.

Für die Temperatur werden häufig vier Bereiche angegeben, die von kommerzieller bis militärischer Elektronik klassifiziert sind. Diese Bereiche sind in der Literatur nicht eindeutig festgelegt und können je nach Autor andere Grenztemperaturen haben. Tabelle 2.1 zeigt die in der Literatur gebräuchlichsten Temperaturbereiche. Um eine elektronische Komponente einer „extremen“ Temperatur auszusetzen, müsste sie also außerhalb des militärischen Grades betrieben werden, da dieser alle anderen Bereiche einschließt [Wri97; McC19; Con99].

Der generelle Einfluss steigender Temperaturen auf CMOS-basierte Schaltungen, lässt sich in dieser Arbeit in drei wesentliche Effekte zusammenfassen:

- Die Reduktion der Ladungsträgermobilität der Elektronen μ_n im Kanal.
- Die Reduktion der Schwellenspannung U_T des Transistors.
- Die steigende Selbsterhitzung durch Erhöhung der Leckströme.

Die ersten beiden Effekte beeinflussen den Drain-Source-Strom, den der Transistor treiben kann, wie in Gleichung 2.1 gegeben. Gleichung 2.2 zeigt den darin enthaltenen Wachstumsfaktor β, der die physikalischen und geometrischen Parameter des Transistors enthält.

Tabelle 2.1: *Namen und Bereiche der aktuell gängigsten Klassifizierungen für Temperaturbereiche elektronischer Komponenten. Jede Zeile schließt den vorherigen Temperaturbereich vollständig ein.*

Name	Temperaturbereich	
	Min.	Max.
Kommerziell	0 °C	70 °C
Industriell	−40 °C	85 °C
Automobil	−40 °C	125 °C
Militär	−55 °C	125 °C

$$I_{ds} = \frac{\beta}{2}(U_{gs} - U_T)^2 \tag{2.1}$$

$$\beta = \beta_\square \frac{W}{L} = \mu_n C_{ox} \frac{W}{L} \tag{2.2}$$

β setzt sich aus den Dimensionen der Weite W und der Länge L des Transistorkanals und dem physikalischen Parameter $\beta_\square = \mu_n C_{ox}$ zusammen. Dabei gibt C_{ox} die Kapazität zwischen Metall-Gate und dotiertem Silizium über das Oxid an [Vee17]. Sowohl μ_n als auch U_T haben dabei temperaturabhängigen Einfluss auf die Schaltverzögerung τ eines Transistors, wie in Gleichung 2.3 gezeigt.

$$\tau = \frac{2CU}{\beta_\square W/L(U_{gs} - U_T)^2} \tag{2.3}$$

U_T skaliert mit etwa −1 mV/K und damit gegenläufig zu $\beta_\square$, das nach Gleichung 2.4 mit einem Exponenten von 1,5 steigt [Cob64].

$$\beta_\square(T) = \beta_\square(298K)\left(\frac{298}{T}\right)^{3/2} \tag{2.4}$$

Zum einen sinkt die Schwellenspannung U_T, was zu einer Erhöhung des Drain-Stroms I_{ds} und der Schaltgeschwindigkeit $1/\tau$ mit steigender Temperatur führt. Zum anderen nimmt die Elektronenbeweglichkeit μ_n ab, was den Drain-Strom und die Schaltgeschwindigkeit wiederum reduziert.

In großen Technologiegrößen ($> 1\,\mu m$) dominiert der Effekt der Reduzierung der Schaltgeschwindigkeit und des Drain-Stroms, da die Schwellenspannungen mit typischerweise mehr als 0,8 V prozentual kaum reduziert wird. In aktuellen Technologien mit Schwellenspannungen $U_T \leq 0{,}3\,V$ ist die prozentuale Reduzierung stärker und die gegenläufigen Effekte heben sich teilweise auf [Vee17].

Der dritte temperaturabhängige Effekt ist die Selbsterhitzung des Chips, die sich durch die exponentiell steigenden Leckströme mit steigender Temperatur erhöht. Diese Selbsterhitzung hängt im Wesentlichen von der Leistungsaufnahme P_D nach Gleichung 2.5 ab.

$$P_D = P_{\mathrm{SC}} + P_{\mathrm{leak}} + P_{\mathrm{dyn}} \tag{2.5}$$

Dabei ist P_{SC} der Querstrom beim Schalten, P_{leak} die Leckströme und $P_{\mathrm{dyn}} = \sigma U^2 f$ die dynamische Verlustleistung. P_{SC} und P_{leak} sind technologische Parameter, die sich durch die Wahl der ASIC-Technologie beeinflussen lassen. Da in dieser Arbeit eine SOI-CMOS-Technologie verwendet wird (siehe Abschnitt 2.3), ist die statische Verlustleistung sehr gering und die Selbsterhitzung wird fast ausschließlich durch die dynamische Verlustleistung P_{dyn} bestimmt. Diese hängt von der Spannungsversorgung U als Technologieparameter, von der Taktfrequenz der Hardware-Architektur f und der Schaltaktivität des Systems σ ab. Mit Hilfe dieser Parameter kann die Leistungsaufnahme des Chips und die damit verbundene Selbsterhitzung in einer Hochtemperaturumgebung gesteuert werden.

$$T_J = T_A + R_{\mathrm{J/A}} P_D \tag{2.6}$$

Gleichung 2.6 zeigt den Zusammenhang der Sperrschichttemperatur T_J (engl. *Junction Temperature*), der Umgebungstemperatur T_A, dem thermischen Widerstand zwischen Sperrschicht und Atmosphäre $R_{\mathrm{J/A}}$ und der Leistungsaufnahme P_D des Chips. Durch eine hohe dynamische Leistungsaufnahme kann die Sperrschichttemperatur deutlich über die maximal spezifizierte Umgebungstemperatur ansteigen und so den Einsatzbereich des Chips einschränken [Gon18; Van04].

2.1. Extreme Umweltbedingungen in der Tiefbohrtechnik

Die extremen Umweltbedingungen in dieser Arbeit entstehen durch den Betrieb der elektronischen Komponenten mehrere Kilometer tief unter der Erde. In der Tiefbohrtechnik erreichen Geothermie-Bohrungen und Bohrungen nach Erdöl oder Erdgas eine Tiefe von bis zu 12 km und durchstoßen dabei verschiedene Erd- und Gesteinsschichten. Abbildung 2.1 zeigt schematisch den Aufbau eines Tiefbohrsystems bei der Erschließung (Teil a) und anschließender Förderung (Teil b) eines Reservoirs. Von einer Basisstation an der Erdoberfläche wird eine Bohrung zunächst senkrecht in den Boden gestartet und kann im Verlauf der Erschließung auch horizontal oder sogar spiralförmig fortgesetzt werden. Der Bohrstrang selbst besteht aus einem Bohrkopf mit Bohrmeißel, gefolgt von dem Bohrgestänge. Dieses Bohrgestänge besteht aus einzelnen, bis zu 30 m langen Teilstücken, die während des Bohrvorgangs von der Oberfläche aus mit Verbindern zusammengesetzt werden. Der gesamte Bohrstrang wird von einer Bohrflüssigkeit durchspült, die von der Basisstation mit Hilfe einer Pumpe in die Tiefe gepumpt wird. Diese kühlt den Bohrkopf während des Bohrvorganges und spült die zerspanten Gesteinsreste entlang der Außenwand des Bohrgestänges an die Oberfläche.

Nach der Erschließung eines Öl-Reservoirs kann der Bohrstrang aus dem Bohrloch gezogen werden und dessen Wand mit Beton stabilisiert werden. Anschließend kann mit der Förderung des erschlossenen Reservoirs begonnen werden. Dies geschieht anfangs über den Eigendruck

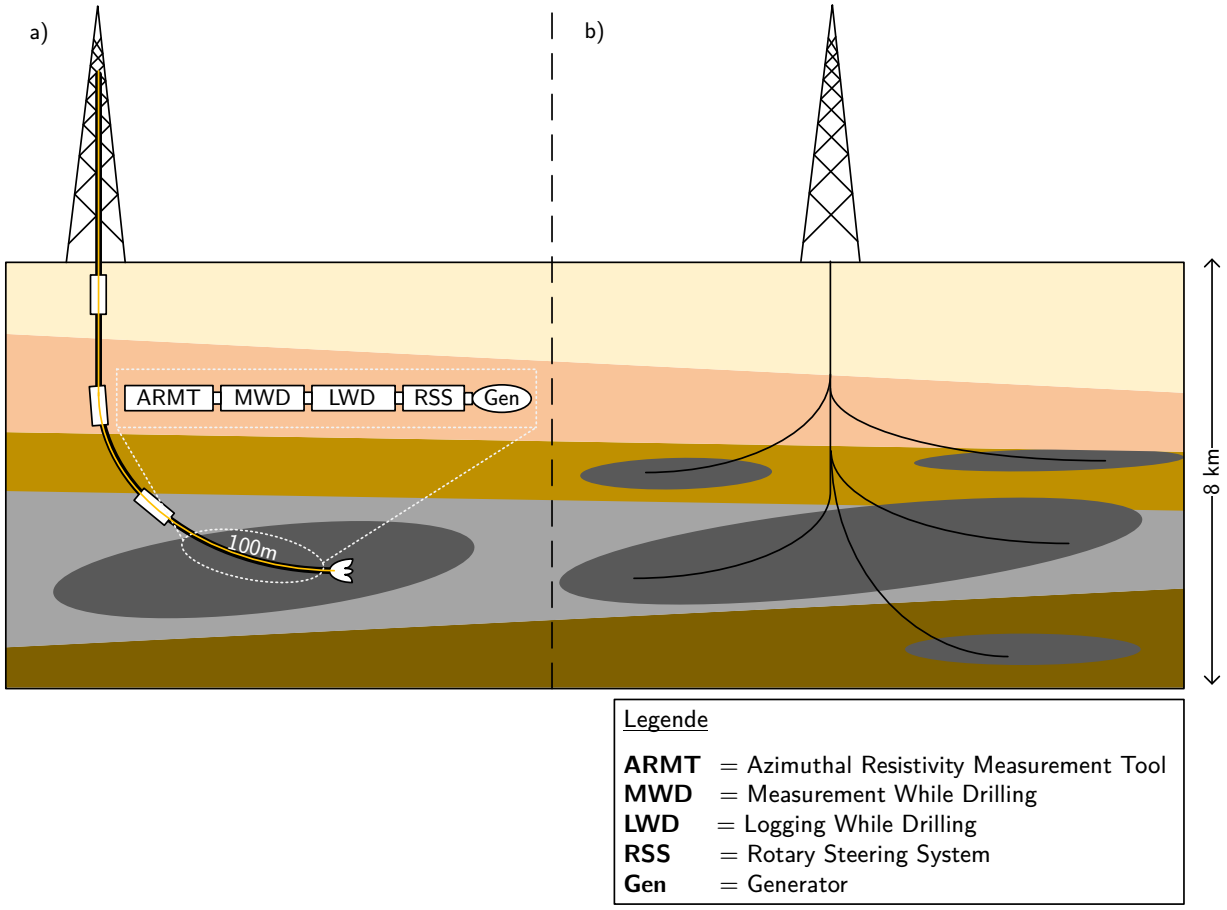

Abbildung 2.1: *Schematische Darstellung des Tiefbohrsystems durch verschiedene Erd- und Gesteinsschichten: in a) bei der Erschließung eines fossilen Reservoirs und in b) bei dessen anschließender Förderung.*

des Resevoirs und befördert etwa 10-15 % des Inhalts an die Erdoberfläche. Danach kann mit Tiefenpumpen der Druck im Reservoir künstlich erhöht und durchschnittlich weitere 18 % des Reservoirs gefördert werden. Mit Hilfe von Einpresssonden lassen sich dann nochmals 30-40 % des Resevoirs ausschöpfen, indem Wasser eingepresst und das Öl an die Oberfläche gespült wird, wo es dann wieder vom Wasser getrennt werden muss. Die restlichen 30-40 % des vorhandenen, zähen und dichten Öls lassen sich nur sehr aufwendig fördern und verbleiben meist ungefördert im Boden.

Die Umweltbedingungen unter Tage sind dabei abhängig von der Tiefe und der geografischen Position des Bohrlochs. Je geringer die Dicke des Erdmantels, desto schneller steigt die Temperatur pro Kilometer Tiefe. Näherungsweise kann eine Steigerung der Temperatur von 3 °C pro 100 m Tiefe angenommen werden [Zei56]. Die durchschnittliche Fördertiefe von Ölvorkommen lag 2008 bei 2371 m, was einem Temperaturanstieg von 71,13 °C entspricht [US 20]. Heutzutage werden auch Vorkommen in mehr als 4000 m Tiefe erschlossen, die eine Temperaturdifferenz von 120 °C und atmosphärische Drücke von bis zu 40 MPa aufweisen. Zusätzlich zu den geothermisch schwierigen Umweltbedingungen kommt es beim Zerspanen des Gesteins zu mechanischen Schocks und Vibrationen.

Entlang der letzten 100 m des Bohrstrangs befinden sich diverse Werkzeuge zum Vermessen der Umgebung und Steuern des Bohrkopfes. Je nach Tiefe, Art der Gesteinsschichten oder anderer

Umwelteinflüsse kann der Aufbau dieses System adaptiert werden. Unter anderem befindet sich dort ein Generator für die lokale Stromerzeugung, der von der Bohrflüssigkeit angetrieben wird. Weiter Komponenten können eine Steuereinheit für die Bohrrichtung (engl.: *Rotary Steering System*, RSS), eine Messeinheit für Widerstände an Schichtgrenzen (engl. *Azimuthal Resistivity Measurement Tool*, ARMT) oder diverse Log- und Messeinheiten (engl. *Logging/Measurement While Drilling*, LWD/MWD) sein [Chi18]. Alle diese Komponenten haben die Aufgabe, die Umgebung möglichst genau zu charakterisieren, um (teil-)autonom Entscheidung über den weiteren Bohrverlauf treffen zu können, da eine direkte Kommunikation mit der Basisstation mit nur mit maximal 20 bit/s möglich ist [Fra14]. Diese Kommunikation ist mit Hilfe eines Werkzeuges möglich, das unter Tage Druckpulse generiert, die als Schwingungen der Flüssigkeitsoberfläche der Basisstation durch optische Messverfahren dekodiert werden können.

Auf Grund dieser geringen Datenübertragungsraten ist es nicht möglich, eine zentrale Steuereinheit über Tage zu betreiben, die alle Komponenten unter Tage steuert. Die Auswertung von Sensorsignalen und die Vermessung der Umgebung wird aktuell nur lokal ausgewertet und die resultierenden Daten lokal gespeichert [Goo19]. Ausgelesen werden diese Informationen erst, wenn der Bohrstrang aus dem Bohrloch gezogen wurde. Eine mögliche Lösung von dedizierter Kommunikationsverdrahtung der Komponenten untereinander ist dabei nicht sinnvoll, da durch die Rotation des Bohrstrangs und die mechanischen Schocks die Wahrscheinlichkeit eines Kabeldefektes sehr groß ist. In diesem Fall müsste der gesamte Bohrvorgang abgebrochen und das defekte Teil ausgetauscht werden, was zu erheblichen Mehrkosten führt und in jedem Fall vermieden werden sollte [Kal14; Wat15]. Eine drahtlose Kommunikation ist ebenfalls nur schwer möglich, da die Schallausbreitung in Gestein und Flüssigkeiten, verglichen mit der Ausbreitung durch die Luft, eine erheblich größere Dämpfung erfährt und damit eine deutlich kürzere Reichweite hat [Sto97].

Die Idee in dieser Arbeit ist die Nutzung der bereits vorhandenen monolithischen Stromversorgungsleitung als Kommunikationsmedium, auch Powerline-Kommunikation genannt. Diese kann ein robustes Kommunikationsnetz mit einer Reichweite von bis zu 100 m realisieren. Nach dem heutigen Stand der Forschung gibt es kein veröffentlichtes Powerline-Modem, das für die oben beschriebenen extremen Umweltbedingungen entworfen oder evaluiert wurde. Dabei muss das Powerline-Kommunikation, neben den extremen Umweltbedingungen, die folgenden Anforderungen erfüllen:

Geringe Leistungsaufnahme: Durch den Generator steht nur ein begrenztes Energie-Budget zur Verfügung, das sich alle Komponenten aufteilen müssen.

Geringe Latenz: Steuerinformationen müssen schnell übertragen werden, um auf unerwartete Änderungen rechtzeitig reagieren zu können.

Gute Skalierbarkeit: Reihenfolge und Anzahl der Werkzeuge entlang des Bohrstrangs sind nicht festgelegt. Somit muss auch die Anzahl und Reihenfolge der Kommunikationsknoten adaptierbar sein.

Zusätzlich gibt es nach heutigem Stand der Forschung keine real gemessenen Übertragungskanäle für eine Powerline-Kommunikation unter den oben gegebenen extremen Umweltbedingungen. Es kann lediglich abgeleitet werden, dass die extremen Temperaturen hohes thermisches Rauschen verursachen und die Verbinder der Werkzeuge, ähnlich wie bei einer Gebäudeverdrahtung, starke Reflexionen durch Impedanzsprünge erzeugen. In dieser Arbeit wird zunächst von einem allgemein stark gestörten Mehrwege-Ausbreitungskanal für die Systemkonzeptionierung ausgegangen. Die

Auswahl eines geeigneten Powerline-Standards ist daher entscheidend und wird im Folgenden diskutiert.

2.2. Standards der Powerline-Kommunikation

Bei der Powerline-basierten Datenübertragung werden Informationen über bereits existierende Spannungsversorgungsleitungen übertragen. Dazu werden freie Frequenzbänder oberhalb der Netzfrequenz von bis zu 400 Hz verwendet, um Daten auf diese aufzumodulieren. Das Prinzip der Powerline-basierten Datenübertragung gibt es schon seit Beginn des 20. Jahrhunderts und wurde bereits 1922 für die Übertragung von Statussignalen verwendet. Hierbei wurde das Frequenzband von 15 kHz bis 500 kHz genutzt [Dos97].

Seit Beginn der 1990er-Jahre wurden Powerline-Kommunikationssysteme vermehrt für den Einsatz in privaten Haushalten entwickelt. Der Hauptgrund hierfür waren der Ausbau der Breitband-Internetverbindung und die Verdopplung der Leistungsfähigkeit von integrierten Schaltungen alle 18 Monate [Moo06]. Der Fokus lag auf der Übertragung von Informationen über die bereits existierende Spannungsversorgung des Hauses, da das nachträgliche Verlegen von Kommunikationsleitungen innerhalb der Wände sehr kostenintensiv ist. Dieser Grundgedanke ist bis heute geblieben, wobei aktuelle Powerline-Systeme Übertragungsgeschwindigkeiten mehr als 2 Gbit/s erreichen [Can16].

Eine Übersicht der aktuellen Standards für die Powerline-Kommunikation ist in Tabelle 2.2 gegeben. Prinzipiell unterscheidet man zwischen Powerline-Systemen mit Schmalband-basierter Kommunikation im Frequenzbereich unterhalb von 500 kHz und der Breitband-basierten Kommunikation oberhalb von 1 MHz. Die Schmalband-Powerline-Kommunikation hat, bedingt durch die niedrigeren Trägerfrequenzen, eine relativ große Wellenlänge und erreicht so eine Kommunikationsdistanz von mehreren Kilometern. Dabei ist sowohl die Komplexität der Hardware als auch die Datenrate von aktuell maximal 500 kbit/s gering verglichen mit der Breitband-Powerline-Kommunikation. Der schnellste Schmallband-Standard ist *Distribution Line Carrier* (DLC) mit 576 kbit/s und nutzt dafür die gesamte verfügbare Bandbreite. Anwendung finden diese Schmalbandsysteme vorwiegend in der Haussteuerung, dem sogenannten „Smart Home“, wie zum Beispiel bei der Steuerung von Heizungen und Rollläden oder auch bei der Fernablesung von Zählerständen durch den Energieversorger [Pap13; Mly15; Can16].

Breitband-Powerline-Kommunikationssysteme hingegen können in aktuellen Standards Geschwindigkeiten von mehr als 1 Gbit/s erreichen und werden vor allem für die Vernetzung von Multimedia-Geräten innerhalb des Hauses verwendet [Buc15]. Zu erkennen ist, dass die Datenrate und die Bandbreite mit den aktuelleren Standards ansteigt. Dabei verwenden alle hier aufgelisteten Breitband-Standards und die vier aktuellsten Schmalband-Standards PRIME, PLC-G3, ITU-T G.hnem und IEEE P1901.2 das Orthogonale Frequenzmultiplexverfahren (OFDM). Dieses teilt das vorhandene Frequenzband in äquidistante Träger, die separat moduliert werden können. Die Vorteile dieses Verfahrens sind seine Robustheit gegenüber Schmalbandstörungen, eine hohe Bandbreiten-Effizienz und adaptierbare Datenraten durch individuelle Modulation der Träger abhängig von der Kanalqualität [Pro83].

Diese Eigenschaften eignen sich gut für den Einsatz von Powerline-Kommunikation in der beschriebenen Tiefbohrtechnik. Die Datenrate eines OFDM-basierten Schmalband-Kommunikationsstandards mit 500 kbit/s wäre für wenige Teilnehmer entlang des Bohrstranges zwar

Tabelle 2.2: *Liste aktueller Powerline-Kommunikationsstandards für die breitbandige und schmalbandige Kommunikation mit ihren jeweiligen Bandbreiten und Durchsatzraten.*

	Name	Bandbreite	Durchsatz	
Schmalband	X10	120 kHz	20 bit/s	[Hae07]
	LonTalk	100–140 kHz	5,4 kbit/s	[Ech07]
	PRIME	42–89 kHz	128 kbit/s	[Hoc11]
	PLC-G3	10–488 kHz	298 kbit/s	[Ber13]
	ITU-T G.hnem	3–500 kHz	500 kbit/s	[Oks11]
	IEEE P1901.2	3–500 kHz	500 kbit/s	[Jim13]
Breitband	HomePlug 1.0/TIA-1113	4,5–21 MHz	10 Mbit/s	[Lee03]
	HomePlug Green PHY	1,8–30 MHz	10 Mbit/s	[Zyr11]
	HomePlug AV/IEEE 1901	1,8–30 MHz	200 Mbit/s	[Rah11]
	HD-PLC/IEEE 1901	1.8–28 MHz	240 Mbit/s	[Rah11]
	HomePlug AV2 SISO	1.8–86 MHz	600 Mbit/s	[Yon13]
	ITU-T G.9972	1.8–80 MHz	1200 Mbit/s	[Ber15]
	HomePlug AV2 MIMO	1.8–86 MHz	2000 Mbit/s	[Yon13]

ausreichend, durch dessen modularen Aufbau kann aber bei vielen Busteilnehmer die effektive Datenrate pro Nutzer sehr gering werden. Um das System flexibel und modular zu halten und auch zukünftige Bohrwerkzeuge mit höheren Datenraten zu unterstützen, wird in dieser Arbeit ein Breitband-Kommunikationsverfahren verwendet. Es wurde der HomePlug-1.0-Standard ausgewählt, da dieser verglichen mit den durchsatzstärkeren Standards eine geringe Hardware-Komplexität aufweist und die Datenrate von 10 Mbit/s ausreichend für das Tiefbohrsystem ist [Zyr11; Lee03].

2.3. Hochtemperaturfesten ASICs

Um den HomePlug 1.0-Standard in einem Tiefbohrsystem verwenden zu können, wird eine performante, hochtemperaturfeste Hardware-Einheit benötigt. Diese muss den Standard echtzeitfähig ausführen können, den oben beschriebenen extremen Umweltbedingungen standhalten und eine geringe Leistungsaufnahme besitzen.

Tabelle 2.3 zeigt eine Auswahl der performantesten, aktuell auf dem Markt verfügbaren Designs, die mindesten 150 °C Umgebungstemperatur standhalten können. Allgemein gibt es nur wenige Anbieter, die diese Designs herstellen und vertreiben. Als größte Hersteller in diesem Bereich sind die Firmen Texas Instruments, Honeywell und Microchip tätig. VORAGON und Tekmos haben sich speziell auf die Konvertierung existierender Design spezialisiert und verwenden dafür eine eigens entwickelte ASIC-Technologie(-Erweiterung).

Tabelle 2.3: *Auswahl der performantesten kommerziell erhältlichen oder wissenschaftlich veröffentlichten Designs mit einer Zieltemperatur von mindestens 150 °C. Die Tabelle ist nach der maximalen Temperatur sortiert.*

Hersteller	Typ	Temperatur	Technologiegröße	Taktrate
Microchip	16 bit RISC	150 °C	–	80 MHz
Texas Instruments	32 bit ARM9 + DSP	175 °C	–	300 MHz
VORAGO	32 bit ARM-M4	200 °C	–	100 MHz
Honeywell	32 kLE FPGA	200 °C	800 nm	–
Texas Instruments	16 bit DSP	200 °C	–	20 MHz
Texas Instruments	32 bit RISC	210 °C	–	100 MHz
Texas Instruments	32 bit DSP	220 °C	–	150 MHz
Honeywell	8 bit RISC	225 °C	800 nm	16 MHz
Honeywell	4 kByte EEPROM	250 °C	800 nm	–
Tekmos	8 bit RISC	250 °C	–	16 MHz

Die Gemeinsamkeit dieser Designs ist, dass die Technologiegröße (soweit diese angegeben ist) verglichen mit aktuellen CPU- oder GPU-Chips sehr groß ist. Einerseits sind die Leckströme bei großen Technologien geringer als bei kleinen Technologiegrößen, besonders bei hohen Temperaturen. Andererseits resultiert die Wahl einer größeren Technologiegröße auch in einer größeren Chipfläche und geringeren Taktrate [Kri96]. Dieser Effekt wird durch die erhöhte Elektronenbeweglichkeit bei steigender Temperatur noch verstärkt und die Länge des kritischen Pfades erhöht sich [Pet17]. Nur ein Design der in Tabelle 2.3 aufgeführten Chips erreicht eine Taktfrequenz von über 150 MHz. Dies ist der TI OMAPL137-HT, der mit einem 300 MHz-Takt betrieben werden kann und sowohl eine ARM-CPU als auch einen DSP besitzt.

Für die Implementierung eines Powerline-Kommunikationsstandards eignet sich keines der in Tabelle 2.3 gezeigten Designs. Die Gesamtperformance der CPU-basierten Komponenten ist durch die maximale Taktrate zu stark begrenzt und würde nicht ausreichen, um den komplexen Homeplug 1.0-Standard rein Software-basiert mit 10 Mbit/s zu betreiben. Vor allem die OFDM-basierte, parallele Verarbeitung kann nicht ohne Hardware-Erweiterungen auf einer CPU mit geringer Taktrate echtzeitfähig ausgeführt werden. Das rekonfigurierbare FPGA-Designs der Firma Honeywell wäre zwar geeignet, um Teile dieser Verarbeitungskette parallel zu berechnen, ist aber kommerziell nicht erhältlich und mit 32 kLE auch vergleichsweise klein gegenüber modernen FPGAs. Deshalb wird in dieser Arbeit für die Implementierung des Homeplug 1.0-Standards der Designraum verschiedener Hardware-Plattformen für die Fertigung in einer modernen Hochtemperatur-ASIC-Technologie untersucht. Dabei sollte diese ASIC-Technologie möglichst hohen Temperaturen standhalten können und gleichzeitig eine kleine Technologiegröße haben, um hohe Taktraten und kleine Chipflächen zu erreichen.

Tabelle 2.4 gibt einen Überblick über aktuell verfügbare ASIC-Technologien für extreme Temperaturbereiche und deren zugrundeliegende Siliziumstruktur. Im Vergleich zu aktuellen Konsumerelektronik-Technologien sind die Strukturgrößen dieser Prozesse sehr groß. Dies liegt vorwiegend an der oben erwähnten Reduzierung der Leckströme und der Verwendung einer zusätzlichen *Silicon-On-Insulator* (SOI) Sperrschicht, die die Leistungsaufnahme durch Leckströme

Tabelle 2.4: *Aktuelle Anbieter und Strukturgrößen von ASIC-Technologien für extreme Temperaturen oberhalb von 150 °C. Mit steigender Temperatur steigt prinzipiell auch die Strukturgröße.*

Name	Größe	Technologie	max. Temperatur	Referenz
XFAB XT018	180 nm	SOI-CMOS	175 °C	[Hao15]
Tekmos XA35	350 nm	SOI-CMOS	175 °C	[Tek19]
Honeywell HTMOS	800 nm	SOI-CMOS	225 °C	[Hon21]
Tekmos XI10	1000 nm	SOI-CMOS	225 °C	[Tek19]
Frauenhofer H035	350 nm	SOI-CMOS	300 °C	[Bol21]
VORAGON HardSil	–*	Bulk-CMOS	300 °C	[VOR16]

*HardSil ist keine eigenständige Technologie, sondern eine Erweiterung bestehender Bulk-Technologien um einen Schutz vor Latch-up-Effekten.

reduziert und die Schaltzeiten verkürzt [Sha99]. Zu erkennen ist auch hier, dass die maximale Betriebstemperatur der Technologien mit steigender Strukturgröße zunimmt. Eine Ausnahme bildet die Frauenhofer H035-Technologie, die bei nur 350 nm Strukturgröße Temperaturen von 300 °C widerstehen kann. Die von der Firma VORAGON bereitgestellte HardSil-Technologie ist keine eigenständige ASIC-Technologie, sondern eine Erweiterung existierender Fertigungsprozesse, um den Chip temperaturfest zu machen. Da diese Erweiterung vor allem auf Bulk-CMOS-Technologien aufbaut und dafür die Anschaffung dieser Bulk-Technologie und die Fertigung und Erweiterung durch die HardSil-Technologie notwendig ist, wird aus Kosten- und Komplexitätsgründen auf diese Technologieerweiterung verzichtet.

Auf Grund der geringen Strukturgröße wurde in dieser Arbeit die XFAB XT018-Technologie gewählt. Diese hat mit 180 nm die geringste Strukturgröße, also auch somit den geringsten Platzbedarf. Die maximale Temperatur von 175 °C ist ausreichend für die in dieser Arbeit betrachteten Tiefen der Bohrtechnik. Die Versorgungsspannung ist mit 1,8 V niedriger, als bei den anderen oben aufgelisteten Technologien und sollte somit eine geringere dynamische Verlustleistungsaufnahme und Selbsterhitzung aufweisen.

Nachfolgend wird in Kapitel 3 zunächst der HomePlug-1.0-Standard vorgestellt und anschließend in Kapitel 4 der Hardware-Entwurfsraum für die Chipfertigung des Powerline-Systems in der XT018-Technologie untersucht.

KAPITEL 3

Entwurf des Powerline-Modems

In diesem Kapitel wird der Entwurf eines Powerline-Modems nach dem HomePlug-1.0-Standard präsentiert und evaluiert. Dabei gilt es, die in Kapitel 3 vorgestellten Randbedingungen der Tiefbohrtechnik zu berücksichtigen:

- Geringe Verlustleistung durch begrenztes Energiebudget
- Starke Störungen auf dem Übertragungskanal durch Mehrwegeausbreitung
- Hohes thermisches Rauschen

Diese Randbedingungen haben dabei Einfluss auf die Wahl eines Synchronisationsalgorithmus oder die Fehler-Korrekturkapazität der Datendecodierung.

Zunächst wird in Abschnitt 3.1 der HomePlug-1.0-Standard zusammenfassend vorgestellt und einen Überblick über den Kommunikationsablauf durch die MAC- und PHY-Schicht gegeben. Abschnitt 3.2 stellt Algorithmen für den nicht näher im Standard spezifizierten PHY-Schicht-Decodierung nach dem aktuellen Stand der Forschung vor. Anschließend wird in Abschnitt 3.3 eine MATLAB-Referenzimplementierung anhand des Standards und der am besten geeigneten Algorithmen für den Decoder erstellt und verifiziert. Abschließend wird eine Entwurfsraumexploration der algorithmischen Parameter für den Einsatz in der Tiefbohrtechnik durchgeführt.

3.1. HomePlug-1.0-Standard

Der HomePlug-1.0-Standard wurde 2001 von der HomePlug Powerline Alliance veröffentlicht und wird seit 2008 von der Telecommunication Industry Association (TIA) als TIA-1113 bereitgestellt [Hom08]. Bestandteil des HomePlug-1.0-Standards sind die untersten beiden Schichten des *Open System Interconnection*-Modells (OSI-Model) nach Zimmermann [Zim80]. Abbildung 3.1 zeigt den Aufbau des OSI-Modells und die Kommunikation der insgesamt sieben Schichten.

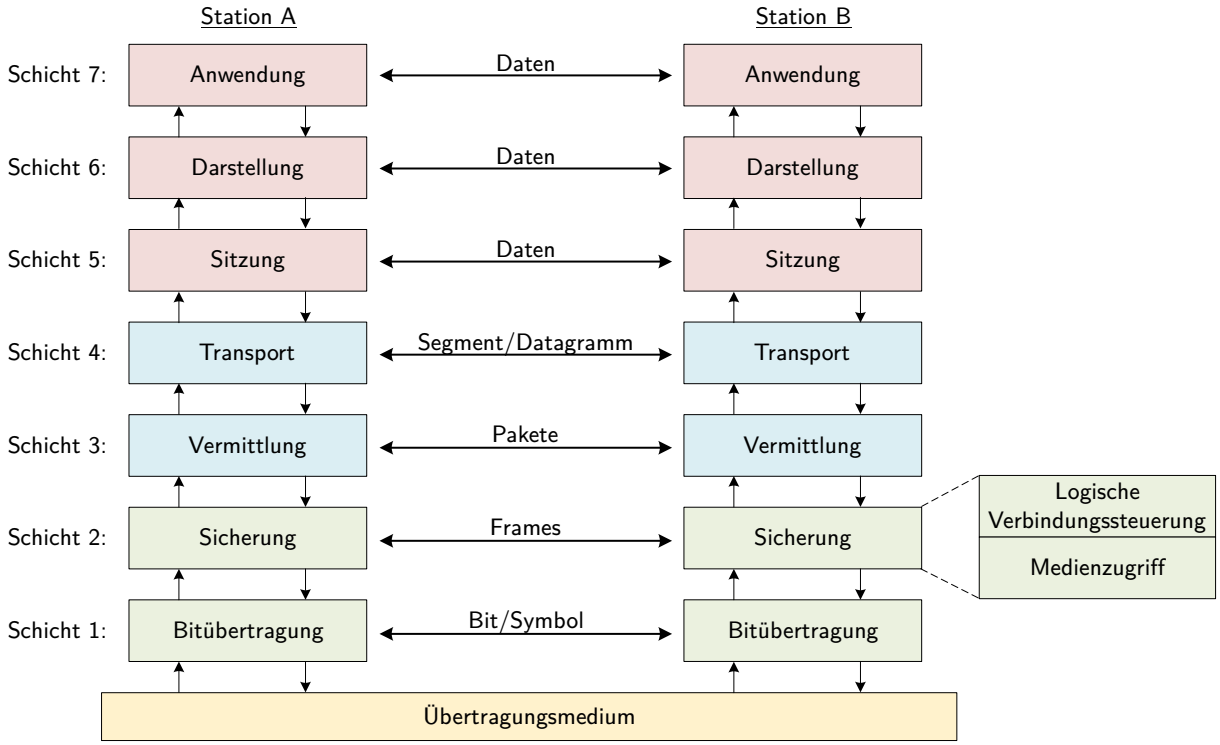

Abbildung 3.1: *OSI-Schichtenmodell nach Zimmermann [Zim80]. Jede Schicht verarbeitet Daten vertikal und kommuniziert horizontal mit der identischen Schicht der Gegenseite. Rechts neben der Sicherungsschicht ist dessen Aufteilung in zwei Teilschichten gezeigt. Die Namen der horizontalen Pfeile geben die Bezeichnung der ausgetauschten Informationen dieser Schicht an.*

Jede Schicht kommuniziert horizontal mit der entsprechenden Gegenstelle und gibt die zu versendenden Daten an die direkten Nachbarschichten weiter oder erhält Daten von diesen. Das heißt, dass eine Schicht keine Kenntnis über den internen Aufbau der anderen Schichten hat und unabhängig von diesen arbeitet. Die Namen auf den horizontalen Pfeilen sind die Bezeichnungen der Informationen, die in dieser Schicht ausgetauscht werden. Die Sicherungsschicht (engl. *Data Link Layer*, DLL) unterteilt sich in die Teilschichten der logischen Verbindungssteuerung (engl.: *Logical Link Control*, LLC) und des Medienzugriffs (engl.: *Medium Access Control*, MAC). Der HomePlug-1.0-Standard spezifiziert die Bitübertragungsschicht (engl. *Physical Layer* PHY), die MAC-Schicht und die Verbindungen zur LLC-Schicht.

Die MAC-Schicht hat die Aufgabe, den Datenfluss zu kontrollieren, Pakete zu vermitteln und Kollisionen zu vermeiden. Zusätzlich erzeugt sie Steuer- und Codierinformationen, die zusammen mit den Nutzdaten an die PHY-Schicht übergeben werden. Der Aufbau und die Funktionsblöcke dieser Schicht werden im Abschnitt 3.1.1 detailliert beschrieben. Die PHY-Schicht hat die Aufgabe, Informationsbits in ein physikalisches analoges Signal zu codieren. Hierfür werden dem Signal Redundanzen hinzugefügt, um Übertragungsfehler am Empfänger erkennen und korrigieren zu können. Ein für die Arbeit besonders interessanter Betriebsmodus ist der sogenannte „Robuste OFDM“-Modus (engl.: *Robust OFDM*, ROBO), der eine Kommunikation über einen stark gestörten Übertragungskanal ermöglicht, der bei herkömmlicher Modulation keine Kommunikation erlauben würden. Eine detaillierte Beschreibung dieser Modulation und der Aufbau der PHY-Schicht folgt in Abschnitt 3.1.2.

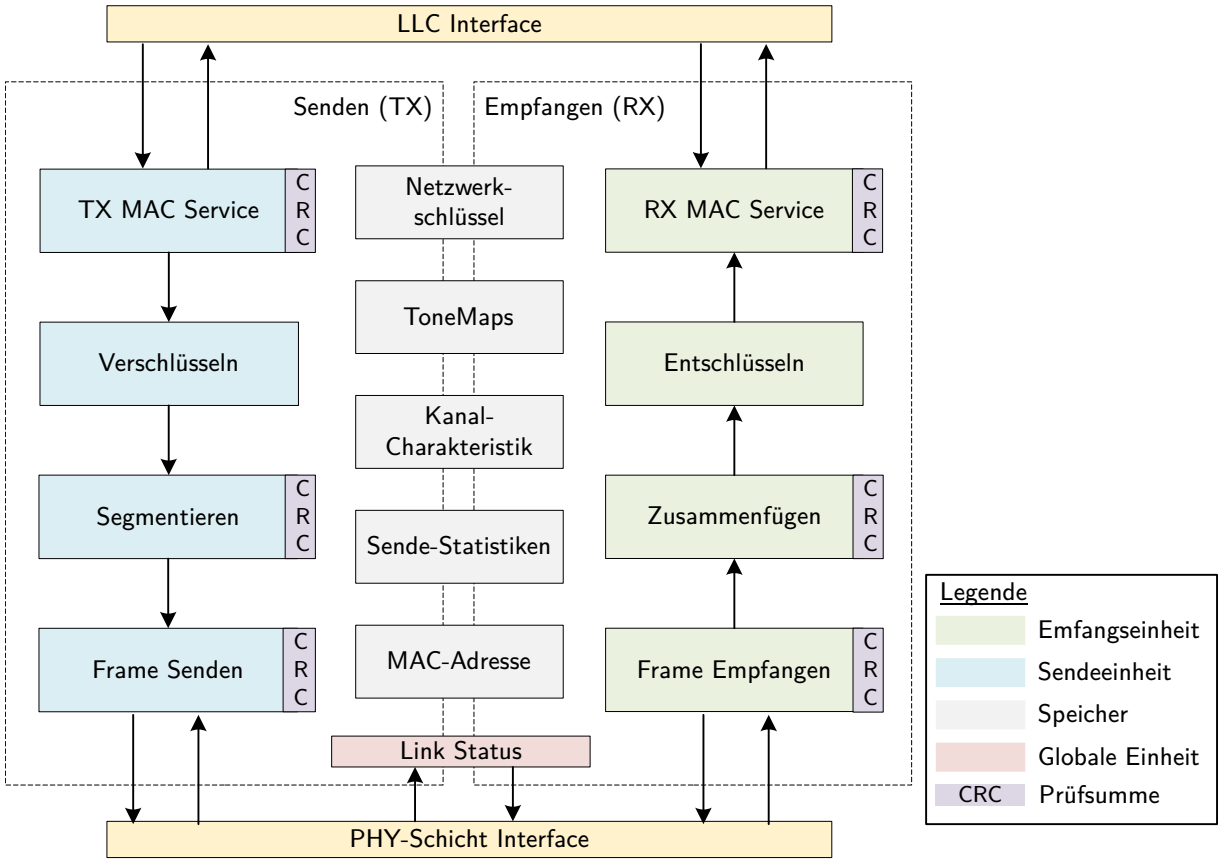

Abbildung 3.2: *Aufbau der MAC-Schicht nach dem HomePlug-1.0-Standard. Die Verarbeitung des Sendepfades (blau) erfolgt von der LLC-Schicht (oben) zur PHY-Schicht (unten) und vice versa für den Empfangspfad (grün). In grau sind die gemeinsam genutzten Speicher gezeigt [Hom08].*

3.1.1. MAC-Schicht

Die MAC-Schicht des HomePlug-1.0-Standards hat die Aufgaben der Paketvermittlung, Kollisionsvermeidung, Paketpriorisierung, Verschlüsselung, Segmentierung und Steuerung des Medienzugriffs. Abbildung 3.2 zeigt die im Standard definierten Verarbeitungsblöcke, die für das Senden und Empfangen eines Paketes zu implementieren sind [Lee03]. Insgesamt gibt es vier Teilbereiche je Sende- und Empfangsteil, die das Paket senderseitig encodieren und empfangsseitig decodieren. Die interne Struktur dieser Verarbeitungsblöcke und deren Kommunikation untereinander sind als Pseudocode im Anhang des Standards gegeben und als Ablaufdiagramme im Anhang C dieser Arbeit beschrieben.

Zwischen den Verarbeitungsblöcken in Abbildung 3.2 sind in grau die gemeinsam verwendeten Speicherblöcke gegeben. Diese enthalten die Netzwerkschlüssel zur Verschlüsselung der Daten, die eigene MAC-Adresse, Sende- und Empfangsstatistiken und die aktuellen Informationen über die Qualität des physikalischen Übertragungskanals (ToneMaps). Als globales Modul ist in rot der Link-Status abgebildet, der kontinuierlich überprüft, ob eine Verbindung zum Netzwerk existiert.

Nachfolgend wird getrennt auf den Ablauf des Sendens und Empfangens eines Paketes durch die MAC-Schicht eingegangen. Dafür dient Abbildung 3.2 als Grundlage.

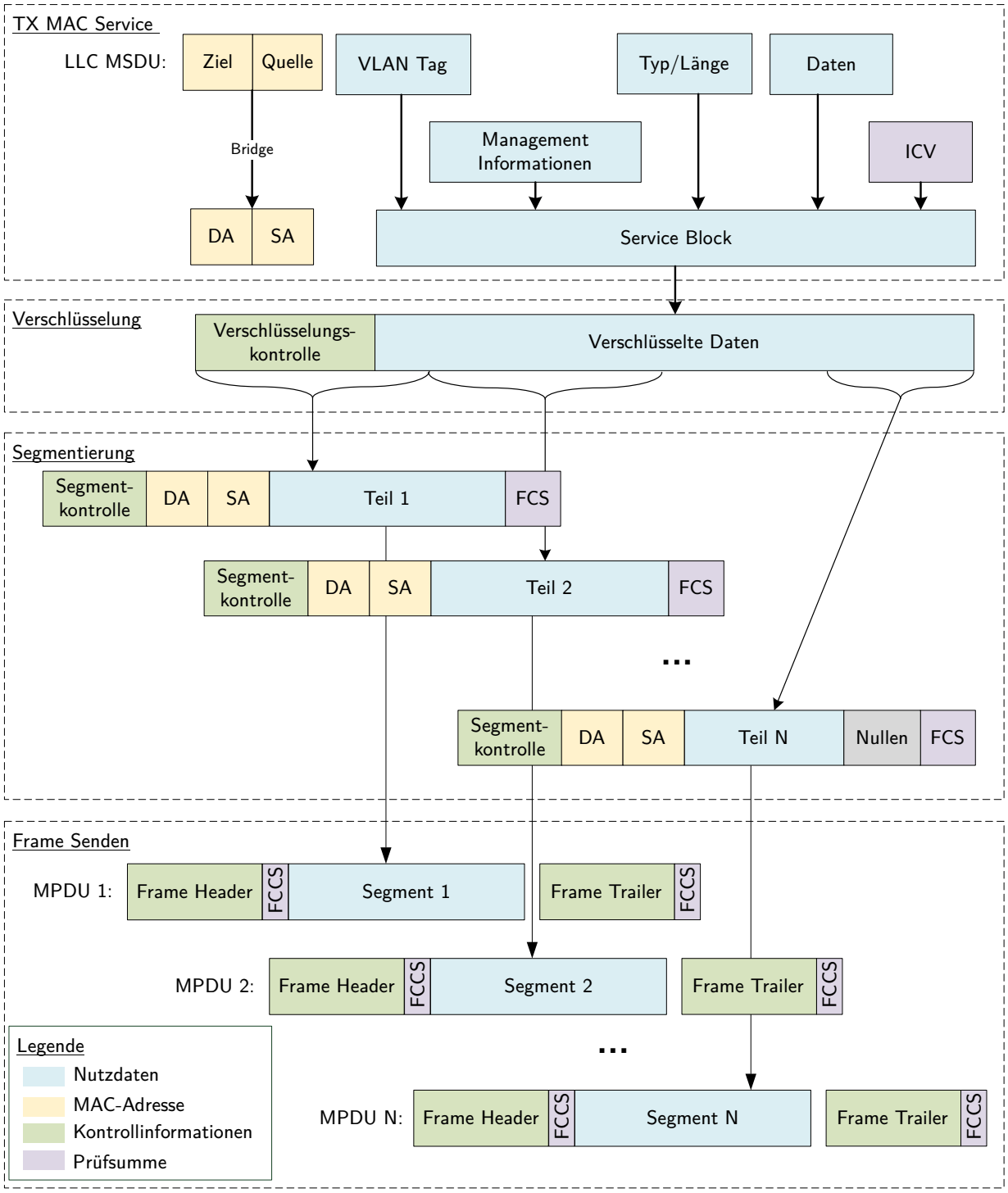

Abbildung 3.3: *Verarbeitung der Sendedaten (MSDU) durch die vier MAC-Schicht-Blöcke des TX-Pfades nach Abbildung 3.2. Die Übergabe der Daten an die PHY-Schicht erfolgt als eine oder mehrere Frames (MPDU) [Hom08].*

Sender

Der Ablauf und die Encodierung eines Paketes auf der Sendeseite ist in Abbildung 3.3 dargestellt. Startpunkt ist ein MAC-Service-Datenpaket (engl.: *MAC Service Data Unit*, MSDU), das über das

LLC-Interface an den TX-MAC-Service übergeben wird. Als Erstes werden alle Teilinformationen zu einem Service-Block zusammengesetzt. Dabei werden Management-Information für die MAC-Schicht des Empfängers hinzugefügt und die Itegritätsprüfsummme (engl. *Integrity Check Value*, ICV) über den gesamten Service-Block gebildet. Diese Prüfsumme berechnet sich mit Hilfe des Generator-Polynoms G_{ICV} nach Gleichung 3.1 und hat eine Länge von 32 bit.

$$G_{\mathrm{ICV}}(x) = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1 \quad (3.1)$$

Zuletzt werden noch die Quell- und Zieladressen überprüft und eventuell überbrückt (engl. *Bridge*). Damit kann die Station neben Paketen für die eigene MAC-Adresse auch Pakete für weitere MAC-Adressen des darüber liegenden Hardware-Systems empfangen und an die LLC-Schicht weitergeben. Zum Beispiel kann das Powerline-System als Brücke für eine *Ethernet*-Verbindung arbeiten und dabei die *Ethernet*-MAC-Adressen überbrücken. Dazu werden *Bridge*-Listen ausgetauscht, bei denen jede Station mitteilt, welche MAC-Adressen, außer der eigenen, im System bekannt sind.

Der gesamte Service-Block wird anschließend der Verschlüsselung übergeben. Dabei wird von der LLC-Schicht festgelegt, ob eine Verschlüsselung durchgeführt werden soll und mit welchem Netzwerkschlüssel dies geschehen soll. Als Verschlüsselungsverfahren verwendet der Standard den 56 bit *Data Encryption Standards* (DES). Dieser arbeitet blockweise auf 64 bit großen Datenblöcken und verwendet den *Cypher Block Chaining*-Modus (CBC). Dieser Modus verwendet die Ausgangsdaten eines Blockes, um die Eingangsdaten des nächsten Blockes per Exklusiv-Oder-Funktion zu chiffrieren. Zu Beginn dieser Kette wird ein zufälliger 64 bit Initialisierungsvektor (engl.: *Inititialization Vector*, IV) verwendet, um den ersten Block zu chiffrieren. Dieser IV wird im Klartext als Teil der vorangestellten Verschlüsselungskontrolle übertragen. Zusätzlich enthält diese Kontrolle die Informationen, ob das Paket verschlüsselt ist und welcher Netzwerkschlüssel dafür verwendet wurde [Bur01; Bel94].

Die Segmentierung überprüft zuerst, ob die verschlüsselten Daten inklusive Kontrollinformationen in einem Segment übertragen werden können oder ob die Daten in mehrere Segmente segmentiert werden müssen. Die Größe dieser Segmente ist abhängig von den Codierparametern der ausgehandelten *ToneMap*, die Modulationsrate, die Anzahl verwendbarer Frequenzträger und die Menge der verwendeten Redundanzen enthält. Diese *ToneMap* zeigt an, welche Frequenzträger, abhängig von der aktuellen Kanalschätzung, für die Übertragung genutzt werden können. Diese *ToneMap* wird zwischen zwei Stationen ausgehandelt und ist eine Untermenge der *ToneMask*. Die Größe der Segmente liegt zwischen 152 und 2076 Byte pro Segment für reguläre Modulationen und zwischen 40 und 172 Byte für den ROBO-Modus.

Auch die Segmentierung erstellt pro Segment eine Segmentkontrolle, die zusammen mit der Zieladresse (engl. *Destination Address*, DA) und der Quelladresse (engl. *Source Address*, SA) vorangestellt wird. Diese Adressen werden vom MAC-Service-Block kopiert und sind für alle Segmente identisch. Die Segmentkontrolle bestehend aus einer Sequenznummer, einem Segmentzähler, der Segmentlänge, einem *Flag* für das letzte Segment der Sequenz und weiteren Steuerinformationen. Alle Segmente einer Übertragung haben die gleiche Sequenznummer und einen fortlaufenden Segmentzähler pro Segment. Das letzte Segment wird eventuell mit Nullen aufgefüllt, bis die erforderlichen Segmentlänge erreicht ist (*Block-Padding*, BPAD). Über jedes Segment wird abschließend die Frame-Prüfsumme berechnet (engl. *Frame Check Sequence*,

FCS). Diese ist 16 bit lang und berechnet sich mit Hilfe des Generator-Polynoms G_{FCS} nach Gleichung 3.2.

$$G_{\mathrm{FCS}}(x) = x^{16} + x^{12} + x^5 + 1 \tag{3.2}$$

Der Frame-Senden-Block generiert für jedes Segment zwei 25 bit lange Framekontrollen, einen *Frame-Header* und einen *Frame-Trailer*. Diese enthalten Steuerinformationen, die für die Synchronisierung des ablaufenden Protokolls zur Kollisionsvermeidung und die Decodierung auf PHY-Schicht-Ebene am Empfänger relevant sind. Auch die Framekontrollen enthalten eine Prüfsumme, die als *Frame Control Check Sequence* (FCCS) bezeichnet wird. Jede FCCS ist 8 bit lang und berechnet sich mit Hilfe des Generator-Polynoms G_{FCCS} in Gleichung 3.3 Die resultierenden Frames werden als *MAC Protocol Data Unit* (MPDU) bezeichnet und als einzelne Frames der PHY-Schicht übergeben.

$$G_{\mathrm{FCCS}}(x) = x^8 + x^2 + x + 1 \tag{3.3}$$

Um diese MPDUs senden zu können, muss der Frame-Senden-Block entsprechend des Kanalzugriffsprotokolls die Prioritätsaushandlung gewinnen und die niedrigste zufällige Wartezeit ziehen. Das in diesem Standard verwendete Protokoll ist der Mehrfachzugriff mit Trägerprüfung und Kollisionsvermeidung (engl. *Carrier Sense Multiple Access/Collision Avoidance* CSMA/CA) und zusätzlicher Prioritätssignalisierung. Bei diesem wird der physikalische Kanal kontinuierlich abgehört und dabei geprüft, ob eine Station gerade Daten überträgt oder übertragen will. In Abbildung 3.4 sind drei typische Sequenzen einer Paketübertragung nach diesem Protokoll dargestellt.

Sequenz a) zeigt den Ablauf der Prioritätsaushandlung und der Kollisionsvermeidung nach der Übertragung eines Frames ohne erwartete Antwort. Die Zeit zwischen dem Ende einer Übertragung und der Prioritätsaushandlung wird als *Contention Interframe Space* (CIFS) bezeichnet. Danach signalisieren alle Stationen, die ein Paket übertragen wollen, ihre Priorität. Diese kann Werte zwischen 0 – 3 annehmen und wird von der LLC-Schicht vorgegeben. Die Ausnahme bilden intern generierte Managementpakete, wie beispielsweise die Aushandlung von Kanalübertragungsparametern, die eine vom Standard vorgegebene Priorität haben.

Anschließend ziehen in der *Backoff*-Phase alle Stationen der aktuell höchsten Priorität eine Zufallszahl N aus einem Zufallszahlenfenster (engl. *Contention Window*, CW). Die Größe dieses Fensters ist dabei abhängig von der Priorität und der Anzahl vorangegangener Kollisionen. Das minimale Fenster ist $N \in [0,7]$ und das maximale $N \in [15,63]$, wobei N die Anzahl der 35,84 µs langen *Slot*-Zeiten angibt. Während dieser Wartezeit scannen die Stationen kontinuierlich das Übertragungsmedium, um festzustellen, ob eine andere Station mit einer kleineren Zufallszahl bereits angefangen hat, Daten zu übertragen. Ist dies der Fall, wird die aktuelle Anzahl noch zu wartender *Slots* gespeichert und für die nächste *Backoff*-Phase verwendet. Dies stellt sicher, dass Stationen mit einer hohen *Slot*-Zahl nicht dauerhaft von anderen Stationen überholt werden können.

Die anderen beiden Sequenzen in Abbildung 3.4 zeigen beide eine Übertragung mit erwarteter Antwort, was durch die *Frame-Header* und *Trailer* signalisiert wird. In Abbildung 3.4 b) ist eine erfolgreiche Paketübertragung mit Antwort des Empfängers gezeigt. Dazu gibt es im Ablauf zusätzlich die 26 µs lange *Response Interframe Space*-Zeit (RIFS) zwischen dem Ende der

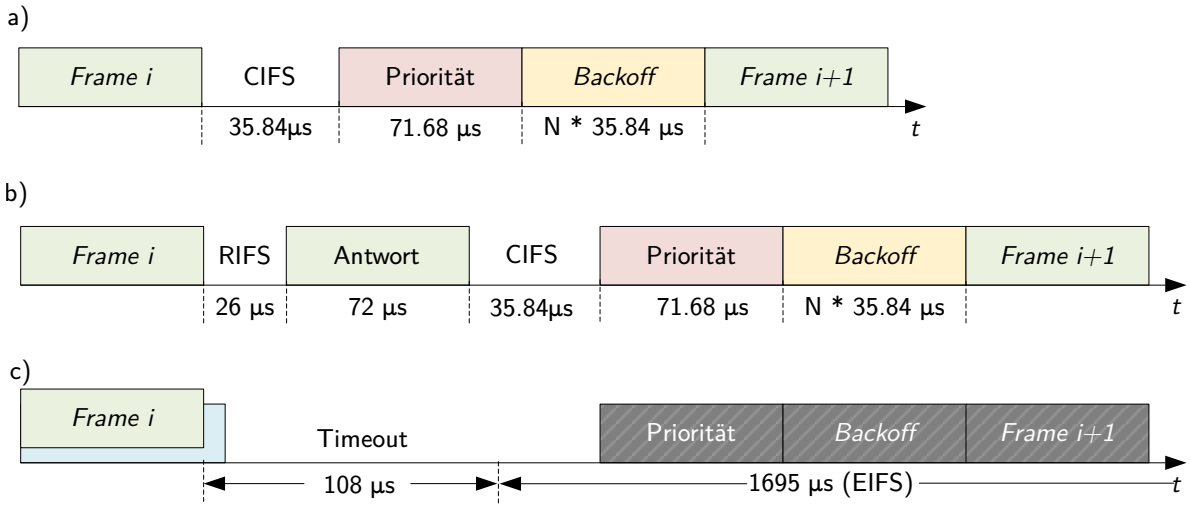

Abbildung 3.4: *Sequenz konsekutiver Paketübertragungen. In a) ohne erwartete Antwort des Empfängers, in b) mit erwarteter Antwort und in c) mit Paketkollision auf dem Übertragungskanal bei erwarteter Antwort. Die in dunkelgrau dargestellten Blöcke zeigen, dass die Station diese nur beobachtet und nicht aktiv daran teilnimmt [Hom08].*

Übertragung und der dazugehörigen Antwort. Die Antwort ist 72 µs lang und enthält eine 25 bit Framekontrolle, die den Typ der Antwort und ebenfalls eine FCCS-Prüfsumme enthält.

Die letzte Sequenz in Abbildung 3.4 c) zeigt eine Kollision zweier Pakete auf dem Kanal, die nur bei einer erwarteten Antwort detektiert werden kann. Enthält der Sender innerhalb von 108 µs keine Antwort, wird dies als Kollision behandelt, bei der entweder zwei Pakete kollidiert sind oder die Antwort verloren gegangen ist. In diesem Fall wartet der Sender eine 1685 µs lange *Extended Interframe Space*-Zeit (EIFS) bis er erneut versucht das Paket zuzustellen. Während dieser Zeit beobachtet die Station den Kanal, um sich mit dem Protokoll auf dem Kanal zu synchronisieren (Abbildung 3.4 in dunkelgrau).

Ob ein Sender die Antwort des Empfängers erwartet, ist durch ein *Flag* im *Header* und *Trailer* angezeigt. Die Verwendung von Antworten kann für jedes Datenpaket von der LLC-Schicht eingestellt werden und sollte je nach Applikation angepasst werden. Beispielsweise kann es sinnvoll sein, für verbindungsorientierte Anwendungen auch auf MAC-Ebene eine gesicherte Übertragung herzustellen, wohingegen bei verbindungslosen Anwendungen auf MAC-Ebene keine Antwort der Gegenstelle erwartete werden sollte. Allgemein ist dies ein Abtausch der maximalen Datenrate gegenüber der Paketverlustrate.

Empfänger

Empfängt die PHY-Schicht ein Paket, wird dieses durch die in Abbildung 3.3 gezeigten RX-Blöcke in inverser Reihenfolge zu den TX-Blöcken decodiert. Zunächst werden die FCCS- und die FCS-Prüfsummen überprüft, indem dieselbe Prüfsummenberechung wie auf Sendeseite durchgeführt wird, inklusive der Prüfsumme des Senders selbst. Dies wird auch als Zyklische Redundanzprüfungen (engl.: *Cyclic Redundancy Check*, CRC) bezeichnet. Sind Fehler in einem Segment enthalten, wird diese durch die Empfängerprüfsumme angezeigt und das Paket muss vom Sender (eventuell) erneut angefordert werden. Dazu verwendet der Standard die automatische

Wiederholungsanfrage (engl.: *Automatic Repeat Request*, ARQ) ohne Kumulation von Antworten (*Stop-and-Wait*), so dass nach jedem Paket auf eine Antwort gewartet wird. Insgesamt gibt es drei verschiedene Antworten, die ein Sender erhalten kann:

- *Acknowledge* (ACK): Paket erfolgreich empfangen und FCS war korrekt.
- *Negative ACK* (NACK): Paket erfolgreich empfangen, aber FCS war inkorrekt.
- *Failure* (FAIL): Empfänger hat keine Kapazitäten das Paket anzunehmen.

Wird ein ACK empfangen, kann der Sender sofort das nächste Paket versenden. Beim Erhalt eines NACK wird dasselbe Paket sofort zum erneuten Versenden bereitgestellt. Bei einem FAIL wartet der Sender 10 ms, bevor er erneut versucht das Paket zuzustellen (sofern es sich um das erste oder einzige Segment eines Paketes handelt).

Sind die FCCS- und FCS-Prüfsumme korrekt, wird das Teilsegment in einem Pufferspeicher zusammengesetzt. Mit einem *Last Segment Flag* in der Segmentkontrolle wird dem Empfänger signalisiert, dass alle Teile der Sequenz empfangen wurden und die Daten der Entschlüsselung übergeben werden können. Diese entschlüsselt das Paket anhand der Informationen in der Verschlüsselungskontrolle und gibt die Klartextinformationen an den RX-MAC-Service-Block weiter. Als letzter Schritt überprüft dieser die ICV-Prüfsumme und extrahiert die Management-Informationen des Paketes. Der Austausch von diesen Management-Informationen geschieht mittels MAC-Management-Einträgen (engl. *MAC Management Entries*, MME), die den Nutzdaten vorangestellt übertragen werden (vgl. Abbildung 3.3). Dabei werden die folgenden neun Einträge im Standard unterstützt:

- Kanalschätzungs-Anfrage/-Antwort
- *Multicast*-Paket mit Antwort
- *Bridge*-Adresse ersetzen
- Sendestatistik-Anfrage/-Antwort
- Netwerkschlüssel ersetzen (und bestätigen)
- Herstellerspezifische Anfragen

Die ersten vier genannten Einträge sind MAC-interne MMEs, die nicht an die LLC-Schicht weitergegeben oder von dieser angefordert werden können. Die Kanalschätzung wird verwendet, um die Codierungsparameter für die PHY-Schicht auszuhandeln, die unter anderem die Größe eines zu übertragenden Paketes festlegen (siehe oben). Der *Multicast*-Paket-MME gibt die Zieladresse des Empfängers an, der bei einer *Multi-* oder *Broadcast*-Übertragung auf das Paket antworten soll. Der *Bridge*-Adressen-MME enthält die Ziel- und Quelladressen der originalen Sender- und Empfänger-MAC-Adressen. Die letzten drei Einträge werden als Serviceanfrage von der LLC-Schicht gestellt und können entweder an die Station selbst oder eine andere Station im Powerline-Netz gerichtet sein. Hier können die Sende- und Empfangs-Statistiken einer Station ausgelesen oder ein neuer, für die Verschlüsselung notwendiger, Netzwerkschlüssel festgelegt werden.

Zusätzlich lassen sich noch herstellerspezifische Anfragen implementieren, die sowohl intern als auch extern Informationen verarbeiten können. Diese Anfragen beginnen immer mit der eindeutigen Herstellerkennung (engl.: *Organizationally Unique Identifier*, OUI), um erkennen zu können, ob eine Station diese Anfrage unterstützt oder ignoriert.

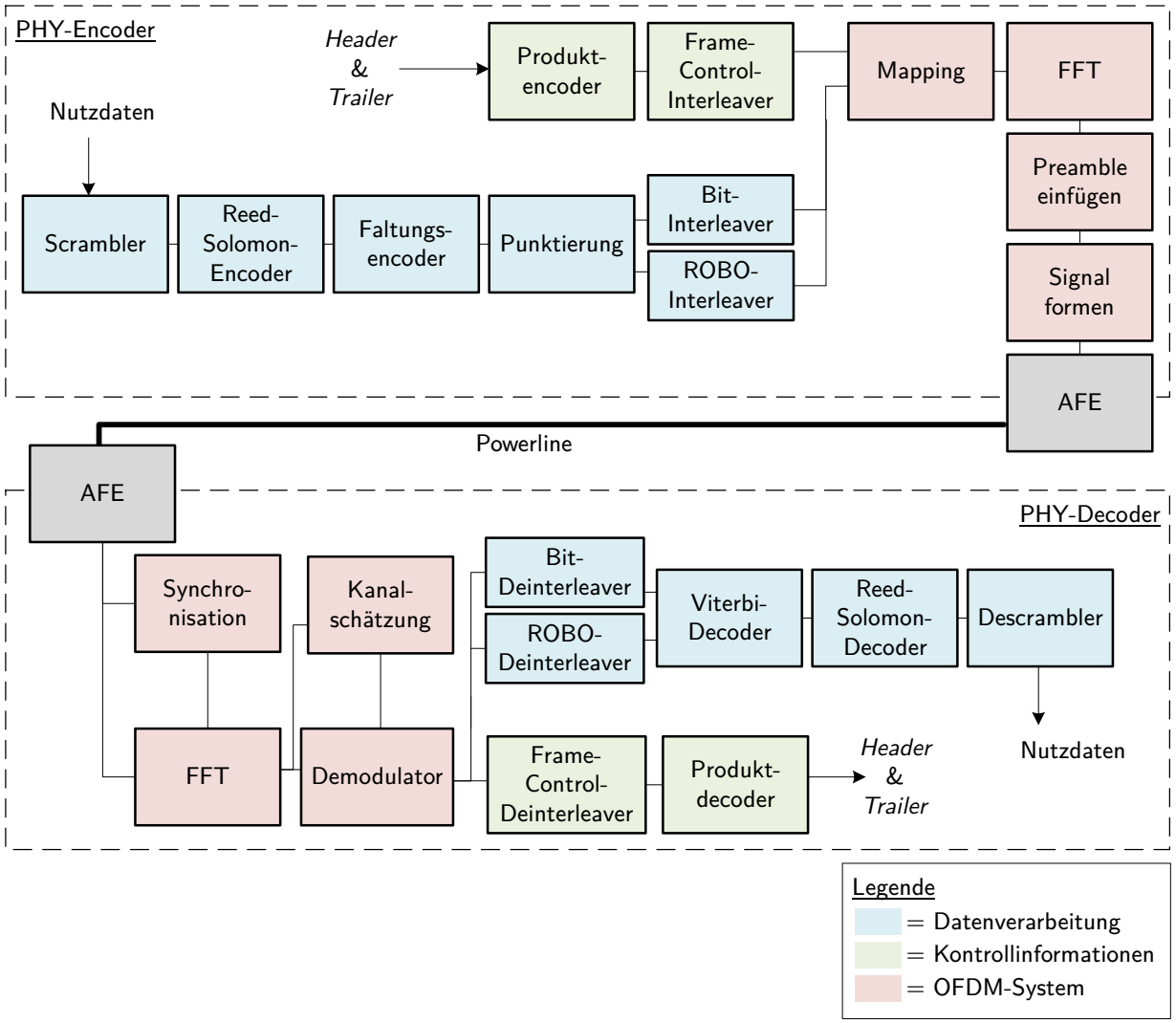

Abbildung 3.5: *Blockdiagramm der PHY-Schicht unterteilt in Encoder (oben) und Decoder (unten). Die AFEs (grau) sind durch eine physikalische Powerline verbunden [Hom08].*

3.1.2. PHY-Schicht

Die Aufgaben der PHY-Schicht des HomePlug-1.0-Standards sind die Encodierung von MPDUs als analoges, kontinuierliches Signal auf dem Powerline-Bus und die Decodierung empfangener Signale zurück in digitale Informationen. Dabei werden bei der Encodierung der Kontroll- und Nutzdaten Redundanzen hinzugefügt, die am Empfänger genutzt werden, um Fehler bei der Übertragung zu korrigieren. Die Verarbeitungsblöcke und deren Reihenfolge ist durch den HomePlug-1.0-Standard vorgegeben und in Abbildung 3.5 dargestellt. Der obere Teil der Abbildung zeigt den Encoder und der untere den Decoder. In blau dargestellt sind die Blöcke zur Codierung der Nutzdaten und in grün die Blöcke zur Codierung der Kontrollinformationen. Die roten Blöcke zeigen den Aufbau des OFDM-Systems, das von beiden Codierpfaden gemeinsam genutzt wird. In grau ist das analoge Frontend (engl. *Analog Frontend*, AFE) dargestellt, das das digitale Powerline-Signal in ein analoges transformiert oder am Empfänger das analoge Signal abtastet, quantisiert und durch Filter bandbegrenzt.

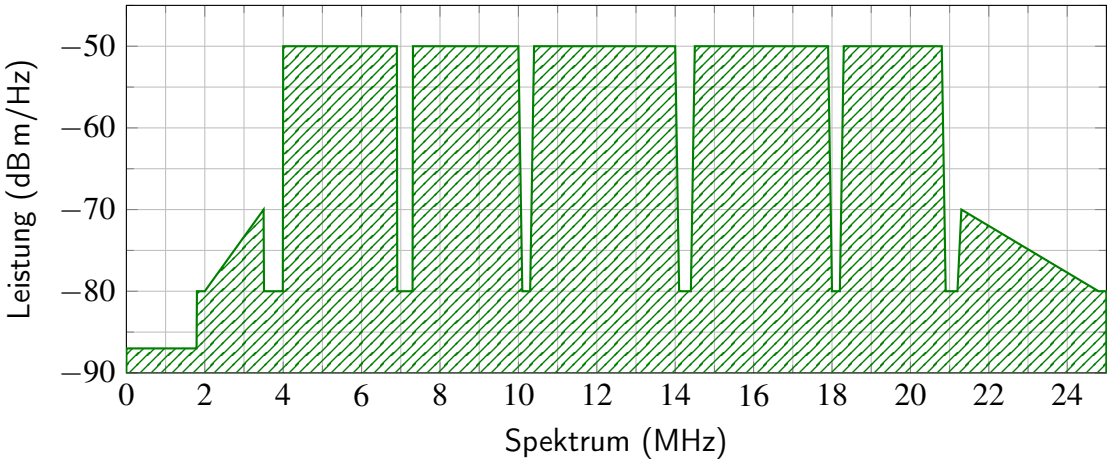

Abbildung 3.6: *Spektrale Maske der Sendeleistung für den Frequenzbereich der Powerline-Kommunikation nach dem HomePlug-1.0-Standard. Die Übertragungsbandbreite reicht von 4 MHz bis 21 MHz. Abgebildet ist die Frequenzmaske des US-amerikanischen Marktes, reguliert durch die FCC [Hom08].*

Zunächst wird die Encodierung eines Powerline-Frames vorgestellt. Diese ist im HomePlug-1.0-Standard vollständig beschrieben und wird in diesem Abschnitt zusammengefasst [Hom08]. Die für die Encodierung verwendeten Verarbeitungsblöcke sind im oberen Teil des Blockschaltbildes in Abbildung 3.5 abgebildet und umfassen zwei Bereiche: die Vorwärtsfehlerkorrektur (engl. *Forward Error Correction*, FEC) der Framekontrollen (grün) und der Nutzdaten (blau) sowie das OFDM-System (rot). Die Kontroll- und Nutzdaten werden in separaten parallelen Blöcken codiert und mit Redundanzen versehen. Anschließend werden durch das OFDM-System die Bitinformationen in Phaseninformationen codiert, in den Zeitbereich transformiert und ein kontinuierliches Zeitsignal erstellt.

Bei der Codierung der Framekontrolle (engl. *Frame Control*, FC) werden die eingehenden 25 bit zunächst mit Hilfe eines systematischen ($n = 100$, $k = 25$, $w = 16$) Produktcodes im Produktencoder um Redundanz ergänzt. Dieser Produktcode lässt sich in zwei identische Matrizen G der Größe ($n = 10$, $k = 5$, $w = 4$) zerlegen. Diese identischen Matrizen enthalten verkürzt-erweiterte Hamming-Codes (engl. *Shortened Extended Hamming-Codes*) [Sch12]. Dabei werden die $k = 25$ bit am Eingang in eine $5x5$-Matrix I umsortiert und anschließend jeweils von links und rechts mit der $10x5$-Generatormatrix G multipliziert: $V = G^T I G$.

$$G = \left(\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ \underbrace{\qquad\qquad\qquad\qquad}_{E} & & & & & \underbrace{\qquad\qquad\qquad\qquad}_{P} & & & & \end{array}\right)$$

Die Matrix G lässt sich in zwei Untermatrizen unterteilt: die Einheitsmatrix E und die Paritätsmatrix P. Das Ergebnis V der Matrixmultiplikation ist eine $n = 100$ bit-Matrix mit einer Hamming-Distanz von $w = 16$. Die entstandene Redundanz von 75 bit unterteilt sich in die Zeilenparität P_z, die Spaltenparität P_s und die Parität der Parität P_p, die aus der zweifachen

Multiplikation der Matrix G von rechts und links entsteht. Am Empfänger können diese Paritäten genutzt werden, um eventuelle Fehler bei der Kanalübertragung zu korrigieren. Die Korrekturkapazität dieses Codes ist $\lfloor \frac{w-1}{2} \rfloor = 7$ und die Anzahl an detektierbaren Fehler ist $w-1 = 15$ [Ham50; Wic95].

$$V = \left(\begin{array}{ccccc|ccccc} \overbrace{\phantom{I_0\ I_5\ I_{10}\ I_{15}\ I_{20}}}^{\text{FC}} & & & & & \overbrace{}^{\text{Zeilenparität}} & & & & \\ I_0 & I_5 & I_{10} & I_{15} & I_{20} & P_z & P_z & P_z & P_z & P_z \\ I_1 & I_6 & I_{11} & I_{16} & I_{21} & P_z & P_z & P_z & P_z & P_z \\ I_2 & I_7 & I_{12} & I_{17} & I_{22} & P_z & P_z & P_z & P_z & P_z \\ I_3 & I_8 & I_{13} & I_{18} & I_{23} & P_z & P_z & P_z & P_z & P_z \\ I_4 & I_9 & I_{14} & I_{19} & I_{24} & P_z & P_z & P_z & P_z & P_z \\ \hline P_s & P_s & P_s & P_s & P_s & P_p & P_p & P_p & P_p & P_p \\ P_s & P_s & P_s & P_s & P_s & P_p & P_p & P_p & P_p & P_p \\ P_s & P_s & P_s & P_s & P_s & P_p & P_p & P_p & P_p & P_p \\ P_s & P_s & P_s & P_s & P_s & P_p & P_p & P_p & P_p & P_p \\ \underbrace{P_s\ P_s\ P_s\ P_s\ P_s}_{\text{Spaltenparität}} & & & & & \underbrace{P_p\ P_p\ P_p\ P_p\ P_p}_{\text{Paritätsparität}} & & & & \end{array}\right)$$

Im anschließenden *Interleaver* wird die Matrix V systematisch auf vier Frequenzträgermasken (engl. *Tone Mask*) der Länge 84 abgebildet. Diese Masken geben an, welche der insgesamt 84 Frequenzträger im Frequenzband[1] von 2 MHz bis 25 MHz verwendet werden dürfen und welche nicht. Abbildung 3.6 zeigt das Spektrum[2] des HomePlug-1.0-Standards, mit nutzbaren Frequenzanteilen bei maximal −50 dB m/Hz und nicht-nutzbaren Frequenzen bei −80 dB m/Hz. Die Frequenzträger verteilen sich dabei äquidistant mit einem Trägerabstand von 195,3125 kHz im Frequenzbereich von 4,49 MHz bis 20,70 MHz. Die maskierten Träger in dieser Arbeit sind die Träger 13, 14, 29, 49, 59, 51, 69 und 70, da diese in reservierte Amateurfunkfrequenzen fallen.

Der *Interleaver* der Framekontrolle (*FC-Interleaver*) verteilt die Informationsbits der Matrix V spaltenweise und zyklisch auf alle vier Frequenzmasken. Abbildung 3.7 zeigt exemplarisch dessen Funktionsweise mit einem maskierten Träger 3 (grau). Diese maskierten Träger werden bei der Belegung übersprungen, so dass die tatsächliche Anzahl an Informationsbits pro Symbol als $w/x/y/z = 84 - N_{\text{masked}}$ berechnet. Jedes der vier Symbole startet mit einem Versatz von 25 bit gegenüber dem vorherigen, beginnend mit dem Index null für das Erste. Diese Verteilung führt zu einer Zeit- und Frequenzdispersion der Informationsbits und erzeugt Redundanz der Daten mit einem Faktor zwischen 3–4 (abhängig von der Anzahl benutzbarer Frequenzträger). Diese Verteilung schützt die Informationen gegen schmalbandige Störungen und Bündelfehler auf dem Übertragungskanal, indem zum Beispiel empfangsseitig eine Mehrheitsentscheidung der verteilten Bits diese Fehler kompensiert werden [Shi04]. Die gesamte Redundanz der FC-Datencodierung liegt für diesen Standard bei einem Faktor von 12.

[1] Die Regulierung und Verteilung dieser Frequenzbänder obliegt in Deutschland der Bundesnetzagentur und muss von den Kommunikationsstandards zwingend eingehalten werden.

[2] Hier: Spektrum für den US-amerikanischen Markt nach Regulierung durch die *Federal Communications Commission* (FCC).

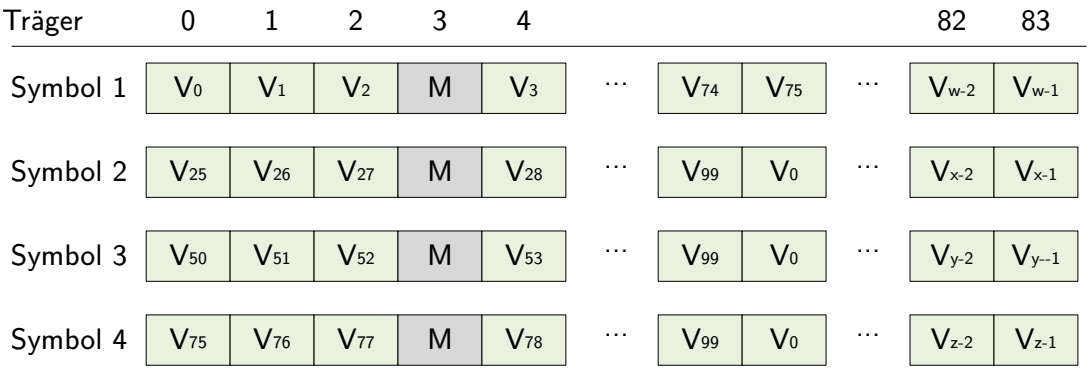

Abbildung 3.7: *Verteilung der Bits des Produkt Encoders auf die vier Symbole des FC-Interleavers. Nach dem ersten Symbol mit 25 bit Versatz pro weiterem Symbol. Das M bei Trägernummer 3 zeigt eine Maskierung an [Hom08].*

Die Codierung der Nutzdaten arbeitet nicht wie die FC-FEC auf allen Bits parallel, sondern blockweise mit einer Größe von 20 oder 40 resultierenden Symbolen am *Interleaver*. Abhängig von der Anzahl der Nutzdaten können bis zu vier Blöcke mit je 40 Symbolen in einem Paket codiert werden (vgl. Abschnitt 3.1.1 und Anhang A). Der erste Schritt in der Verarbeitung ist ein *Scrambler*, der auf Basis eines linear rückgekoppelten Schieberegisters (engl. *Linear Feedback Shift Register*, LSFR) die Nutzdatenbits in eine pseudozufällige Bitsequenz umwandelt. Diese Sequenz hat die Eigenschaft einer Gleichverteilung, bei der die Auftrittswahrscheinlichkeiten für binäre Einsen und Nullen nahezu identisch ist. Diese Eigenschaft ist für die Signalübertragung wichtig, da eine Ungleichverteilung der Bits zu einem Gleichanteil des Sendesignals führen kann, der sich eventuell nicht über den Kommunikationskanal übertragen lässt [Gi 94].

Anschließend wird mit Hilfe eines Reed-Solomon-Encoders eine Vorwärtsfehlerkorrektur durchgeführt. Durch seine byteweise Verarbeitung der Daten eignet sich der RS-Code besonders zur Korrektur von Bündelfehlern. Der Reed-Solomon-Encoder in diesem Standard hat dafür zwei verschiedene Codiermodi:

- ein ($n = 255$, $k = 239$, $t = 8$)-Code für reguläre Codierung
- ein ($n = 255$, $k = 247$, $t = 4$)-Code für den ROBO-Modus

Dabei ist n die Blockgröße in Byte, k die Anzahl an Informationsbytes und t die Anzahl korrigierbarer Fehler. Die zur Fehlerkorrektur angehängten $2t$ Paritätsbytes pro Block berechnen sich mit Hilfe der Generatorpolynome $G(x)$ für die reguläre Codierung nach Gleichung 3.4 und $G_{\mathrm{ROBO}}(x)$ für den ROBO-Modus nach Gleichung 3.5.

$$G(x) = (x - \alpha^1)(x - \alpha^2)...(x - \alpha^{16}) \tag{3.4}$$

$$G_{\mathrm{ROBO}}(x) = (x - \alpha^1)(x - \alpha^2)...(x - \alpha^{8}) \tag{3.5}$$

Da ein Block, bestehend aus 40 Symbolen, je nach Modulierungsparametern bis zu 629 Byte enthalten kann, muss der Reed-Solomon-Encoder bis zu drei RS-Blöcke berechnen, die anschließend konkateniert werden. Im ROBO-Modus ist die maximale Größe eines 40-Symbolblocks mit 152 Byte pro Block deutlich unter der Kapazität von $k = 247$ und es wird immer nur ein RS-Block benötigt.

Die redundanten RS-Byte-Blöcke werden anschließend serialisiert und als Bits durch einen Faltungsencoder (engl. *Convolutional Encoder*) verarbeitet. Dieser ist in Abbildung 3.8 gezeigt

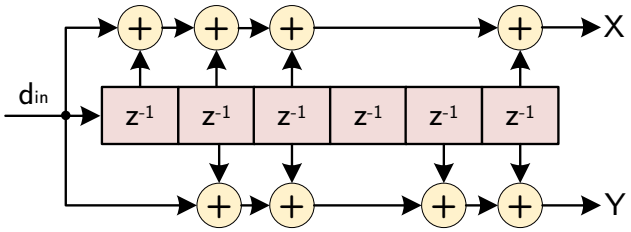

Abbildung 3.8: *Struktur des Faltungsencoders mit zwei Ausgängen und Gedächtnislänge $K = 7$. Jede +-Operationen stellt eine Exklusiv-Oder-Funktion zweier Bits dar [Hom08].*

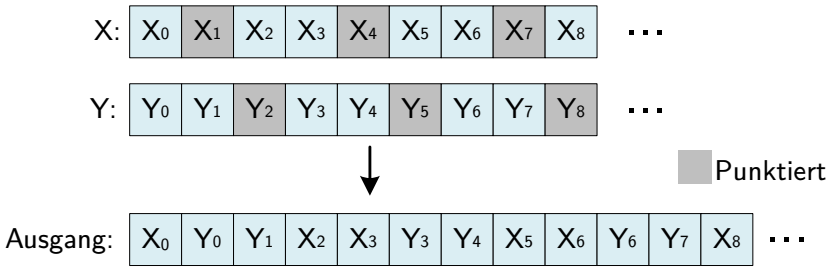

Abbildung 3.9: *Punktierung und Serialisierung der Ausgangsdaten des Faltungsencoders. Die grau markierten Blöcke zeigen die Daten an, die punktiert und dementsprechend verworfen werden [Hom08].*

und fügt zusätzliche systematische Redundanz hinzu. Pro Eingangsbit d_{in} werden zwei Ausgangsbits X und Y berechnet, indem an verschiedenen Stellen entlang eines Schieberegisters die Bits mit Exklusiv-Oder-Bausteinen verrechnet werden. Das Schieberegister hat sechs Register und erzeugt damit ein Gedächtnis der Länge $K = 7$ mit maximal $2^{K-1} = 64$ möglichen Zuständen und jeweils zwei Folgezuständen. Mit Hilfe dieser Codierung können auf Basis des Gedächtnisses in den codierten Informationen Einzelfehler am Empfänger korrigiert werden. Die resultierende Codierrate ist hier $R = \frac{1}{2}$, da für jedes Eingangsbit zwei Ausgangsbits erzeugt werden. Damit verdoppelt sich auch die Anzahl Informationsbits durch diesen Faltungsencoder.

Um den Anteil der Redundanz zu reduzieren und die damit verbundene Nutzdatenrate zu erhöhen, kann optional eine Punktierung (engl. *Puncturing*) des Bitstroms durchgeführt werden. Abbildung 3.9 zeigt die im Standard spezifizierte Punktierung. Dazu werden systematisch $\frac{1}{3}$ aller Bits verworfen, was zu einer erhörten Codierrate von $R = \frac{1}{2} \times \frac{3}{2} = \frac{3}{4}$ führt. Die Punktierung kann je nach Qualität des Übertragungskanals aktiviert oder deaktiviert werden, um den Anteil der Redundanz für den Empfänger zu erhöhen oder zu senken. Ohne Punktierung der Daten serialisiert der Punktierer nur die Eingänge X und Y, indem abwechselnd ein Bit von X und eines von Y ausgegeben wird.

Als letzter Schritt der Nutzdatencodierung verteilen zwei *Interleaver* die codierten Informationen auf die Symbolblöcke für die OFDM-Verarbeitung. Der lineare Bitstrom aus der Punktierung wird zyklisch auf alle 20 oder 40 Symbole verteilt, indem die Informationen zeilenweise in eine Matrix geschrieben und anschließend spaltenweise ausgelesen werden. Der ROBO-*Interleaver* fügt zusätzliche Redundanz hinzu, indem er, genau wie beim *FC-Interleaver*, die Informationsbits auf vier Symbole zyklisch verschoben verteilt. Die einzelnen Symbole haben hierbei einen Versatz von jeweils ein Viertel der Anzahl Frequenzträger in der Frequenzmaske. Damit entsteht, wie beim *FC-Interleaver*, eine zusätzliche Zeit- und Frequenzdispersion der Informationsbits, die

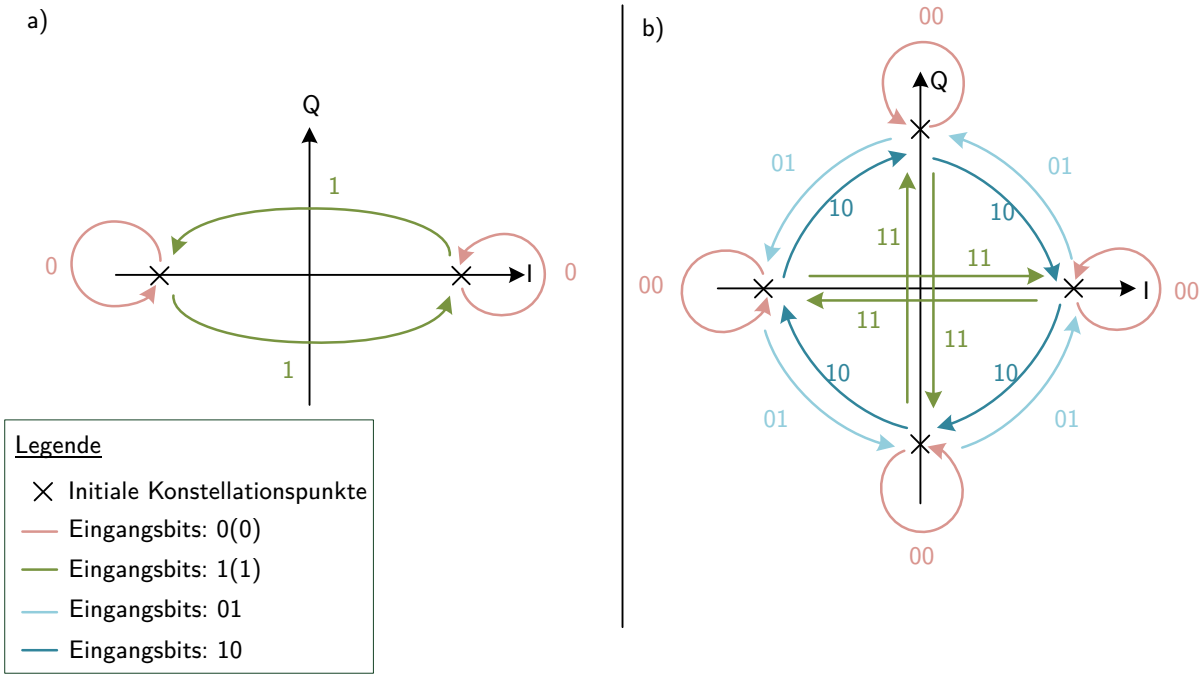

Abbildung 3.10: *Differentielle Modulation von PSK-Signalen: in a) als DBPSK und in b) als DQPSK. Die Übergänge zwischen den Konstellationspunkten zeigen die Übersetzung der Bitinformation in eine Phasenrotation.*

am Empfänger verwendet werden kann, um starke Kanalstörungen zu kompensieren. Durch diese vierfache Zuweisung von Bits sind im ROBO-Modus nur ein Viertel der Informationsbits pro Symbolblock enthalten, verglichen mit der regulären Codierung. Dies reduziert auch die Datenrate um den Faktor 4, ermöglicht aber eine Kommunikation bei Kanalstörungen, bei der eine Kommunikation mit regulärer Codierung nicht mehr möglich ist.

Nach der FEC-Codierung der Kontroll- und Nutzdaten folgt die Modulation und Transformation der digitalen Informationsbits auf den analogen Übertragungskanal. Als erster Schritt werden die Bits in Phaseninformationen abgebildet (engl. *Mapping*). Dazu wird in diesem Standard die Phasenumtastung (engl.: *Phase Shift-Keying*, PSK) verwendet, wobei je nach Übertragungskanalqualität eine binäre Phasenumtastung (engl.: *Binary Phase Shift-Keying*, BPSK) oder eine Quadraturphasenumtastung (engl.: *Quadrature Phase Shift-Keying*, QPSK) verwendet werden kann. Die codierten Kontrollinformationsbits werden immer mit kohärenter BPSK moduliert, die feste Referenzkonstellationspunkte für die Phasenumtastung vorgibt [Hom08]. Die codierten Nutzdatenbits werden entweder mit differentieller BPSK (DBPSK) oder differenzieller QPSK (DQPSK) moduliert. Diese beiden Modulationen sind in Abbildung 3.10 mit allen möglichen Übergängen zu den angegebenen Eingangsbits dargestellt. In Abbildung 3.10 a) ist die DBPSK abgebildet mit den initialen Konstellationspunkten auf der I-Achse. Ist das zu modulierende Bit eine Null, wird der gleiche Konstellationspunkt des vorangegangenen Symbols verwendet und bei einer Eins dieser Punkt um 180° gedreht. Bei der DQPSK in Abbildung 3.10 b) gibt es vier Konstellationspunkte, die sich ebenfalls mit der Eingabe 00 nicht ändern und mit 11 um 180° drehen. Durch eine 01 als Eingang dreht sich der Konstellationspunkt um 90° und mit 10 um 270°. Dabei wird als Referenz die Phase des identischen Frequenzträgers des vorangegangenen Symbols verwendet [Das10]. Bei der Kommunikation im ROBO-Modus wird nur DBPSK-Modulation unterstützt.

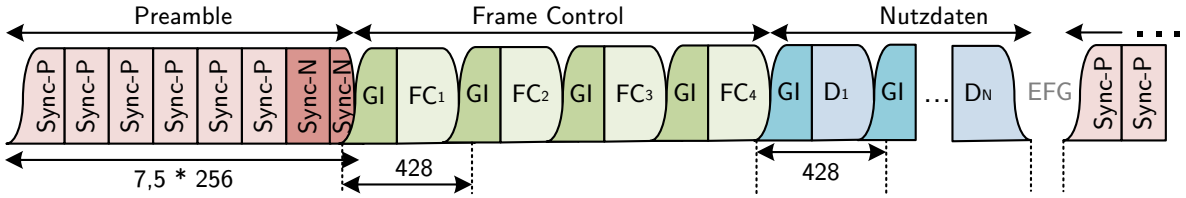

Abbildung 3.11: *Aufbau des analogen Sendesignals mit sich überlappenden OFDM-Symbolen der Präambel, der Frame Control und der Nutzdaten. Unterhalb der Symbole ist deren Länge in Samples angegeben [Hom08].*

Anschließend werden die Phaseninformationen blockweise durch eine inverse schnelle Fourier-Transformation (engl.: *Inverse Fast Fourier Transformation*, IFFT) in den Zeitbereich transformiert. Diese IFFT ist komplexwertig und transformiert die 84 Frequenzträger jedes Symbols in ein 256 Werte kontinuierliches digitales Zeitsignal. Dafür wird eine 256-Punkt IFFT verwendet, die für die Erzeugung eines reellen Ausgangssignals ein symmetrisches Frequenzspektrum benötigt. Daher werden die 84 Frequenzträger entsprechend der Spektralmaske in Abbildung 3.6 auf 128 Werte mit Nullen aufgefüllt und anschließend diese Werte komplex-konjugiert gespiegelt [And12].

Nach der Transformation in den Zeitbereich werden die letzten 172 Werte jedes Symbols kopiert und als Schutzintervall (engl.: *Guard Interval*, GI) vorangestellt. Damit kann die, für eine OFDM-basierte Übertragung typische, lange Kanalimpulsantwort ausklingen, bevor ein weiteres Symbol beginnt. Zusätzlich gibt die zyklische Erweiterung des Signals der Zeit-Synchronisation am Empfänger mehr Freiheiten bei der Wahl des Synchronisationszeitpunktes (siehe Abschnitt 3.2.1) [Zai18].

Abbildung 3.11 zeigt den detaillierten Aufbau des Sendesignals, bestehend aus den einzelnen Symbolen inklusive dessen Schutzintervallen. Dem Signal wird eine Präambel vorangestellt, die allen HomePlug-1.0-Standard-kompatiblen Stationen bekannt ist. Diese besteht aus 7,5 Wiederholungen eines 256 Werte langen Sync-Signals, dessen spektrale Eigenschaften alle Frequenzen des Standards enthält. Dieses Signal wird vom Empfänger verwendet, um die Synchronisation und die Schätzung des Übertragungssignals durchzuführen. Die ersten sechs Wiederholungen des Sync-Signals haben ein positives Vorzeichen (Sync-P) und die folgenden 1,5 Wiederholungen ein negatives Vorzeichen (Sync-N).

Anschließend folgen die vier Symbole des *FC-Headers* mit ihrem Schutzintervall. Die Gesamtlänge eines Symbols mit Schutzintervall beträgt 428 Werte, wobei sich die ersten und letzten acht Werte mit dem vorangegangenen und dem nachfolgenden Symbol überlappen. Diese Überlappung verhindern Amplitudensprünge an den Symbolgrenzen. Dazu werden die acht Werte an den Grenzen mit einem Cosinus-Roll-Off-Faktor multipliziert, um Inter-Symbol-Interferenz und eine Amplitudenüberhöhung zu vermeiden [And05]. Nach den FC-Symbolen folgen direkt überlappend die Nutzdaten-Symbole. Insgesamt entsteht so ein kontinuierliches Signal von Präambel bis EFG, der das Ende der Nutzdaten anzeigt. Abschließend folgt ein weiterer *Delimiter*, der den *FC-Trailer* enthält und identisch zum ersten *Delimiter* aufgebaut ist.

Dieses Signal wird dem analogen Frontend übergeben, das die einzelnen Werte digital-analog wandelt (engl. *Digital-to-Analog-Converter*, DAC). Der Standard schreibt eine Abtastfrequenz von 50 MHz vor, so dass das AFE eine Ausgangsdatenrate von 50 MSamples/s hat. Zusätzlich befinden sich im AFE noch Bandpassfilter, die den Frequenzbereich des Signals auf 4,5 MHz bis 21 MHz bandbegrenzen. Beim Empfang von Daten hat das AFE die Aufgabe, das analoge Signal

in ein digitales zu wandeln (engl. *Analog-to-Digital-Converter*, ADC). Moderne AFEs verwenden dabei zusätzlich eine automatische Verstärkungsregelung (engl. *Automatic Gain Control*, AGC), um den Wertebereich des ADC möglichst optimal auszusteuern [Pér11].

Der Decoder ist, bis auf das Blockschaltbild in Abbildung 3.5, im Standard nicht näher spezifiziert. Im Wesentlichen gilt es hier die im Encoder erzeugten Redundanzen und Signaleigenschaften zu nutzen, um Übertragungsfehler zu korrigieren und die übertragenen Bits möglichst exakt zu rekonstruieren. Eine Untersuchung und Auswahl von Algorithmen für den Decoder nach dem aktuellen Stand der Forschung erfolgt im nächsten Abschnitt.

3.2. Entwurf des PHY-Schicht-Decoders

Die Verarbeitungsblöcke des PHY-Schicht-Decoders sind im unteren Teil von Abbildung 3.5 gezeigt, im Standard selbst aber nicht weiter spezifiziert. In diesem Abschnitt werden anhand des aktuellen Stands der Forschung verschiedene Algorithmen und Möglichkeiten der Decodierung der einzelnen Blöcke vorgestellt. Da nicht alle Verarbeitungsblöcke Redundanzen erzeugen, gibt es Blöcke, wie den *Scramblers* oder den *Interleaver*, deren Funktionen am Empfänger nur invers ausgeführt werden müssen.

Bei allen Senderblöcken, die Redundanz erzeugt haben, gibt es hingegen keine eindeutige Lösung der Decodierung. Je nach Beschaffenheit des Übertragungskanals und der Struktur des Kommunikationsnetzes, in dem sich das Powerline-System befindet, können hier mehr oder weniger komplexe Algorithmen verwendet werden, um Übertragungsfehler korrigieren zu können. Dies führt zu einem generellen Abtausch von Aufwand der Decodierung, der Geschwindigkeit und Größe des Gesamtsystems und der Empfindlichkeit gegenüber Störungen. Diese Parameter spannen einen Entwurfsraum auf, den es in dieser Arbeit nach dem aktuellen Stand der Forschung zu analysieren gilt. Nachfolgend werden die einzelnen Blöcke in der Reihenfolge bei der Decodierung, beginnend mit dem digital abgetasteten Signal am AFE, vorgestellt und die jeweiligen Möglichkeiten ihrer Parametrierung präsentiert.

Nachdem das AFE die analogen Daten abgetastet und gefiltert hat, werden die digitalen Werte dem Synchronisationsmodul übergeben. Dieses korreliert kontinuierlich das vom Powerline-Bus empfangene Signal mit der bekannten Präambel, um festzustellen, ob ein Powerline-Paket vorhanden oder nur Rauschen auf dem Kanal ist. Zusätzlich wird der Zeit- und Phasenversatz des Empfangssignals geschätzt und korrigiert. Der detaillierte Aufbau der Synchronisation im Zeitbereich und Kanalschätzung im Frequenzbereich wird in dem folgenden Abschnitt 3.2.1 gezeigt, da der Aufbau und die Parametrierung des Synchronisationssystems sehr umfangreich ist. Das synchronisierte Signal wird anschließend durch eine schnelle Fourier-Transformation (FFT) in den Frequenzbereich transformiert und die Phaseninformationen mit Hilfe der geschätzten Kanalfrequenzantwort korrigiert. Anschließend werden die 84 Frequenzträger aus dem Spektrum extrahiert und der Demodulation übergeben.

Bei der Demodulation werden die Phaseninformationen zurück in Bitinformationen transformiert. Dabei werden der *FC-Header* und -*Trailer* kohärent nach der Referenzphase und die Nutzdaten differenziell demoduliert. Dazu kann entweder eine harte Entscheidung (engl. *Hard Decision*) mit fester Grenze oder eine weiche Entscheidung (engl. *Soft Decision*) mit quantisierter Wahrscheinlichkeiten verwendet werden [Das10; Pro83]. Bei *Hard Decision* wird an

der Grenze entschieden, ob die empfangene Phase zu einer logischen Eins oder logischen Null gehört und entsprechend decodiert. Dabei wird die Information über die Distanz zur Grenze, also die Wahrscheinlichkeit der korrekten Decodierung, verworfen. Bei *Soft Decision* wird diese Wahrscheinlichkeit teilweise erhalten, indem die Ausgabe des Demodulators eine quantisierte Version dieser Wahrscheinlichkeit ist. Nach heutigem Stand der Forschung wird dafür häufig das logarithmische Plausibilitätsverhältnis (engl. *Log-Likelihood Ratio*, LLR) verwendet, das sich nach Gleichung 3.6 berechnet [Hel71].

$$L(b) = \log_e\left(\frac{p(b=0|r)}{p(b=1|r)}\right) = \log_e\left(\frac{\sum_{s\in S_0} e^{\frac{-1}{\sigma^2}((x-s_x)^2+(y-s_y)^2)}}{\sum_{s\in S_1} e^{\frac{-1}{\sigma^2}((x-s_x)^2+(y-s_y)^2)}}\right) \tag{3.6}$$

Dabei ist b das ursprünglich gesendete Bit, σ^2 die Rauschleistung im Signal, $r = x + iy$ die empfangene, komplexwertige Phase und $s = s_x + is_y$ der Referenzkonstellationspunkt. Die Mengen $S_0 \subset S$ und $S_1 \subset S$ sind diejenigen Konstellationspunkte, bei denen das gesendete Bit null oder eins ist und für die gilt $S = S_0 \cup S_1$. In den beiden Summen über die Exponentialfunktionen ist also die Distanz zu allen Konstellationspunkten der Modulation betrachtet.

$$L(b) = \frac{-1}{\sigma^2}\left(\min_{s\in S_0}((x-s_x)^2+(y-s_y)^2) - \min_{s\in S_1}((x-s_x)^2+(y-s_y)^2)\right) \tag{3.7}$$

Um die algorithmische Komplexität der Berechnung in Gleichung 3.6 zu reduzieren, haben Viterbi et al. in [Vit98] eine approximative Berechnung des LLR vorgestellt. Diese ist in Gleichung 3.7 gezeigt und betrachtetet für die Berechnung des LLR nur die jeweils nächstgelegenen Konstellationspunkt für eine Eins oder Null. Für die Implementierung in dieser Arbeit ergeben sich also die folgenden Parameter:

- Wahl zwischen *Hard Decision* und *Soft Decision*
- Quantisierung der *Soft-Decision*-Berechnung

Nach der Demodulation teilt sich der Datenpfad, wie auch beim Encoder, in die Decodierung der Nutzdaten und Kontrollinformationen.

Die Decodierung der Kontrollinformationen beginnt mit dem *FC-Deinterleaver*, der die verteilten Bits der vier FC-Symbole aufsummiert. Anschließend kann mit Hilfe eines Schwellenwertes dasjenige Bit decodiert werden, das am wahrscheinlichsten übertragen wurde (*Hard Decision*). Alternativ kann auch hier ein *Soft-Decision*-Ansatz gewählt werden, bei dem die aufsummierten Werte nicht decodiert und direkt an den Produktdecoder weitergegeben werden.

Abschließend werden mit Hilfe des Produktdecoders Bitfehler detektiert und korrigiert. Die Algorithmen zur Decodierung eines Produktcodes unterteilen sich ebenfalls in die Gruppe der *Hard-Decision*- und *Soft-Decision*-Decoder. Eine Übersicht der aktuellen *Soft-Decision*-Decoder wurde in Al-Askary et al. [Al-03] gezeigt und unterteilt sich in *Maximum Likelihood* (ML)-Decoder, *Maximum Aposteriori Probability* (MAP)-Decoder oder *Generalized Minimun Distance* (GMD)-Decoder. Für die Implementierung in dieser Arbeit kann also aus folgenden Parametern gewählt werden:

- Wahl der *Hard-Decision*- oder *Soft-Decision*-Decodierung des gesamten Kontrollpfades
- Bei *Soft Decision*, Auswahl aus ML-, MAP- oder GMD-Decoder.

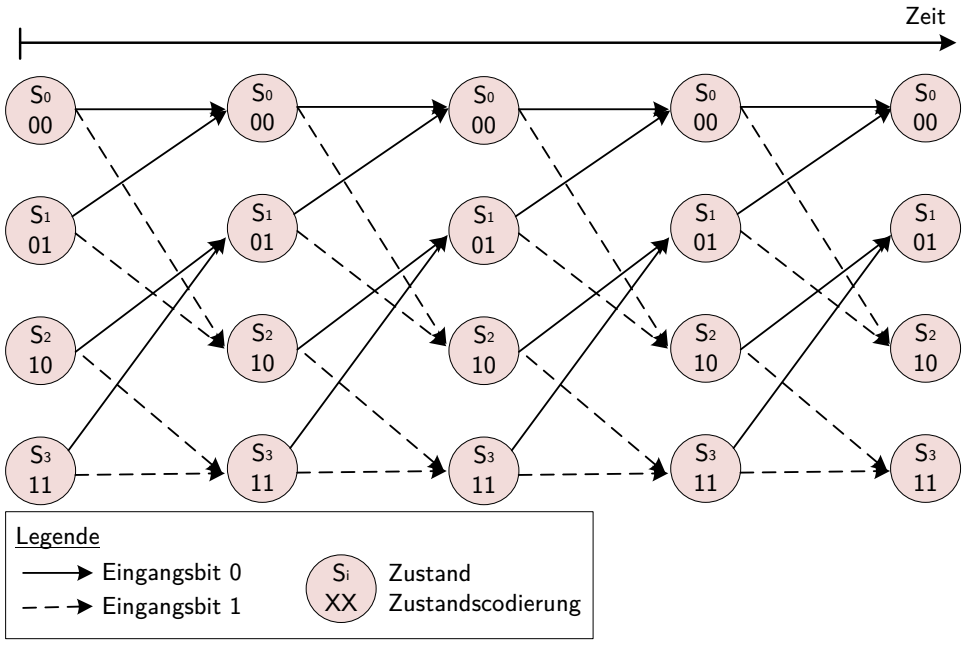

Abbildung 3.12: *Trellis-Diagramm für einen Faltungsencoder mit zwei Registern, also vier Zuständen. Die durchgehenden Pfeile zeigen eine eingehende null und die gestrichelten Pfeile eine eingehende eins am Eingang des Faltungsencoders.*

Auch die Nutzdaten werden zuerst mit Hilfe des entsprechenden *Deinterleavers* zurück in ihre ursprüngliche Reihenfolge gebracht, wobei Bündelfehler auf Einzelfehler verteilt werden. Zusätzlich summiert der ROBO-*Deinterleaver* die vier redundanten Kopien auf und führt, im Falle einer *Hard-Decision*-Decodierung, eine Schwellwert-basierte Quantisierung durch. Alternativ kann auch hier wieder eine *Soft-Decision*-basierte Weitergabe ohne Quantisierung erfolgen.

Danach folgt die Decodierung und Fehlerkorrektur durch den Viterbi-Decoder. Dieser bestimmt auf Basis des Gedächtnisses des Faltungsencoders die wahrscheinlichste Bitfolge, die gesendet wurde. Dazu werden alle möglichen Zustände und Folgezustände parallel über die Bitfolge betrachtet. Die Basis dafür ist ein Trellis-Diagramm, anhand dessen mit Hilfe von gewichteten Kanten alle möglichen Pfade berechnet und verfolgt werden können. Abbildung 3.12 zeigt exemplarisch ein Trellis-Diagramm für eine Gedächtnislänge von $K = 3$ und daraus resultierenden vier Registerzuständen. Pro Zustand gibt es zwei abgehende Kanten mit den Eingangsbits Null (durchgezogener Pfeil) oder Eins (gestrichelter Pfeil) und zwei ankommende Kanten. Das Trellis-Diagramm ist die Basis für die Decodierung des Viterbi-Decoders, der anhand der Gewichtung der Kanten, der Entscheidung von überlebenden Pfaden und rückwärtiger Suche entlang des Diagramms Fehler korrigieren kann.

Der Viterbi-Algorithmus wurde zuerst 1967 von Andrew J. Viterbi vorgestellt und wird immer noch in aktuellen Datenübertragungsstandards verwendet (zum Beispiel DVB-T) [Vit67]. 1973 veröffentlichte Forney einen dreischrittigen Algorithmus zur iterativen Berechnung des Viterbi-Algorithmus, der bis heute Grundlage der meisten Implementierungen ist [For73]. Seit dieser Veröffentlichung gab es viele Implementierungen und Adaptionen, mit dem Ziel den Durchsatz oder die Flächeneffizienz dieses Algorithmus zu verbessern [Fet91; Gem01]. Eine Übersicht über die aktuellen Versionen und Implementierungen des Viterbi-Algorithmus finden sich in Habib et al. [Hab10]. Unter anderem fanden Hagenauer et al. [Hag89] heraus, dass *Soft-Decision*-

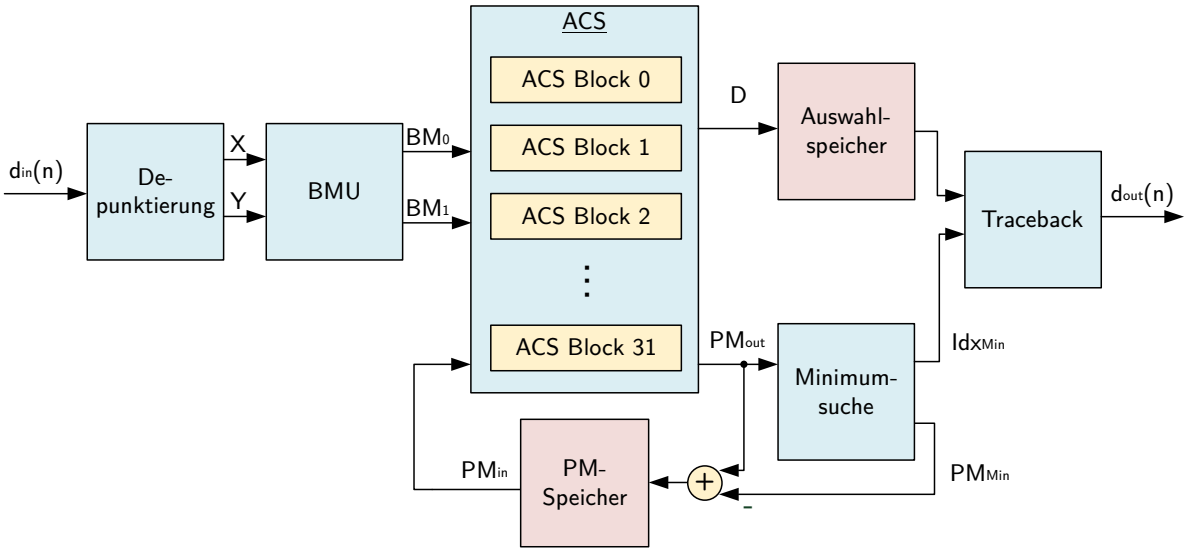

Abbildung 3.13: *Aufbau des klassischen Dreischritt-Viterbi-Decoders mit Traceback-Decodierung und initialer Depunktierung des Bitstroms. Dieser wurde um eine Überlaufkontrolle, bestehend aus Suche und Subtraktion des Minimums ergänzt.*

Informationen anstelle von *Hard-Decision*-Informationen am Eingang des Viterbi-Decoders die Robustheit gegenüber Rauschen um etwa 3 dB erhöht. Darauf aufbauend entdeckten Joeressen et al.[Joe93], dass eine 3 bit-Quantisierung der Eingangswerte ausreichend ist, um eine nur minimal geringere Robustheit gegenüber dem Rauschen zu erreichen, als es bei unquantisierten Eingangswerten der Fall ist.

Auf Basis dieser Erkenntnisse, wurde der in Abbildung 3.13 gezeigte *Traceback*-basierten Viterbi-Decoder erstellt, der dem Schema des klassischen Dreischritt-Algorithmus von Forney folgt. Zusätzlich wurde dieser um eine Überlaufkontrolle der Pfad-Metrik nach Fettweis et al. [Fet90] und eine Depunktierung des eingehenden Bitstroms erweitert. Dieser Viterbi-Decoder kann sowohl 3 bit-quantisierte *Soft Decision*-, also auch *Hard Decision*-Eingangsdaten verarbeiten und bietet eine parametrierbare Plattform zur Skalierung des Durchsatzes und der Latenz. Zuerst wird mit Hilfe einer Depunktierung die Ausgangsreihenfolge der Bits aus dem Faltungsencoder wiederhergestellt. Dabei werden Dummy-Bits an punktierten Stellen eingefügt und der kontinuierliche Bitstrom in die beiden Ausgänge X und Y des Faltungsencoders zerlegt. Diese Ausgänge werden verwendet, um vier Kantengewichte (engl.: *Branch Metric*, BM) zu berechnen, die durch ein Distanzmaß, wie beispielsweise die Hamming-Distanz oder die euklidische Distanz, die Differenz zu den fehlerfreien Pfaden angeben. Die eventuell eingefügten Dummy-Bits werden in diese Berechnung ignoriert. Anschließend werden in 32 parallelen Addition-Vergleich-Auswahl-Einheiten (engl. *Add-Compare-Select*, ACS) zuerst die vier Kantengewichte auf die Pfadgewichte (engl. *Path Metric*, PM) addiert, dann je zwei eingehende Kanten an den beiden Folgezuständen verglichen und das jeweils geringere ausgewählt (vgl. Zustände in Abbildung 3.12). Die dabei getroffene Entscheidung, ob eine Eins- oder Nullkante das geringere Gewicht hat, wird im Auswahlspeicher gespeichert und die aktuelle Pfad-Metrik in den PM-Speicher zurückgeschrieben.

Für die Überlaufkontrolle wurde eine Minimumsuche implementiert, die sowohl den minimalen Wert für die Subtraktion vor der Speicherung findet, als auch den Index des minimalen Wertes als Startpunkt für die Decodierung bestimmt. Zur Decodierung der Informationsbits

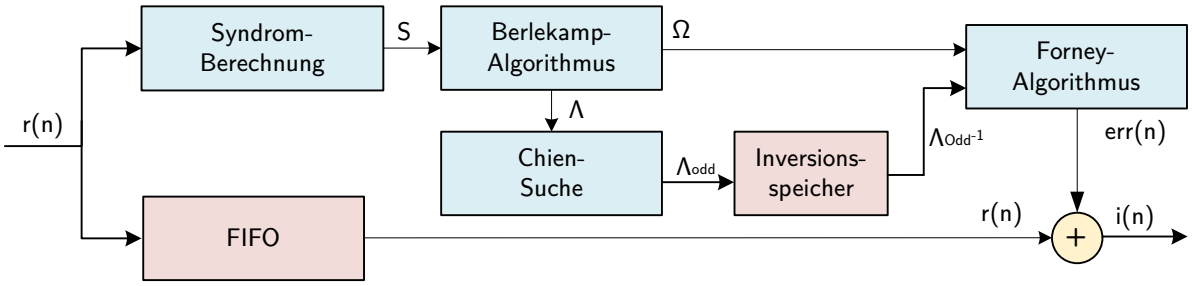

Abbildung 3.14: *Verwendeten Algorithmen und deren Reihenfolge zum Finden und Korrigieren von Fehlern bei der Reed-Solomon-Decodierung nach Flocke et al. [Flo05].*

wird ein sogenannter *Traceback* verwendet, der anhand des aktuell geringsten Pfadgewichtes und den getroffenen Entscheidungen im Auswahlspeicher eine rückwärtige Suche durch das Trellis-Diagramm durchführt. Dabei gilt, je länger der rückwärtige Pfad, desto geringer die Restfehlerwahrscheinlichkeit. Dabei hängt die Länge dieses Pfades direkt linear mit der Latenz der Decodierung und dem Hardware-Aufwand zusammen. Untersuchungen in der Literatur zeigen, dass für punktierte Codes eine Länge von mindestens $L = 10(K-1)$ Schritten verwendet werden sollte, also hier $L = 60$ [Col89]. Diese Länge bestimmt auch die Größe des Auswahlspeichers, der hier $L2^(K-1) = 10(K-1)2^(K-1) = 1800\,\text{bit}$ groß ist.

Enthält eine Bitsequenz mehr Einzelfehler als der Viterbi-Decoder decodieren kann, kann es zu Bündelfehlern am Ausgang kommen. Eine Fehlentscheidung entlang eines Pfades führt dazu, dass eventuell mehrere konsekutive Bits falsch ausgegeben werden, bis die *Traceback*-Einheit wieder auf dem korrekten Sendepfad ankommt. Um diese Bündelfehler zu korrigieren, wird ein Reed-Solomon-Decoder verwendet, der sich gut für die Korrektur dieser Fehler eignet. Wie der Reed-Solomon-Encoder arbeitet auch der Decoder blockweise auf 255 Byte großen Blöcken und verwendet abgeschlossene Körper der Galois-Feld-Arithmetik zum Finden und Korrigieren von Fehlern [Des09]. Die Algorithmen in dieser Arithmetik sind sehr komplex und deren Implementierung und Maximierung der Geschwindigkeit ist bis heute Gegenstand der Forschung. Ein Vergleich der Ansätze zur Hardware-effizienten Berechnung verschiedener Algorithmen und deren effizienten Implementierungen in Hardware findet sich in Kumar et al. [Kum05], Sarwate et al. [Sar01] oder Wu et al. [Wu15]. Aktuellere Implementierungen des Algorithmus haben häufig das Ziel einer sehr hohen Datenrate von mehr als 1 Gbit/s [Per19; Mon18] oder sind Adaptionen für Spezialanwendungen [Wan17; Liu17]. Der Reed-Solomon-Decoder,der in dieser Arbeit verwendet wird, basiert auf einer Implementierung von Flocke et al. [Flo05] und ist optimiert für eine hohe Flächeneffizienz und Taktrate. Der Vorteil dieser Architektur ist, dass sie generisch und adaptierbar ist und damit beide Generatorpolynome des Standards (siehe Gleichungen 3.4 und 3.5) in einer Architektur kombiniert verarbeitet werden können.

Der Aufbau dieses Reed-Solomon-Decoders ist in Abbildung 3.14 gezeigt. Die Eingangsbytes $r(n)$ werden in einem *First-In-First-Out*-Speicher (FIFO) gespeichert und parallel zur Berechnung des Syndroms S verwendet. Hier wird mit Hilfe des Generatorpolynoms $G(x)$ oder $G_{\text{ROBO}}(x)$ die identische Encoderberechnung über einen RS-Block durchgeführt, um zu überprüfen, ob das Syndrom $S = 0$ ist. Ist dieses der Fall, liegt kein Fehler im Block vor und die Daten können ohne weitere Berechnungen aus dem FIFO an den Scrambler weitergegeben werden. Wenn $S \neq 0$ ist, wird mit Hilfe des Berlekamp-Massey-Algorithmus [Ber72; Mas69] und des Syndroms ein Fehlerstellen-Polynom Λ und ein Fehlerwertpolynom Ω bestimmt. Dabei geben die Nullstellen

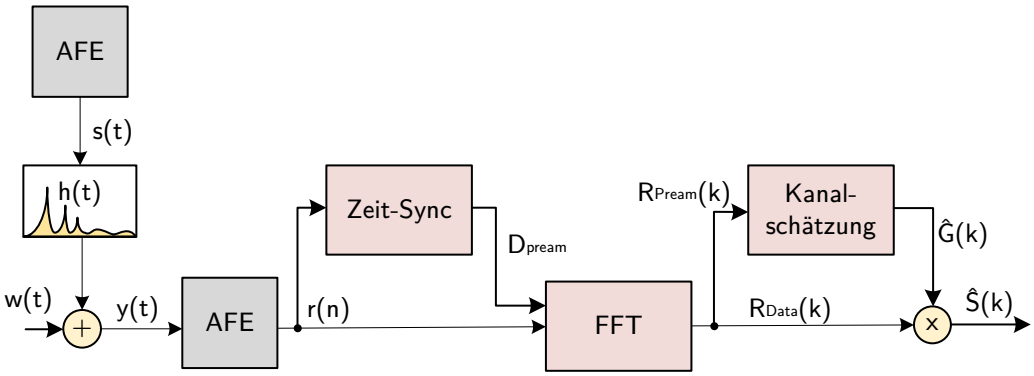

Abbildung 3.15: *Aufbau des OFDM-basierten Kanal- und Empfangsmodells in dieser Arbeit. $h(t)$ stellt den Mehrwege-übetragungkanal und $w(t)$ das additive weiße Rauschen dar.*

von Λ die Fehlerpositionen, die sich mit mit Hilfe einer Chien-Suche finden lassen [Chi64]. Anschließend werden die Nullstellen mit Hilfe einer Tabelle invertiert und zusammen mit dem Fehlerwertpolynom Ω durch den Forney-Algorithmus verarbeitet [For65]. Das Ergebnis ist eine Folge $e(n)$ mit 255 Byte, die an allen fehlerfreien Positionen null ist und an fehlerhaften Positionen den entsprechenden Korrekturwert enthalten. Diese Folge wird auf die im FIFO gespeicherten Werte aufaddiert und seriell als $i(n)$ ausgegeben.

Als letzter Schritt der Nutzdaten-Decodierung werden die Daten durch den Descrambler verarbeitet. Hier kann das gleiche LFSR wie bei der Encodierung der Daten verwendet werden, da durch die doppelte Anwendung der Exklusiv-Oder-Funktion wieder die gleichen Daten wie am Srambler-Eingang erzeugt werden. Die decodierten FC- und Nutzdaten werden abschließend der MAC-Schicht als MPDU übergeben.

3.2.1. OFDM-basierte Synchronisierung

In Breitband-Powerline-Systemen lässt sich die Kommunikationsstrecke innerhalb eines Gebäudes als Mehrwege-Dämpfungskanal darstellen (engl. *Multi-Path Fading Channel*) [Zim02; Got04]. Dieser hat eine Tiefpass-Charakteristik, schmalbandige starke Dämpfung durch Interferenzen und impulshaftes, periodisches Rauschen. Um trotz dieser Effekte eine stabile Kommunikation zu ermöglichen, ist eine robuste Synchronisation notwendig. Schon bei geringen Verletzungen der Orthogonalität der Frequenzträger kommt es zu Inter-Träger-Interferenzen (engl. *Inter-Carrier Interference*, ICI) und Inter-Symbol-Interferenze (engl. *Inter-Symbol Interference*, ISI) aufweist. Dies geschieht überwiegend bei der Wahl falscher Abtastzeitpunkte, bei einem Drift der Oszillatoren oder durch Addition von Reflexionen bei einem Mehrwegekanal [Wu04].

Abbildung 3.15 zeigt das in dieser Arbeit angenommene Empfängermodell. Das gesendete Signal $s(t)$ wird durch den Mehrwege-Übertragungskanal $h(t)$ gesendete und additives Rauschen $w(t)$ hinzugefügt. Das am Empfänger anliegende Signal $y(t) = s(t) * h(t) + w(t)$ wird durch das AFE bandbegrenzt, abgetastet und erzeugt das komplexwertige Basisbandsignal $r(n) = r_r(n) + jr_i(n)$. Die Zeit-Synchronisation berechnet aus diesem Signal den Zeitpunkt des ersten verwendbaren Präambelsymbols D_{pream}, der als Startpunkt für die FFT dient. Die komplexwertige 128-Punkt-FFT transformiert $r(n)$ in den Frequenzbereich und erzeugt das Frequenzsignal $R(k)$, wobei $k \in [-63, 64]$ der Subträger-Index im Basisband ist und über den gesamten Frequenzbereich von 25 MHz in Abbildung 3.6 läuft. Aus der transformierten Präambel $R_{\text{Pream}(k)}$ schätzt die

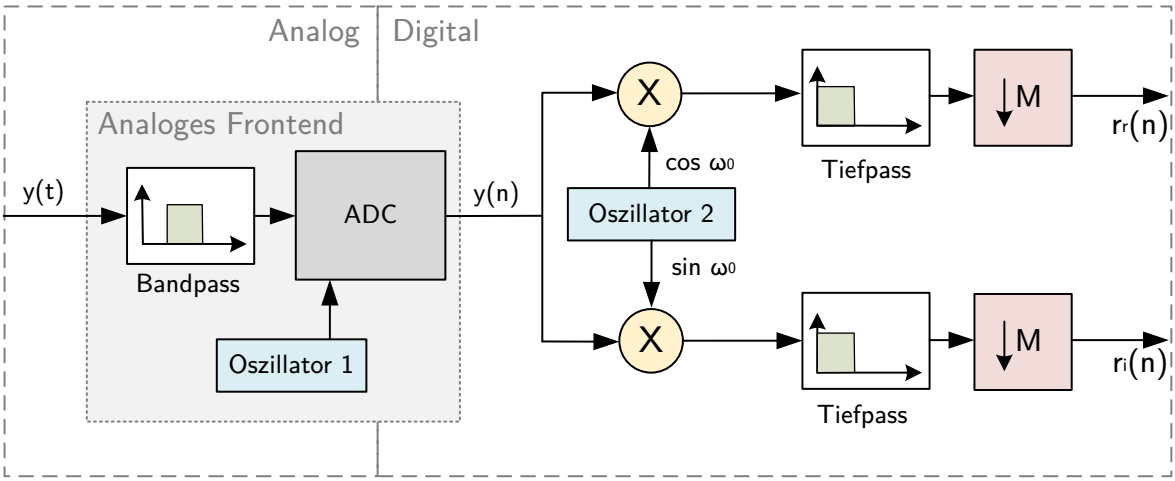

Abbildung 3.16: *Schematischer Aufbau des Analog-Frontends und der anschließenden Basisbandkonvertierung durch einen Digital Down Converter. Dabei werden der Realteil (oben) und der Imaginärteil (unten) getrennt voneinander gefiltert und Unterabgetastet [Tei09].*

Kanalschätzung die Kanalkorrekturkoeffizienten $\hat{G}(k)$ für alle Frequenzträger k. Diese werden mit den nachfolgenden Kontrollinformations- und Nutzdatensymbole $R_{\text{Data}}(k)$ multipliziert. Abschließend wird das synchronisierte Signal $\hat{S}(k)$ an die Demodulation übergeben und von dort, wie in Abschnitt 3.2 beschrieben, weiter decodiert.

Die Verschiebung des Empfangssignals $y(t)$ ins Basisband kann entweder analog vor oder digital nach der Analog-Digital-Wandlung geschehen [Löh00]. Da das Ziel dieser Arbeit ein möglichst vollständig integriertes System ist, wird hier ein digitaler Basisbandkonvertierer (engl. *Digital Down Converter*, DDC) verwendet. Abbildung 3.16 zeigt schematisch das analoge Frontend und den Aufbau des hier verwendeten DDC nach Teichert et al. [Tei09]. Das empfangene analoge Signal wird zunächst durch das AFE verarbeitet, wobei es durch einen Bandpass-Filter bandbegrenzt und anschließend mit Hilfe eines ADC abgetastet wird. Für die Abtastung wird eine Abtastfrequenz von 50 MHz durch den Oszillator 1 bereitgestellt. Das diskrete Signal $y(n)$ wird dann ins Basisband verschoben, indem mit einer komplexen Exponentialfunktion $e^{j\omega_0} = \cos(\omega_0) + j\sin(\omega_0)$ multipliziert wird. Die Frequenz des zweiten Oszillators ist hierbei die Trägerfrequenz des OFDM-Signals und beträgt 12,5 MHz. Anschließend werden Real- und Imaginärteil des Signals separat Tiefpass-gefiltert und unterabgetastet. Der Faktor der Unterabtastung ist hier $M = 2$, da die Bandbreite von 50 MHz im Passband auf 25 MHz im Basisband reduziert wird. Das resultierende komplexe Basisbandsignal $r(n)$ kann anschließend zur Zeit-Synchronisation verwendet werden.

Im Folgenden werden die möglichen Fehler, die nach der Basisbandkonvertierung im Empfangssignals vorhanden sein können, aufgezeigt und die Algorithmen zu deren Korrektur nach dem aktuellen Stand der Forschung vorgestellt.

1. Amplitudenskalierung und Phasendrehung der Subträger
2. Zeitlicher Symbolversatz
3. Versatz der Trägerfrequenzen (engl. *Carrier Frequency Offset*, CFO)
4. Versatz der Abtastratenfrequenzen (engl. *Sampling Clock Offset*, SCO)
5. IQ-Ungleichgewicht

Amplitudenskalierung und Phasendrehung der Subträger

Dieser Fehler wird im Wesentlichen durch den Einfluss der Kanalimpulsantwort und des Rauschens verursacht. Gleichung 3.8 zeigt diesen Einfluss auf das Basisbandsignal im Frequenzbereich nach dem Modell aus Abbildung 3.15.

$$R(k) = S(k)H(k) + W \qquad \forall k \in [-63, 64] \tag{3.8}$$

Der Frequenzträgerindex k gibt hier die Träger im Basisband an, wobei $k = 0$ die ursprüngliche Passband-Trägerfrequenz von 12,5 MHz repräsentiert. Der empfangene Frequenzträger $R(k)$ ist eine Multiplikation aus dem gesendeten Träger $S(k)$ und dem Übertragungskanalkoeffizienten $H(k)$ plus additivem weißen Rauschen W (engl. *Additive White Guassian Noise*, AWGN), das unabhängig vom Subträgerindex k ist. Eine Schätzung $\hat{H}(k)$ des Kanalkoeffizienten kann nach Gleichung 3.9 aus dem Empfangssignal $R(k)$ und den bekannten Präambelträgern $P(k)$ erfolgen.

$$\hat{H}(k) = R(k)\frac{1}{P(k)} \qquad \forall k \in [-63, 64] \tag{3.9}$$

Durch Aufteilung der gesamten Frequenzbandbreite in einzelne orthogonale Frequenzträger, kann der Einfluss der Kanalfrequenzantwort $H(k)$ für einen Frequenzträger k durch einen einzelnen komplexen Kanalkoeffizienten repräsentiert werden [Hsi98]. Zur Korrektur des Kanaleinflusses kann somit ein einfacher Equalizer, der jeden Träger individuell korrigiert, verwendet werden. Diese Korrektur ist in Gleichung 3.10 gezeigt, indem ein Frequenträger $R(K)$ durch einen geschätzten Kanalkoeffizienten $\hat{H}(k)$ dividiert wird und so die Amplitudenskalierung und Phasendrehung korrigiert.

$$\hat{R}(k) = \underbrace{\frac{1}{\hat{H}(k)}}_{\hat{G}(k)} R(k) = \hat{S}(k) + \hat{W}(k) \qquad \forall k \in [-63, 64] \tag{3.10}$$

Dabei ist $\hat{S}(k)$ das geschätzte fehlerbehaftete Sendesignal und $\hat{W}(k) = W\hat{G}(k)$ das durch den Equalizer gefärbte Rauschen, das nun vom Subträger k abhängt [Chi12]. Die Rauschskalierung ist vor allem bei *Fading*-Kanälen mit steilen Einbrüchen in der Kanalfrequenzantwort problematisch, da die Inversion der geschätzten Kanalkoeffizienten $\hat{G}(k) = \hat{H}(k)^{-1}$ zu einer hohen Verstärkung des Rauschens führt. Gleichung 3.11 zeigt einen weiteren Equalizer, der zusätzlich das SNR als Skalierungsfaktor mit berücksichtigt, um diese Verstärkung zu vermeiden.

$$\hat{R}(k) = \underbrace{\frac{\hat{H}^*(k)}{|\hat{H}(k)|^2 + \frac{1}{\mathrm{SNR}}}}_{\hat{G}(k)} R(k) \qquad \forall k \in [-41, 42] \tag{3.11}$$

Dabei ist $(\cdot)^*$ die komplexe Konjugation eines Wertes und $|\cdot|$ der Betrag einer komplexen Zahl. Bei geringer Sendeleistung oder starkem Rauschen wird eine Überhöhung der Amplitude durch das inverse SNR im Nenner limitiert. Für geringes Rauschen oder großes SNR konvertiert $\hat{G}(K)$

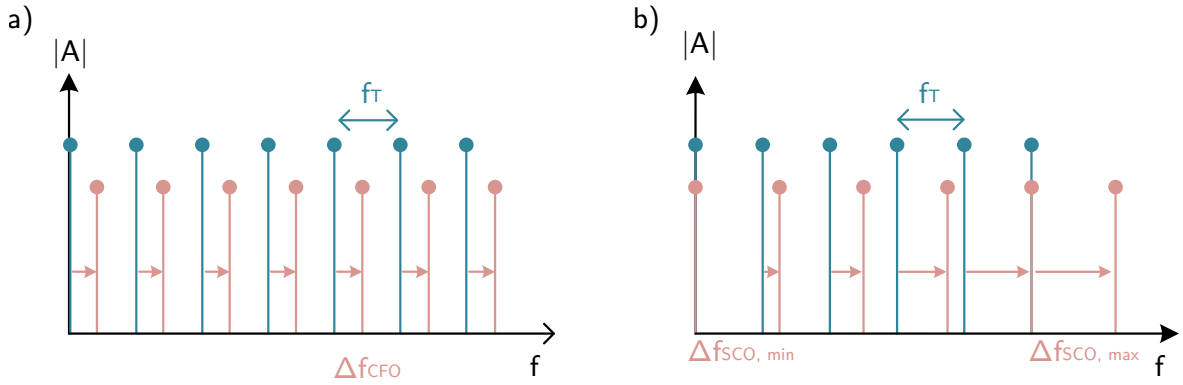

Abbildung 3.17: *Gesendete (blau) und empfangene Frequenzträger (rot) mit konstantem Frequenzträgerversatz durch SCO in a) und inkrementellem Trägerversatz durch CFO in b). Die Amplitudenskalierung dient der besseren Unterscheidung zwischen gesendeten und empfangenen Trägern.*

in Gleichung 3.11 gegen das $\hat{G}(K)$ in Gleichung 3.10 [Chi12]. Die dafür benötigte Schätzung des SNR pro Träger lässt sich nach Ren *et al.* [Ren09] aus Gleichung 3.12 berechnen.

$$\mathrm{SNR}_k = \frac{P_{R,k}}{P_W} - 1 \tag{3.12}$$

Dabei ist k der Subträgerindex, $P_{R(k)}$ die Leistung des Empfangssignals $R(k)$ und P_W der Rauschleistungsanteil des Signals $R(k)$. Die Gesamt- und Rauschleistung können mit Hilfe der Gleichungen 3.13 und 3.14 geschätzt werden.

$$P_{R(k)} = \frac{1}{U} \sum_{u=0}^{U-1} \|R_{u,k}\|^2 \tag{3.13}$$

$$P_W = \frac{1}{2K} \sum_{k=0}^{K-1} \|R_{u,k} - R_{u+1,k}\|^2 \tag{3.14}$$

Hier ist U die Anzahl der Wiederholungen der Präambel, die verwendet wurden und K die Anzahl der Subträger. Das Rauschen wird als weißes Rauschen angenommen und ist somit unabhängig von den Trägern und wird über diese summiert. Bei beiden Gleichungen wird die Differenz aufeinander folgender Kopien der Präambel verwendet, um einerseits durch Subtraktion das Rauschen P_W und anderseits durch Mittelwertbildung die Empfangsleistung $P_{R(k)}$ zu berechnen.

Zeitlicher Symbolversatz

Der zeitliche Symbolversatz ist ein Fehler, bei dem der Start eines Symbols um ein oder mehrere Werte versetzt falsch detektiert wird. Der korrekte Startzeitpunkt eines Signals kann mit Hilfe der Synchronisation im Zeitbereich detektiert werden. Für die in diesem Standard definierte Präambel mit 6 positiven und 1,5 negativen Wiederholungen (vgl. Abbildung 3.11) eignen sich korrelationsbasierte Verfahren zur Detektion am besten. Dabei wird zwischen autokorrelationsbasierten,

kreuzkorrelationsbasierten oder kombinierten Auto- und Kreuzkorrelationen unterschieden. Keller et al. [Kel01] geben eine Übersicht über Algorithmen für die Implementierung dieser Korrelationen. Da die Präambelsymbole allen Stationen bekannt sind, eignen sich Optimalfilter (engl. *Matched Filter*, MF) am besten für das hier zu implementierende System, da sie das Empfänger-SNR optimieren [Pus07].

$$C_{\mathrm{NCC}}(m) = \frac{\overbrace{\sum_{i=0}^{N-1} r(n-m)p(n)}^{C_{rp}(m)}}{\underbrace{\sqrt{\sum_{i=0}^{N-1} r(n-m)^2}}_{\sigma_r(m)} \underbrace{\sqrt{\sum_{i=0}^{N-1} p(n)^2}}_{\sigma_p}} = \frac{C_{rp}(m)}{\sigma_p \sigma_r(m)} \tag{3.15}$$

$$D_{\mathrm{Pream}} = \begin{cases} 1 & \text{wenn } C_{NCC}(m) \geq S_{th}, \\ 0 & \text{sonst.} \end{cases} \tag{3.16}$$

Der *Matched Filter* in dieser Arbeit basiert auf einer normierten Kreuzkorrelation (engl. *Normalized Cross-Correlation*, NCC) nach Gleichung 3.15. Dabei ist $p(n)$ die bekannte $N = 128$ Werte lange, komplexe Präambel in Basisband-Repräsentation und $r(n) = s(n) * h(n) + w(n)$ das komplexe Basisbandsignal nach dem DDC. Der Nenner von Gleichung 3.15 sind die Leistungen der Präambel σ_p und des aktuellen Ausschnitts des Empfangssignals $\sigma_r(m)$. Da $p(n)$ konstant ist, ist auch σ_p konstant und kann als Konstante vorab berechnet werden. Der Wertebereich von $C_{\mathrm{NCC}}(m)$ liegt wegen der Energienormalisierung zwischen null und eins. Ob eine Präambel im Empfangsfenster vorliegt, wird durch eine schwellenwertbasierte Entscheidung festgelegt. Gleichung 3.16 zeigt die binäre Entscheidung D_{Pream} zur Detektion einer Präambel in Abhängigkeit der Schwelle S_{th} [Tur60].

Versatz der Träger- und Abtastfrequenzen

Der Versatz von Träger- und Abtastfrequenzen entsteht durch Ungenauigkeiten der realen Oszillatoren auf Sende- und Empfangsseite (vgl. Abbildung 3.16). Dies resultiert in einer Verschiebung und Skalierung der Frequenzträger (CFO) und einem Drift der Frequenzträger über das Frequenzband (SCO), wie in Abbildung 3.17 gezeigt. Der CFO in Abbildung 3.17 a) entsteht durch Oszillatorungenauigkeiten der Mischerfrequenz im DDC (siehe Oszillator 2 in Abbildung 3.16), so dass das Passbandsignal nicht exakt in den Ursprung verschoben wird. Die Ungenauigkeit eines Oszillators wird typischerweise in Anteilen pro Millionen (engl. *parts per million*, ppm) angegeben. Pollet et al. [Pol95] geben zur Berechnung der Degradation des SNR durch CFO eine Approximation nach Gleichung 3.17 an.

$$D_{SNR,CFO}(\varepsilon) \approx \frac{10\pi^2}{3\ln(10)} \left(\frac{f_0}{\Delta_f}\varepsilon\right)^2 \frac{E_s}{N_0} \tag{3.17}$$

Hierbei ist $ln(\cdot)$ der natürliche Logarithmus, $f_0 = \omega_0/(2\pi)$ die Trägerfrequenz, E_s/N_0 das SNR des Symbols und ε der relative Frequenzfehler des Oszillators.

Der SCO in Abbildung 3.17 b) entsteht durch die Abweichung der Oszillatorfrequenzen zwischen Sender und Empfänger. Dies resultiert in unterschiedlichen Abtastfrequenzen am AFE (siehe Oszillator 1 in Abbildung 3.16), was zu einer Stauchung oder Streckung des Frequenzbandes führt. Für die Degradation des SNR durch SCO geben Pollet et al. [Pol94] ebenfalls eine approximative Berechnung nach Gleichung 3.18 an. Dabei ist k der Frequenzträgerindex im Frequenzbereich.

$$D_{\text{SNR, SCO}}(\varepsilon,k) \approx \frac{10\pi^2}{3\ln(10)} \left(\frac{k}{N}\frac{f_0}{\Delta_f}\varepsilon\right)^2 \frac{E_s}{N_0} \tag{3.18}$$

Um CFO und SCO kombiniert zu kompensieren, gibt es in der Literatur verschiedene Ansätze. Ein für diesen Standard vielversprechender Algorithmus ist der *Best Linear Unbiased Estimator* (BLUE), der auf sich wiederholenden, identischen Präambelsymbolen arbeitet [Mor99]. Dieser arbeitet im Zeitbereich und berechnet die auftretenden Phasendifferenzen der Präambelsymbole. Da der HomePlug-1.0-Standard $U = 7$ identische Symbole als Präambel vorgibt, eignet sich der BLUE-Algorithmus besonders gut, da er einen großen Korrekturbereich von bis zu $U/2 = 3{,}5$ Frequenzträgern hat. Die Phasendifferenz $\varphi(u)$ berechnet sich im BLUE-Algorithmus nach den Gleichungen 3.19 und 3.20.

$$\Phi_{\text{BLUE}}(u) = \frac{1}{R-uN} \sum_{m=uN}^{R-1} r(m)r^*(m-uN) \qquad 0 \le u \le K < U \tag{3.19}$$

$$\varphi(u) = [\angle\Phi_{\text{BLUE}}(u) - \angle\Phi_{\text{BLUE}}(u-1)]_{2\pi} \qquad 1 \le u \le K \tag{3.20}$$

Dabei ist N die Länge eines Präambelsymbols, R die Länge aller Präambelsymbole, K die Anzahl zu verwendender Präambelsymbole und $\Phi_{\text{BLUE}}(u)$ die Kreuzkorrelation zwischen zwei Präambelsymbolen. Mit steigendem Index u steigt auch der Abstand zwischen den Symbolen. $\Phi_{\text{BLUE}}(K)$ ist die Korrelation zwischen dem ersten und dem letzten Präambelsymbol. Mit Hilfe von Gleichung 3.21 kann der mittlere Subträgerversatz für alle Subträger ermittelt werden. Die Gewichte w_u der einzelnen Phasendifferenzen berechnen sich dabei nach Gleichung 3.22.

$$\hat{\Delta f}/f_s = \frac{U}{2\pi} \sum_{u=1}^{K} w_u \varphi(u) \tag{3.21}$$

$$w_u = 3\frac{(U-u)(U-u+1) - K(U-K)}{K(4K^2 - 6UK + 3U^2 - 1)} \tag{3.22}$$

Die Korrektur des geschätzten Subträgerversatzes $\Delta\hat{f}$ kann anschließend durch eine Multiplikation des Empfangssignals mit $e^{-j2\pi\Delta\hat{f}}$ erreicht werden.

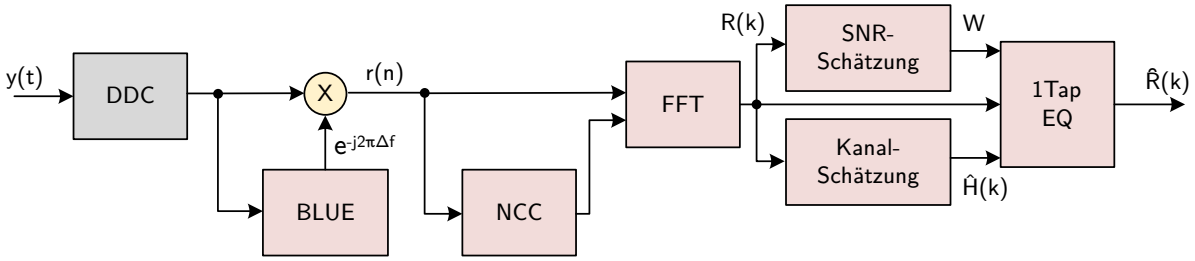

Abbildung 3.18: *Resultierendes Synchronisationssystem für diese Arbeit nach dem aktuellen Stand der Forschung vom analogen Eingangssignal bis zur Übergabe der korrigierten Frequenzträger an die Demodulation.*

IQ-Ungleichgewicht

IQ-Ungleichgewicht entsteht durch separate Verarbeitung des In-Phase- (I) und Quadratur-Anteils (Q) des Empfangssignals als analoge Komponenten. Hierbei wird das Q-Signal um 90° gegenüber dem I-Signal gedreht, was durch die Verwendung von analogen Komponenten störungsbehaftet ist. Diese nicht-perfekte Drehung um $90° + \Delta_Q$ führt zu einer Verzerrung der Konstellationspunkte und somit zu möglichen Fehlern bei der Demodulation. Da in dieser Arbeit ein DDC-System verwendet wird, das eine rein digitale Mischung der I- und Q-Komponenten unter Verwendung des identischen Oszillatorsignals durchführt, kann ein IQ-Ungleichgewicht in dieser Arbeit vernachlässigt werden.

Das resultierende Synchronisationssystem des Empfängers zur Kompensation der oben beschriebenen Effekte ist in Abbildung 3.18 gezeigt. Nach der Konvertierung des Empfangssignals ins Basisband wird mit Hilfe des BLUE eine CFO- und SCO-Schätzung durchgeführt und korrigiert. Anschließend wird auf dem entzerrten Signal mit Hilfe einer normierten Kreuzkorrelation der Startpunkt für das FFT-Fenster bestimmt und das komplexe 128-Werte lange Signal transformiert. Nach der FFT wird im Frequenzbereich eine Schätzung des SNR und des Übertragungskanals für jeden Träger durchgeführt und mit Hilfe eines 1-Tap-Equalizers nach Gleichung 3.11 kompensiert. Die daraus geschätzten Subträger $\hat{R}(k)$ können anschließend demoduliert und decodiert werden.

3.3. MATLAB-Referenzimplementierung

Um die im vorherigen Abschnitt beschriebenen Algorithmen charakterisieren und testen zu können, wurde zunächst eine Implementierung der PHY-Schicht als Referenz in MATLAB erstellt. Diese mathematisch orientierte Sprache erlaubt es, Algorithmen schnell umzusetzen (engl. *Rapid Prototyping*), da durch diverse Erweiterungsmodule bereits viele Standardalgorithmen der Signalverarbeitung enthalten sind, wie z.B. FFT, BPSK-/QPSK-Modulatoren oder Viterbi-Decoder. Damit eignet sich MATLAB besonders, um die in Abschnitt 3.2.1 vorgestellten Algorithmen zur Synchronisation im PHY-Schicht-Decoder zu charakterisieren und zu verifizieren. Hierbei werden die verschiedenen Freiheitsgrade der Implementierung bezogen auf ihre Fehlerkorrekturkapazität, Ausführungszeit oder Rechengenauigkeit untersucht. Zusätzlich dient die MATLAB-Implementierung als Referenz der Genauigkeit und zur Verifikation der Hardware-Implementierungen in den folgenden Kapiteln.

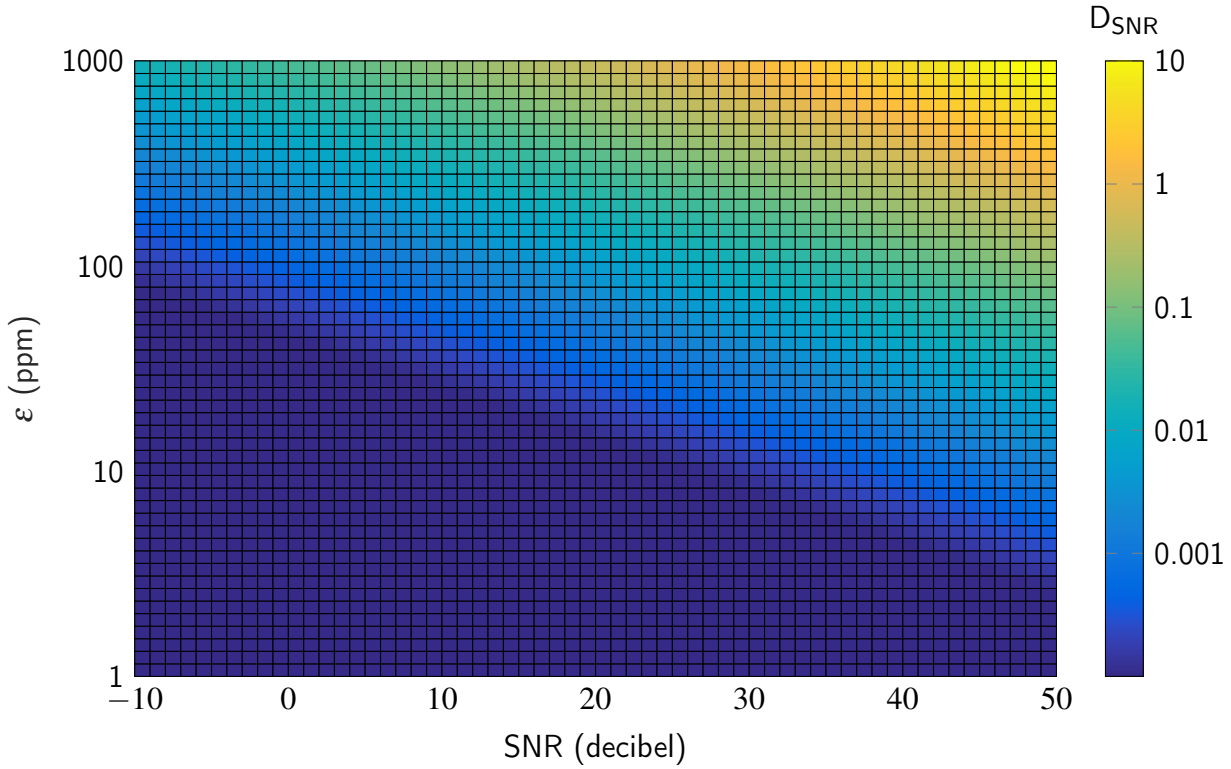

Abbildung 3.19: *SNR-Degradation durch CFO über die Oszillator-Ungenauigkeit ε und SNR am Empfänger. Alle drei Achsen sind in logarithmischer Darstellung.*

Der Encoder wurde nach den Vorgaben des Standards, wie in Abschnitt 3.1.2 gezeigt, umgesetzt und arbeitet blockweise die einzelnen Funktionen für die FC-Daten und Nutzdaten ab (siehe Abbildung 3.5). Als Eingabe werden diesem die *Header*- und *Trailer*-Informationen sowie die Nutzdaten und die Übertragungsparameter übergeben. Die Ausgabe ist ein nach Abbildung 3.11 geformtes digitales Powerline-Signal ohne Berücksichtigung des AFE. Da diese Blöcke im Standard definiert sind gibt es hier keine Freiheitsgrade für die Implementierung. Die Verifikation aller Blöcke erfolgte gegen die im Standard beschriebenen Algorithmen. Lediglich die Quantisierung der Phaseninformationen und des reellen Sendesignals konnte frei gewählt werden und wurde hier für eine Genauigkeit von 10 bit am AFE ausgelgt, was einer Sendeleistung von maximal $20 * log_{10}(2^{-10}) = -60\,\text{dB}$ entspricht. Hierfür wurde eine Konvertierung der Werte mit doppelter Genauigkeit in entsprechende Fixpunktformate durchgeführt, so dass an jeder Stelle ab dem *Mapping* diese Genauigkeit erreicht wird.

Der Decoder arbeitet ebenfalls blockweise auf den Eingangsdaten und erhält als einzigen Übergabeparameter die aktuelle *ToneMap*. Alle weiteren Parameter lassen sich aus dem Empfangssignal decodieren. Die Decodierung beginnt mit dem in Abschnitt 3.2.1 vorgestellten Synchronisationssystem. Die verfügbaren Freiheitsgrade sind hier das Design des Tiefpassfilters in der DDC, der Schwellenwert für die Detektion eines Präambelsymbols in der NCC und die Genauigkeit der SNR- und Kanalschätzung im Frequenzbereich. Vor der Evaluation dieser Parameter kann das Synchronisationssystem in Abbildung 3.18 mit Hilfe einiger Randbedingungen reduziert werden.

Abbildung 3.19 zeigt exemplarisch die SNR-Degradation nach Gleichung 3.17 aus Abschnitt 3.2.1 für eine Oszillatorungenauigkeit zwischen 1 bis 1000 ppm und SNR-Werte von −10 dB

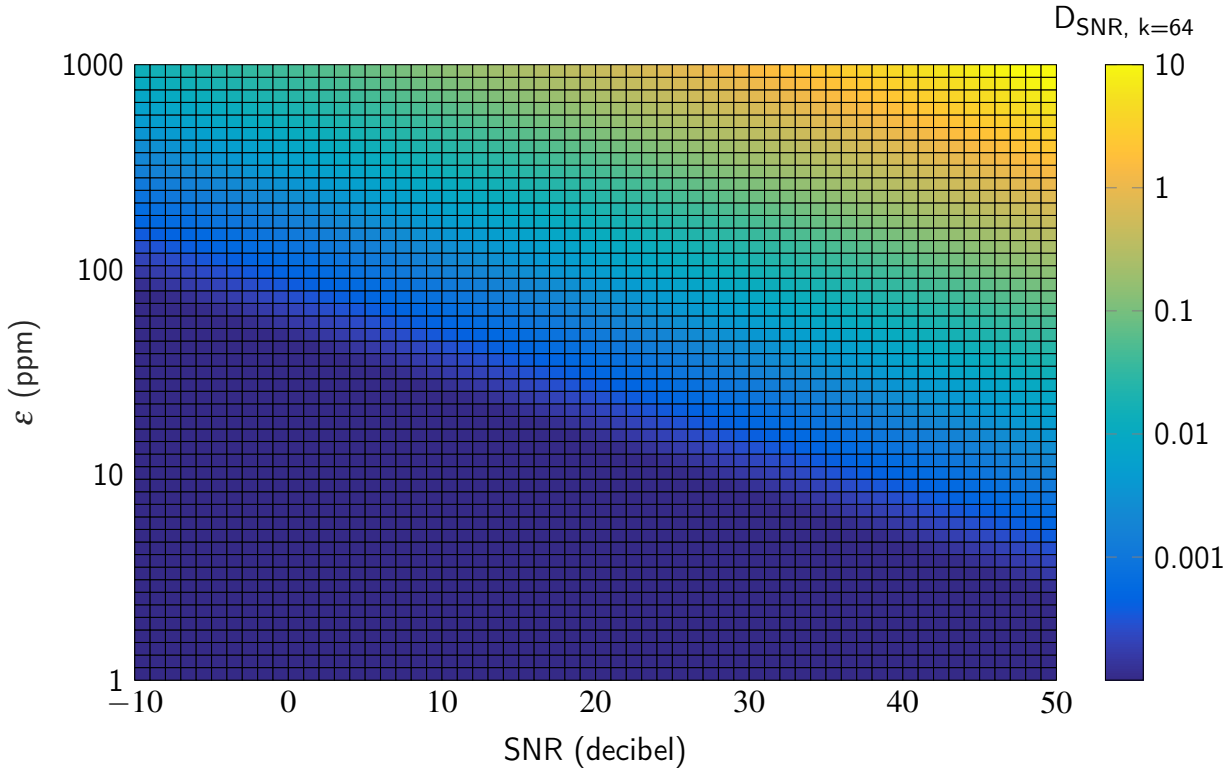

Abbildung 3.20: *SNR-Degradation durch SCO über Oszillator-Ungenauigkeit ε und SNR am Empfänger für den Frequenträger $k = 64$ mit höchster Frequenz. Alle drei Achsen sind in logarithmischer Darstellung.*

bis 50 dB bei gegebenem Trägerabstand von $\Delta_f = 195{,}31\,\text{kHz}$. Generell ist der CFO dabei unabhängig von der Trägerposition im Frequenzspektrum. Zu erkennen ist, dass die Degradation mit steigendem SNR und mit höherer Oszillatorungenauigkeit zwar zunimmt, aber für Werte von $\varepsilon \leq 250$ oder $\text{SNR} \leq 30\,\text{dB}$ kleiner als 1 dB ist.

Ein ähnlicher Zusammenhang kann in Abbildung 3.20 für SCO gezeigt werden, wo die Degradation des SNR über die Oszillatorungenauigkeit ε und das Empfangs-SNR für den höchsten Frequenzträgerindex $k = 64$ gezeigt ist. Dieser Träger ist am stärksten vom SCO betroffen, da, wie in Abbildung 3.17 b) gezeigt, die Skalierung der Trägerfrequenzen zu den höheren Frequenzen zunimmt. Die gezeigten Ergebnisse der SCO-Degradation sind vergleichbar gering, wie die Werte aus Abbildung 3.19.

In Abbildung 3.21 ist zusätzlich die Degradation des SNR für alle Frequenzträger nach Gleichung 3.18 gezeigt. Hierbei wurde für ein festes SNR von 40 dB die Degradation über alle Frequenzträger k und die Oszillatorungenauigkeit ε berechnet. Erkennbar ist, dass die Degradation über die Träger symmetrisch von $k = 0$ zu den höheren Frequenzen ansteigt. Frequenzträger $k = 0$ hat dabei unabhängig der Oszillatorungenauigkeit keine Verschiebung und auch keine Degradation.

In dieser Arbeit kann angenommen werden, dass der Oszillatordrift selbst bei Temperaturen von 175 °C einen Wert von 100 ppm nicht überschreitet. Damit liegt die SNR-Degradation durch SCO und CFO addiert selbst für hohe SNR-Werte noch unter 1 dB. Das entspricht einer maximalen Degradation von 2 % und einer mittleren Degradation von 0,05 % für den SNR-Bereich von −10 dB bis 20 dB, der realistisch für das Tiefbohrsystem angenommen werden kann. Zu erwarten

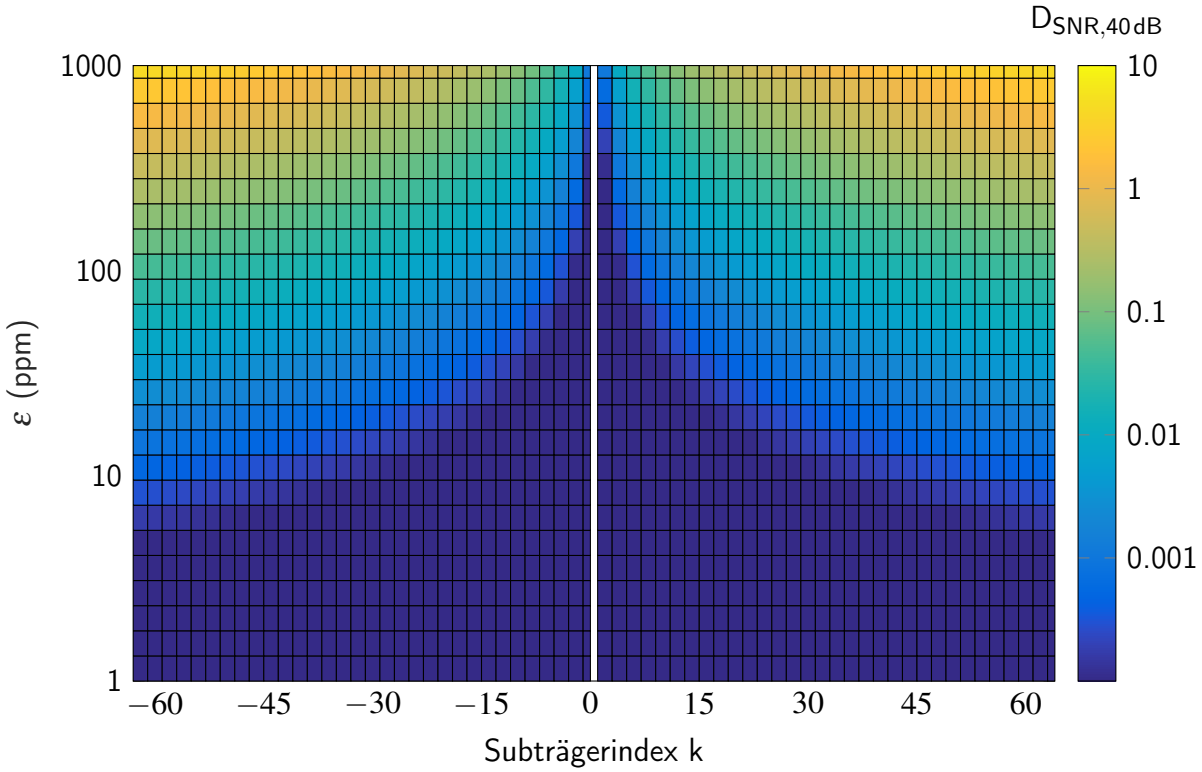

Abbildung 3.21: *SNR-Degradation durch SCO über Oszillator-Ungenauigkeit ε und Subträgerindex k. Alle drei Achsen sind in logarithmischer Darstellung. Der Träger mit Index $k = 0$ ist nicht vom SCO betroffen.*

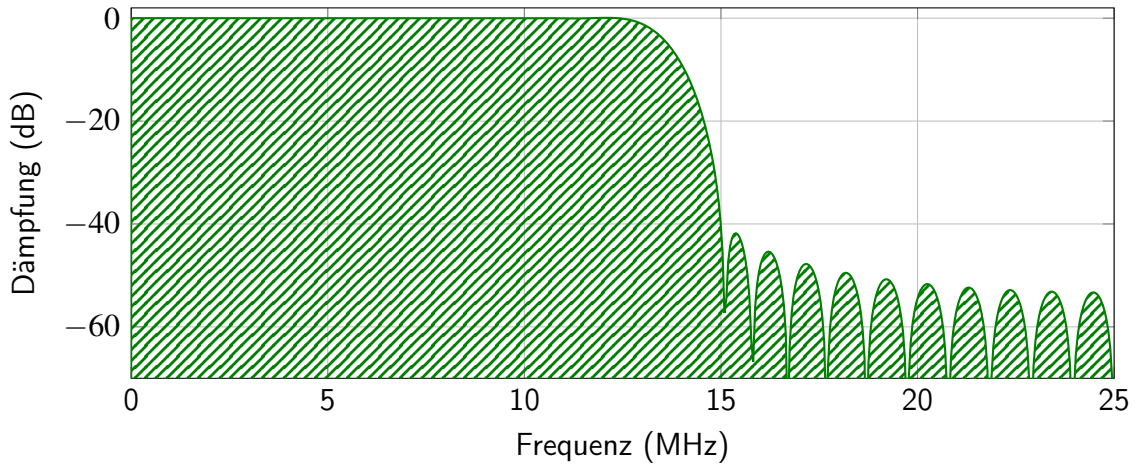

Abbildung 3.22: *Übertragungsfunktion des DDC-Tiefpassfilters zweiter Ordnung mit Greunzfrequenz 12,5 MHz und mindestens 40 dB Dämpfung der Nebenmaxima.*

ist, dass das thermische Grundrauschen der Hochtemperaturumgebung diese Degradationseffekte überdeckt. Grund für diese geringen Störungen ist hauptsächlich der große Trägerabstand von $\Delta_f = 195{,}3125\,\text{kHz}$ im HomePlug-1.0-Standard. Modernere Standards, wie der HomePlug AV2 verwenden im fast identischen Frequenzspektrum 917 Frequenzträger mit einem Abstand von nur 24,414 kHz. Daher kann in dieser Arbeit auf die Korrektur dieser beiden Effekte und die Implementierung des BLUE-Algorithmus in Abbildung 3.18 verzichtet werden.

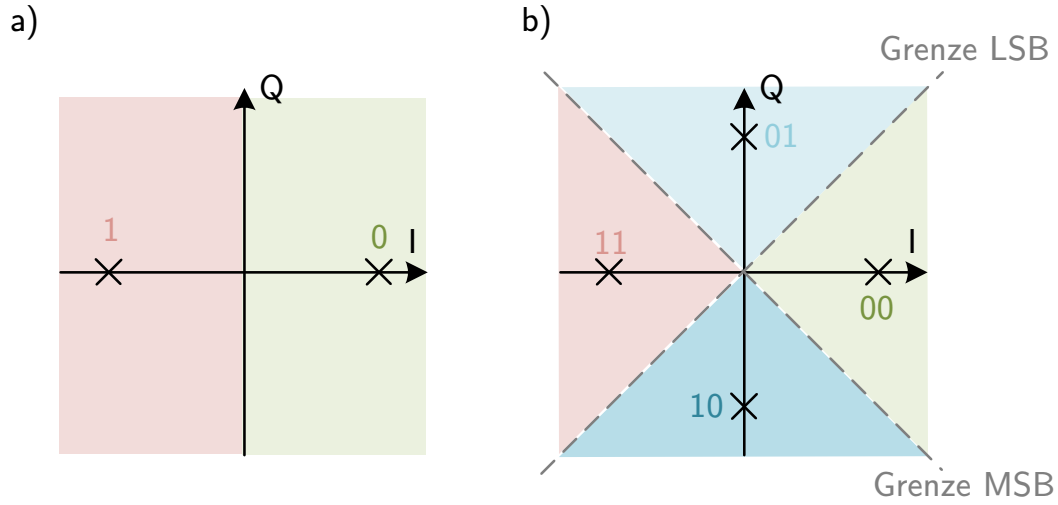

Abbildung 3.23: *Decodierbereiche der Phaseninformationen am Demodulator: in a) für die BPSK und in b) für die QPSK. Die farbigen Bereiche zeigen den Zuordnungsbereich der jeweiligen Konstellationspunkte und die Bereichesübergänge die Grenzen der Decodierung für ein Bit(-paar).*

Für das Filter im DDC-Modul wurde der Filter-Designer von MATLAB verwendet, um ein Filter mit minimaler Länge, aber mindestens 40 dB Nebenmaximaunterdrückung zu erreichen. Die Grenzfrequenz wurde auf 12,5 MHz festgelegt und die Steilflankigkeit des Filters als Freiheitsgrad für die Reduzierung der Koeffizientenanzahl verwendet. Das resultierende linearphasige Filter zweiter Ordnung ist in Abbildung 3.22 dargestellt und verwendet nur 46 Koeffizienten für die Filterung. Das Filter hat eine geringe Welligkeit im Durchlassbereich und −41,8 dB Dämpfung der Nebenmaxima. Die restlichen Komponenten des DDC wurden in MATLAB, wie in Abschnitt 3.2.1 beschrieben, implementiert.

Die normierte Kreuzkorrelation wurde nach Gleichung 3.15 direkt in MATLAB umgesetzt. Die Evaluation des Schwellwertes für die Entscheidung, ob eine gültige Präambel anliegt, ergab einen Wert von $S_{th} = 0,5$, für verschiedene Kanaldämpfungen und Rauschleistungen am Eingang des Systems. Eine Präambel wird dabei als gültig erkannt, wenn mindestens zwei positive und ein negativer Wert die Schwelle S_{th} über- oder im negativen Fall unterschreiten. Die Ausgabe dieser Zeit-Synchronisation ist der Startpunkt der Präambel und der FC-Daten.

Die detektierte Präambel wird anschließend Fourier-transformiert und im Frequenzbereich für die Kanal- und SNR-Schätzung verwendet. Diese sind, wie in Abschnitt 3.2.1 gezeigt, nach den Gleichung 3.11 bis 3.14 implementiert und bestimmen die Faktoren zur Kanalkorrektur der FC- und Nutzdaten.

Die anschließende Demodulation wurde sowohl als *Hard Decision*- als auch als *Soft Decision*-Variante umgesetzt. Hier soll der Einfluss der Entscheidung und die Anzahl der verwendeten Bits bei der *Soft Decision*-Decodierung für den nachfolgenden Viterbi-Decoder untersucht werden. Dafür wird zunächst die Referenzphase bei kohärenter Modulation oder die vorangegangene Phase bei der differenziellen Modulation subtrahiert und dann eine Zuordnung zu den Konstellationspunkten anhand von Entscheidungsgrenzen durchgeführt.

Abbildung 3.23 zeigt die Decodierbereiche der BPSK- oder QPSK-Modulation. Die farbigen Bereiche zeigen die Zuordnung der empfangenen Werte zu den zwei oder vier Konstellationspunkten. Die Übergänge der farbigen Bereiche geben damit die Entscheidungsgrenzen der Demodulation

Tabelle 3.1: *Hier verwendete Codierung und Bedeutung der Soft-Decision-Bits für eine 3 bit-Quantisierung.*

Dezimal	Binär	Bedeutung
0	000	Sicherste null
1	001	Sichere null
2	010	Schwache null
3	011	Schwächste null
4	100	Schwächste eins
5	101	Schwache eins
6	110	Sichere eins
7	111	Sicherste eins

an. Abbildung 3.23 a) zeigt diese Grenze für die BPSK-Demodulation, die entlang der Ordinate verläuft. Damit werden alle Träger mit positivem Realteil als logische null und alle Träger mit negativem Realteil als logische eins decodiert.

Bei der in Abbildung 3.23 b) gezeigten QPSK-Demodulation werden zwei Grenzen für die beiden zu decodierenden Bits verwendet. Anhand dieser Grenzen lassen sich das höchstwertige Bit (engl. *Most Significant Bit*, MSB) und das niederwertigste Bit (engl. *Least Significant Bit*, LSB) separat demodulieren. Auf Grund dieser beiden Grenzen entstehen vier Quadranten (rot, gelb, grün, blau), die die vier möglichen Bitkonstellationen decodieren.

Der Störabstand Δ_d von einem Referenzkonstellationspunkt zur nächsten Entscheidungsgrenze entspricht bei BPSK-Modulation der Amplitude des Realteils und bei QPSK-Modulation $1/\sqrt{2}$-mal dieser Amplitude, also $20\log_{10}(1/\sqrt{2} = 3\,\text{dB}$ [Ul 13]. Wird der Amplituden- oder Phasenfehler größer als dieser Abstand, führt eine hart decodierte Entscheidung direkt zu einem falsch decodierten Bit. Die passiert insbesondere bei einem geringen SNR oder starkem Phasenrauschen.

Implementiert wurde die *Soft Decision*-Demodulation als approximierte LLR nach Gleichung 3.7 und unterstützt sowohl eine Quantisierung bis 3 bit als auch eine unquantisierte Demodulation. Die sich aus den 3 bit ergebenden Werte von null bis sieben, geben an, wie sicher die Decodierung des jeweiligen Bits auf Basis des Abstands zur nächsten Entscheidungsgrenze war. Die Codierung einer 3 bit-Quantisierung ist in Tabelle 3.1 dargestellt von der sichersten null zur sichersten eins in aufsteigender Reihenfolge.

Der anschließende FC-*Deinterleaver* ist, wie in Abschnitt 3.2 beschrieben, umgesetzt. Dabei ist der Schwellwert des *Deinterleavers* nach der Bit-Summierung für *Hard-Decision*-demodulierte Informationen $S_{\text{DI, hart}} > 1$ eine Eins oder $S_{\text{DI, hart}} \leq 1$ für eine Null. Für *Soft Decision* demodulierte Informationen ist der Schwellwert $S_{\text{DI, soft}} \geq 0$ für eine Null und $S_{\text{DI, soft}} < 0$ für eine Eins. Die Ausgabe des *Deinterleaver* ist dabei immer eine harte Entscheidung mit binären Werten. Auf Grund der bereits sehr hohen Redundanz der Kontrollinformationen, mit einem Faktor von 12, wurde hier auf eine *Soft-Decision*-Decodierung verzichtet und nur mit Hilfe der Redundanz Fehler detektiert und korrigiert.

Der für den Produktdecoder verwendete Algorithmus ist ein *Hard Decision* GMD-Decoder, der zeilen- und spaltenweise Syndrome berechnet, die Fehlerstellen direkt anzeigen [Eli54; Bio19].

Dieser arbeitet am effizientesten (bezogen auf Latenz und Komplexität) für die Decodierung von *Hard-Decision*-Informationen. Dieser wird mit zwei Iterationen pro Zeilen- und Spaltendecodierung berechnet und kann somit insgesamt bis zu 20 Einzelfehler detektieren und bis zu 10 Einzelfehler korrigieren. Zur Decodierung wird die Decodermatrix H verwendet, die die Bedingung $H^T \cdot G = \overrightarrow{0}$ erfüllt. Damit ergibt sich die Matrix H zu:

$$H = \left(\begin{array}{ccccc|ccccc} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ \underbrace{1 \quad 0 \quad 0 \quad 1 \quad 1}_{P} & \multicolumn{5}{c}{\underbrace{0 \quad 0 \quad 0 \quad 0 \quad 1}_{E}} \end{array}\right)$$

Eine Iteration des Decoders besteht aus einer zeilenweisen Multiplikation der die Bits vom *Deinterleaver* als $10x10$-Matrix K mit H, also $S_{\text{row}} = K \cdot H^T$. Ist das Syndrom S_{row} ungleich null, wurde ein Fehler in der korrespondierenden Zeile entdeckt. Der von null verschiedene Wert entspricht bei einem Einzelfehler exakt einer Spalte von H und kann entsprechend dieser Spalte direkt korrigiert werden. Ist das Syndrom ungleich der Spalten von H, ist in dieser Zeile mehr als ein Fehler enthalten und kann in dieser Iteration nicht korrigiert werden. Danach wird die korrigierte Matrix K spaltenweise überprüft und das Syndrom $S_{col} = H \cdot K$ berechnet. Genau wie bei der Zeile werden auch die Spalten des Syndroms, die ungleich null sind, gegen die Zeilen von H verglichen und eventuelle Fehler korrigiert. Diese Decodierung wird jeweils einmal pro Zeile und Spalte wieder holt.

Die beiden *Deinterleaver* und der *Descrambler* im Nutzdatenpfad wurden, wie in Abschnitt 3.2 beschrieben, umgesetzt. Für den ROBO-*Deinterleaver* wurde im Falle von *Hard-Decision*-Demodulation die identische Grenze, wie für den FC-*Deinterleaver* gewählt. Im Fall von *Soft Decision*-Demodulation wird der Mittelwert über alle vier Kopien der Daten gebildet und der Wert unquantisiert weitergegeben.

Für den Viterbi- und Reed-Solomon-Decoder wurden zu Evaluation die MATLAB-internen Funktionen der *Communications Toolbox* verwendet. Diese lassen sich durch Übergabeparameter auf im Standard geforderten Parameter anpassen und können sowohl mit *Hard-Decision*- als auch *Soft Decision*-Informationen arbeiten.

3.3.1. Evaluation der Parameter

Zur Evaluation der oben beschriebenen algorithmischen Parameter, wurde eine Designraumexploration (engl. *Design Space Exploration*, DSE) durchgeführt. Dabei wurden Parameterkombinationen getestet, mit dem Ziel einer möglichst geringe Bitfehlerrate (engl. *Bit Error Rate*, BER) bei niedrigen Signal-Rausch-Verhältnissen. Die folgenden Parameter wurden dabei untersucht:

- Modulationsart: ROBO, BPSK, QPSK, QPSK mit Punktierung
- *Hard-/Soft-Decision* Decodierung
- Quantisierung der *Soft-Decision* Informationen: 1, 2, 3 bit

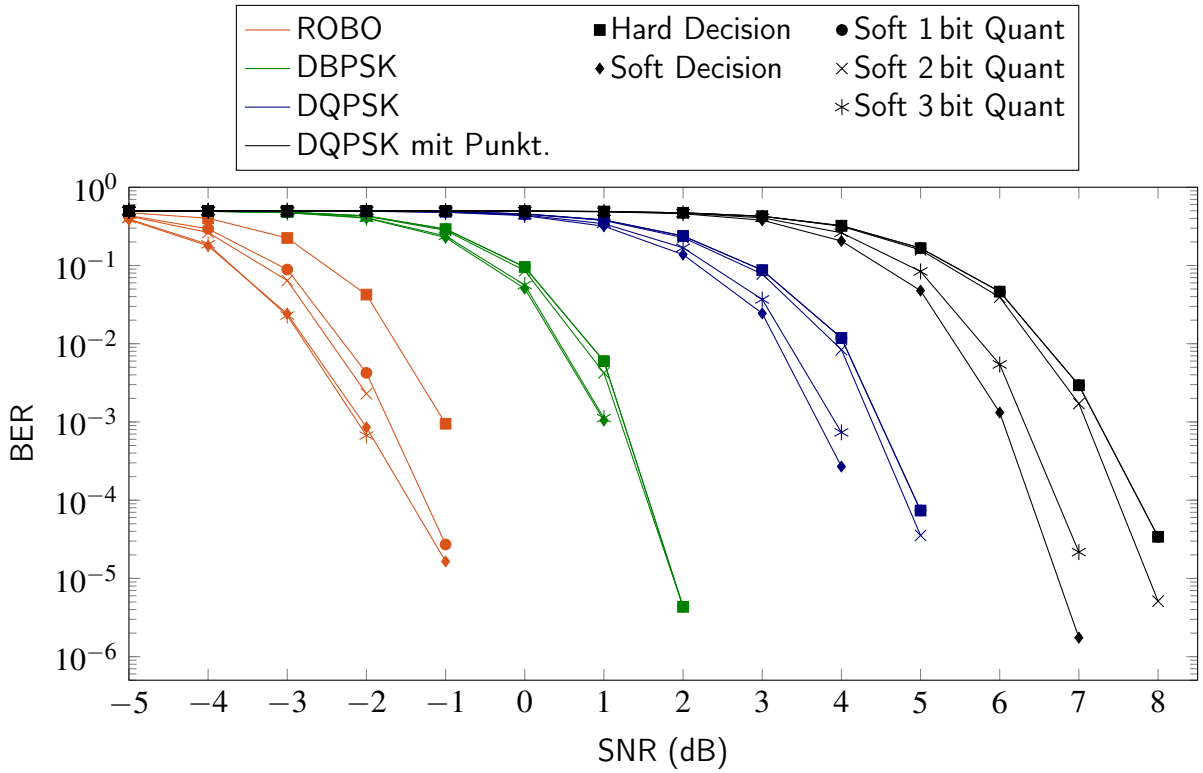

Abbildung 3.24: *Bitfehlerrate der einzelnen Modulationsarten über das Signal-Rausch-Verhältnis jeweils für einen Hard-Decision-, einen unquantisierten Soft-Decision- und drei quantisierte Soft-Decision-Decoder mit jeweils* $1-3$ bit.

Die Evaluierung erfolgt über das SNR am Empfänger, die Modulationsart und die Quantisierung am Demodulator und wertet die Bitfehlerrate aus, die am Ausgang des Decoders noch vorhanden sind. Dazu werden Pakete maximaler Länge (160 OFDM-Symbole) encodiert und parallel von fünf verschieden parametrisierten Decodern decodiert. Diese sind: ein *Hard-Decision*-Decoder, ein *Soft-Decision*-Decoder ohne Quantisierung und drei *Soft-Decision*-Decoder mit jeweils eins, zwei oder drei Bit quantisierten Wahrscheinlichkeitswerten. Die Quantisierung am Viterbi-Decoder ist für eine spätere Hardware-Implementierung besonders relevant, da dieser Block das größte Teildesign im Decoder darstellt. Hier gibt es einen Abtausch der Größe des Hardware-Designs gegenüber der Paketfehlerrate am Decoder. Für das Empfänger-SNR wurde der Bereich von -8 bis 8 dB untersucht und für jeden SNR-Wert 5000 Iteration mit zufälligen Eingangsinformationen ausgeführt. In dieser ersten Untersuchung wurde nur additives weißes Rauschen verwendet und kein Übertragungskanal.

Die Ergebnisse der Evaluation sind in Abbildung 3.24 dargestellt. Diese zeigt vier Gruppen von Kurven, die den eingestellten Modulationsarten am Sender entsprechen. Die rote Gruppe zeigt den ROBO-Modus, die grüne BPSK-Modulation, die blaue QPSK-Modulation und die schwarze QPSK-Modulation mit Punktierung. Die BPSK- und QPSK-Modulation haben, einen Abstand von etwa 3 dB, was sich auf den oben gezeigten Störabstand der Modulationsverfahren zurückführen lässt. Der ROBO-Modus ist weitere 3 dB robuster gegenüber Rauschen, was vor allem an dem Faktor vier an Redundanz liegt. Am *Deinterleaver* wird der Mittelwert über die zyklisch identischen Symbole gebildet und so das Rauschen reduziert. Die Punktierung verliert gegenüber der reinen QPSK-Modulation etwa 3 dB an Robustheit, da hier ein Drittel der

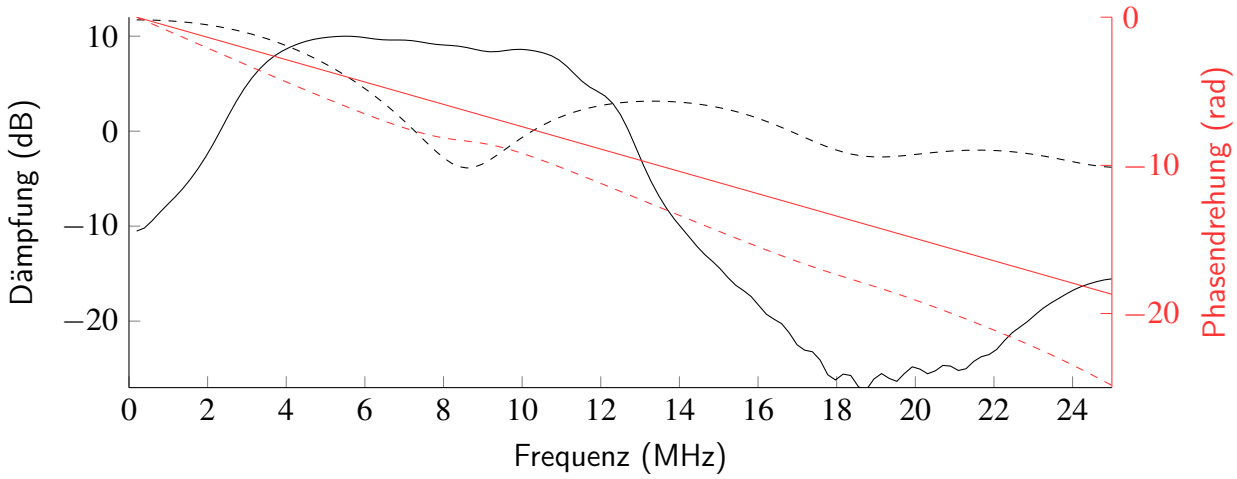

Abbildung 3.25: *Dämpfung (schwarz) und Phase (rot) zweier realer Powerline-Übertragungskanäle: eine 100 m-langer Stromversorgungsleitung entlang eines Bohrstranges bei 175 °C (durchgezogene Linie) und ein Mehrwege-Rayleigh-Fading-Kanal mit gemessenen Dämpfungsparametern nach [Zim02] (gestrichelte Linie).*

am Faltungsencodierer erzeugten Redundanz verworfen werden und nicht zu Fehlerkorrektur verwendet werden können.

Jede der farbigen Gruppen besteht aus fünf Messreihen, die die fünf beschriebenen Decoder zeigen. In allen Gruppen haben die *Hard-Decision*-Decoder die geringste Robustheit gegenüber Rauschstörungen und der unquantisierte *Soft-Decision*-Decoder die Beste. Die Quantisierung der *Soft-Decision*-Informationen hat dabei einen linearen Einfluss auf die Qualität der Decodierung. Bei BPSK, QPSK und punktierter QPSK sind *Hard Decision*- und 1 bit-*Soft Decision*-Decodierung nahezu identisch in ihrer Robustheit. Im ROBO-Modus gibt es einen 1 dB-Abstand zwischen diesen beiden Decodern. Dies liegt daran, dass die Quantisierung der Informationen erst nach dem ROBO-*Deinterleaver* durchgeführt wird und somit der Decoder von der Mittelung der nicht quantisierten Amplitudenwerte profitiert. Bei der *Hard Decision*-Variante wird hier nur ein Mehrheitsentscheid über die Bits durchgeführt. Die 2 bit- und 3 bit-quantisierten Decoder sind robuster als die 1 bit-Quantisierung und im Fall der BPSK sogar schon identisch zu nicht quantisiertem *Soft Decision*.

Insgesamt zeigt diese Auswertung, dass die *Soft Decision*-Decodierung bis zu 2 dB mehr Robustheit gegenüber der *Hard Decision*-Decodierung bietet und das 3 bit-quantisierte Amplitudenwerte eine ähnliche Fehlerrate bietet, wie unquantisierte Werte.

Um das Verhalten der Decoder unter möglichst realitätsnahen Bedingungen zu evaluieren, wurden zwei weitere Untersuchungen mit Übertragungskanälen durchgeführt. Abbildung 3.25 zeigt zwei Übertragungskanäle mit ihrer Dämpfung (schwarz) und Phase (rot) über das Frequenzspektrum der Powerline-Kommunikation. Die durchgehenden Linien für Dämpfung und Phasendrehung sind ein realer gemessener Einwege-Powerline-Kommunikationskanal entlang eines Bohrstranges dargestellt. Dieser wurde unter Laborbedingungen bei 175 °C gemessen und weist starke Dämpfungen der Frequenzen ab 15 MHz auf. Die Phase dieses Übertagungskanals ist linear und somit die Gruppenlaufzeit null.

Die gestrichelten Linien sind ein Mehrwege-Rayleigh-Dämpfungskanal nach Messparametern, die von Zimmermann et al. [Zim02] präsentiert wurden. Dieses Modell hat 15 Pfade entlang

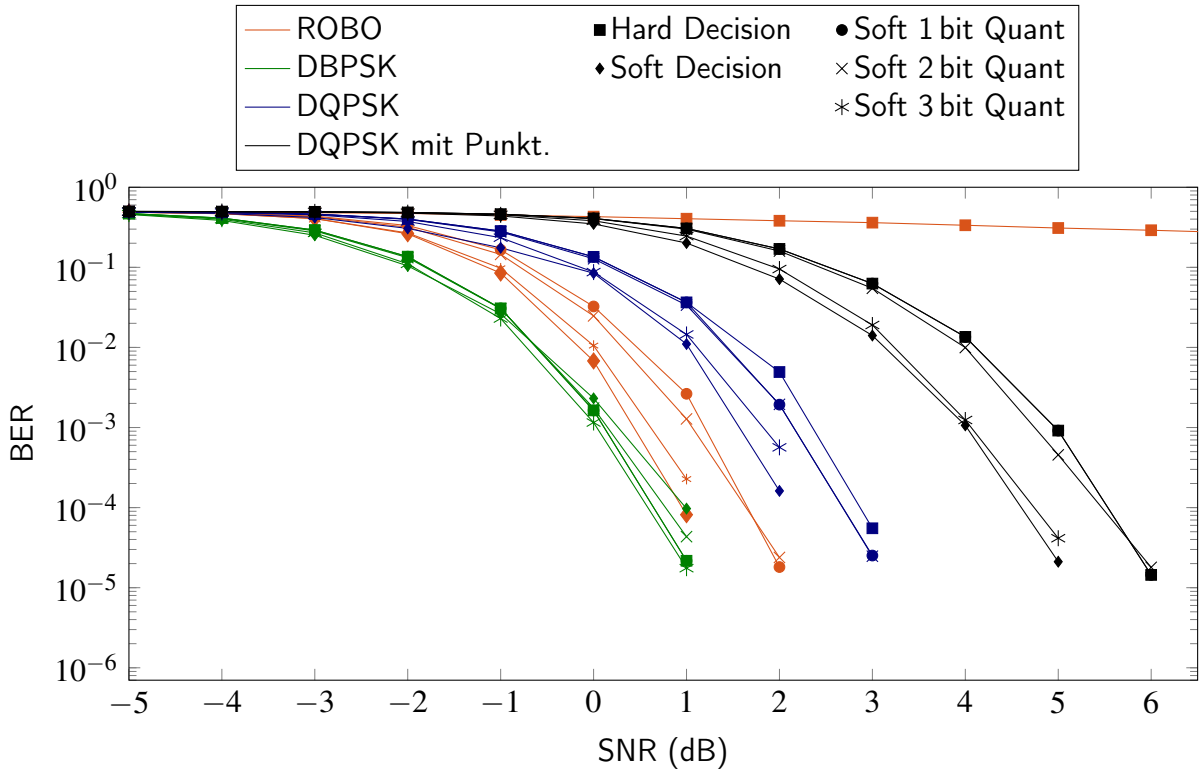

Abbildung 3.26: *Bitfehlerrate der einzelnen Modulationsarten gegenüber dem Signal-Rausch-Verhältnis für ein 100 m-Powerline-Kanal jeweils für einen Hard-Decision-, einen unquanisierten Soft-Decision- und drei quantisierte Soft-Decision-Decoder mit jeweils* $1-3$ bit.

der Powerline, die zu Dämpfung und Reflexionen führen. In dieser Arbeit wurden die ersten acht dieser Pfade als Kanalmodell verwendet. Der Kanal weist, die für Mehrwege-Ausbreitung typische, Kammstruktur auf, mit einem tiefen Einbruch bei 8 MHz und einer stärker werdenden Dämpfung zu den hohen Frequenzen. Die Phase ist überwiegend linear, außer an den Einbrüchen bei 8 MHz und 18 MHz.

Die Ergebnisse der Evaluationen mit den beiden Kanälen sind in Abbildung 3.26 für den 100 m-Bohrstrangkanal und in Abbildung 3.27 für den Mehrwege-Rayleigh-Dämpfungskanal dargestellt. Für den Einwegekanal wurde die *Tone Map* auf diejenigen Frequenzträger reduziert, die weniger als −50 dB Dämpfung aufweisen. Dies wäre in einer realen Anwendung die Aufgabe der MAC-Schicht, der die verwendbaren Träger aus der Kanalschätzung der PHY-Schicht bestimmt. Für alle Modulationsarten außer dem ROBO-Modus können vergleichbare Bitfehlerraten, wie bei einer Übertragung ohne Kanal in Abbildung 3.24, ermittelt werden. Lediglich die Datenraten haben sich durch die Verwendung von etwa der Hälfte der Träger halbiert. Auch der Einfluss von *Hard-Decision*-Decodierung, *Soft-Decision*-Decodierung und der Quantisierung ist vergleichbar zu den Ergebnissen aus Abbildung 3.24. Die Fehlerraten des ROBO-Modus sind hingegen um etwa 4 dB gestiegen. Der Grund dafür ist, dass der ROBO-Modus ungeachtet der *Tone Map* alle in der *Tone Mask* verwendbaren Träger nutzt, um die vierfach redundanten Informationen zu verteilen. Da durch den Übertragungskanal in Abbildung 3.25 a) etwa die Hälfte der Träger um mehr als 50 dB gedämpft wird, dominiert hier das additive Rauschen den Kanal. Bei der *Hard Decision*-Decodierung im ROBO-Modus ist es sogar überhaupt nicht möglich über diesen Kanal

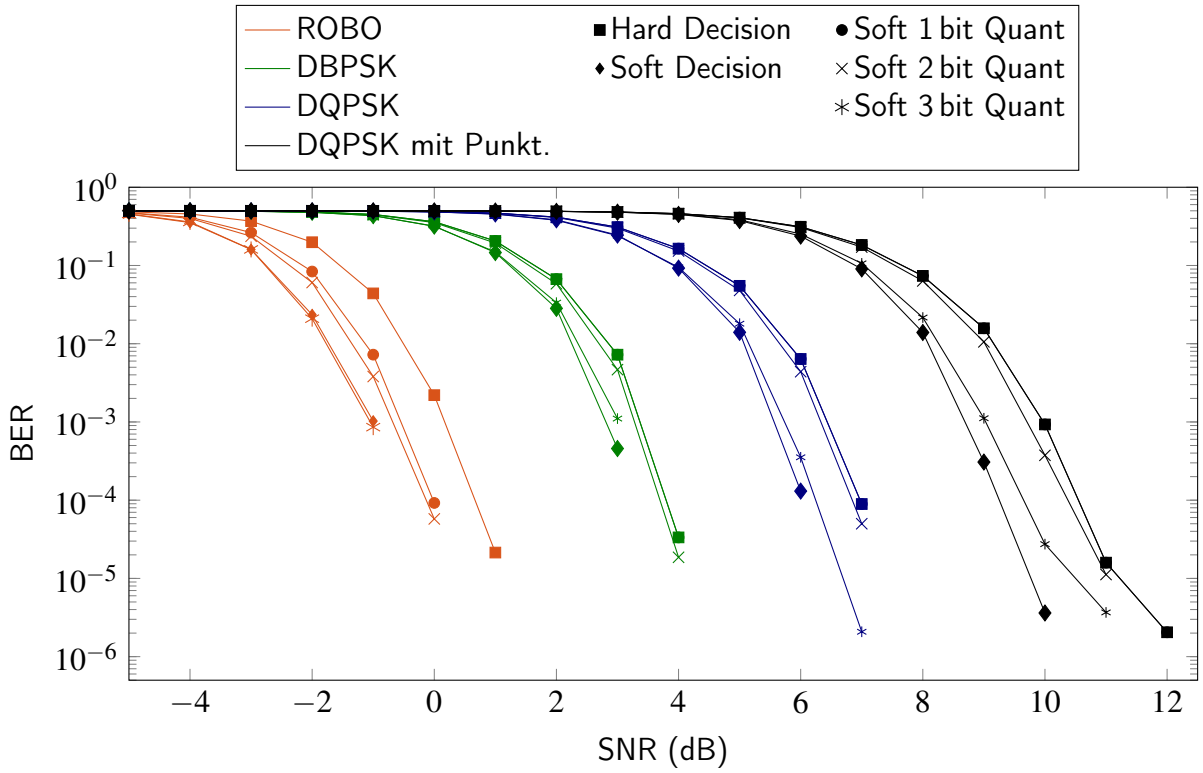

Abbildung 3.27: *Bitfehlerrate der einzelnen Modulationsarten gegenüber dem Signal-Rausch-Verhältnis für einen Mehrwege-Rayleigh-Dämpfungskanal jeweils für einen Hard-Decision-, einen unquanisierten Soft-Decision- und drei quantisierte Soft-Decision-Decoder mit jeweils* $1-3$ bit.

zu kommunizieren, da selbst bei 8 dB die Bitfehlerrate bei 23 % liegt. Erst ab 50 dB SNR wäre hier eine erfolgreiche Übertragung bei einer Bitfehlerrate von 10^{-6} möglich.

Für den Mehrwege-Rayleigh-Dämpfungskanal in Abbildung 3.27 zeigt sich gegenüber der Abbildung 3.24 eine Verschiebung der Sensitivität gegenüber dem SNR um 2 dB. Der Verlauf der Kurven und die Anordnung der Gruppen sind mit der Evaluation ohne Kanal aus Abbildung 3.24 weitestgehend identisch. Dies zeigt, dass das gewählte Kanalschätzverfahren gut funktioniert und den Kanaleinfluss bis auf einen geringen Restfehler eliminieren kann.

Als Ergebnis dieser Evaluation zeigt sich, dass eine Verwendung von *Soft-Decision*-Decodierung sinnvoll ist, vor allem für den in dieser Arbeit besonders interessanten ROBO-Modus. Für einen Übertragungskanal mit Tiefpasscharakteristik kann der ROBO-Modus mit *Hard Decision* keine Kommunikation aufbauen. Dabei hat sich eine Quantisierung der *Soft-Decision*-Informationen von 3 bit als bester Kompromiss aus zusätzlichem Hardware-Aufwand und minimaler Fehlerrate ergeben. In den nachfolgenden Abschnitten wird nur noch der *Soft-Decision*-Decoder mit 3 bit-Quantisierung verwendet.

Abschließend können auf Basis dieser Evaluation die Grenzen für die Wahl der Modulationsart nach Tabelle 3.2 festgelegt werden. Diese Grenzwerte entscheiden, welche Frequenzträger für eine Übertragung verwendet werden und welche dazugehörige Modulationsart darauf moduliert wird. Zusätzlich ist die minimale Trägeranzahl die in einer *ToneMap* enthalten sein müssen für die jeweilige Modulationsart angegeben. Diese Grenzwerte werden in allen folgenden Hardware--Implementierungen verwendet.

Tabelle 3.2: *SNR-Grenzwerte der jeweiligen Modulationsarten und die minimale Anzahl an Frequenzträgern, die bei dieser Modulationsart benötigt werden. Die SNR-Werte sind das Ergebnis der drei Simulationen mit und ohne Kanalimpulsantwort in MATLAB.*

Modulationsart	Min. SNR	Min. Trägeranzahl
ROBO	$\leq 4\,\mathrm{dB}$	–*
DBPSK	$> 4\,\mathrm{dB}$	32
DQPSK ohne Punktierung	$> 7\,\mathrm{dB}$	16
DQPSK mit Punktierung	$> 10\,\mathrm{dB}$	11

*Es wird immer die System-weit festgelegte *ToneMask* verwendet.

Die abschließenden, allgemeinen Erkenntnisse durch die Exploration des Algorithmus in diesem Kapitel sind:

- Die verfügbaren Modulationsarten haben untereinander einen Störabstand von etwa 3 dB Empfangs-SNR.
- Die Verwendung von 3 bit-*Soft-Decision*-Decodierung ist nicht signifikant schlechter als die unquantisierte Decodierung.
- Eine reale Kommunikation im ROBO-Modus ist im besten Fall bis −1 dB-Empfangs-SNR möglich.
- Es existiert ein genereller Abtausch von Hardware-Aufwand gegenüber der Robustheit der Kommunikation gegen Störungen.

KAPITEL 4

Entwurfsraumexploration der Hardware-Architekturen

Das Ziel der Entwurfsraumexploration ist es, performante Hardware-Architekturen für das in Kapitel 3 vorgestellte Powerline-Kommunikationssystem zu identifizieren. Dabei liegt das Hauptaugenmerk auf der ASIC-Fertigung der Architekturen in der XFAB XT018-Technologie und der echtzeitfähigen Kommunikation der resultierenden Powerline-Systeme. Die Plattformen dieses Entwurfsraumes sind in die in Abbildung 4.1 gezeigten Kategorien eingeteilt und reichen von Prozessoren mit Betriebssystemen bis zur Integration als dedizierte Hardware [Blu08]. Allgemein bietet dieser Entwurfsraum einen Abtausch der Flexibilität einer Plattform gegenüber der Geschwindigkeit des ausgeführten Zielalgorithmus. Der Implementierungsaufwand steigt dabei mit der Spezialisierung der Plattform an. Gleichzeitig sinkt die Leistungsaufnahme, da weniger Hardware für die Systemumgebung benötigt wird, wie beispielsweise ein Betriebssystem oder Kontroll- und Programmierstrukturen.

Der Universalprozessor (engl. *General Purpose Processor*, GPP) ist die flexibelste und langsamste Plattform, bezogen auf die Ausführung eines Algorithmus. Dieser Prozessor ist auf keine Anwendung beschränkt und kann beliebigen Programmcode ausführen. Die am weitesten verbreiteten GPPs sind Prozessoren der Firmen Intel und AMD, wie zum Beispiel der Intel i7-10700k oder der AMD Ryzen 9 5950x. Diese werden häufig durch Betriebssysteme verwaltet.

Der applikationsspezifische Instruktionssatz-Prozessor (engl. *Application-Specific Instrucion-Set Processor*, ASIP) ist eine Erweiterung eines GPP mit zusätzlichen funktionalen Einheiten (engl. *Function Units*, FU) oder Co-Prozessoren und den zugehörigen Instruktionen, um spezifische Anwendungen beschleunigt ausführen zu können.

Der digitale Signalverarbeitungsprozessor (engl. *Digital Signal Processor*, DSP) ist ein Mikroprozessor, der auf die schritthaltende Verarbeitung von Algorithmen zur Signalverarbeitung optimiert ist. Dafür verwendet dieser dedizierte Beschleuniger für zum Beispiel die Audio-Signalverarbeitung oder das Ausführen von Kommunikationsprotokollen.

Das *Field Programmable Gate Array* (FPGA) ist eine programmierbares Array aus Logikbausteinen, das beliebige Hardware-Strukturen emulieren kann. Dazu wird in einer Hardware-

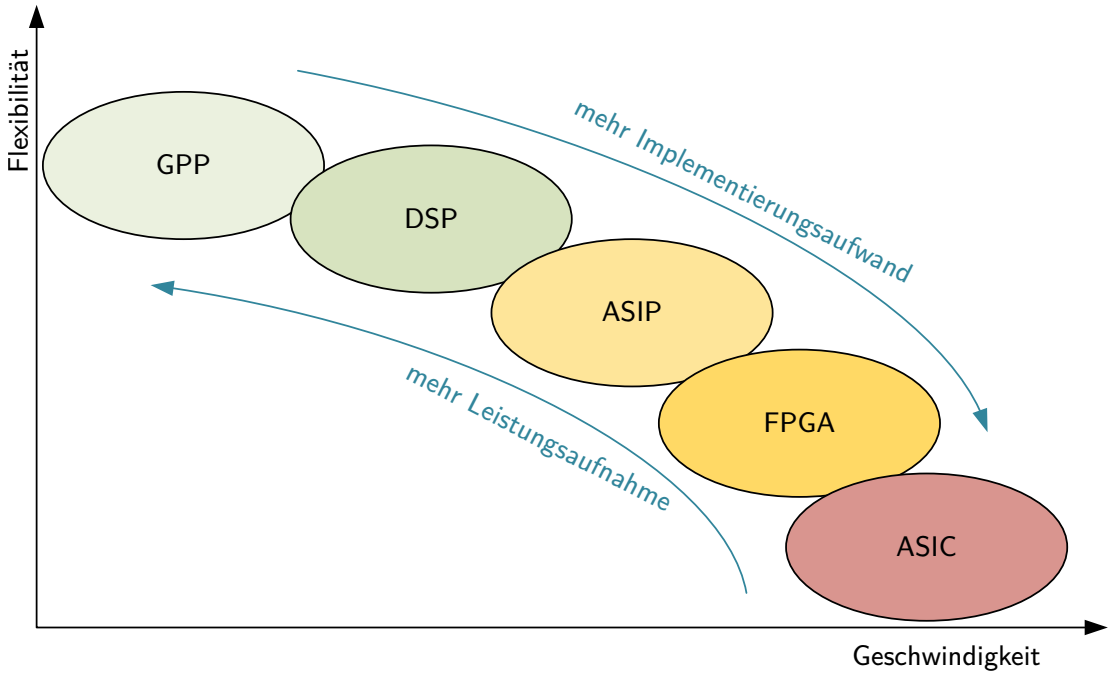

Abbildung 4.1: *Entwurfsraum der Hardware-Plattformen mit Abtausch der Flexibilität, der Geschwindigkeit, der Leistungsaufnahme und des Implementierungsaufwandes nach H. Blume und T. Noll [Blu08; Blu05].*

Beschreibungssprache (engl. *Hardware Description Language*, HDL) die Funktionalität und das zeitliche Verhalten der Hardware beschrieben und mit Hilfe eines Synthese-Werkzeuges in eine Konfigurationsdatenstrom (engl. *Bitstream*) übersetzt. Dieser *Bitstream* wird anschließend auf das FPGA geladen und ermöglicht eine echtzeitfähige Ausführung und Verifikation der beschriebenen Hardware. Da es auf dem Markt aktuell kein hochtemperaturfestes FPGA für die Implementierung des HomePlug-1.0-Systems unter den in Kapitel 2 vorgestellten extremen Umweltbedingungen gibt und eine Implementierung einer eigenen FPGA-Plattform als dedizierte Hardware-Architektur zu aufwändig ist, wird in dieser Arbeit das FPGA als Zielplattform nicht weiter betrachtet.

Der applikationsspezifische integrierte Schaltkreis (engl. *Application-Specific Integrated Circuit*, ASIC) ist die starrste und schnellste Optionen ein Algorithmus umzusetzen. Hierfür wird, ähnlich wie bei einem FPGA, der Algorithmus in einer HDL beschrieben und anschließend mit Hilfe eines Synthesewerkzeuges in eine Struktur aus Standard-Zellen einer ASIC-Technologie übersetzt (*Semi-Custom*-Implementierung) oder durch manuelle Platzierung der einzelnen Transistoren und deren Verbindungen (*Full-Custom*-Implementierung).

Zum Beginn dieser Arbeit gab es nach dem aktuellen Stand der Technik keine veröffentlichten Architekturen für die Implementierung eines kompletten HomePlug-1.0-Systems, lediglich Teile des Standards wurden auf Mikrocontrollern abgebildet [Ges19; Ges21]. Weiterhin wurde keine Veröffentlichung gefunden, die die Implementierung eines Powerline-Systems für die Tiefbohrtechnik zeigt. Auf Grund dieser fehlenden Referenzen wird in den folgenden Abschnitten der oben gezeigte Entwurfsraum für die Implementierung der MAC- und PHY-Schicht umfassend exploriert

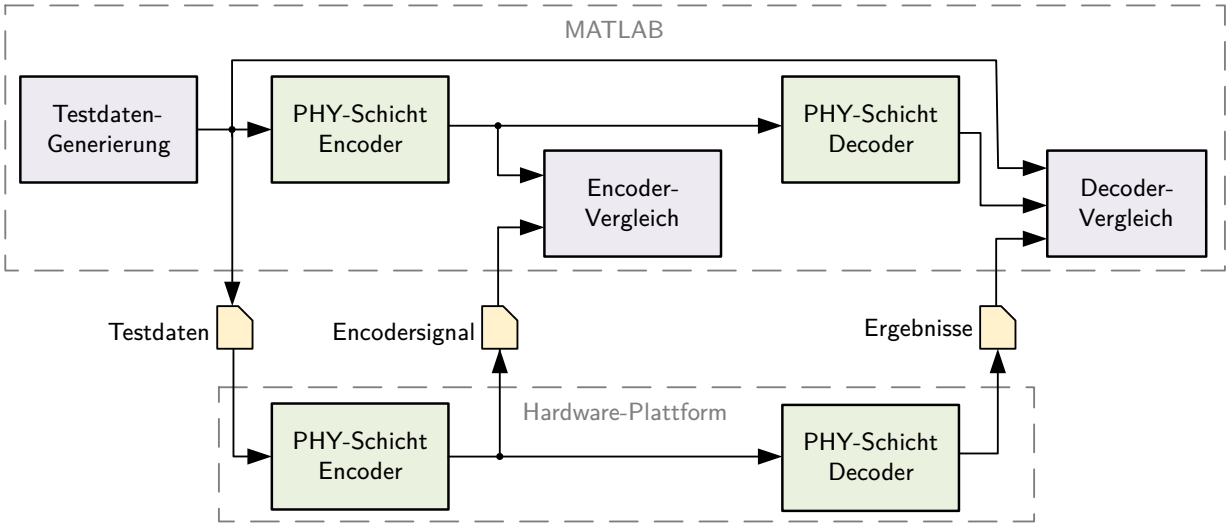

Abbildung 4.2: *MATLAB-basiertes Testsystem zur Verifikation der Hardware-Plattformen. Die zu testenden Designs sind in grün und die Blöcke zur Verifikation in lila gegeben. Für die Verifikation werden Testdaten, Encoder- und Decoder-Ergebnisse über Dateien (gelb) ausgetauscht.*

und die Architekturen hinsichtlich ihres Durchsatzes, ihrer Größe und ihrer Leistungsaufnahme evaluiert.

Zunächst werden in Abschnitt 4.1 die Implementierungen des Powerline-Systems auf den verschiedenen Plattformen vorgestellt, optimiert und in einer Entwurfsraumexploration evaluiert. Anschließend wird in Abschnitt 4.2 auch die MAC-Schicht implementiert, optimiert und evaluiert. In Abschnitt 4.3 wird die Performance der dedizierten Implementierung des Gesamtsystems, bestehend aus MAC- und PHY-Schicht, in Emulation gegen ein kommerzielles HomePlug-1.0-Modem verifiziert und die Leistungsfähigkeit echtzeitfähig vermessen. Abschließend wird die Skalierbarkeit des Systems bezogen auf Latenz und Paketverlustrate mit bis zu 60 Kommunikationsteilnehmer emuliert und evaluiert. Eine Zusammenfassung der Ergebnisse ist in Abschnitt 4.4 gegeben.

4.1. Exploration der PHY-Schicht

Für die Exploration der Hardware-Architekturen der PHY-Schicht wird ein Testsystem zur Verifikation der berechneten Ergebnisse sowie eine Implementierung der Algorithmen in C benötigt. Zunächst wurde der MATLAB-Referenzalgorithmus in eine C-Implementierung portiert und gegen die MATLAB-Version verifiziert. Anschließend wurde ein MALTAB-basiertes Verifikationssystem erstellt, das für die nachfolgenden Implementierungen auf den Hardware-Plattformen als Testsystems dient. Dieses ist in Abbildung 4.2 dargestellt.

Zuerst werden in MATLAB die Kontrollinformationen, Nutzdaten und Übertragungsparameter generiert und an den MATLAB-PHY-Encoder übergeben. Parallel werden diese Werte in eine Datei geschrieben, die von der Hardware-Plattform eingelesen und ebenfalls dem zu testenden PHY-Encoder übergeben wird. Die in der Testplattform und in MATLAB erzeugten Powerline-Signale werden dann einer Vergleichseinheit in MATLAB übergeben, die statistische Fehler wie die mittleren quadratischen (engl. *Root Mean Square*, RMS) und die maximale Abweichung auswertet. Dazu schreibt die Testplattform diese Werte in eine Datei zurück, die von MATLAB

eingelesen wird. Abschließend werden auch die Ergebnisse der beiden PHY-Decoder gegen die Eingangsdaten verglichen und überprüft, ob sich beide Decoder identisch gegenüber Fehlern verhalten beziehungsweise diese detektieren und korrigieren.

Die verwendeten Hardware-Plattformen für eine echtzeitfähige Powerline-Kommunikation werden in den folgenden Abschnitten präsentiert. In Abschnitt 4.1.1 wird die Implementierung auf der RISC-V-basierten PULP-Plattform der ETH Zürich [Ros15; Con16] vorgestellt. Anschließend wird in Abschnitt 4.1.2 der Powerline-Algorithmus auf verschiedene Tensilica LX7-ASIPs portiert und evaluiert. Abschnitt 4.1.3 zeigt die Implementierung als dedizierte Hardware in *Very High Speed Integrated Circuit Hardware Description Language* (VHDL). Abschließend wird in Abschnitt 4.1.4 die Exploration der PHY-Schicht zusammengefasst und die Ergebnisse diskutiert.

4.1.1. CV32E40P-basierte PULP-Plattform

Als erste Implementierung für den zu untersuchenden Entwurfsraum wird der MATLAB-Code auf den CV32E40P-Prozessor mit 32 bit Wortbreite und vier *Pipeline-Stages* portiert. Dieser RV32IMC-Prozessor ist ein RISC-V-Instruktionssatz-kompatibler GPP. Als Besonderheiten des Prozessors werden Instruktionssatzerweiterungen in Form von z.B. DSP-Operationen, Bit-Manipulation oder Hardware-Schleifen unterstützt. Speziell die DSP-Erweiterung mit einer *Multiply Accumulate*-Einheit (MAC) und die *Single Instruction Multiple Data*-Erweiterung (SIMD) sind für die hier zu implementierenden Algorithmen der Signalverarbeitung interessant. Der CV32E40P-Prozessor wird in dieser Arbeit innerhalb des PULPissimo-*System-on-Chip* (SoC) verwendet, das Teil der PULP-Plattform der ETH Zürich ist. Eine schematische Darstellung des SoC ist in Abbildung 4.3 gegeben [Gau17].

Die PULP-Plattform besitzt eine eigene Software-Entwicklungsumgebung (engl. *Software Development Kit*, SDK), ist skalierbar in Größe und Performance und unterstützt verschiedene Prozessoren. Neben dem Einkernprozessor PULPissimo werden auch Mehrkernprozessoren und Cluster von Mehrkernprozessoren unterstützt [Con16]. Für diese Arbeit besonders interessant sind die Schnittstellen der Hardware-Erweiterung: die *Hardware Processing Engine* (HWPE) und das Register-Interface. Die HWPE ist eine Streaming-basierte Schnittstelle zum Anschluss von Co-Prozessoren oder dedizierten Hardware-Erweiterungen. Das Register-Interface ist eine über den *Advanced eXtensible Interface 4* (AXI4) Bus erreichbare Schnittstelle, um Berechnungen parallel zum Betrieb des Prozessors durchführen zu können. Dazu werden die Operanden in Eingangsregister geschrieben und das Ergebnis aus Ausgangsregistern wieder abgeholt. Diese Erweiterung eignet sich vor allem für kleinere funktionale Einheiten, die häufig wiederverwendet werden.

Eine ASIC-Synthese des Systems mit Hilfe von Cadence Genus 19.13 für die schlechtest möglichen Betriebsparameter in der Zieltechnologie XT018 der Firma XFAB mit 175 °C und einer Kern-Spannung von 1,62 V (siehe Kapitel 2) zeigt einen kritischen Pfad von 23,5 ns inklusive PULP-Erweiterungen. Das erlaubt eine maximale Taktfrequenz von 42,5 MHz für den Betrieb in dieser Technologie. Die Größe der gesamten Plattform beträgt 6,73 mm^2 und davon entfallen 1,37 mm^2 auf den CV32E40P-Kern. Da das Synthesewerkzeug Speicher generell als Flip-Flops umsetzt, wurden die Größen der Instruktions- und Datenspeicher anhand des Programmcodes abgeschätzt. Hierfür wurde der *Memory-Compiler* der Firma XFAB verwendet und ein hochtemperaturfester *Static Random-Access Memory* (SRAM) der XT018-Technologie gewählt. Der Instruktionsspeicher hat eine Größe von 16 kByte bei einer Fläche von 2,96 mm^2

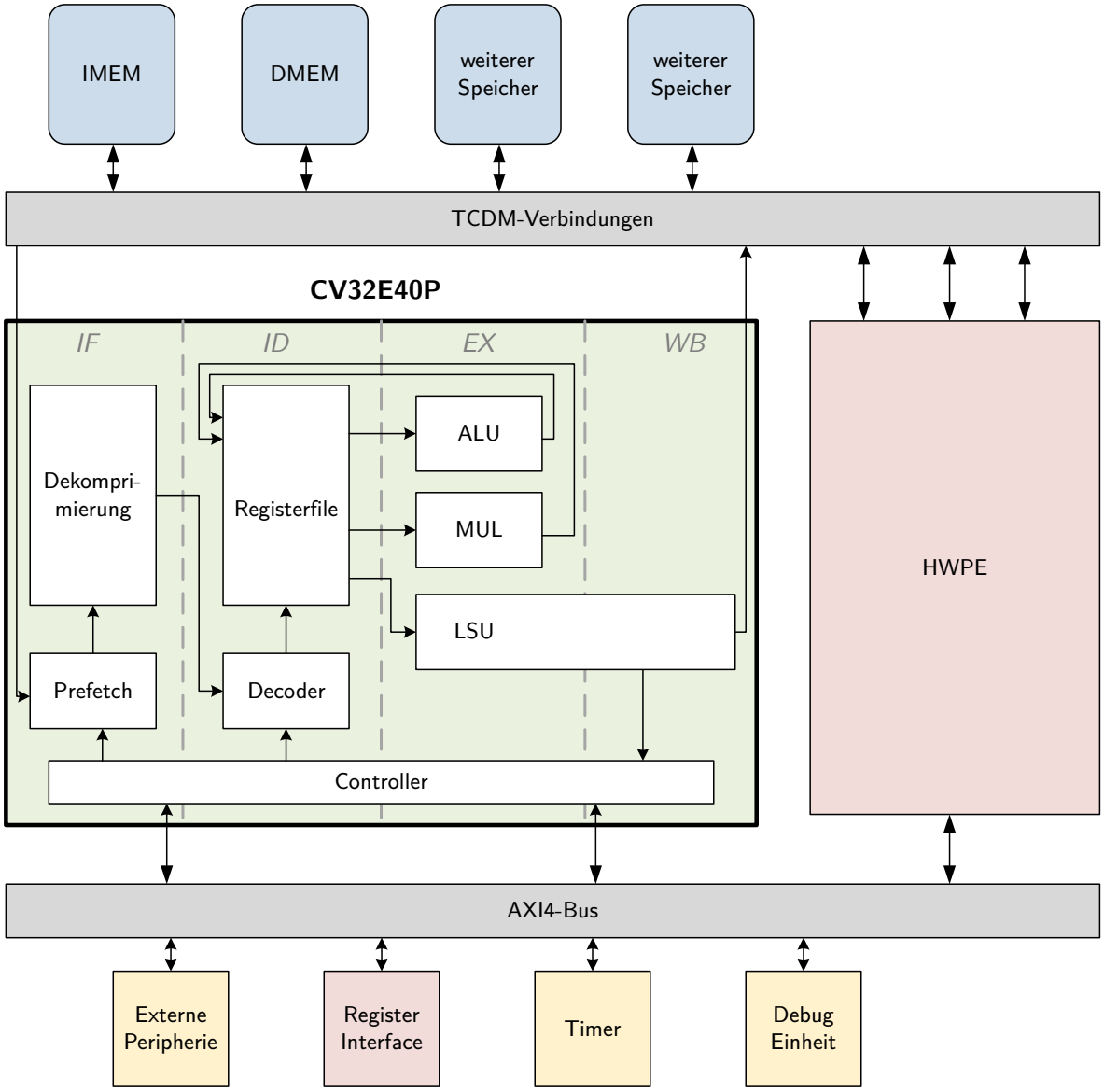

Abbildung 4.3: *Ausschnitt der PULPissimo-Plattform mit internem Aufbau des CV32E40P-Prozessors (grün), Anschlüssen an Speicher (blau), externe Peripherie (gelb) und programmierbarer Hardware-Erweiterung (rot) für große Co-Prozessoren (Hardware Processing Engine, HWPE) und kleinere funktionale Einheiten (Register-Interface) [Sch18].*

und der Datenspeicher eine Größe von 4 kByte bei einer Fläche von 0,95 mm^2. Damit sind die Speicherblöcke zusammen etwa doppelt so groß wie der CV32E40P-Kern. Der Rest der Fläche wird durch Peripherie, Debugging-Schnittstellen und den AXI-Interconnect-Bus belegt.

Für die Portierung des MATLAB-Codes auf die PULP-Plattform wurde eine Fixpunkt-C-Implementierung erstellt, die direkt von der MATLAB-Implementierung abgeleitet wurde. Dabei wurde das Fixpunkt-Format nur bei der OFDM-Verarbeitung benötigt, da erst ab dem Modulator bzw. vor dem Demodulator Phasen- statt Bit-Informationen verwendet werden. Die Genauigkeit der Fixpunktformate ist hier auf 16 bit ausgelegt.

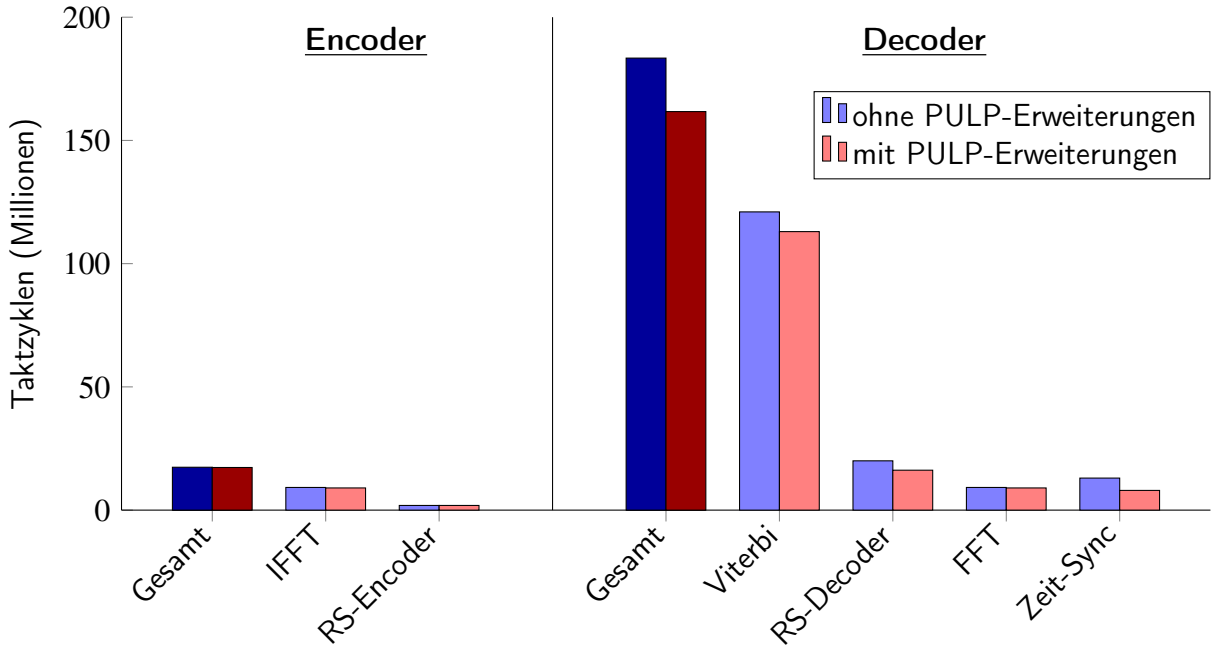

Abbildung 4.4: *Anzahl der benötigten Taktzyklen in Millionen zum Ausführen der jeweiligen Teilfunktion. Linker Teil für den Encoder, rechter Teil für den Decoder.*

Die Ergebnisse dieser Portierung, ohne Software-seitige Verbesserung, sind in Abbildung 4.4 dargestellt. Diese zeigt die Anzahl der Taktzyklen in Millionen, die für die Ausführung des Programmcodes notwendig sind. Der linke Teil des Diagramms zeigt die Takte für den Encoder und der rechte Teil für den Decoder. Die dunkleren Balken auf beiden Seiten zeigen die Gesamtzahl der benötigten Zyklen und die heller gefärbten Balken die anteilig größten Verarbeitungsblöcke. In blau sind die benötigten Taktzyklen für eine RISC-V-kompatible Ausführung ohne PULP-Erweiterungen und in rot mit PULP-Erweiterungen gezeigt. Zu erkennen ist ein großer Unterschied in den Ausführungszeiten von Encoder und Decoder. Der Decoder benötigt 183,4 Mio. Zyklen ohne und 161,7 Mio. Zyklen mit PULP-Erweiterungen und der Encoder in beiden Fällen nur 17,4 Mio. Zyklen. Die maximale erreichbare Datenrate R berechnet sich nach Gleichung 4.1.

$$R[\text{bit/s}] = \frac{P_{\text{max}}[\text{bit}] \times f[\text{Hz}]}{Z_T} \tag{4.1}$$

Dabei ist Z_T die Anzahl der benötigten Taktzyklen für den Encoder oder den Decoder und $P_{\text{max}} = 16.608\,\text{bit}$ die maximal Paketgröße in Bit. Unter der Annahme der besten Codierrate und einer Systemfrequenz von $f = 42{,}5\,\text{MHz}$ ergibt sich eine Datenrate von 34,35 kbit/s für den Encoder und von 4,26 kbit/s für den Decoder. Damit ist der Decoder um den Faktor 8 langsamer und bietet das größere Potential für die Beschleunigung der Gesamtausführungszeit und des Datendurchsatzes der Powerline-Kommunikation.

Die Reduktion der Taktzyklen durch die Verwendung der PULP-Erweiterungen beträgt maximal 11,8 % im Decoder und ist damit nur eine geringe Performance-Steigerung. Eine Analyse des C-Codes zeigt, dass sich nur etwa 13 % des Programmcodes durch die PULP-Erweiterungen beschleunigen lassen, indem die Bit-Manipulation und Hardware-Schleifen-Zähler verwendet werden. Um den Anteil der Instruktionen, der durch die Erweiterungen beschleunigt werden

kann, zu erhöhen muss der Programmcode speziell auf die PULP-Plattform angepasst und die Software optimiert werden.

Software-Optimierungen

Als erster Schritt der Optimierung wurden die Algorithmen auf die PULP-Plattform angepasst und Software-seitige Optimierungen durchgeführt. Das größte Potential hat der Viterbi-Decoder, der als Traceback-Decoder entworfen wurde (siehe Abschnitt 3.2). Dieser verbringt ein Großteil seiner Verarbeitungszeit in der rückwärtigen Suche des Pfad-Minimums. Eine alternativer Algorithmus ist der Register-Exchange-Viterbi-Decoder, der die rückwärts laufende Decodierung während der Berechnung der Pfad-Metrik mitdurchführt und alle möglichen Zwischenergebnisse speichert. Hierdurch reduziert sich die Suche nach dem Pfad-Minimum auf das Auslesen eines Registers, aber die Anzahl der Speicherzugriffe und die damit verbundene Leistungsaufnahme steigt stark an. Der Grund dafür ist, dass die Traceback-Methode nur die aktuell neu dekodierten 64 Kandidaten-Bits im Auswahlspeicher aktualisiert, die Register-Exchange-Methode hingegen in jedem Takt die gesamte Matrix aus 64 60 bit-Registern neu schreiben muss [Lee04]. Da das primäre Ziel dieser Implementierung die Maximierung des Durchsatzes ist, wurde hier der Register-Exchange-Algorithmus gewählt.

Auch die FFT und IFFT wurden angepasst, indem die Wortbreite von initial 32 bit-Eingangswerten auf 16 bit reduziert wurde und somit die 16 bit-PULP-DSP-Erweiterung verwendet werden konnte, um die Butterfly-Gleichungen zu berechnen. Der Genauigkeitsverlust der entfallenen 16 bit ist hierbei nicht signifikant, da die Eingangsdaten des Systems mit maximal 16 bit auch keine höhere Datenauflösung erreichen.

Zusätzlich wurde dieser beschleunigte (I)FFT-Algorithmus verwendet, um die Kreuzkorrelation im Frequenzbereich statt im Zeitbereich zu berechnen. Die in Abschnitt 3.2.1 gezeigte Kreuzkorrelation in Gleichung 3.15 wurde dafür mit einem Schwellwert von $S_{\text{th}} = 0,5$ aus Gleichung 3.16 zur Ungleichung 4.2 kombiniert.

$$\frac{C_{rp}(m)}{\sigma_p \sigma_r} \geq 0,5 \tag{4.2}$$

Durch Quadrierung und Umformung der Ungleichung erhält man die Ungleichung 4.3.

$$C_{rp}(m)^2 \geq 0,25 \sigma_p^2 \sigma_r^2 \tag{4.3}$$

Die Quadrierung eliminiert die Wurzelberechnung und die Auflösung des Bruchs vermeidet die Division. Um die Berechnung von $C_{rp}(m)^2$ zu beschleunigen, wurde diese mit Hilfe der FFT transformiert und kann als Multiplikation im Frequenzbereich berechnet werden [Wer08]. Die resultierende Gleichung 4.4 zeigt die benötigten Transformationen und Berechnungen im Frequenzbereich.

$$\text{IFFT}\{\text{FFT}\{r(n)\}P(k)^*\}^2 \geq 0,25 \sigma_r^2 \sigma_p^2 \tag{4.4}$$

Dabei ist $P(n)^*$ die komplex konjugierte Fourier-transformierte Präambel und kann genau wie σ_P^2 vorberechnet gespeichert werden. Lediglich das Empfangssignal $r(n)$ muss per FFT transformiert, multipliziert und per IFFT zurück transformiert werden.

Als letzte algorithmische Änderung wurde eine Modulo-Berechnung im RS-Decoder optimiert. Durch die Galois-Feld-Berechnungen auf dem Feld von $2^8 = 256$ Elemente muss nach jeder

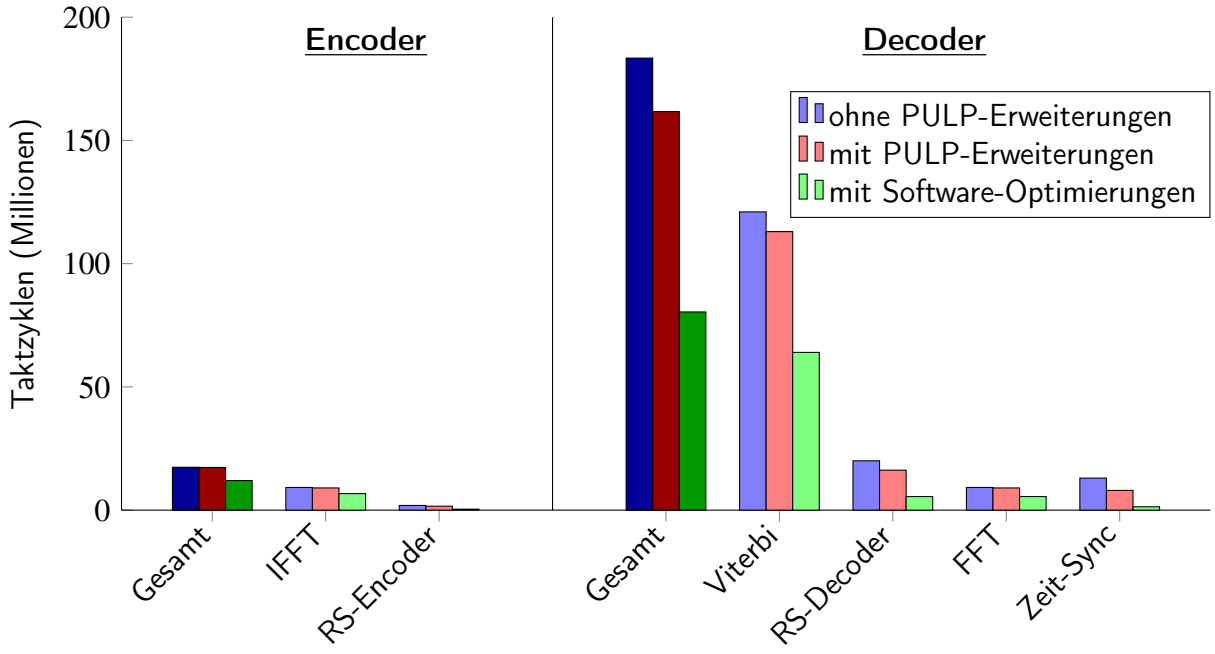

Abbildung 4.5: *Anzahl der benötigten Taktzyklen nach den Software-Optimierungen. Linker Teil für den Encoder, rechter Teil für den Decoder.*

arithmetischen Operation das Ergebnis mittels Modulo-255-Rechnung wieder in den abgeschlossen Galois-Feld-Körper gebracht werden. Diese Modulo-Berechnung lässt sich auch durch eine bedingte Subtraktion darstellen, die nach jeder Operation überprüft, ob das Ergebnis größer als 254 ist.

Die Ergebnisse dieser Software-Optimierungen sind in Abbildung 4.5 dargestellt. Die Gesamtzahl der Taktzyklen hat sich für den Encoder um 31 % und den Decoder um 50 % gegenüber der initialen Version mit PULP-Erweiterungen verringert. Damit steigt die Datenrate des Encoders auf 54,14 kbit/s und der Decoder erreicht 8,57 kbit/s. Der Viterbi-Decoder wird zwar mit einer Reduktion der Taktzyklen um 43,4 % stark beschleunigt, hat aber mit einem Anteil von 80 % an den Gesamttaktzyklen des Decoders immer noch das größte Optimierungspotential. Die Berechnung der FFT und IFFT hat sich um 38,8 % und 25,5 % beschleunigt, wobei der Unterschied in den verbleibenden Taktzyklen zwischen FFT und IFFT sich mit der Umsortierung der Frequenzkoeffizienten vor der Berechnung der IFFT begründet. Der RS-Encoder und der RS-Decoder wurden um 75 % und 70,5 % beschleunigt. Am stärksten hat sich die Zeit-Synchronisation mit 83,1 % verbessert und benötigt als Ergebnis deutlich weniger Taktzyklen als die FFT und IFFT. Der Grund dafür ist, dass der Algorithmus nur auf den beiden Präambeln des Paketes ausgeführt wird und nicht auf den Nutzdaten.

Insgesamt haben die Software-Optimierungen den Algorithmus zwar beschleunigt, dennoch ist die Verarbeitungszeit des Decoders deutlich langsamer als die des Encoders, was zu einem Ungleichgewicht zwischen Senden und Empfangen eines Paket führt. Um den Decoder weiter beschleunigen zu können, wurden Hardware-Erweiterungen für den Viterbi- und den RS-Decoder entworfen und implementiert.

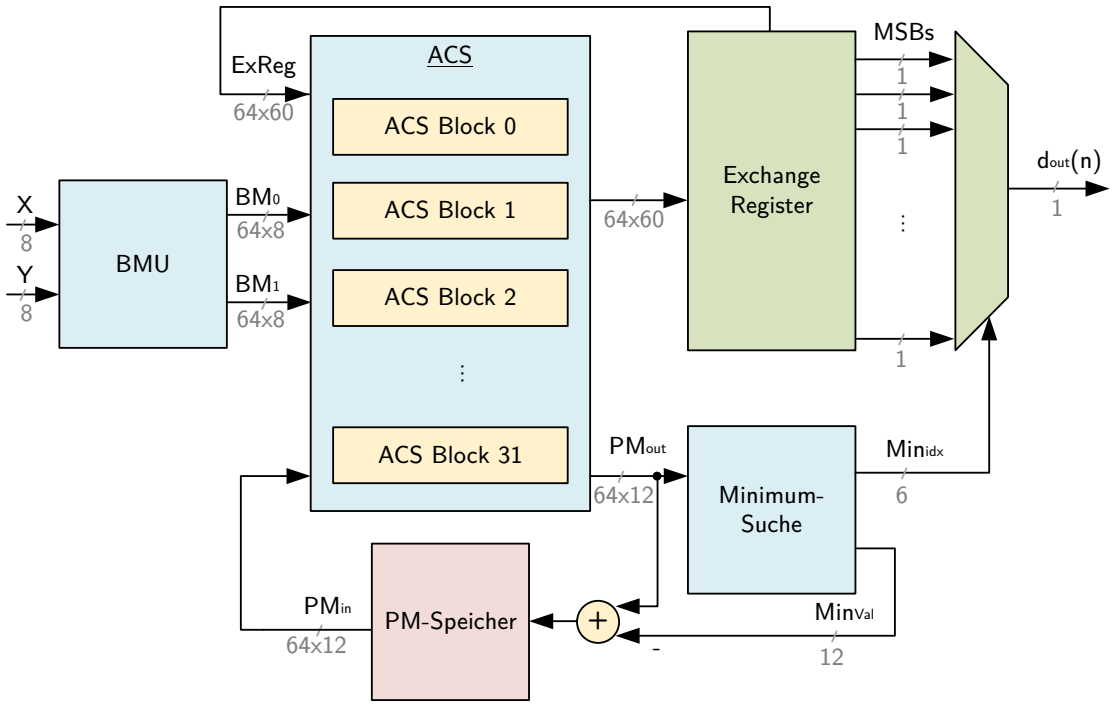

Abbildung 4.6: *Struktur des Viterbi-Decoders mit Register-Exchange. Änderungen zu dem in Abbildung 3.13 gezeigten Traceback-Viterbi-Decoder sind in grün dargestellt.*

Hardware-Erweiterungen

Zunächst wurde der Viterbi-Decoder als gesamte Einheit in SystemVerilog beschrieben und in das PULPissimo-SoC integriert. Dafür wurde das oben beschriebene Register-Interface der Plattform verwendet, um Daten zum und vom Viterbi-Decoder zu transferieren. Der Aufbau und die Bitbreiten des Viterbi-Decoders sind in Abbildung 4.6 gezeigt. Die *Exchange-Register* und der Multiplexer (grün in Abbildung 4.6) ersetzen hier die *Traceback*-Logik aus Abbildung 3.13 in Abschnitt 3.2. Zusätzlich erhält jeder ACS-Block zwei 60 bit große *Exchange-Register* und berechnet anhand der im Selektionsblock getroffenen Entscheidung die aktualisierten Registereinträge. Dazu werden abhängig von der Entscheidung die unteren 59 bit eines Registers kopiert und die aktuelle Entscheidung als LSB angefügt. Die resultierenden 64 Pfadentscheidungen werden im *Register-Echange*-Block gespeichert. Das MSB jedes Registers liegt an einem Multiplexer, der, durch die Suche des Minimums gesteuert, das decodierte Bit auswählt. Eine erste ASIC-Synthese des Viterbi-Modul in der XT018-Technologie ergab einen kritischen Pfad von 51,92 ns. Dieser wurde dann mit insgesamt 4 Pipeline-Registern in der Suche des Minimums auf 13,70 ns verkürzt, so dass er kürzer ist als der kritische Pfad des PULP-Systems. Die resultierende Latenz des *Register-Exchange*-basierten Viterbi-Decoders ist damit 8 Takte für jedes Tuple aus X und Y am Eingang. Diese geringe Decodierlatenz hat als Nachteil eine hohe Schaltaktivität und einen hohen Flächenbedarf, verglichen mit der *Traceback*-Decodierung [Hab10].

Als zweite Hardware-Einheit wurde eine Galois-Feld-MAC-Einheit (GF-MAC) in SystemVerilog umgesetzt, die die Multiplikations- und MAC-Operationen im RS-Decoder weiter beschleunigen. Der hierfür verwendete Algorithmus ist ein Bit-Level-Algorithmus auf dem Galois-Feld 2^n nach Silverman et al. [Sil99]. Abbildung 4.7 zeigt den Aufbau der resultierenden Hardware. Durch die Berechnungen innerhalb des Galois-Feldes, kann eine Multiplikation in *Shift-Add*-Technik auf

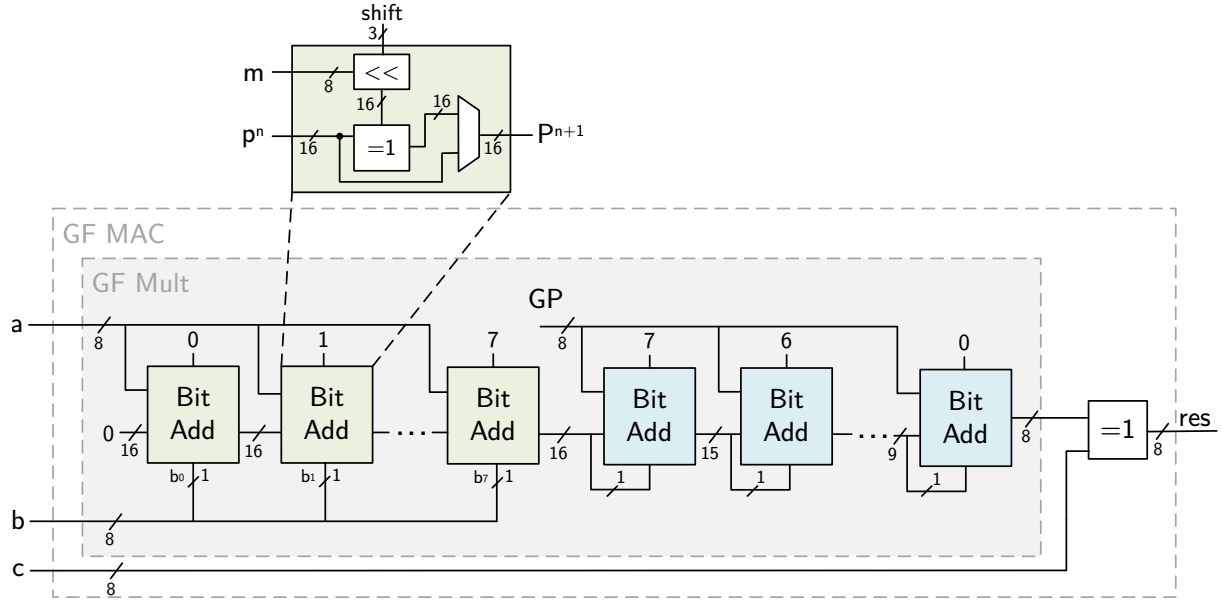

Abbildung 4.7: *Hardware-Struktur der Galois-Feld-MAC-Einheit mit selektierbarem Ausgang für Multiplikations- oder MAC-Ergebnis nach Silverman et al. [Sil99].*

Exklusiv-Oder-Funktionen zurückgeführt werden. Dabei addiert eine *Bit-Add*-Einheit (grün) ein Partialprodukt auf und gibt dieses an die nächste Einheit weiter. Nach der Multiplikation wird durch die Modulo-Berechnung des Generatorpolynoms GP, mit Hilfe derselben *Bit Add*-Blöcke, das Ergebnis wieder in das Galois-Feld transformiert (blau). Am Ende der Berechnung wird das Ergebnis der MAC-Operation $res = a \cdot b + c$ ausgegeben. Mit Hilfe dieser Hardware-Architektur lassen sich folgende Operationen Galois-Feld-Operationen berechnen:

- GF-MAC: Multiplikation aus a und b mit anschließender Akkumulation in c.
- GF-Multiplikation: Multiplikaten in a und b mit $c = 0$.
- GF-Addition: a als erster Summand, $b = 1$ und c als zweiter Summand.
- GF-Modulo: Operand in a mit $b = 1$ und $c = 0$.

Die Synthese dieser Hardware-Erweiterung zeigt einen kritischen Pfad von 7,8 ns, was deutlich schneller ist als der kritische Pfad der PULP-Plattform oder der kritische Pfad des gepipelineten Viterbi-Decoders. Daher müssen keine zusätzlichen Pipeline-Register für diese Einheit implementiert werden.

Die Messungen der Taktzyklen mit den beiden Hardware-Erweiterungen sind in Abbildung 4.8 dargestellt. Die Gesamtzyklenzahl des Encoders hat sich nicht verändert, da beide implementierten Hardware-Einheiten Teil des Decoders sind. Die Anzahl der benötigten Taktzyklen des Decoders hat sich um 80 % auf 15,4 Mio. Zyklen reduziert, was vor allem an der starken Beschleunigung des Viterbi-Decoders liegt. Dieser wurde um 98 % beschleunigt und braucht nun lediglich 0,84 Mio. Taktzyklen. Der RS-Decoder wurde um 48 % beschleunigt und benötigt 2,9 Mio. Taktzyklen.

Neben den benötigten Taktzyklen ist bei den Hardware-Erweiterungen auch der Flächenzuwachs mit zu betrachten. Eine Synthese der beiden Hardware-Einheiten in der XT018-Technologie von XFAB zeigt eine Fläche von 0,84 mm^2 für den Viterbi-Decoder und 0,005 mm^2 für die GF-MAC-Einheit. Gegenüber dem CV32E40P-Kern besitzt der Viterbi-Decoder etwa 61 % von

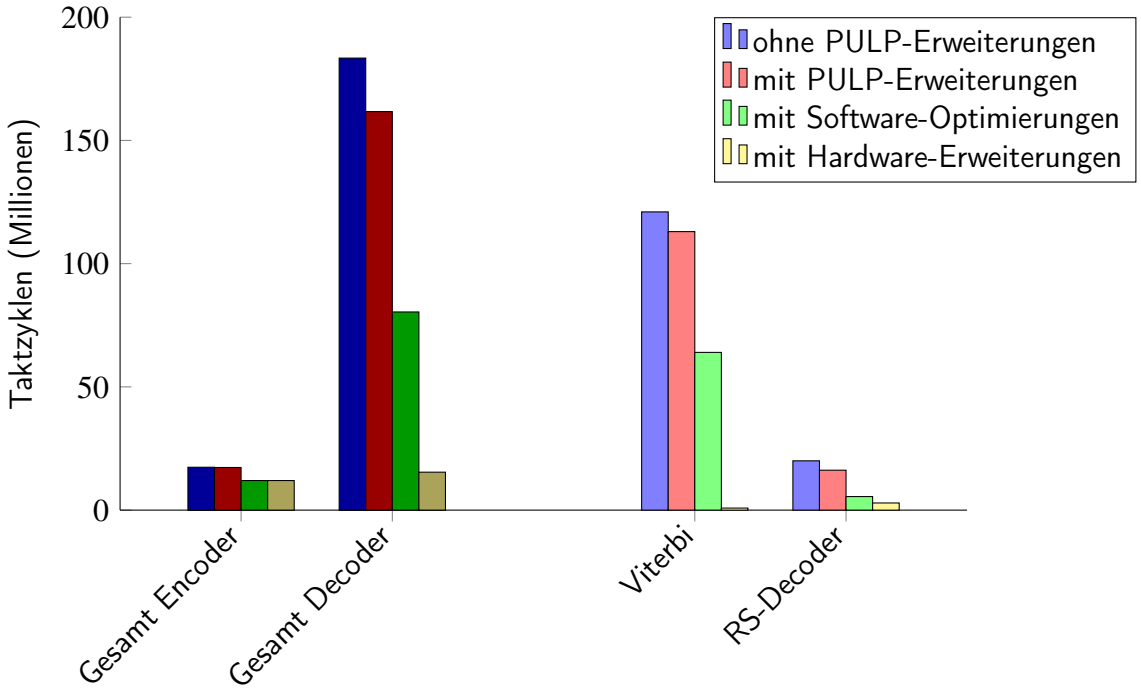

Abbildung 4.8: *Anzahl der benötigten Zyklen mit Viterbi- und Galois-Feld-MAC-Erweiterung. Die dunkleren Balken zeigen die Gesamtzahl und die helleren Balken die Taktzyklenzahlen der Subsysteme.*

dessen Fläche und die GF-MAC-Einheit nur 0,4 %. Letztere ist mit einer Reduzierung von 579 Taktzyklen/µm^2 wesentlich effizienter als der Viterbi-Decoder mit 75 Taktzyklen/µm^2. Die resultierende Kommunikationsgeschwindigkeit des Decoders mit beiden Hardware-Erweiterungen beträgt 45,33 kbit/s und ist damit um den Faktor 5,3 gestiegen.

Abbildung 4.9 zeigt die Exploration des Entwurfsraumes über Fläche und Datenübertragungsrate. Der CV32E40P-Prozessor ist inklusive Instruktions- und Datenspeicher als Referenz für den Mehraufwand der gesamten PULP-Plattform angegeben. Die gesamte PULP-Plattform ist 10,33 mm^2 groß und der CV32E40P-Prozessor belegt mit 8,87 mm^2 etwa 85 % davon. Die oben gezeigten Software-Optimierungen steigern die Datenrate um den Faktor 2 und vergrößern dabei nicht die Fläche der Plattform. Die Hardware-Erweiterungen steigen den Durchsatz um den Faktor 10,6 gegenüber der unoptimierten reinen Softwareversion und vergrößern die benötigte Chipfläche um 35,6 % auf 11,31 mm^2. Damit haben die Hardware-Erweiterungen alleine eine Größe von 1 mm^2 und ergeben eine Steigerung der Leistungsaufnahme um 86,56 mW.

Um den Durchsatz des Systems noch weiter steigern und Datenraten im Mbit/s-Bereich erreichen zu können, müssten weitere Bestandteile des Powerline-Algorithmus in Hardware implementiert und an die PULP-Plattform angehängt werden, wie z.B. die FFT, die IFFT, der RS-Decoder als Ganzes oder die Zeit-Synchronisation. Damit wäre der RISC-V-Prozessor dann nur noch ein Steuerprozessor für die angehängten Hardware-Erweiterungen, um Daten zu transferieren. Da dies nicht Ziel dieser Arbeit ist, wird im folgenden Abschnitt eine Implementierung auf dem Tensilica LX7-ASIP untersucht, der durch seine Parametrisierbarkeit und Instruktionssatzerweiterungen ein deutlich größeres Spektrum an Möglichkeiten der Beschleunigung bietet.

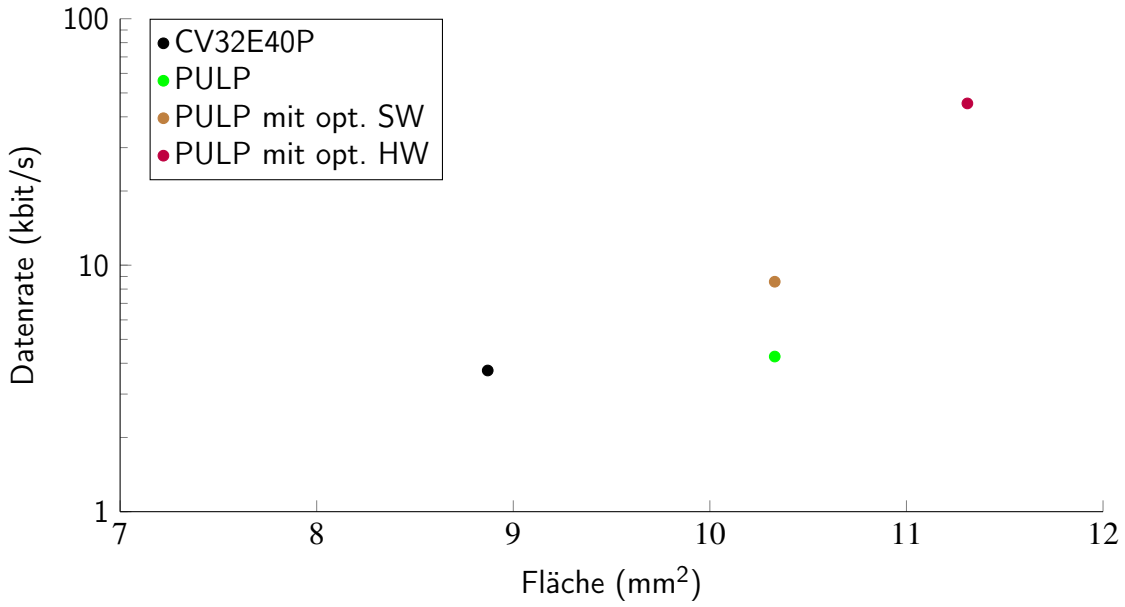

Abbildung 4.9: *Entwurfsraumergebnisse der PULP-Plattform mit Software- und Hardware-Optimierungen.*

4.1.2. Tensilica Xtensa LX7

Nachdem im vorangegangenen Abschnitt 4.1.1 gezeigt wurde, dass ein GPP selbst mit Hardware-Erweiterungen nur sehr geringe Datenraten erreichen kann, wird in diesem Abschnitt die Implementierung des HomePlug 1.0 Standards auf einer ASIP-Plattform vorgestellt. Diese Plattform ist die Tensilica LX7-Prozessorarchitektur, die als Basis einen 32 bit-ASIP mit fünf Pipeline-Stufen hat: *Instruction Fetch* (IF), *Instruction Decode* (ID), *Execute* (EX), *Memory* (MEM) und *Write Back* (WB). Optional kann die Pipeline-Struktur auf sieben Stufen erweitert werden, indem eine weitere Stufe in IF und MEM eingfügt wird, um den kritischen Pfad zu verkürzen.

Die Basisarchitektur kann unter anderem durch funktionale Einheiten (engl. *Functional Unit*, FU), wie Multiplizierer, MAC, Dividierer oder eigens entworfene FUs erweitert werden. Durch die Wahl eines hochperformanten DSP-Coprozessors kann der Basisprozessor auf ein bestimmtes Gebiet der Signalverarbeitung adaptiert werden. Diese Coprozessoren bieten verschiedene Arten der parallelen Datenverarbeitung, wie zum Beispiel *Single Instruction Multiple data* (SIMD), *Very Long Instruction Words* (VLIW) oder Vektorverarbeitungseinheiten. Die Fusion-Coprozessoren bieten hochperformante Beschleuniger für die digitale Signalverarbeitung im Allgemeinen. Die Hifi-Coprozessoren enthalten Beschleuniger für die Audiosignalverarbeitung. Die Vision-Coprozessoren beschleunigen die digitale Bildverarbeitung durch massiv-parallele Datenverarbeitung. Die ConnX-Coprozessoren sind speziell für die Anforderungen von Fahrzeugen an die drahtlose Datenkommunikation und Signalverarbeitung von Sensordaten entwickelt worden. Eine Übersicht der Hardware-Eigenschaften der aktuellsten Generation der Coprozessoren ist in Tabelle 4.1 gegeben. Die Hifi-Reihe bildet dabei die Gruppe der kleinsten Coprozessoren, gefolgt von der Fusion-Reihe, der ConnX-Reihe und der Vision-Reihe. Die Größe der Load-Store-Einheit steigt zusammen mit den verarbeitenden Einheiten linear bis zu einer Breite von 1024 bit für die Vision-Reihe an.

Innerhalb eines Verarbeitungsblockes im C-Codes des HomePlug 1.0-Standards lassen sich Schleifendurchläufe und Bit-Manipulationen gut parallelisieren. Die Verarbeitungsblöcke selbst

Tabelle 4.1: *Vergleich der aktuellsten Coprozessoren für einen Tensilica LX7-Basisprozessor („–“ = keine Informationen veröffentlicht).*

DSP	ALUs	Load/Store Einheit	MUL/MAC pro Zyklus				VLIW Slots
			32x32	24x24	32x16	16x16	
Hifi3	1	64 bit	2	4	4	4	3
Hifi3z	1	2x64 bit	2	4	4	8	3
Hifi4	1	2x64 bit	4	4	8	8	4
FusionF1	1	64 bit	1	2	2	4	2
FusionG3	2	2x128 bit	4	–	–	8	4
FusionG6	2	2x256 bit	8	–	–	16	4
ConnX BBE16EP	4	128 bit	–	–	–	16	5
ConnX BBE32EP	8	256 bit	–	–	–	32	5
ConnX BBE64EP	16	512 bit	–	–	–	64	5
VisionP5	4	1024 bit	16	–	–	64	–
VisionP6	5	1024 bit	16	–	–	64	–

sind aber linear abhängig voneinander und können nicht parallel ausgeführt werden (siehe Abbildung 3.5). Daher werden im Folgenden nur Coprozessoren betrachtet, die parallel zu Ausführungseinheiten in der EX-Phase arbeiten. Für diese Arbeit wurden der Hifi3, der Hifi3z, der Fusion F1, der Fusion G3 und der Fusion G6 ausgewählt.

Der Hifi4 war nicht Teil der Auswahl, da dieser gegenüber dem Hifi3z keine wesentlich performanteren Verarbeitungseinheiten bietet, aber laut Abschätzung der Fläche des Xtensa Xplorer-Tools signifikant größer ist. Die ConnX-Reihe hat zwar dedizierte Beschleuniger für Teile des Algorithmus, diese sind aber meist auf drahtlosen Kommunikationsalgorithmen optimiert und lassen sich nicht eins-zu-eins auf den HomePlug-1.0-Standard abbilden. Die Vision-Reihe profitiert vor allem durch seine massive Parallelisierung von kleinen Recheneinheiten für die Bildverarbeitung. Dieser Grad an Parallelität wird vom hier zu implementierenden Algorithmus nicht erreicht und die Vision-Reihe daher nicht verwendet.

Eine weitere Möglichkeit des LX7-Basisprozessor Algorithmus zu beschleunigen, ist die *Tensilica Instruction Set Extension* (TIE) Hardware-Beschreibungssprache. Diese ist eine proprietäre Sprache zur Erweiterung der ALU um dedizierte funktionale Einheiten. Neben den Möglichkeiten der Erweiterung des LX7-Prozessors können auch die vorhandenen Basisblöcke, wie Cache oder Datenspeicher, konfiguriert werden. Unter anderem lässt sich die Cache-Struktur modifizieren und deren Assoziativität anpassen. Auch können die Größen aller Speicher (Cache, IMEM, DMEM) angepasst und optional ein direkter Speicherzugriff (engl. *Direct Memory Access*, DMA), verschiedene Buffer oder eine Prefetch-Einheiten instantiiert werden.

Um diesen Parameterraum umfassend abdecken zu können, wurden für jeden gewählten Prozessor mehrere Konfigurationen der Speicher- und Pipeline-Architektur getestet. Die verwendeten

Tabelle 4.2: *Liste aller evaluierten Konfigurationen der Hifi- und Fusion-Reihe mit ihrer jeweiligen Hardware-Spezifikation und Abkürzung.*

	DSPs	Pipeline	Cache (I+D)	Lokaler Speicher	Abkürzung
Hifi3	2	5	–	256 kByte	H3LM5
	2	5	12 kByte	–	H3C5 min
	2	5	40 kByte	–	H3C5
	2	7	40 kByte	–	H3C7
Hifi3z	2	5	–	256 kByte	H3zLM5
	2	5	12 kByte	–	H3zC5 min
	2	5	40 kByte	–	H3zC5
	2	7	40 kByte	–	H3zC7
Fusion F1	2	5	–	256 kByte	F1LM5
	2	5	12 kByte	–	F1C5
Fusion G3	3	5	–	192 kByte	G3LM5
Fusion G6	3	5	–	192 kByte	G6LM7

Konfigurationen sind in Tabelle 4.2 aufgelistet. Im Folgenden werden für die Konfigurationen der Prozessoren deren Abkürzungen in der letzten Spalte der Tabelle verwendet. Dabei wird zwischen der Verwendung eines Caches C oder nur lokalem Speicher LM und der Anzahl der Prozessor-Pipeline-Stufen (5 oder 7) unterschieden. Zum Beispiel steht H3zC7 für den Hifi3z-Coprozessor, eine siebenstufige Prozessor-Pipeline und die Verwendung von Caches. Der Zusatz „min“ bezeichnet eine Konfiguration mit minimal großem Cache.

Zunächst wurde der unoptimierte C-Referenz-Code auf die verschiedenen Prozessorkonfigurationen portiert. Als Compiler wird *XT-Clang* mit Optimierungslevel O3 verwendet, der durch einbetten von Funktionen eine Optimierung über die Funktionsgrenzen hinaus erlaubt. Weiterhin wurden folgenden Compiler-Optimierungen aktiviert:

Use DSP Co-Processor:

Aktiviert den internen DSP mit zusätzlichen DSP-Registern (zusätzlich zum DSP-Coprozessor).

Speculatively vectorizing loops with ifs:

Spekulative Vektorisierung von Schleifen mit bedingten Sprüngen.

Create seperate data/function sections:

Separierung von Datensätzen und Funktionen zur Steigerung der Cache-Performance.

Enable long calls:

Erlaubt große Sprünge über den gesamten Adressraum des Prozessors.

Don't serialize volatile memory access:

Neuordnung von Speicherreferenzen für effizienteren Zugriff.

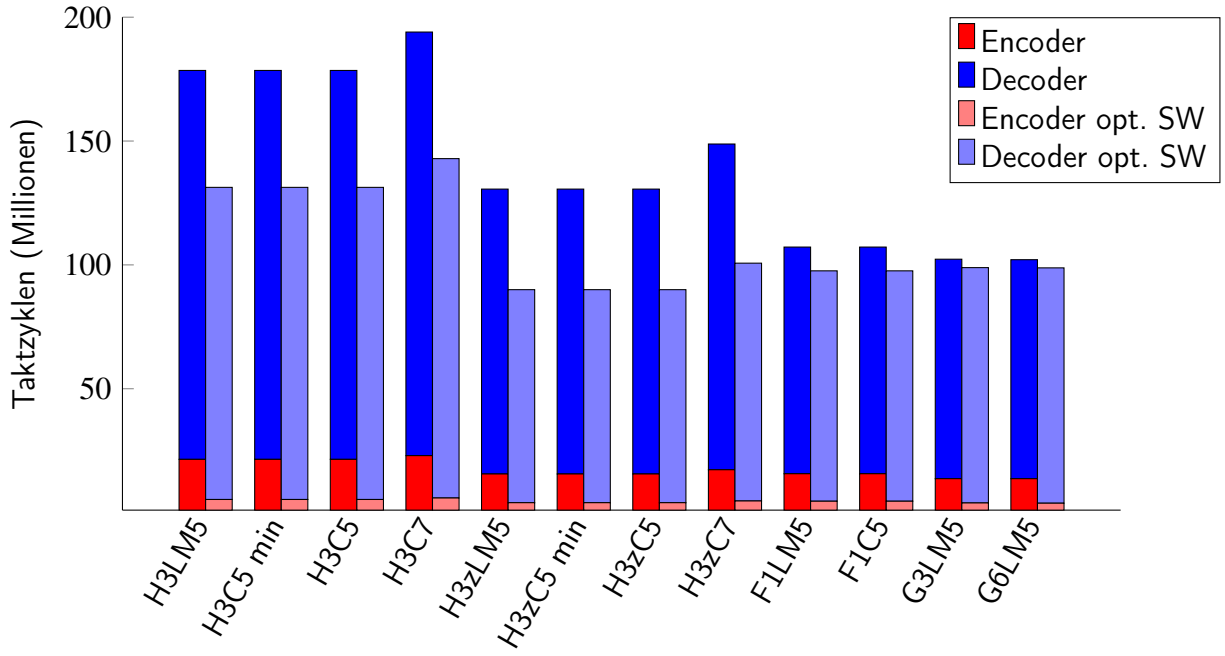

Abbildung 4.10: *Ausführungszeit des Powerline-Algorithmus getrennt nach Encoder und Decoder auf den Konfigurationen des LX7-Prozessors. Linke Balken ohne Software-Optimierung, rechter hellerer Balken mit Software-Optimierung.*

Die Gesamtausführungszeit unter Verwendung dieser Optionen für die einzelnen Konfigurationen ist in Abbildung 4.10 gezeigt.

Die jeweils linken Balken einer jeden Konfiguration zeigen die benötigten Taktzyklen in Millionen für die Berechnung eines Paketes des Encoders (rot) und des Decoders (blau) ohne algorithmische Optimierung. Der hellere rechte Balken zeigt die benötigten Taktzyklen mit den algorithmischen Optimierungen des Viterbi-Decoders, der Galois-Feld-Arithmetik und der Kreuzkorrelation im Frequenzbereich (siehe Abschnitt 4.1.1). Die C7-Konfigurationen der Hifi-Reihe brauchen mit ihrer siebenstufigen Pipeline etwa 8 % mehr Taktzyklen für die gesamte Berechnung, da diese nicht immer voll ausgenutzt werden können. Ob es sinnvoll ist, eine siebenstufige Pipeline zu verwenden, kann erst durch eine ASIC-Synthese und der daraus resultierenden Prozessortaktrate bestimmt werden. Insgesamt benötigt die Hifi3-Reihe die meisten und die Fusion-Reihe die wenigsten Taktzyklen, wie auch durch die Anzahl an parallelen Verarbeitungseinheiten zu erwarten war (siehe Tabelle 4.1).

Mit den Software-Optimierungen des Algorithmus aus Abschnitt 4.1.1 wurden bei der Hifi-Reihe im Schnitt 26 % und bei der Hifi3z-Reihe 31 % der Taktzyklen eingespart. Bei der Fusion-Reihe wurde eine Einsparung von 12 % für die beiden Fusion F1 und nur 7 % für die Fusion G3 und G6 erreicht. Bei genauerer Aufschlüsselung der Taktzyklenreduzierung nach Verarbeitungsblöcken fällt auf, dass sich eine deutlich geringere Verbesserung bei allen algorithmischen Optimierungen der Fusion-Reihe gegenüber der Hifi-Reihen feststellen lässt und beim Viterbi-Decoder sogar eine Verschlechterung. Dies liegt an den, in der Fusion-Reihe bereits vorhandenen, parallelen Verarbeitungseinheiten, die innerhalb von algorithmischen Blöcken voll ausgenutzt werden können. Vor allem der Traceback-basierte Viterbi-Decoder kann von der Fusion-Reihe deutlich effizienter berechnet werden als bei den Hifi-Konfigurationen. Der

Register-Exchange-basierten Viterbi-Decoder bietet deutlich weniger Potential für parallele Berechnung, was zu einer durchschnittlichen Verschlechterung der Taktzyklen der Fusion-Reihe von 15 % gegenüber der Traceback-Variante führt. Der Viterbi-Decoder hat, wie auch bei der Implementierung auf der PULP-Plattform (vgl. Abschnitt 4.1.1), mit 66 % den größten Anteil an der Gesamttaktzyklenzahl. Dadurch kompensiert die Verschlechterung bei Fusion-Reihe den Gewinn der anderen Komponenten fast vollständig. Insgesamt sind nach der Software-Optimierung die H3zC5- und H3zLM5-Konfigurationen am performantesten.

Hardware-Erweiterungen

Um die Performance des System weiter zu steigern, werden, wie auch bei der PULP-Plattform, der Viterbi-Decoder und die Galois-Feld-MAC als anteilig größte Teilkomponenten in dedizierter Hardware umgesetzt. Die Implementierung erfolgt hier in der proprietären Hardware-Beschreibungssprache TIE des Cadence Tensilica Xtensa Xplorer. Diese integriert die beschriebenen funktionale Einheit direkt in die EX-Stufe der Prozessor-Pipeline und erweitert somit den Instruktionssatz von diesem. Software-seitig lässt sich diese erweiterte Einheit direkt mittels intrinsischen Funktionsaufrufen ansprechen. Damit kann die gesamte Einheit in nur einem Takt berechnet werden und spart gegenüber dem Register-Interface der PULP-Plattform die langen Zugriffslatenzen.

Im Falle des Viterbi-Decoders würde die Berechnung in einem Takt zu einem kritischen Pfad von 18 ns führen und damit das System stark drosseln. Daher wurde eine Pipeline-Stufe in der PMU eingefügt, so dass der kritische Pfad nicht mehr durch den Viterbi-Decoder läuft. Die Galois-Feld-MAC ist klein genug, so dass sie in einem Takt ausgeführt werden kann und nicht Teil des kritischen Pfades ist.

Die resultierenden Ausführungszeiten mit Hardware-Erweiterungen sind in Abbildung 4.11 gezeigt. Die linken und die mittleren Balken sind die bereits bekannten Ausführungszeiten mit unoptimierten und optimierten Algorithmen aus Abbildung 4.10. Der rechts Balken zeigt die Ergebnisse mit Hardware-Erweiterungen. Zu erkennen ist, dass sich die Taktzyklenzahl des Encoders nicht verändert hat, was daran liegt, dass beide Hardware-Erweiterungen nur im Decoder Verwendung finden. Der Decoder wurde mit 94 % stark beschleunigt und ist nur noch 25 % langsamer als der Encoder. Mit diesen Taktzyklenzahlen lässt sich ein balanciertes Powerline-Modem betreiben, das ähnlich schnell Pakete aussenden und empfangen kann.

Zum Vergleich der verschiedenen Konfigurationen wurde, wie bei der PULP-Plattform, eine Entwurfsraumexploration auf Basis der XT018-ASIC-Synthesen durchgeführt. Dafür sind mit Hilfe des *Memory Compilers* der Instruktions- und Datenspeicher auf Basis der Größe des Programmcodes flächenmäßig abgeschätzt worden. Für alle Konfigurationen hat der 32 kByte große Instruktionsspeicher einer Fläche von 5,92 mm^2 und der 8 kByte große Datenspeicher einer Fläche von 1,56 mm^2.

Die Ergebnisse der ASIC-Synthesen aller LX7-Konfigurationen sind in Abbildung 4.12 gegeben, die den gleichen Entwurfsraum wie Abbildung 4.9 in Abschnitt 4.1.1 zeigt. Die Datenrate ist auch hier das Maximum der Taktzyklenzahl von Encoder und Decoder und berechnet sich nach Gleichung 4.1. Zu erkennen ist, dass bereits die nicht optimierte Software-Implementierung auf Fusion-G-Konfigurationen doppelt so schnell ausgeführt werden wie die optimierte Software auf der PULP-Plattform. Diese Konfigurationen profitieren dabei aber nicht von der Software-Optimierung, da der Compiler durch das Abrollen der Schleifen und die parallele Ausführung

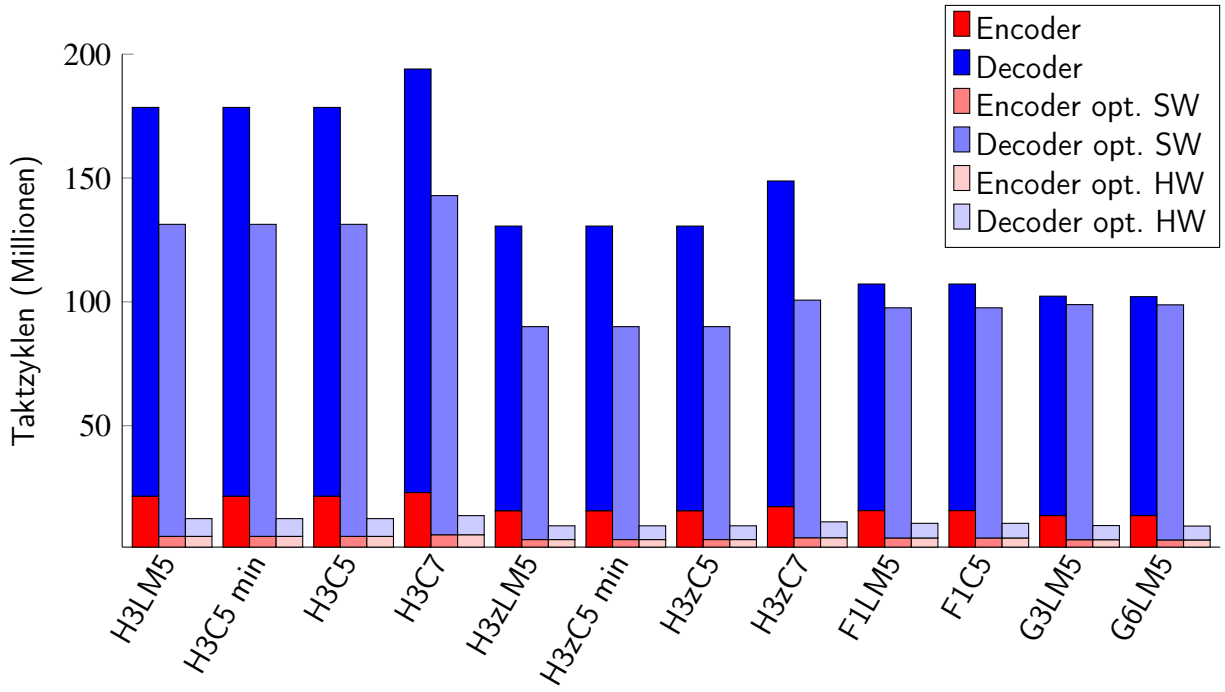

Abbildung 4.11: *Ausführungszeit des Powerline-Algorithmus getrennt nach Encoder und Decoder auf den Konfigurationen des LX7-Prozessors. Linke Balken ohne Software-Optimierung, mittlerer Balken mit Software-Optimierung, rechter Balken mit Hardware-Erweiterungen.*

auf den Vektoreinheiten die Verarbeitungszeit bereits optimiert. Bei der Hifi3-Reihe hingegen verbessert die Software-Optimierung die Datenrate um durchschnittlich 25 % und bei der Hifi3z-Reihen um durchschnittlich 31 %. Die Erweiterung mit den Hardware-Modulen steigert die Datenrate um den Faktor 17 für die Hifi-Reihe, und den Faktor 15 für die Hifi3z- und Fusion-Reihe gegenüber der unoptimierten Version. Der Hifi3-C7 erreicht als langsamste Konfiguration 180 kbit/s und der Hifi3z-C5 min und der Fusion G3 als schnellste Konfiguration 244 kbit/s. Dabei ist der Hifi3z-C5 min um den Faktor 1,7 kleiner als der Fusion G3. Der Flächenzuwachs durch die Hardware-Erweiterungen ist bei fast allen Konfiguration 2 mm^2. Bei den Fusion G- und den C7-Konfigurationen fällt der Zuwachs mit 2,7 mm^2 deutlich größer aus. Durch die direkte Implementierung der Hardware-Einheiten in die Prozessor-Pipeline werden für die Vektor-basierten G-Konfigurationen und die siebenstufigen C7-Konfigurationen breitere Register benötigt, was zu diesem Flächenzuwachs führt.

Folgende Pareto-optimale Punkte lassen sich in diesem Entwurfsraum identifizieren:

- Der CV32E40P-RISC-V-Kern als kleinstes Design mit 4,26 kbit/s Durchsatz.
- Der Fusion F1 mit lokalem Speicher und optimierter Software mit 14,10 kbit/s und 9,86 mm^2.
- Die PULP-Plattform mit Hardware-Erweiterungen mit 45,33 kbit/s und 11,31 mm^2.
- Der Hifi3 mit lokalem Speicher und fünf Pipeline-Stufen mit 205,4 kbit/s und 11,79 mm^2.
- Der Hifi3z mit minimalem Cache und fünf Pipeline-Stufen mit 244 kbit/s und 12,23 mm^2.

Für den Betrieb von einem Systemen mit geringen Datenraten oder großen Toleranzen der Latenz können die in diesem Entwurfsraum gezeigten Implementierungen bereits ausreichend sein.

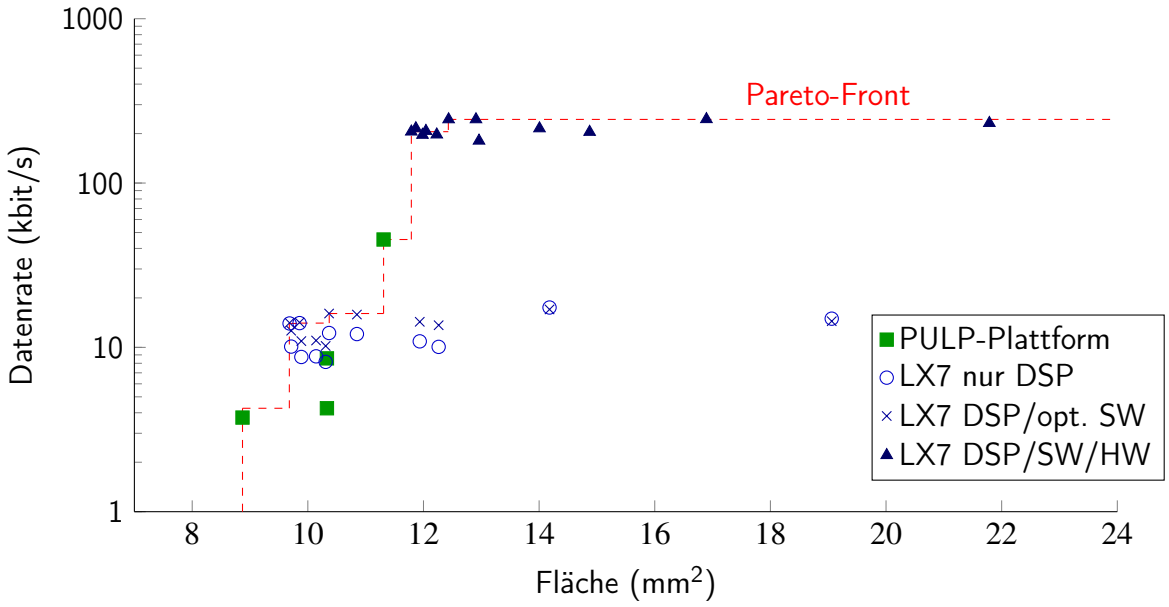

Abbildung 4.12: *Entwurfsraumergebnisse der LX7-Konfigurationen mit Software- und Hardware-Optimierung. Als Referenz sind die Ergebnisse der PULP-Plattform in grün gezeigt.*

Wichtig zu erwähnen ist, dass die Verwendung der HomePlug-1.0-PHY-Schicht mit geringerer Datenrate als im Standard spezifiziert nur möglich ist, wenn es im Decoder eine Detektionsschaltung in Hardware gibt, die den Anfang eines Paketes anzeigt. Diese Detektionsschaltung muss mit 50 MHz laufen und echtzeitfähig auf dem Bus nach Paketen suchen und diese abspeichern.

Um die Datenrate des Systems weiter steigern zu können und die vom Standard geforderten 14 Mbit/s zu erreichen, wurde eine dedizierte Struktur der PHY-Schicht in Hardware implementiert. Diese ist als Streaming-Architektur aufgebaut und wird im folgenden Abschnitt vorgestellt.

4.1.3. Dedizierte Hardware-Architektur

In diesem Abschnitt wird die Implementierung der PHY-Schicht als dedizierte Hardware-Architektur vorgestellt. Da sowohl die PULP-Plattform als auch die LX7-Prozessoren nicht den gewünschten Durchsatz von 14 Mbit/s erreichen konnten, werden alle Verarbeitungsblöcke in Hardware implementiert. Hierzu wird, anders als bei den Prozessor-basierten Implementierungen, ein Streaming-basierter Ansatz verwendet, der schritthaltend die Informationen vom AFE verarbeiten kann, ohne diese als gesamten Block zu speichern.

Der Aufbau der gesamten Hardware-Einheit ist in Abbildung 4.13 gezeigt und unterteilt sich in die zwei großen Bereiche der Encodierung und Decodierung. Das P1-*Interface* ist die im Standard festgelegte Schnittstelle zwischen MAC- und PHY-Schicht. Das AFE-*Interface* steuert das analoge Frontend und streamt Daten bidirektional vom Encoder oder zum Decoder mit $f_s = 2f_{\text{max}} = 50\,\text{MHz}$.

Die Idee der Streaming-Architektur ist die Vermeidung von großen Eingangs- und Ausgangsspeicherblöcken, die ganze Pakete zwischenspeichern müssen. Stattdessen werden die Informationen erst nach ihrer Decodierung am P1-*Interface* gespeichert. Die FC-Informationen

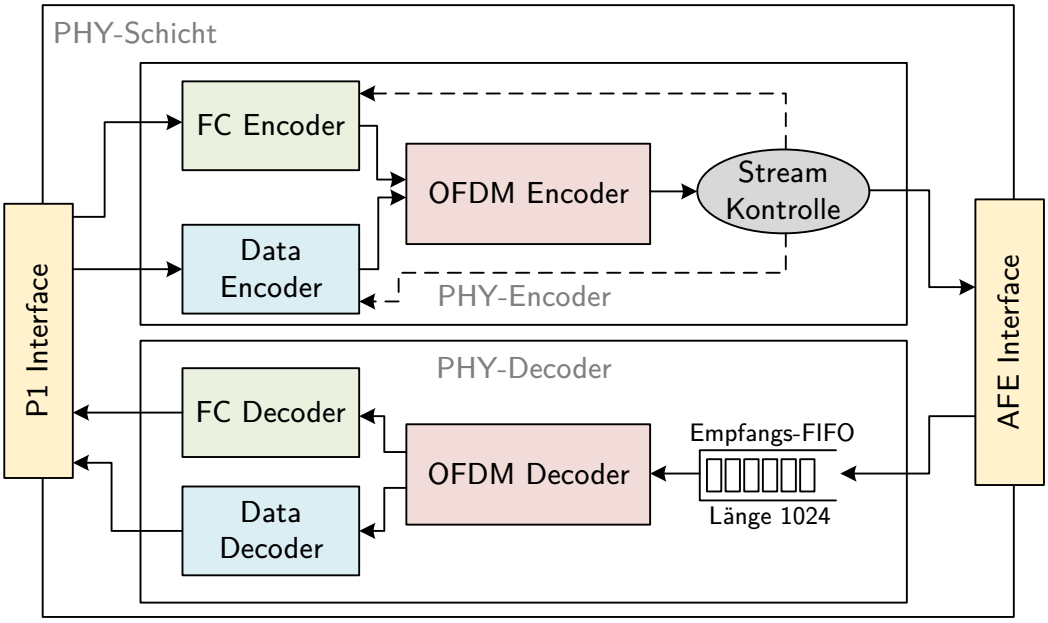

Abbildung 4.13: *Aufbau der dedizierten Hardware-Einheit als Streaming-Architektur unterteilt in Encoder und Decoder und mit P1-Interface zur MAC-Schicht und AFE-Interface zum AFE.*

setzen sich beim Empfang aus vier OFDM-Symbolen zusammen, also 1680 Werten mit 10 bit. Nach deren Decodierung bestehen die FC-Daten aus nur noch 25 bit, was einem Reduktion der Datenmenge um den Faktor 672 entspricht. Für die Nutzdaten kann selbst im Falle des größtmöglichen codierbaren Paketes von 2316 Byte ein Faktor 36 an Speicherbedarf eingespart werden. Auf ein gesamtes Paket gesehen können so die 74.400 Werte mit 10 bit in einem Speicher der Größe 2322 Byte gespeichert werden.

Um auch möglichst viele Zwischenspeicher innerhalb der Architektur zu vermeiden, wurde eine Stream-Kontrolle eingefügt, die die FC- und Nutzdaten-Encoder zyklengenau ansteuert. Dazu muss diese Kontrolleinheit die Verzögerungszeiten der einzelnen Blöcke kennen und entsprechend vorzeitig den Start der Codierung triggern. Dabei lassen sich nicht alle Einheiten als rein Streaming-basierte Struktur entwerfen. Die Bit- und ROBO-*Interleaver* schreiben einen 40er-OFDM-Symbolblock zeilenweise in eine Matrix, um diese dann spaltenweise wieder auszulesen (siehe Abschnitt 3.1.2). Dazu müssen erst alle Bits in der Matrix vorhanden sein, bevor man die erste Spalte auslesen kann. Für aufeinander folgende Blöcke ist nicht genug Zeit, um die Matrix zu leeren, bevor neue Daten in diese geschrieben werden müssen. Hier kann mit Hilfe einer Doppelpufferung (engl. *Double Buffering*) immer ein Speicher gelesen werden, während der andere geschrieben wird. Insgesamt werden dafür zwei 840 Byte große Speicher benötigt.

Für die im Standard nicht näher beschriebenen Blöcke des Reed-Solomon-Encoders und der IFFT wurden ebenfalls Streaming-basierte Architekturen implementiert. Für den Reed-Solomon-Encoder wurde eine Filterstruktur nach Chose [Cho98] verwendet, die die Addition und Multiplikation im Galois-Feld $2^m = 2^8$ berechnet. Das Filter hat eine Länge von 16, das zur Laufzeit über einen Multiplexer entweder die Koeffizienten des (239, 255, 8)-Codes für reguläre Modulation oder die Koeffizienten des (247, 255, 4)-Codes für den ROBO-Modus auswählt. Im

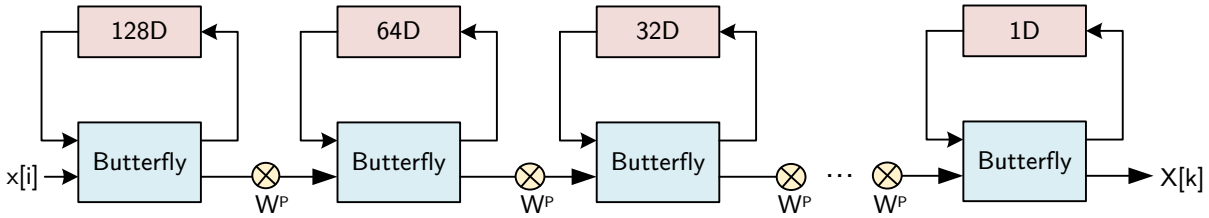

Abbildung 4.14: *Aufbau einer 256-Punkt Radix-2 DIF-FFT mit Single-Path Delay Feedback [He96].*

ROBO-Modus werden dabei die letzten acht Koeffizienten auf null gesetzt, da der Code nur acht Paritätsbytes berechnet.

Für die (I)FFT wurde ebenfalls eine Streaming-basierte Implementierung verwendet, da eine klassische voll-parallele Ausführung einen viel zu hohen Durchsatz erreichen und die meiste Zeit auf das Einlesen und Ausgeben der 256 seriell verarbeiteten Frequenz- oder Zeitwerte warten würde. Für diese Arbeit wurde daher eine Architektur mit rückgekoppelten Schieberegistern basierend auf der Idee von Wold et al. [Wol84] und der darauf aufbauenden Architektur von He et al. [He96] verwendet. Diese Architektur verarbeitet ein Datenwort pro Takt und kann kontinuierlich Daten durch die (I)FFT streamen. Die Struktur der verwendeten FFT ist als Differenzierung in Frequenzen (engl. *Differentiation in Frequency*, DIF) ist in Abbildung 4.14 dargestellt. Hier werden die einzelnen Stufen einer FFT in jeweils einem Butterfly (BF) berechnet und anschließend mit dem entsprechenden Drehfaktor (engl. *Twiddle Factor*) W^P multipliziert. Für die Realisierung der Encoder-seitigen IFFT als Differenzierung in Zeit (engl. *Differentiation in Time*, DIT) wird die gleiche Struktur verwendet mit Schieberegisterlängen in umgekehrter Reihenfolge, also von links nach rechts 1D, 2D, 4D, 8D, ... , 128D. Durch die rückgekoppelten Schieberegister entsteht eine Latenz von 256 Takten vom Eingang zum Ausgang. Nach dieser initialen Latenz liegt der Durchsatz dann bei dem oben erwähnten einem Wert pro Takt.

Alle weiteren Encoder-seitigen Einheiten konnten direkt, wie im Standard beschrieben, als Streaming-Architektur implementiert werden. Diese sind entweder bereits als Streaming-basierte Hardware-Struktur spezifiziert (Faltungsencoder, Produkt Encoder und *FCInterleaver*) oder verarbeiten alle Bits unabhängig voneinander (*Scrambler* und Modulator).

Am Eingang des PHY-Schicht-Decoders befindet sich ein FIFO der Größe 1024, um die Synchronisation und Kanalschätzung parallel auf mehreren Präambelsymbolen zu ermöglichen. Zunächst wird mit Hilfe der normierten Kreuzkorrelation (siehe Abschnitt 3.2.1) der Beginn eines Paketes detektiert. Dazu müssen mindestens zwei positive und ein negatives Präambelsymbol detektiert werden. Die Kreuzkorrelation verwendet dabei nur die obersten 5 bit des eingehenden Signals zur Korrelation mit der bekannten Präambel. Dies ist ausreichend, da für schlechtes SNR die unteren Bits des Signals vom Rauschen dominiert werden. Zusätzlich wird durch die AGC am AFE eine starke Dämpfung des Signals ausgeglichen, so dass die oberen 5 bit des Signals für alle noch decodierbaren Pakete ausgesteuert werden. Für die dedizierte Implementierung der Detektion einer Präambel nach Gleichung 4.3 in Abschnitt 4.1.1 kann diese umgeformt und vereinfacht werden. Die Konstanten können hierbei so gewählt werden, dass $0,25\sigma_p^2 = 115,81 \approx 128 = 2^7$ gilt und deren Multiplikation einen *Bit-Shift* von 7 realisiert. Damit wird der Schwellwert von $S_{th} = 0.5$ auf $S_{th} \approx 0,525$ angehoben, was für die Detektionsrate der Präambel eine vernachlässigbare Erhöhung ist.

Nach erfolgreicher Detektion der Präambel wird diese durch die oben vorgestellte DIF-FFT in den Frequenzbereich transformiert und dort zur Kanalschätzung nach Gleichung 3.11 verwendet. Diese beinhaltet die in Gleichung 3.12 gezeigte Berechnung des SNR. Kombiniert ergibt sich Gleichung 4.5.

$$\hat{G}(k) = \frac{P\hat{H}^*(k) - W\hat{H}^*(k)}{P|\hat{H}(k)|^2 - W|\hat{H}(k)|^2 + W} \tag{4.5}$$

Diese enthält vier Multiplikationen und eine Division, wobei letztere mit Hilfe des Newton-Raphson-Verfahrens approximiert werden kann [Fly70]. Das Verfahren berechnet die Inverse des Nenners, um diese dann mit dem Zähler zu multiplizieren. Da der Nenner reell und der Zähler komplexwertig ist, können Real- und Imaginärteil des Zählers mit dem gleichen Inversionsergebnis multipliziert werden. Der Wertebereich für die Berechnung der Inversion ist $0,5 \leq N \leq 1$ und besitzt den optimalen Startpunkt der Iterationen in Gleichung 4.6. Um auch Werte außerhalb dieses Bereiches berechnen zu können, wird mit Hilfe von Shift-Operationen der Wert in diesen Wertebereich verschoben, dort berechnet und anschließend zurückgeschoben. Um die Berechnung der Startbedingung Hardware-effizient umzusetzen, wird diese, wie in Gleichung 4.6 gezeigt, approximiert.

$$X = \frac{48}{17} - \frac{32}{17}N \approx 3 - 2N \tag{4.6}$$

Anschließend werden zwei Newton-Raphson-Iterationen nach Gleichung 4.7 durchgeführt.

$$X^{i+1} = X^i + X^i(1 - NX^i) \tag{4.7}$$

Mit nur zwei Iterationen ergibt sich eine Genauigkeit von 15 bit der Inversion, was für die erwartungsgemäß thermisch stark verrauschten Eingangswerte am AFE ausreichend ist.

Die resultierenden Koeffizienten zur Korrektur des Kanaleinflusses $\hat{G}(k)$ werden gespeichert und wie in Gleichung 3.11 gezeigt mit allen folgenden Datenblöcken multipliziert. Die korrigierten komplexen Koeffizienten $\hat{R}(k)$ werden anschließend *Soft Decision*-demoduliert (siehe Abschnitt 3.3).

Für die *Deinterleaver* wird, wie bei den *Interleavern*, eine Doppelpufferung verwendet, um ein Streaming zu ermöglichen. Die Größe der zwei Puffer ist mit 2520 Byte dreimal so groß, da der Demodulator 3 bit-quantisierte *Soft-Decision*-Informationen ausgibt. Die anschließenden Traceback-Viterbi- und RS-Decoder sind, wie in Abschnitt 3.2 vorgestellt, implementiert und bereits auf Daten-Streaming ausgelegt. Das Gleiche trifft auf den *Descambler* und der gesamte FC-Decoder zu.

Eine erste Synthese des Gesamtsystems in der XFAB XT018-Technologie zeigt, dass der kritische Pfad durch den Traceback des Viterbi-Decoders verläuft und etwa um den Faktor 6 zu langsam ist, um die vom Standard vorgegebenen 50 MHz zu erreichen. Das gesamte Design wurde in mehreren Iterationen, bestehend aus Synthese und Pipelining, beschleunigt, bis die erforderliche Länge von 20 ns des kritischen Pfades unterschritten wurde (inklusive eines Puffers für Platzierung und Routing von 1 ns). Mit dieser Taktrate wird auch die vom Standard geforderte Bruttodatenrate von 50 MSample/s und die PHY-Schicht-Nettodatenrate von 14 Mbit/s erreicht.

Da diese Implementierung die erste Version ist, die die geforderte Datenrate nach dem HomePlug 1.0-Standard erreicht, wurden auch die Sende- und Empfangslatenzen des Systems bestimmt. Diese betrugen für den gesamten Sendepfad 2 Takte oder 40 ns, für die Nutzdaten im Empfangspfad 1403 Takte oder 28,06 µs und für die *Trailer*-Informationen 668 Takte oder

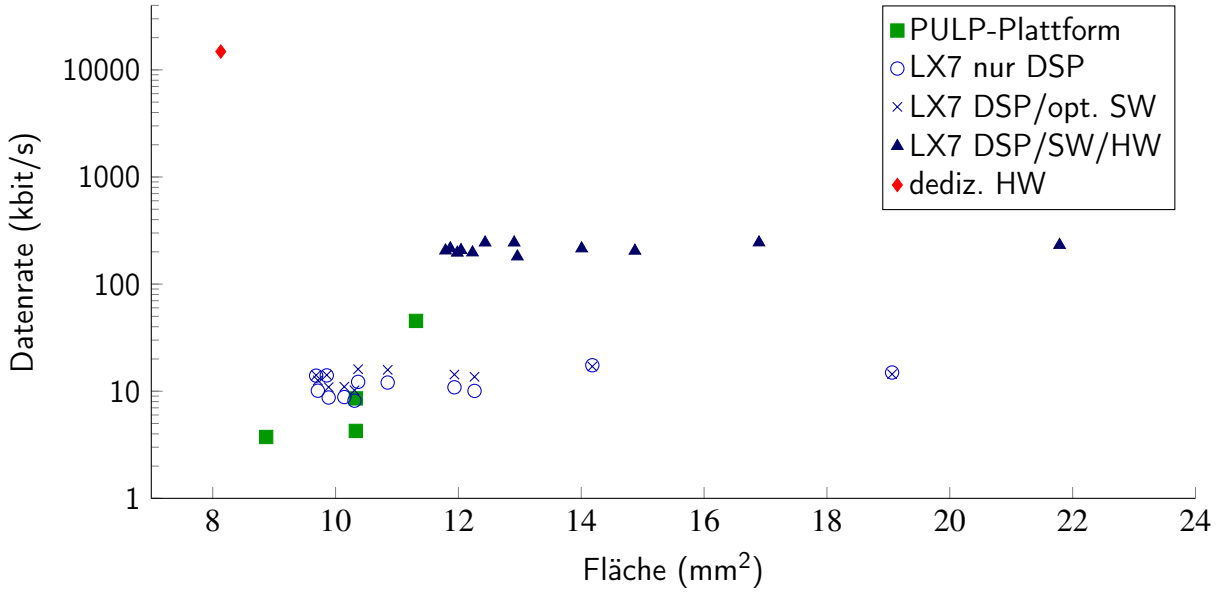

Abbildung 4.15: *Entwurfsraumergebnisse aller PHY-Schicht-Implementierungen: Die Ergebnisse der PULP-Plattform sind in grün und der LX7-Konfigurationen in blau gegeben.*

13,36 µs. Damit überschritt die Decodierung der Nutzdaten die Antwortzeit für ein Paket von 26 µs um 2,06 µs ohne Betrachtung der Verarbeitungszeit in der MAC-Schicht. Da die ARQ-Antwort Informationen über die CRC-Checksumme enthält, die über die Nutzdaten in der MAC-Schicht berechnet werden, musste der Decoder beschleunigt werden. Die größten Latenzen wiesen der Viterbi- und der RS-Decoder auf. Zur Beschleunigung wurde der *Deinterleaver* so modifiziert, dass zwei Bit pro Takt aus dem Speicher ausgelesen werden und alle folgenden Verarbeitungsblöcke ebenfalls zwei Bit parallel verarbeiten können. Dazu mussten im Viterbi-Decoder alle Submodule und im RS-Decoder die Berechnung des Syndroms angepasst werden. Da der RS-Decoder byteweise Daten ausgibt, wurde der *Descrambler* so modifiziert, dass er direkt auf diesen Bytes arbeiten kann und auch die Daten byteweise an die MAC-Schicht übergibt. Durch die simple Schieberegister-XOR-Struktur lassen sich die benötigten acht Iterationen direkt als Logikfunktionen abrollen (siehe Anhang B).

Als Ergebnis dieser Optimierungen erreicht der Empfangspfad des PHY-Decoders eine Latenz von 936 Takten oder 18,72 µs und ist somit um 9,34 µs beschleunigt worden. Auch die Antwortzeit von 26 µs wird um 7,28 µs unterschritten und steht der MAC-Schicht zur Verfügung, um eine Antwort auf das empfangene Paket zu generieren. Die Einordnung der Geschwindigkeit und der Größe in den bereits aufgespannten Entwurfsraum wird im folgenden Abschnitt vorgenommen und die Ergebnisse aller Implementierungen diskutiert.

4.1.4. Diskussion der Ergebnisse

Um die Implementierung der dedizierten Hardware einordnen zu können, wurde eine ASIC-Synthese für die XFAB XT018-Technologie durchgeführt und der Entwurfsraum aus Abschnitt 4.1.2 um die resultierende Fläche und Datenrate erweitert. Das Ergebnis ist in Abbildung 4.15 zu sehen. Wie zu erwarten, ist die dedizierte Implementierung mit 8,13 mm^2 die kleinste und

Tabelle 4.3: *Vergleich der optimalen Punkte bezüglich Größe, Geschwindigkeit und Flächenzuwachs pro Durchsatz. Dabei ist gilt für Fläche/Durchsatz: je kleiner, desto besser.*

Name	Fläche [mm^2]	Durchsatz [kbit/s]	Fläche/Durchsatz [$\mu m^2/(bit\,s)$]
CV32E40P	8,87	4,26	2033,36
Fusion F1 LM5 SW	9,68	13,98	676,19
PULP HW	11,31	45,33	243,66
Hifi3 LM5 HW	11,79	205,40	56,05
Hifi3z LM5 HW	12,41	243,92	49,68
Dedizierte HW	8,13	14.829,89	0,54

mit 14.830 kbit/s die schnellste Variante. Dabei ist der Durchsatz bei einer maximalen Taktrate von 68,88 MHz gegenüber der schnellsten Prozessorvariante um den Faktor 60 höher.

Tabelle 4.3 zeigt die Pareto-optimalen Punkte des Entwurfsraumes aus Abbildung 4.12 und der dedizierten Implementierung mit ihrer Fläche, Durchsatz und dem Quotienten aus beidem. Dabei gilt für den Quotienten: je kleiner der Wert, desto besser skaliert die Architektur. Die Einträge der Tabelle sind, mit Ausnahme der dedizierten Implementierung, monoton steigend in Fläche und Durchsatz, wobei der Durchsatz überproportional zur Fläche ansteigt. Dies liegt vor allem an den implementierten Hardware-Erweiterungen, die den Durchsatz stark erhöhen.

Den geringsten Quotienten aus Fläche und Durchsatz hat die dedizierte Hardware-Implementierung mit 0,54, was um den Faktor 92 besser ist als der schnellste ASIP Hifi3z mit Hardware-Erweiterungen mit 49,68. Der beste Prozessor ohne zusätzliche Hardware-Erweiterungen ist der Fusion F1 LM5 mit einem Quotienten von 676,19.

Basierend auf den Ergebnissen dieses Abschnittes lassen sich für den Einsatz in realen Powerline-basierten Kommunikationssystemen nach dem HomePlug 1.0-Standard drei optimale Architekturen angeben. Die erste Architektur ist der Xtensa LX7 Fusion F1 mit lokalem Speicher. Dieser ist mit 14,10 kbit/s für langsame Systeme mit geringer Auslastung geeignet, wie zum Beispiel ein Netzwerk mit wenigen Busteilnehmer, die Messwerte austauschen. Die zweite Architektur ist der Xtensa LX7 Hifi3z mit lokalem Speicher. Diese erreicht 244 kbit/s und eignet sich damit für Systeme mit vielen Teilnehmern, die nur sporadisch kommunizieren, oder mit wenigen Teilnehmer und hoher Busauslastung, wie zum Beispiel das Streamen von Sensordaten. Die letzte Architektur ist die dedizierte Hardware-Implementierung, die die maximale Datenrate von 14,48 Mbit/s erreicht und sich damit sowohl für Systeme mit hoher Auslastung als auch vielen Teilnehmern eignet. Diese Implementierung erlaubt das kontinuierliche Streamen von Messdaten vieler Teilnehmer, um zum Beispiel Sensordatenfusion über verschiedenste Sensoren zu realisieren.

Wichtig zu erwähnen ist, dass ein System mit weniger als 14 Mbit/s Übertragungsdatenrate nur realisiert werden kann, indem Pakete ohne erwartete Antwort gesendet werden. Diese Option lässt sich standardkonform umsetzten, indem Software-seitig das Antwort-Flag in der MAC-Schicht deaktiviert wird. Außerdem wird zusätzlich eine Korrelationseinheit als dedizierte Hardware benötigt, die das Powerline-Signal bei Erkennen eines Paketes aufzeichnet und den

Prozessoren bereitstellt. Hierfür könnte zum Beispiel die sehr Hardware-effiziente iterative Korrelation nach Schmidl und Cox [Sch97] verwendet werden. Diese benötigt nur ein Schieberegister in Länge der Präambel, zwei Multiplizierer und zwei Register. Werden diese beiden zusätzlichen Einschränkungen beachtet, ist eine Realisierung mit reduzierter Datenrate Standard-konform möglich.

Für die Integration des Gesamtsystems in dieser Arbeit und die anschließende Fertigung des ASIC wird die dedizierte Hardware-Implementierung verwendet, da diese ohne Adaption oder Einschränkung nach dem HomePlug 1.0-Standard kommunizieren kann. Darüber hinaus erreicht das System mit 14 Mbit/s eine Datenrate, die auch für zukünftige Systeme in der Tiefbohrtechnik mit vielen Teilnehmern, Sensoren und Aktoren noch eine ausreichende Bandbreite bereitstellen sollte.

4.2. Exploration der MAC Schicht

In diesem Abschnitt wird das in Abschnitt 3.1.1 vorgestellte Protokoll der MAC-Schicht als echtzeitfähige Hardware-Implementierung realisiert. Dafür wurde zunächst der im Anhang des Standard gegebene Pseudocode als C-Referenz-Implementierung umgesetzt. Die Herausforderung hierbei war es, die unvollständigen Abschnitte des Pseudocodes anhand der textuellen Beschreibungen im Standard umzusetzen und zu verifizieren. Unter anderem ist für die Berechnung einer *ToneMap* nur die Anweisung „*Calculate {VT,RATE,MOD} based on Channel Characterisitcs*“gegeben. Dies bedeutet, dass der physikalische Kanal geschätzt, evaluiert und die Übertragungsparameter VT (verwendbare Frequenzträger), RATE (Punktierungsrate) und MOD (Modulationsart) abgeleitet werden sollen.

Diese gesamte C-Implementierung wurde händisch gegen die im Standard definierten Prozesse und den Pseudocode verglichen (siehe Anhang C) und gilt für alle folgenden Hardware-Implementierung als ausführbare Referenz. Zur Verifikation der Hardware-Implementierungen wurde ein Testsystem, vergleichbar zu dem in Abschnitt 4.1, verwendet, wobei die Generierung der Testdaten und jeweiligen Vergleiche in C ausgeführt werden statt in MATLAB. Wie auch bei der PHY-Schicht werden in der MAC-Schicht Sender und Empfänger separat verifiziert und evaluiert.

Um zu bestimmen, ob eine Implementierung echtzeitfähig ist, muss zunächst die maximal mögliche Datendurchsatzrate der MAC-Schicht bestimmt werden. Die Ausgabedatenrate an die PHY-Schicht ist aus Abschnitt 4.1 bekannt und beträgt 14 Mbit/s. Die Datenrate am LLC-Interface lässt sich anhand des Standards und der PHY-Schicht-Datenrate unter Berücksichtigung des CSMA/CA-Protokolls berechnen. Der Standard gibt eine maximale Nutzdatengröße von $s_{Paket,max} = 1500\,\text{Byte}$ am Eingang der MAC-Schicht und eine maximale Länge des Serviceblocks am Ausgang von 1626 Byte an, inklusive Segment-*Header*. Unter der Annahme, dass Pakete als *Burst* ohne Wartezeit gesendet werden, ergeben sich die maximalen Datenraten durch die MAC-Schicht nach den Gleichungen 4.8 und 4.9.

$$D_{\text{max}} = \frac{s_{\text{Paket, max}}}{t_{\text{Protokoll, min}}} = \frac{1500\,\text{Byte}}{1261\,\mu\text{s}} = 9{,}07\,\text{Mbit/s} \tag{4.8}$$

$$D_{\text{max, AW}} = \frac{s_{\text{Paket, max}}}{t_{\text{Protokoll, AW}}} = \frac{1500\,\text{Byte}}{1359\,\mu\text{s}} = 8{,}42\,\text{Mbit/s} \tag{4.9}$$

Dabei ist D_{max} die absolute maximale Datenrate bei verbindungsloser (keine Antwort vom Empfänger) und $D_{\text{max, AW}}$ bei verbindungsorientierter Kommunikation (mit Antwort vom Empfänger). $t_{\text{Protokoll, min}}$ ist die minimale Länge eines Paketzykluses inklusive aller Wartezeiten ohne erwartete Antwort und $t_{\text{Protokoll, AW}}$ mit erwarteter Antwort. Diese Zeiten setzen sich aus der minimalen Paketdauer für ein 1626 Byte großes Paket mit 120 OFDM-Symbolen $t_{P,120} = 1153{,}5\,\mu\text{s}$, der Wartezeit von $3t_{\text{Slot}} = 107{,}52\,\mu\text{s}$ und der Zeit für die Antwort $t_{\text{AW}} = t_{\text{RIFS}} + t_{\text{Delimiter}} = 98\,\mu\text{s}$ zusammen.

Da die MAC-Schicht maximal 1626 Byte am Ausgang erreicht und nicht die maximal mögliche Segmentgröße der PHY-Schicht von 2076 Byte ausreizt, liegt die reale Datenrate am Eingang der PHY-Schicht bei 9,84 Mbit/s. Damit ist das Ziel für eine echtzeitfähige Implementierung eine Datendurchsatzrate von mindestens 10 Mbit/s zu erreichen. Auf Basis der Erkenntnis, dass nur die dedizierte Implementierung der PHY-Schicht mehr als 10 Mbit/s erreichen konnte, wird im folgenden Abschnitt die MAC-Schicht zunächst als dedizierte Hardware implementiert.

4.2.1. Dedizierte Implementierung

Ziel dieser Implementierung ist es, eine echtzeitfähige Kommunikation über die MAC-Schicht mit 10 Mbit/s in der XFAB XT018-Technologie zu erreichen. Dazu wurden zunächst die in Abschnitt 3.1.1 Abbildung 3.2 vorgestellten Zustandsdiagramme als Hardware-Blöcke beschrieben. Der detaillierte interne Aufbau der Diagramme ist im Anhang C in den beiden Abbildungen C.1 und C.2 gezeigt. Dabei orientiert sich die Struktur der Hardware direkt an diesen Zustandsdiagrammen und es wurden jeweils an den Zustandsübergängen Pipeline-Register eingefügt. Die aufwendigste und nicht im Standard beschriebene Struktur ist der DES-Algorithmus. Hier wurde eine Implementierung direkt nach der Funktionsbeschreibung im *Federal Information Processing Standard* (FIPS) 46-3 gewählt [Tec99].

Abbildung 4.16 zeigt den internen Aufbau der Verschlüsselung und Entschlüsselung im *Cipher-Block-Chaining*-Modus. In diesem werden die verschlüsselten Ausgangsdaten als initialer Vektor mit dem jeweils nächsten Block verrechnet. Für den ersten Block wird ein zufälliger Initialisierungsvektor (IV) verwendet, der dem Empfänger im Klartext mit übertragen wird. Nach einer initialen Permutation werden 16 identische Iterationen auf der linken und rechten Hälfte der Daten ausgeführt der permutierten Daten ausgeführt. Dabei führt jede Iteration die Feistel-Funktion (siehe Abbildung 4.16 rechts) aus, die die rechte Blockhälfte mit einem Subschlüssel verrechnet. Anschließend wird mit Hilfe einer nicht-linearen Transformation das Ergebnis verschlüsselt, permutiert und abschließend mit der linken Hälfte des Eingangsblocks verarbeitet. Insgesamt werden aus dem Netzwerkschlüssel 16 Subschlüssel berechnet, indem linke und rechte Hälfte des Schlüssels unabhängig zyklisch rotiert und anschließend davon je 24 bit ausgeschnitten werden. Am Ende wird eine weitere Permutation ausgeführt und der verschlüsselte Block ausgegeben. Für die Entschlüsselung eines Blocks müssen die Subschlüssel in umgekehrter Reihenfolge angewendet und das CBC vor der Ausgabe der Daten anstatt am Eingang ausgeführt werden.

Da die resultierende Hardware ohne Pipelining einen kritischen Pfad besitzt, der durch alle 16 Feistel-Funktionen verläuft, wurden initial Register nach je vier Iterationen eingefügt. Anschließend wurde eine erste Synthese des gesamten MAC-Schicht-Designs für die XFAB XT018-Technologie durchgeführt. Zusätzlich wurde mit Hilfe der Simulation in Cadence INCISIVE der Durchsatz für

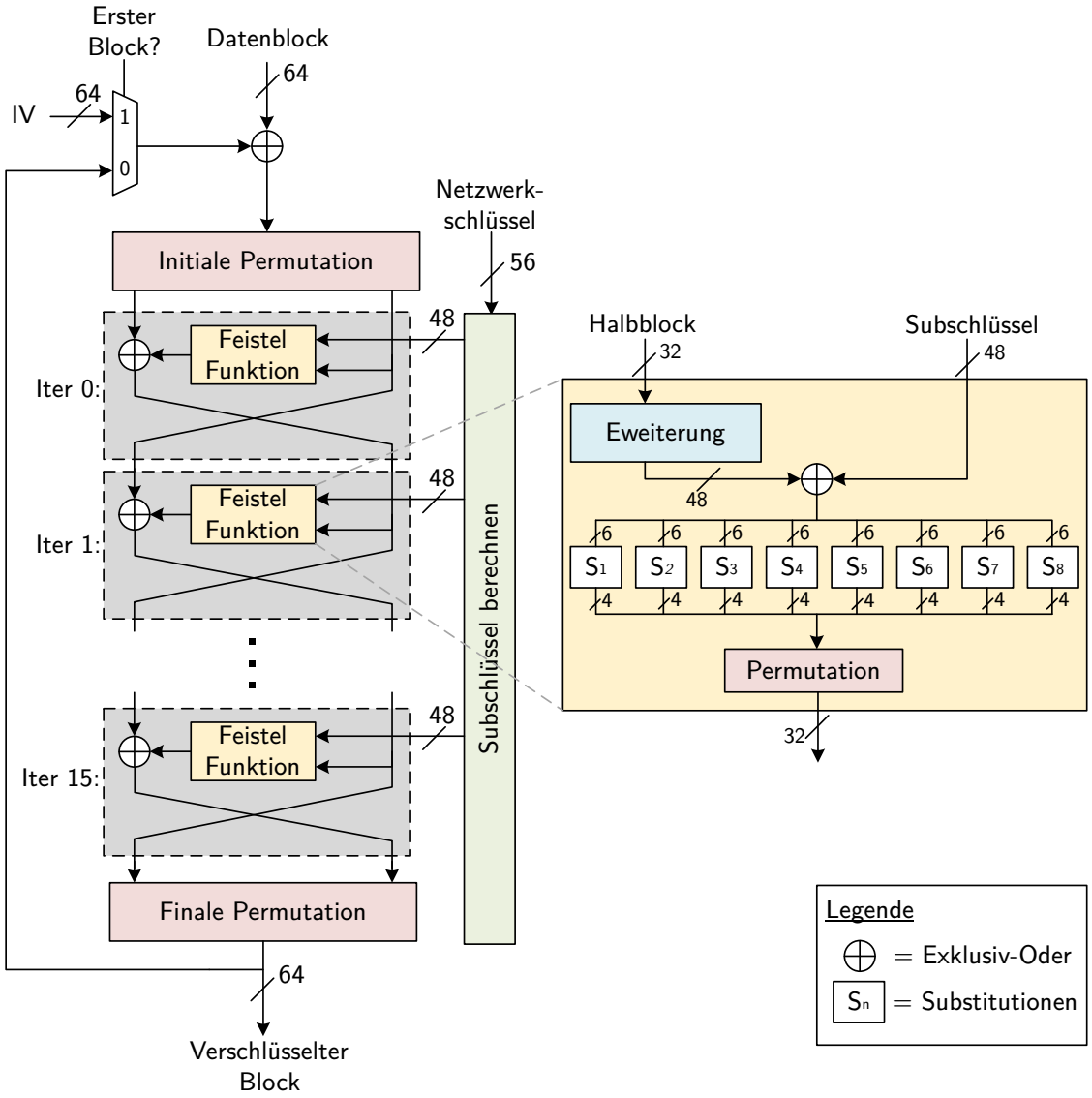

Abbildung 4.16: *Aufbau des DES nach dem* Federal Information Processing Standard *46 (FIPS-46) mit* Cypher Block Chaining *(CBC) und insgesamt 16 Iterationen der Kerngleichungen [Tec99]. Der rechte Teil zeigt den internen Aufbau der Feistel-Funktion mit den nicht-linearen Transformatoren S_n.*

ein maximal großes Paket des jeweiligen Moduls bestimmt. Die Ergebnisse dieser Synthese sind in Tabelle 4.4 dargestellt.

Die flächenmäßig größten Module sind mit 0,5 mm^2 die DES-Module für die Ver- und Entschlüsselung der Daten. Diese haben mit 14,5 ns den längsten kritischen Pfad aller Module und somit den geringsten Durchsatz von 180 Mbit/s. Der Durchsatz der Module Frame Senden und Empfangen ist mit 10 Mbit/s zwar geringer, wird aber durch das dort ausgeführte CSMA/CA-Protokoll bestimmt, das feste Sende- und Wartezeiten vorgibt. Für eine Implementierung müssen lediglich die im Standard festgelegten Toleranzen dieser Zeiten eingehalten werden. Den höchste Durchsatz erreichen der MAC Service Block im Sender mit 420 Mbit/s, gefolgt von dem Zusammenfügen, der Segmentierung und dem MAC Service im Empfänger.

Insgesamt erreicht die dedizierte Hardware-Implementierung der MAC-Schicht zwar die geforderten 10 Mbit/s, ist aber in ihren Submodulen bis zu 41-mal schneller als erforderlich. Die

Tabelle 4.4: *Ergebnisse der Synthese und Simulation der dedizierten MAC-Schicht-Implementierung: Fläche und kritischer Pfad von der Synthese in der XT018-Technologie, Durchsatz berechnet aus der Anzahl Takte für ein maximal großes Paket in der Simulation mit einem Takt von 50 MHz. Die Teilmodule zeigen den jeweils längsten kritischen Pfad und geringsten Durchsatz ihrer Komponenten.*

	Zustandsdiagramm	Fläche [mm^2]	Durchsatz [Mbit/s]	Krit. Pfad [ns]
Sender	MAC Service	0,08	420	7,9
	Verschlüsseln	0,29	180	14,5
	Segmentieren	0,04	380	7,7
	Frame Senden	0,04	10*	9,5
	Link Status	0,02	–	11,2
Empfänger	MAC Service	0,30	340	8,8
	Entschlüsseln	0,31	180	14,5
	Zusammenfügen	0,09	400	10,5
	Frame Empfangen	0,06	10*	9,4
Teilmodule	Sender inklusive Speicher	0,65	10*	14,5
	Empfänger inklusive Speicher	2,07	10*	14,5
	Gemeinsame Speicher	3,67	–	–
Gesamt	MAC-Schicht	6,40	10*	14,5

*Die Datenrate ist durch das CSMA/CA-Protokolls festgelegt.

gesamte Fläche der MAC-Schicht beträgt 6,40 mm^2 und besteht aus 83 % Speicherblöcken für das Speichern des aktuellen Datenblocks und aus nur 17 % Logik für die eigentlichen Berechnungen der Zustandsdiagramme. Damit ist diese Implementierung zwar echtzeitfähig einsetzbar, aber nicht optimal in Bezug auf die Auslastung der Teilmodule. Ebenfalls zeigt eine statische Analyse der Verlustleistung, unter der Annahme einer Schaltaktivität von 50 %, eine Leistungsaufnahme von 206 mW. Diese Schaltaktivität ist zwar unrealistisch hoch für den realen Einsatz, zeigt jedoch das die Leistungsaufnahme der MAC-Schicht etwa 41 % der Leistungsaufnahme der PHY-Schicht beträgt. Da die PHY-Schicht deutlich rechenintensivere Algorithmen wie den Viterbi-Decoder oder die komplexe Basisband-Synchronisation ausführt, ist dieser Wert für die MAC-Schicht zu hoch.

Zur Balancierung der Durchsatzraten der Module und Verringerung des Flächen- und Leistungsbedarfs wird im folgenden Abschnitt eine Implementierung als Prozessor-basierte Streaming-Plattform vorgestellt.

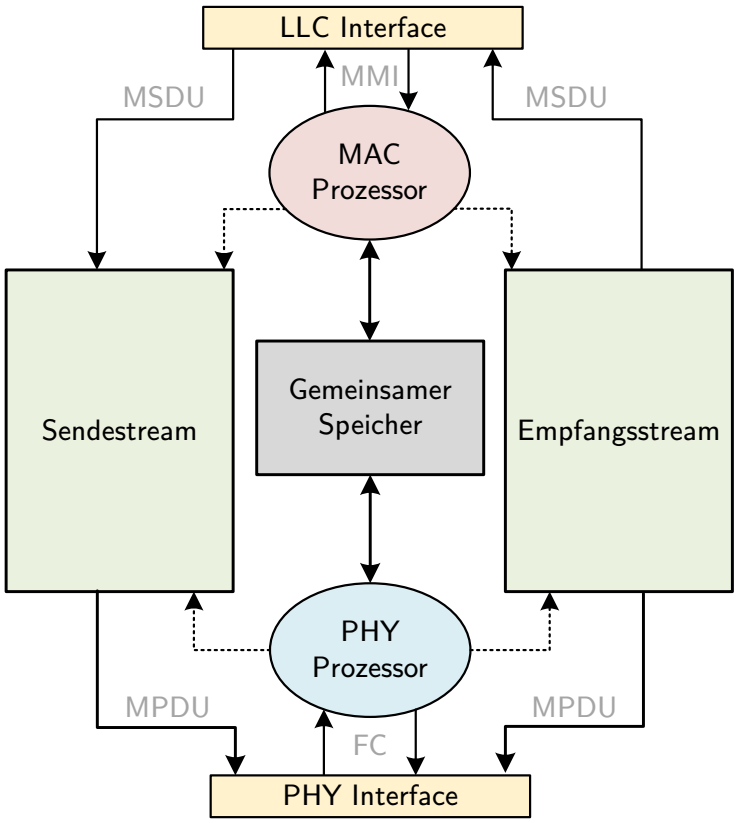

Abbildung 4.17: *Aufbau der Prozessor-basierten MAC-Schicht-Implementierung bestehend aus zwei Prozessoren, zwei Datenverarbeitungsstreams und einem gemeinsamen Speicher.*

4.2.2. Prozessor-basierte Streaming-Plattform

Da die direkte dedizierte Implementierung sehr viel Speicher benötigt und in einigen Modulen deutlich höhere Durchsatzraten erreicht als notwendig, wurde die Integration von Prozessoren untersucht. Auf diesen Prozessoren werden die Kontrollfluss-basierten Zustandsdiagramme ausgeführt, die wenig Logik und viel Speicher verwenden. Dies sind die beiden MAC-Service-Module und das Frame-Senden- und Frame-Empfangen-Modul. Dabei haben diese sehr unterschiedliche Anforderungen an ihre Ausführungszeit: Die Frame-Senden und Frame-Empfangen-Module sind zeitkritisch und simpel aufgebaut. Die MAC-Service-Module sind komplex, aber nicht zeitkritisch. Um die zeitkritischen Prozesse in ihrer Ausführung nicht zu unterbrechen, werden in dieser Implementierung zwei Prozessoren verwendet, die jeweils die beiden MAC-Service-Module (MAC-Prozessor) und die beiden Frame-Module (PHY-Prozessor) ausführen [Rot18].

Der Aufbau der Implementierung ist in Abbildung 4.17 dargestellt. Die beiden Prozessoren kommunizieren jeweils mit dem LLC- und PHY-Interface. Der MAC-Prozessor verarbeitet die *MAC Management*-Informationen (MMI), verwaltet die *Bridge*-Adressen und steuert die Sende- und Empfangsstreams. Diese beiden Sende- und Emfpangsstreams verarbeiten die zu sendenden oder zu empfangenen Daten zwischen den beiden Interfaces und berechnen aus einer MSDU eine MPDU und vice versa (vgl. Abbildung 3.3). Der PHY-Prozessor erstellt und verarbeitet die *FC-Header*- und *FC-Trailer*-Informationen und verfolgt die Aktivität auf dem Kanal, indem er die bereitgestellten Informationen der PHY-Schicht auswertet. Zwischen den beiden Prozessoren existiert ein gemeinsamer 32 Byte-großer Speicher zum Austausch von Kontrollinformationen,

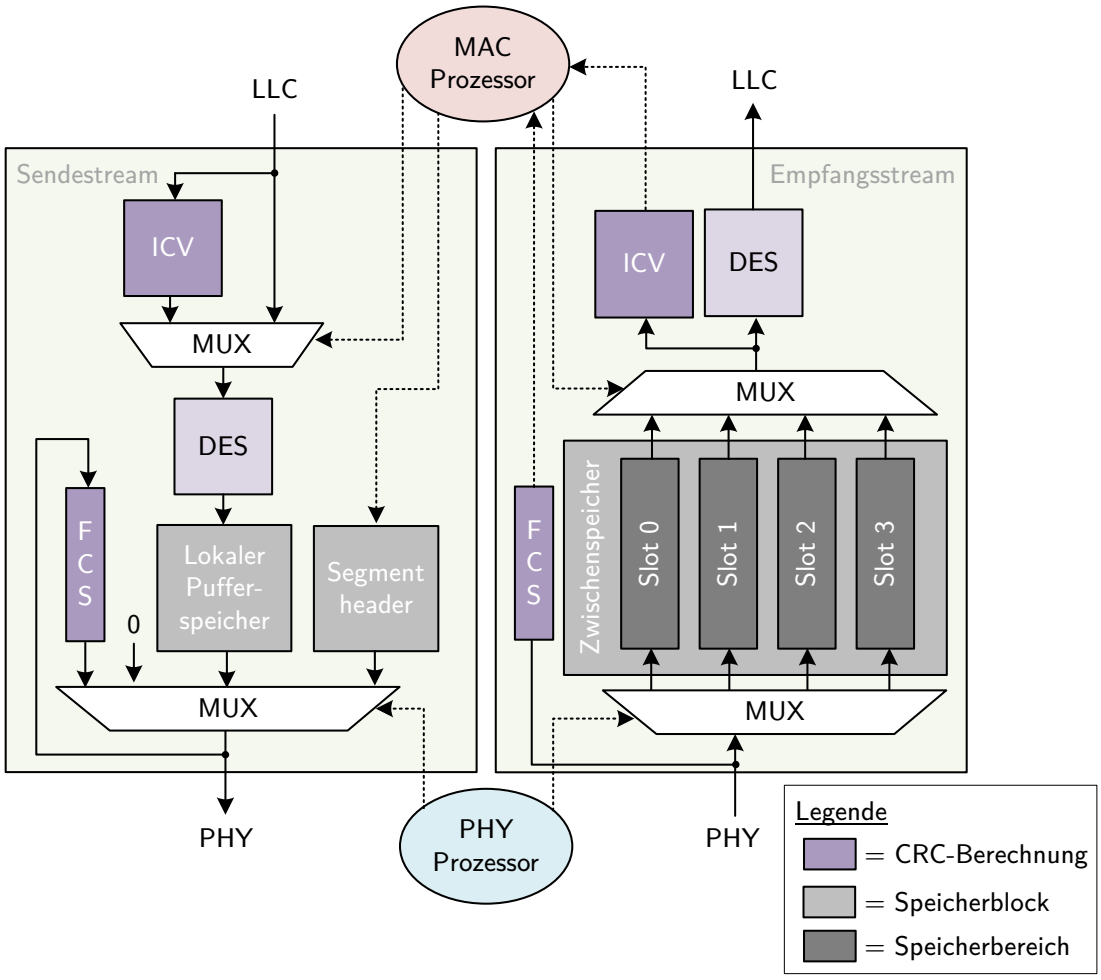

Abbildung 4.18: *Detaillierter Aufbau des Sende- und Empfangsstreams. Links der Pfad zur Verschlüsselung und Segmentierung und rechts das Zusammensetzen und Entschlüsseln eines Paketes. Zusätzlich sind in gestrichelten Linien die Kontrollsignale zu und von den Prozessoren dargestellt.*

wie zum Beispiel der Information, dass ein Paket zum Senden bereit ist oder die Priorität ein- und ausgehender Pakete. Beide Prozessoren können den gesamten gemeinsamen Speicher lesen, aber jeweils nur disjunkte 16 Byte schreiben. Somit wird verhindert, dass es zu Schreibkonflikten in den Speichern kommt.

Die beiden Stream-Einheiten enthalten die rechenintensiven Algorithmen ohne Steuerfunktion, wie den DES-Algorithmus und die CRC-Berechnungen. Der detaillierte Aufbau dieser Einheiten und deren Kommunikation mit den beiden Prozessoren ist in Abbildung 4.18 gezeigt. Beide Streams verwenden die Hardware-Module des DES-Algorithmus aus dem vorherigen Abschnitt 4.2.1. Um den zu hohen Durchsatz und die Fläche der voll gepipelineten DES-Architektur zu verringern, wurde das Design auf vier serielle Kern-Iteration reduziert (siehe Abbildung 4.16), die in einem Takt ausgeführt werden. Dies resultiert in einer Mehrzyklenarchitektur, die in vier Takten die sechzehn Iterationen pro 64 bit-DES-Block berechnet und das Gesamtergebnis ausgibt. Die dabei entstandene Latenz von vier Takten wird kompensiert von der byteweisen Zu- und Abführung der 64 bit-Verarbeitungsblöcke.

Die in Abbildung 4.18 in lila dargestellten Blöcke sind die Berechnung und Überprüfung der beiden CRC-Prüfsummen ICV und FCS. Der ICV wird byteweise parallel zur Berechnung der

Ver- und Entschlüsselung generiert und empfangsseitig vom MAC-Prozessor verifiziert. Die FCS wird parallel zur Übertragung des Paketes zu und von der PHY-Schicht berechnet und ebenfalls empfangsseitig vom MAC-Prozessor überprüft. Für jedes Segment, das gesendet werden soll, erzeugt die MAC-Schicht einen Segment-*Header* und legt diesen im gleichnamigen Speicher ab. Der PHY-Prozessor steuert die Generierung der Segmente, indem er für jedes Segment zuerst den aktuellen *Header* ausliest, dann den aktuellen Segmentinhalt und zum Schluss die FCS über das Segment anfügt. Dabei werden *Header* und FCS für jedes Segment neu berechnet.

Empfangsseitig kann ein Segment in vier verschiedenen Slots gespeichert und zusammengesetzt werden. Diese Slots sind notwendig, um Pakete von verschiedenen Sendern zusammenzusetzen, falls nicht alle Teilsegmente einer Station direkt aufeinander folgen. Zusätzlich dient der Zwischenspeicher als Puffer für die Pakete, bis sie von der LLC-Schicht entgegen genommenese übergeben.

Insgesamt enthält der Sendestream nur einen 17 Byte großen *Header*-Speicher und einen 2 kByte großen Frame-Puffer. Der Emfangsstream braucht auf Grund seiner vier Empfangsslots, die je ein maximal großes Paket enthalten können, insgesamt 8 kByte Speicher. Für die Verbindung zur PHY-Schicht wird durch die Stream-basierte Architektur kein Speicher mehr benötigt und die MPDU kann schritthaltend zur oder von der PHY-Schicht gestreamt werden. Die Steuerung der PHY-Schicht-Anbindung übernimmt hierbei der PHY-Prozessor.

Für die beiden zu integrierenden MAC- und PHY-Prozessoren wurde jeweils passende Prozessoren als existierende Hardware-Beschreibungen gesucht. Die Anforderungen an die beiden Prozessoren waren dabei: ein möglichst geringer Flächenbedarf, eine leichte Erweiterbarkeit um zusätzliche Hardware-Komponenten und eine flexible Bus-Infrastruktur zur Steuerung dieser Komponenten. Bei der Suche nach Prozessoren, die diese Anforderungen erfüllen, wurde für diese Arbeit der NEO430-Prozessor ausgewählt. Dieser ist instruktionssatzkompatibel zum MSP430 der Firma Texas Instruments, hat eine Instruktionssatzbreite von 16 bit und ist programmierbar in C bei Verwendung der offenen MSP430 -*GNU-Compiler-Collection* (GCC) [Nol22]. Eine Besonderheit des Prozessors ist seine Flexibilität und Erweiterbarkeit durch einen Systembus mit monolithischem Adressraum. Der Basisprozessor besteht nur aus dem Prozessorkern, einem Instruktions- und einem Datenspeicher. Zusätzlich können 15 Erweiterungsmodule Hardwareseitig zugeschaltet werden, wie z.B. Multiplizierer und Dividierer, externe Interrupts, Timer oder diverse Kommunikationsinterfaces. Außerdem werden alle Komponenten als *Memory-Mapped I/O* angesprochen, also durch definierte Adressen am zentralen Systembus. Dies erlaubt es, den Prozessor flexibel durch eigene Komponenten zu erweitern und so die beiden Streams direkt aus dem Prozessor anzusprechen.

Für den MAC-Prozessor wurde der NEO430 um eine Multiplizierereinheit und einen Timer erweitert und mit insgesamt 32 kByte Instruktions- und 4 kByte Datenspeicher ausgestattet. Auf diesem Prozessor laufen die beiden großen MAC Service-Module, das Link-Status-Modul und die Schnittstellen zum LLC-Interface. Die Ansteuerung der beiden Streams wurde durch Interrupts und Speichereinblendung realisiert.

Der PHY-Prozessor wurde ebenfalls als NEO430 realisiert und um eine Multiplizierereinheit und zwei Timer erweitert. Dieser besitzt 8 kByte Instruktions- und 2 kByte Datenspeicher. Die zwei Timer sind notwendig, um dem Protokoll auf dem Powerline-Bus folgen zu können. Auf diesem Prozessor laufen die beiden Module des Frame senden und Frame empfangen, die beide echtzeitfähig dem CSMA/CA-Protokoll folgen müssen. Um die Ausführung von parallelen

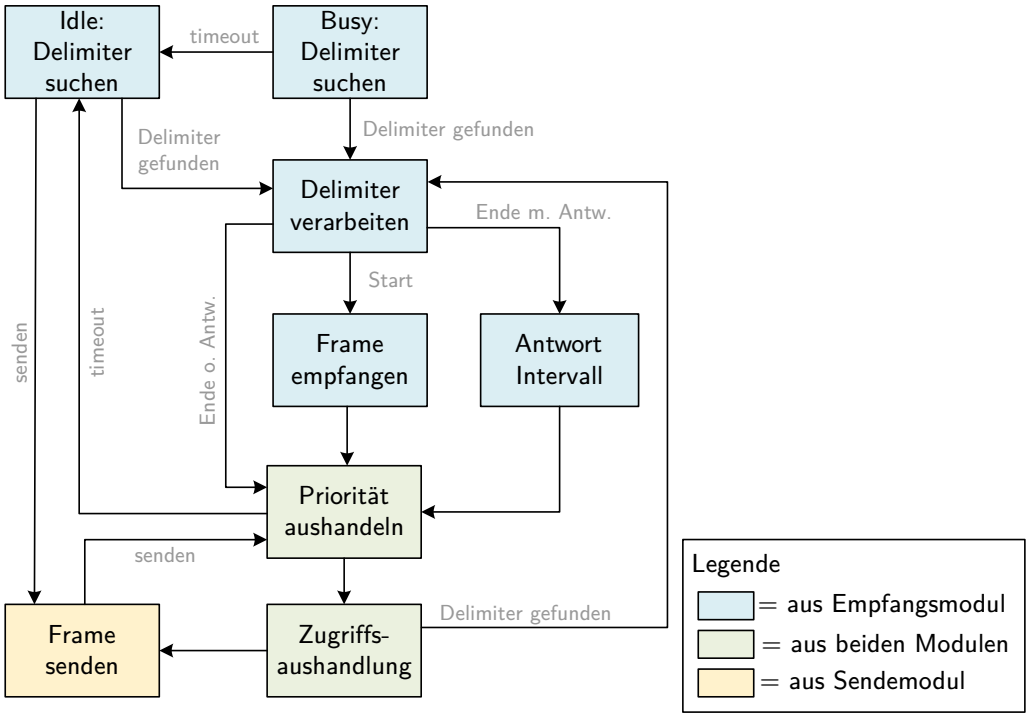

Abbildung 4.19: *Zusammengeführtes Zustandsdiagramm aus Frame senden und Frame empfangen. Die Farben zeigen jeweils den Ursprung der entsprechenden Teilaufgabe an. Die Informationen auf den Kanten geben den Zustand des Protokolls an.*

Threads auf dem Prozessor zu vermeiden, wurden die Diagramme aus den Abbildungen C.1 und C.2 im Anhang C kombiniert und resultieren in dem Diagramm in Abbildung 4.19.

Die Farben geben hierbei an, aus welchem der Zustandsdiagramme die jeweilige Aufgabe stammt. Die blauen Blöcke stammen aus dem Frame-empfangen- und und das gelbe aus dem Frame-senden-Modul. Die beiden grünen Blöcke sind zusammengeführte Aufgaben, die Teile von beiden Modulen enthalten. Das resultierende Zustandsdiagramm kann als Endlosschleife auf dem PHY-Prozessor ausgeführt werden und benötigt keine weiteren parallelen Algorithmen.

Nachdem alle Hardware- und Software-Komponenten integriert und verifiziert wurden, wurde erneut eine ASIC-Synthese in der XT018-Technologie durchgeführt. Die Ergebnisse der Synthese sind in Tabelle 4.5 dargestellt. Die Werte in Klammern zeigen die Unterschiede in Fläche und kritischem Pfad verglichen mit der rein dedizierten Implementierung. Für die Vergleichbarkeit bei den Prozessoren wurden jeweils die darauf ausgeführten Module flächenmäßig addiert und gegen den jeweiligen Prozessorkerne inklusive deren Instruktions- und Datenspeicher verglichen.

Zu erkennen ist, dass die beiden Streams deutlich kleiner sind als die jeweiligen Module der dedizierten Implementierung. Der kritische Pfad verläuft weiterhin durch die DES-Module, die auf Grund der Mehrzyklenausführung geringfügig mehr Kontrolllogik benötigen und somit 5 % langsamer sind. Auch die Prozessorkerne sind inklusive ihrer Instruktions- und Datenspeicher kleiner als die dedizierten Versionen der darauf ausgeführten Module. Der MAC-Prozessor hat einen um 29 % und der PHY-Prozessor um 40 % reduzierten Flächenbedarf. Wie auch bei den Streams verlangsamt sich der kritische Takt auf 11,4 ns und verläuft durch die Befehlsabrufeinheit (engl. *Instruction Fetch*, IF) der Prozessoren.

Tabelle 4.5: *Syntheseergebnisse der Prozessor-basierten Streaming-Implementierung und die Differenzen der Fläche und des kritischen Pfades zur rein dedizierten Implementierung. Die prozentualen Werte in Klammern geben eine Verbesserung (-) oder eine Verschlechterung (+) an. Die Größe der Prozessoren ist inklusive Instruktions- und Datenspeicher angegeben.*

	Fläche [mm^2]	krit. Pfad [ns]
MAC-Prozessor	1,88 (−29 %)	11,4 (+30 %)
Sendestream	0,23 (−54 %)	15,3 (+5 %)
Empfangsstream	0,23 (−60 %)	15,3 (+5 %)
PHY-Prozessor	0,61 (−34 %)	11,4 (+20 %)
Gesamt	3,61 (−43 %)	15,3 (+5 %)

Insgesamt kann mit dieser Implementierung 43 % der Fläche eingespart werden, bei nur 5 % langsamerer Ausführung (ohne Betrachtung der Module zum Frame senden und Frame empfangen). Die geschätzte Verlustleistung, bei eine Schaltaktivität von 50 %, ist mit 85 mW um 59 % gesunken. Der Anteil des Speichers am Gesamtsystem ist mit 84 % aber unverändert hoch. Zusätzlich zu den Ersparnissen in Fläche und Verlustleistung können die auf den Prozessoren laufenden Module schnell angepasst oder erweitert werden, da diese in C programmiert sind. Dies ist vor allem im Bezug auf die spätere ASIC-Integration des System relevant, da sich nach der Fertigung die Hardware nicht mehr ändern lässt.

Bei der Evaluation des Systems wurde festgestellt, dass der PHY-Prozessor für die Bestimmung der zu sendenden Antwort an die Gegenstelle langsamer ist als die geforderte RIFS-Zeit von 26 µs. Dies liegt unter anderem an ineffizient erzeugtem Assemblercode für Shift-Operationen durch den GCC und an der Mehrzyklenarchitektur des NEO430. Kombiniert ergibt sich eine nicht-deterministische, ineffiziente Ausführungszeit. Mit Hilfe von *Inline*-Assemblerbefehlen wurde die Ausführungszeit beschleunigt und teilweise deterministisch festgelegt. Die resultierende Ausführungszeit beträgt 16 µs und ist damit 10 µs schnell als vom Standard gefordert.

Da aber während der RIFS-Zeit auch die PHY-Schicht teilweise noch bis zu 18,72 µs lang die Nutzdaten decodiert (siehe Abschnitt 4.1.3), ist auch diese beschleunigte Variante noch nicht schnell genug, um dem Protokoll zu folgen. Die Ausführungszeit der Algorithmen auf dem PHY-Prozessor muss deterministisch sein und in weniger als 9 µs geschehen. Daher wurde ein alternativer Prozessor gesucht, der diese Kriterien erfüllen kann.

Als geeignetster Kandidat hat sich der PauloBlaze-Prozessor herausgestellt. Dieser ist ein *Open Source* 8 bit-Mikrocontroller, der den Instruktionssatz des proprietären PicoBlaze-Prozessors der Firma Xilinx verwendet [Gen21]. Dieser Prozessor besitzt nur zwei Pipeline-Stufen und der Code wird in Assemblersprache beschrieben. Da dieser Prozessor für jede Instruktion nur zwei Takte benötigt und durch die Assemblerbeschreibung keine dynamische Ausführungsreihenfolge hat, ist die benötigte Ausführungszeit für ein Programm immer identisch. Weiterhin besitzt der Prozessor keinen Datenspeicher, sondern nur einen 32 Byte-großen Register-basierten *Scratchpad*-Speicher. Diese Größe ist ausreichend, da der Prozessor keinen *Stack* oder *Heap* benötigt, sondern nur Register-basiert arbeitet.

Tabelle 4.6: *Verarbeitungszeit und Durchsatz der Module der Prozessor-basierten Streaming-Implementierung.*

	Modul	Verarbeitungszeit [µs]	Durchsatz [Mbit/s]
Sender	MAC-Prozessor	261	43,8
	Sendestream	100	114,4
	PHY-Prozessor	–*	10,0
Empfänger	MAC-Prozessor	95	120,4
	Empfangsstream	78	146,7
	PHY-Prozessor	–*	10,0
Gesamt	Sender	315	10,0
	Empfänger	173	10,0

*Keine eigene Verarbeitungszeit, da nur dem vom Standard festgelegten CSMA/CA-Protokoll gefolgt wird.

Der C-Code des NEO430-PHY-Prozessors wurde in Assemblerbefehle portiert und anschließend der PauloBlaze integriert. Eine ASIC-Synthese für die XT018-Technologie ergibt eine Fläche von 0,04 mm^2 für den Prozessorkern und damit eine Reduktion von 45 % gegenüber dem NEO430. Inklusive aller Speicher hat der PauloBlaze eine Größe von 0,54 mm^2 und ist damit nur 14 % kleiner als der NEO430 als PHY-Prozessor. Dies liegt vor allem an dem 2 kByte großen IMEM, der für beide Prozessoren benötigt wird und alleine schon eine Fläche von 0,47 mm^2 aufweist. Die gesamte Größe und Leistungsaufnahme der MAC-Schicht hat sich somit nur marginal verringert auf 3,56 mm^2 und 78 mW.

Die Zeit zur Bestimmung der Antwort für den Sender beträgt mit dem PauloBlaze-Prozessor nur noch 2 µs und ist damit um den Faktor 8 beschleunigt worden. In Summe benötigen PHY- und MAC-Schicht nun 20,72 µs zur Decodierung des Paketes und Generierung der Antwort. Damit existieren 5,28 µs Puffer für eventuelle zusätzliche Aufgaben des PHY-Prozessors.

Abschließend wurden die Verarbeitungszeit und der Durchsatz der einzelnen Komponenten für ein 1500 Byte großes Paket bestimmt. Diese sind in Tabelle 4.6 gezeigt. Der Durchsatz des gesamten Systems beträgt 10 Mbit/s und wird weiterhin vom im HomePlug-1.0-Standard definierten CSMA/CA-Protokoll bestimmt, das auf dem PHY-Prozessor ausgeführt wird. Die Sendezeit des MAC-Service-Moduls von 261 µs und des Sendestreams von 100 µs überlappen sich teilweise und sind deshalb in der Gesamtverarbeitungszeit von 315 µs kleiner als die Summe der Einzelzeiten. Der Durchsatz der beiden Module ist dabei aber deutlich über den geforderten 10 Mbit/s. Die Latenzen des MAC-Service-Moduls und des Empfangsstreams sind deutlich geringer und erreichen ebenfalls Datenraten, die deutlich über den 10 Mbit/s liegen.

In Summe ist die Verarbeitungszeit des Senders zu groß, um konsekutive Pakete maximaler Größe ohne Wartezeit zu senden. Diese darf eine Latenz von 120 µs nicht überschreiten. Da sich die hier gezeigte Verarbeitungszeit aber auf das erste zu sendende Paket bezieht und in 195 µs für die Vorbereitung und 120 µs für das anschließende Senden unterteilen lässt, ist die Sendelatenz ausreichend gering. Ab dem zweiten Paket kann die Vorbereitung parallel zur Verschlüsselung

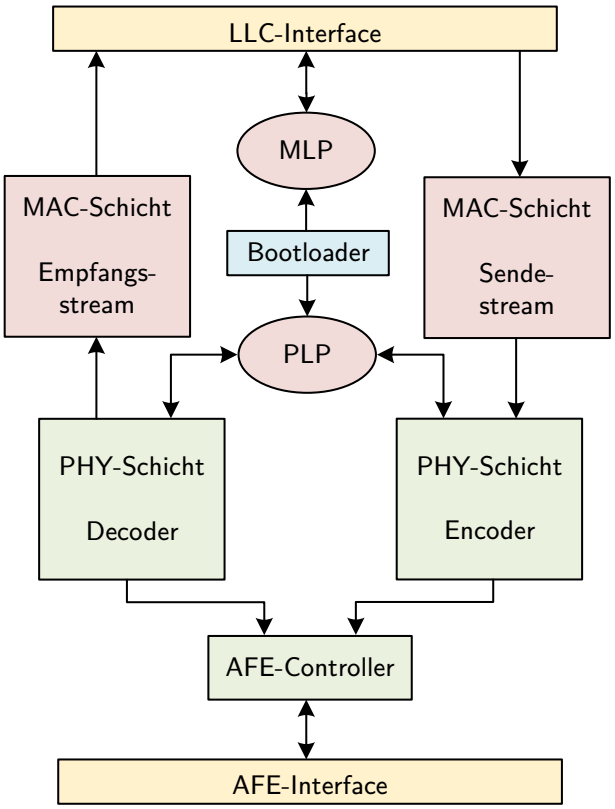

Abbildung 4.20: *Aufbau des gesamten Powerline-Modems bestehend aus der MAC- und der PHY-Schicht, einem AFE-Controller und einem extern steuerbaren Bootloader. Die PHY-Schicht-Module sind in grün und die MAC-Schicht-Module in rot gegeben.*

und zum Senden durch die PHY-Schicht erfolgen, so dass die effektive Sendelatenz nur 120 µs beträgt.

Mit dieser Implementierung existiert nun auch für die MAC-Schicht ein echtzeitfähiges, flexibles System, das für die Integration eines Gesamtsystem verwendet werden kann. Diese Integration und echtzeitfähige Evaluation auf Basis von FPGA-Emulationssytemen erfolgt im folgenden Abschnitt 4.3.

4.3. FPGA-basierte Evaluation

Nach der Optimierung der Streaming-basierten Implementierungen für die MAC- und die PHY-Schicht wird in diesem Abschnitt das gesamte Powerline-System evaluiert und echtzeitfähig charakterisiert. Dazu wird zunächst ein Gesamtsystem bestehend aus der MAC- und PHY-Schicht vorgestellt und anschließend in verschiedenen Kommunikationsszenarien getestet. Das gesamte Powerline-Modem ist in Abbildung 4.20 dargestellt und besteht aus der in Abschnitt 4.1.3 vorgestellten dedizierte PHY-Schicht-Implementierung (grüne Blöcke) und der in Abschnitt 4.2.2 vorgestellten Prozessor-Streaming-Version der MAC-Schicht (rote Blöcke). Für den Bootvorgang der Prozessoren wurde ein *Bootloader* ergänzt, der den Instruktionscode vor dem Start der beiden Prozessoren in deren Instruktionsspeicher lädt. Dieser *Bootloader* verfügt über ein Interface, das extern angesteuert wird und direkt die Instruktionsspeicher der beiden Prozessoren lesen und

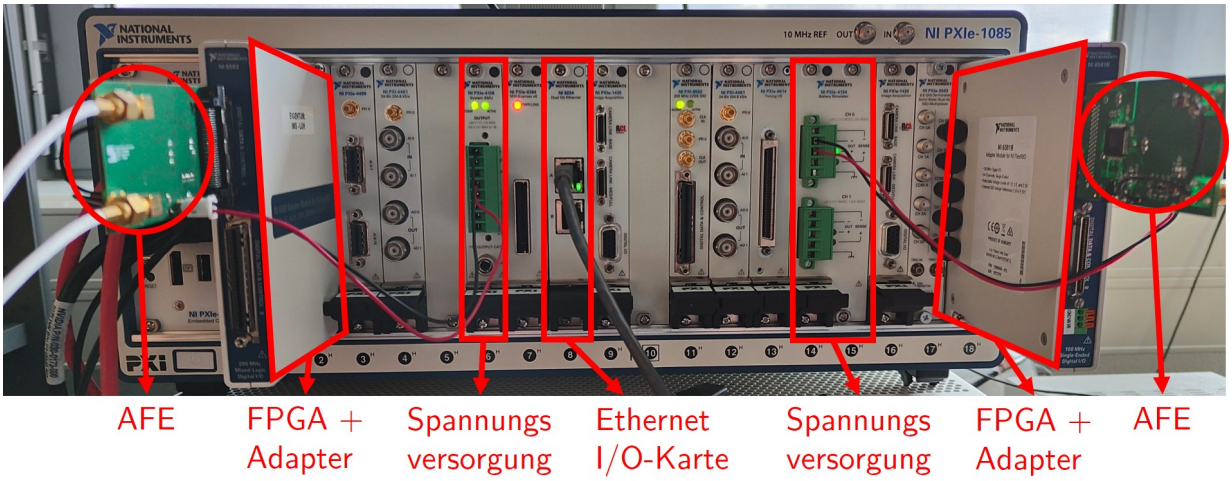

Abbildung 4.21: *Foto des PXI-Systems mit Markierungen der verwendeten Einsteckkarten. Links im Bild ist die Host-PC-Controller-Karte zu erkennen.*

schreiben kann. Dessen Adressbereich hat eine Tiefe von 14 bit, mit Datenworten der Breite 16 bit, was genau der Größe des NEO430-IMEM entspricht. Über ein zusätzliches Steuerbit kann zwischen dem NEO430- und dem PauloBlaze-IMEM umgeschaltet werden, wobei für den PauloBlaze nur die unteren 12 bit des Adressraums benötigt werden.

Die beiden PHY-Schicht Encoder und Decoder wurden direkt an den Sende- und Empfangsstream angeschlossen und vom PHY-Schicht-Prozessor (PLP) gesteuert. Zum AFE-Interface wurde ein AFE-Controller ergänzt, der den hier verwendeten Maxim MAX2981-AFE-Chip ansteuert. Für die Verwendung eines anderes AFEs muss der Controller Hardware-seitig adaptiert werden. Der AFE-Controller wählt unter anderem die Sende- und Empfangsrichtung aus, stellt die Signalverstärker ein und aktiviert oder deaktiviert die AGC. Außerdem wird das vorzeichenbehaftete digitale Signal im Wertebereich verschoben und als vorzeichenlose 10 bit-Zahl an das MAX2981 übergeben.

Das Gesamtsystem wurde mit Cadence INCISIVE 15.2 simuliert und gegen die MATLAB-Referenzimplementierung verifiziert. Dabei wurden alle Kommunikationsparameter, wie die Anzahl der Frequenzträger, die Modulationsarten oder die MAC-Managementanfragen getestet. In den folgenden beiden Abschnitten wird das verifizierte System auf FPGAs echtzeitfähig charakterisiert. Dazu wird in Abschnitt 4.3.1 die Kommunikation mit proprietären HomePlug-1.0-Modems getestet, die Latenz der Datenübertragung ermittelt und der bidirektionale Durchsatz gemessen. Anschließend wird in Abschnitt 4.3.2 die Skalierbarkeit des HomePlug-1.0-Powerline-Kommunikationssystems evaluiert, indem mit bis zu 60 Stationen parallel die Latenz und die Paketverluste in Echtzeit vermessen und ausgewertet wird.

4.3.1. Latenz- und Durchsatzmessung auf einem PXIe-System

Für die echtzeitfähige Verifikation und Charakterisierung des hier vorgestellten Gesamtsystems wurde ein National Instruments PXIe-System verwendet, das sich modular mit PXIe-Einsteckkarten aufbauen lässt [Nat15]. Dieses besitzt als Einschub einen Host-PC (PXIe-8135) mit Windows 10 und LabVIEW 2019 als Controller des gesamten PXIe-Systems. Außerdem

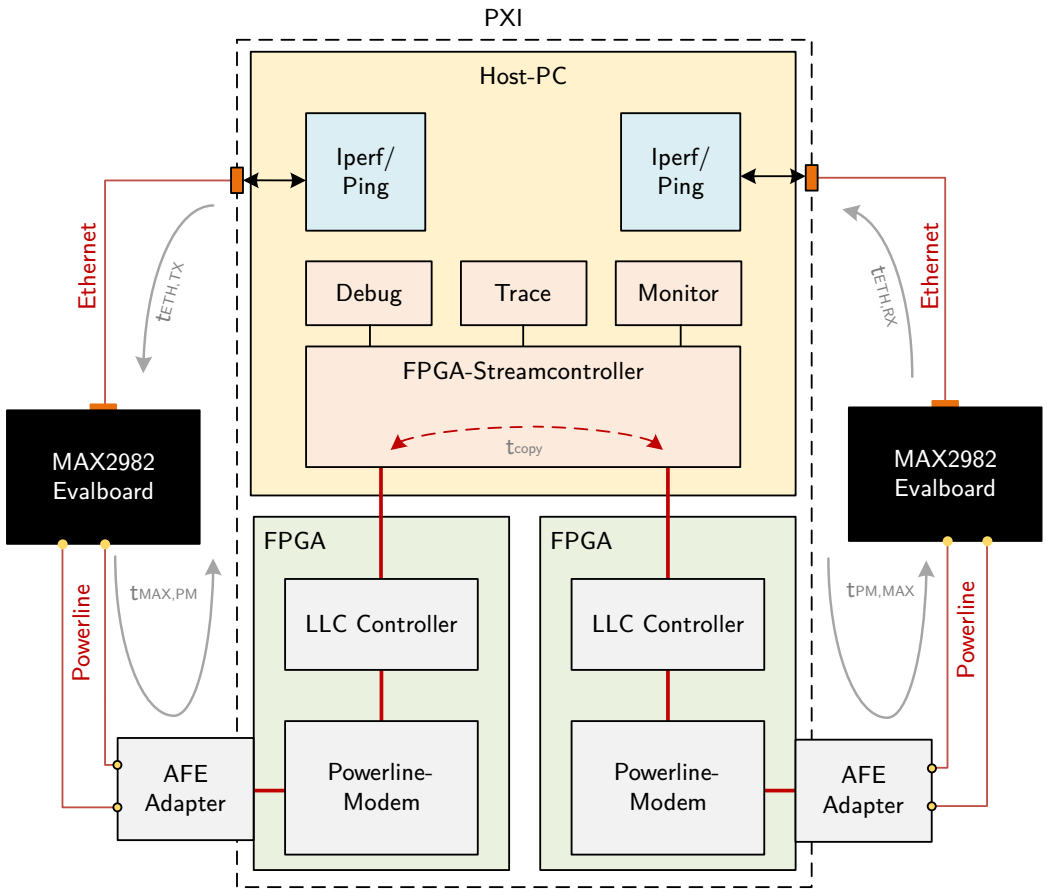

Abbildung 4.22: *Schematischer Aufbau und Ablauf der Kommunikation auf dem PXI-System zur Verifikation und Evaluation der Powerline-Modems gegen proprietäre Maxim MAX2982 Evaluations-Boards. Die roten Verbindungen zeigen den Kommunikationsweg der Ping- und IPerf-Tests.*

werden zwei FlexRIO-FPGA-Karten (PXIe-7975R) mit ein Kintex-7 K410T verwendet, um das Powerline-Modem zu emulieren. Diese FlexRIO-FPGA-Karten können mit einem Aufsteckmodul ausgestattet werden, das die externe Peripherie festlegt. Hier wurde zwei digitale I/O-Karte (NI 6581B und NI 6583) mit VHDCI-Steckern verwendet, an denen je eine Platine mit einem Maxim MAX2981-AFE steckt. Ein Foto dieses Aufbaus ist in Abbildung 4.21 gezeigt, wobei nur die rot markierten PXIe-Karten in diesem Versuchsaufbau verwendet werden. Die beiden Spannungsversorgungskarten sind jeweils an die PCBs der AFEs angeschlossen, um diese separat zu versorgen und deren Leistungsaufnahme zu messen.

Abbildung 4.22 zeigt den schematischen Aufbau des gesamten Testsystems inklusive der Beschaltung der beiden Maxim MAX2982-Evalboards. Letztere enthalten je ein vollständiges HomePlug-1.0-basiertes Powerline-Modem (MAX2982) inklusive AFE (MAX2981) und dienen als Referenz für die Verifikation der standardkonformen Implementierung dieser Arbeit. Außerdem lassen sich die MAX2982-Evalboards mit Hilfe einer *Ethernet*-Schnittstelle mit einem Host-PC verbinden, um eine Ende-zu-Ende-Kommunikation auf Applikationsebene zu realisieren. Jeweils eines dieser Boards wird über Coaxial-Kabel Powerline-seitig mit den AFE-Platinen am FPGA verbunden.

Auf dem FPGA läuft neben dem Powerline-Modem noch ein LLC-Controller, der sowohl das LLC-Interface des Modems ansteuern als auch Daten mit dem Host-PC per FIFO-Interface

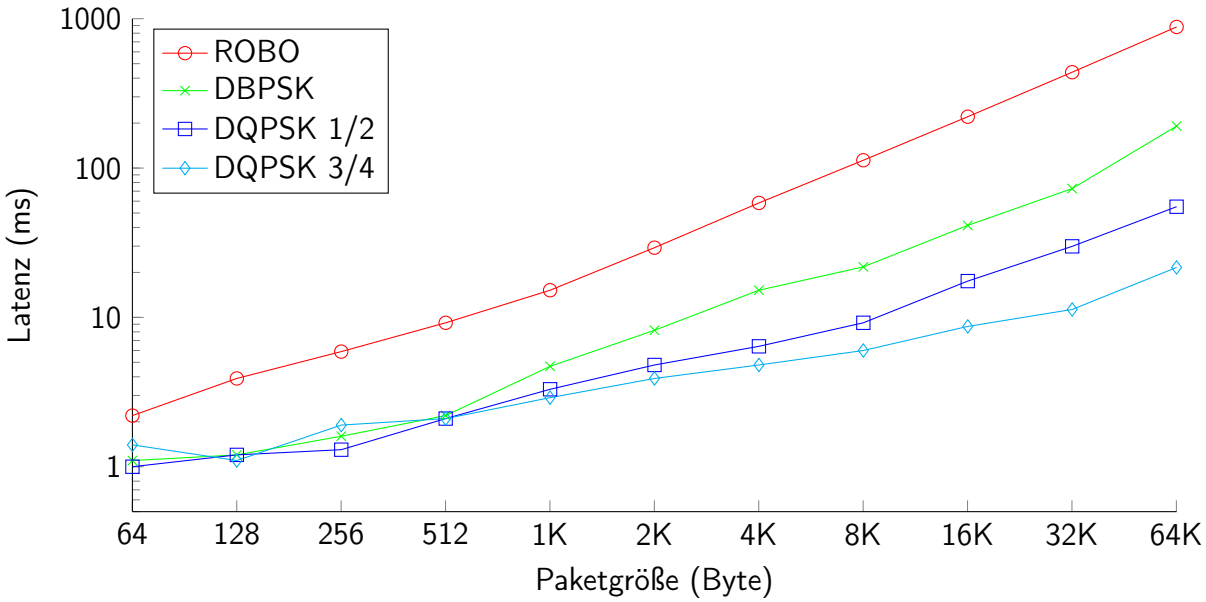

Abbildung 4.23: *Messung der Latenz durch ein FPGA-Powerline-System aus Abbildung 4.22 über die vier einstellbaren Modulationsarten.*

austauschen kann. Dieser Aufbau ist symmetrisch für beide FPGAs und wird nur über den Host-PC im PXIe-System gesteuert. Dieser übernimmt hierbei die Ausführung der Testanwendung als Windows-Programm (blau in Abbildung 4.22) und die Ansteuerung der beiden FPGAs in LAbVIEW (rot in Abbildung 4.22). Der FPGA-Stream-Controller kopiert die empfangenen LLC-Pakete zwischen den FPGAs, so dass pro Paket immer ein FPGA dieses empfängt und das andere dieses wieder aussendet. Zusätzlich kann der Controller über die Debug-Schnittstelle Pakete in Dateien abspeichern und über eine Trace-Einheit Debugausgaben der beiden MLPs anzeigen. Über einen Monitor kann auch die Kommunikation über die LLC-Interfaces beobachtet und verifiziert werden.

Durch diesen Aufbau wird auch bei unidirektionaler Ende-zu-Ende-Kommunikation die Sende- und Empfangsrichtung der Powerline-Modems getestet. Der gesamte Kommunikationsweg ist in Abbildung 4.22 in rot gegeben und beinhaltet zwei vollwertige Powerline-Kommunikationsnetzwerke mit je zwei Teilnehmern. Zunächst wurde mit Hilfe des Ping-Programms unter Windows die Ende-zu-Ende-Verbindung getestet und dessen Latenz als Mittelwert über 1000 Messungen bestimmt.

Dabei ist die gemessene Latenz die Paketumlaufzeit (engl. *Round Trip Time*, RTT) T_{RTT} entlang des roten Sendepfades in Abbildung 4.22. Gleichung 4.10 zeigt die darin enthaltenen Teilverzögerungszeiten.

$$t_{\mathrm{RTT}} = 2(t_{\mathrm{ETH,TX}} + t_{\mathrm{MAX,PM}} + t_{\mathrm{copy}} + t_{\mathrm{PM,MAX}} + t_{\mathrm{ETH,RX}}) \tag{4.10}$$

Dabei ist $t_{\mathrm{MAX,PM}}$ die Übertragungszeit über die Powerline-Verbindung von MAX2982 Evalboard nach FPGA-Powerline-Modem und $t_{\mathrm{PM,MAX}}$ die umgekehrte Übertragungsrichtung. $t_{\mathrm{ETH,TX/RX}}$ sind die Zeiten für das Senden (TX) und Empfangen (RX) der *Ethernet*-Pakete und t_{copy} die Kopierzeit des FPGA-Stream-Controllers, die jeweils für eine Paketgröße konstant

ist. Dabei kann nur für die FPGA-Powerline-Modems die Modulationsart angepasst werden und die MAX2982-Evalboards verwenden bei allen Messungen DQPSK mit Punktierung (3/4) als Modulationsart. Daher ist die Übertragungszeit $t_{\mathsf{MAX,PM}}$ für eine Paketgröße über alle Modulationsarten konstant.

$$t_{\mathsf{RTT,Eval}} = 2(t_{\mathsf{ETH,TX}} + t_{\mathsf{MAX,MAX}} + t_{\mathsf{ETH,RX}}) \tag{4.11}$$

Um nur die Latenz des FPGA-basierten Powerline-Modems zu erhalten, wurden die beiden MAX2982-Evalboards direkt per Powerline verbunden und die RTT dieser Teilstrecke nach Gleichung 4.11 bei allen Paketgrößen gemessen. Die Ergebnisse wurden von den Messungen der gesamten Kommunikationsstrecke subtrahiert, so dass nur noch die in Gleichung 4.12 enthaltenen Teilverzögerungen in Abbildung 4.23 dargestellt sind. Dabei kann angenommen werden, dass $t_{\mathsf{MAX,MAX}}$ ungefähr gleich groß ist wie $t_{\mathsf{MAX,PM}}$ und dass die Kopierzeit t_{copy} vernachlässigbar klein gegenüber der Powerline-Kommunikation ist, da die Pakete per DMA über eine PCIe-Verbindung mit vier *Lanes* (x4) kopiert werden.

$$t_{\mathsf{RTT,FPGA}} = \frac{t_{\mathsf{RTT}} - t_{\mathsf{RTT,Eval}}}{2} = t_{\mathsf{copy}} + t_{\mathsf{PM,MAX}} \tag{4.12}$$

Die Messungen in Abbildung 4.23 zeigen, dass die Latenz der Pakete im ROBO-Modus linear mit durchschnittlich 14,8 µs/Byte ansteigt und für kleine Pakete etwa doppelt so groß ist wie bei allen anderen Modulationsarten. Dies liegt vor allem an der hohen Redundanz der Codierung, wodurch nur 152 Byte an Nutzdaten in einem maximal großen Paket codiert werden können. Daher werden bereits ab 128 Byte Ping-Datengröße (plus Header aller OSI-Schichten) Pakete auf MAC-Ebene segmentiert und in mehreren Powerline-Paketen übertragen.

Bei DBPSK und DQPSK wurde eine geringere Latenz von etwa 1–2 ms für Pakete kleiner als 1 kByte gemessen. Ab 1 kByte ist für DBPSK die Grenze der Segmentierung von 692 Byte überschritten und ab 1,5 kByte kommt es zusätzlich zu Fragmentierung auf IP-Ebene. Der Anstieg der Latenz ab 1 kByte beträgt durchschnittlich 13,5 µs/Byte. Auch bei DQPSK ohne Punktierung (1/2) kommt es ab 1384 Byte noch zu Segmentierung und erst bei DQPSK mit Punktierung (3/4) ist die Grenze der Segmentierung mit 2076 Byte größer als die Grenze der Fragmentierung. Der Anstieg der Latenz ohne Punktierung liegt bei 0,9 µs/Byte und ist mit 0,4 µs/Byte am geringsten, wenn auch Punktierung verwendet wird.

Für den Einsatz des Systems entlang des unteren Endes eines Bohrstrangs sind vor allem kleine Paketgrößen ($\leq$512 Byte) relevant, da Sensoren und Aktoren meist nur geringe Datenmengen versenden müssen, dafür aber mit geringer Latenz. Für diesen Bereich der Latenz muss im hier implementierten Powerline-System nur zwischen ROBO- oder nicht ROBO-Modus unterscheiden werden, da die restlichen Latenzen nahezu identisch sind. Da DBPSK gegenüber DQPSK nach dem HomePlug-1.0-Standard einen um $\sqrt{2}$ größeren Störabstand in der Amplitude besitzt, ist für zukünftige Powerline in diesem Anwendungsbereich die Verwendung vom ROBO-Modus und DBPSK-Modulation ausreichend.

Zusätzlich zu der Latenz wurde auch der Durchsatz durch das Gesamtsystem über die Paketgröße und die verschiedenen Modulationsarten evaluiert. Anders als bei der Messung der Latenz können die MAX2982-Evalboards und die Kopierzeit vernachlässigt werden, da nur der Flaschenhals des Durchsatzes die Datenrate bestimmt. Da die MAX2982-Evalboards immer mit maximaler Datenrate (DQPSK mit Punktierung) senden und empfangen, liegt der

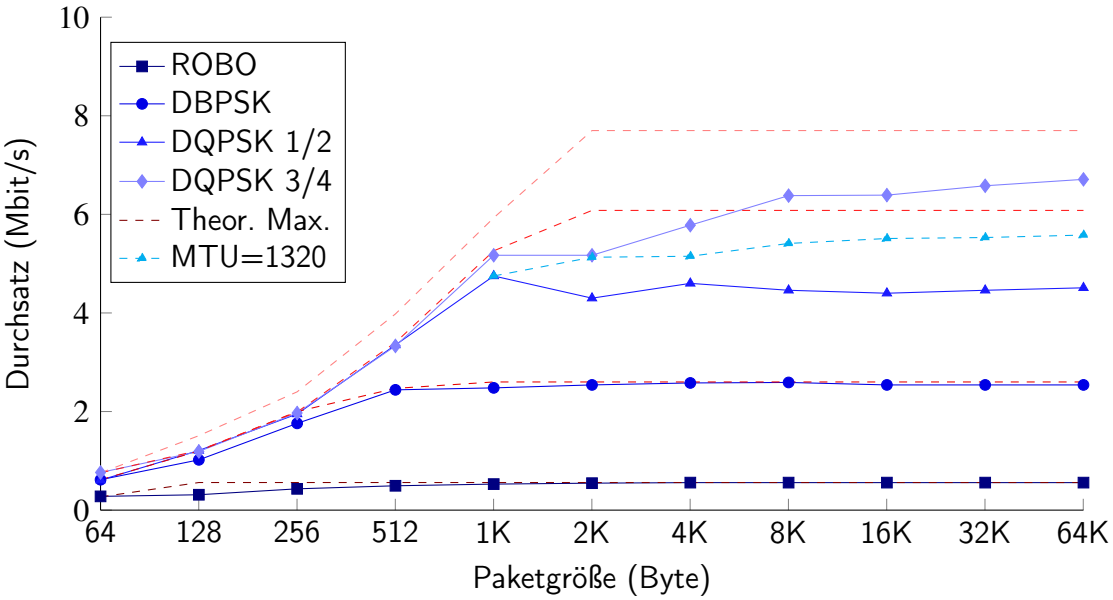

Abbildung 4.24: *Messung des UDP-Durchsatzes auf dem PXI-System für den ROBO-Modus und die drei verschiedenen Modulationsarten des Powerline-Systems. Die roten Kurven zeigen das jeweilige theoretische Maximum des Durchsatzes der MAC-Schicht für alle Messpunkte und Modulationsarten.*

gemessene Flaschenhals in der FPGA-basierten Implementierung. Die Messungen wurden mit dem Programm IPerf3 für TCP- und UDP-Kommunikation durchgeführt, das den maximalen Durchsatz einer Kommunikationsstrecke ermittelt, indem versucht wird, so viele Datenpakete wie möglich innerhalb eines gegebenen Zeitfensters zu übertragen. Die Bandbreite wurde auf 10 Mbit/s festgelegt und die Messdauer beträgt 10 s pro Protokoll und Paketgröße. für die UDP-Kommunikation sind die Ergebnisse der Messungen in Abbildung 4.24 gezeigt.

$$D_{\max}(S) = \frac{\min(S_{\text{Frame}}, S_{\text{Segment}})}{t_{\text{Segment}} + t_{\text{RIFS}} + t_{\text{AW}} + t_{\text{CIFS}} + 2t_{\text{PRS}} + 3,5t_{\text{Slot}}} \left[\frac{\text{bit}}{\text{s}}\right] \tag{4.13}$$

Die roten Kurven in den Abbildungen zeigen das theoretische Maximum des Durchsatzes auf MAC-Ebene bei Kommunikation mit erwarteter Antwort und Prioritätsaushandlung nach Gleichung 4.13. S ist die Größe und t_{Segment} die Dauer eines Segmentes. Dabei ändert sich t_{Segment} nur in diskreten Schritten, je nachdem, wie viele 20er- und 40er-OFDM-Symbolblöcke in einem Paket enthalten sind. $3,5t_{\text{Slot}}$ ist die mittlere Wartezeit der *Contention*-Phase ohne vorherige Kollisionen.

Die Kurven gehen ab dem Erreichen der maximalen Segmentgröße oder der Fragmentierungsgrenze von 1500 Byte Paketgröße in Sättigung. Die maximale Segmentgröße ist für den ROBO-Modus 152 Byte, für DBPSK-Modulation 692 Byte, für DQPSK ohne Punktierung 1452 Byte und für DQPSK mit Punktierung 2212 Byte, was größer ist als die Fragmentgröße.

Für den ROBO-Modus und Modulation mit DBPSK ist in Abbildung 4.24 zu erkennen, dass der UDP-Durchsatz das jeweilige theoretische Maximum ab 512 Byte Blockgröße erreicht. Für kleinere Datenblöcke liegt die Datenrate unterhalb des Maximums, da UDP-*Header* (8 Byte), IP-*Header* (24 Byte) und MAC-*Header* (32 Byte) relativ groß im Vergleich zu den Nutzdaten sind. Die DQPSK-Modulation ohne Punktierung erreicht das theoretische Maximum des Durchsatzes

bis 1 kByte Paketgröße und saturiert anschließend für größere Pakete bei einem Durchsatz von 4,5 Mbit/s. Dies liegt an der maximalen Segmentgröße von 1452 Byte, die bei einer MTU von 1500 Byte in zwei Segmenten pro Paket resultiert. Dieses zweite Segment enthält mehr *Header*-Daten als Nutzdaten und reduziert somit die Datenrate um etwa 1 Mbit/s.

Diese Reduzierung betrifft alle Pakete größer als 1 kByte und kann kompensiert werden, indem eine $MTU_{\text{red}} = 1320$, wie in Gleichung 4.14 berechnet, eingestellt wird. Die Messergebnisse mit dieser MTU sind in hellblau in Abbildung 4.24 ab einer Paketgröße von 1 kByte dargestellt. Mit dieser Maßnahme kann die Datenrate auf 5,58 Mbit/s bei maximaler Paketgröße angehoben und so um 24 % gesteigert werden.

$$\text{MTU}_{\text{red}} = \underbrace{1452\,\text{Byte}}_{\text{Segmentgröße}} - \underbrace{32\,\text{Byte}}_{\text{MAC-Header}} - \underbrace{87\,\text{Byte}}_{\text{MAC-Management}} - \underbrace{13\,\text{Byte}}_{\text{MAC-Trailer}} = 1320\,\text{Byte} \tag{4.14}$$

Der maximale Durchsatz von 6,71 Mbit/s wird mit einer Paketgröße von 64 kByte und DQPSK-Modulation mit Punktierung erreicht. Dabei erreichen die DQPSK-Modulation mit und ohne Punktierung nicht die theoretischen Maxima, auch nicht mit reduzierter MTU. Dies liegt an der Kombination aus der Sendelatenz von 261 µs (vgl. Tabelle 4.6) und der Verwendung von nur einem Sendepuffer. Letzterer kann erst für ein neues Datenpaket freigegeben werden, wenn ein ACK des Empfängers decodiert wurde, das heißt während der CIFS-Zeit vor der Prioritätsaushandlung (siehe Abbildung 3.4 b). Der MAC-Schicht verbleiben der Decodierung der Antwort von 13,5 µs durch die PHY-Schicht noch 94,02 µs, um den Sendepuffer freizugeben, ein neues Paket vorzubereiten und dieses zu verschlüsseln. Diese Zeit wird um 167 µs verpasst und führt zu der gezeigten Reduzierung der Datenrate für DQPSK-Modulation in Abbildung 4.24. Dieser Effekt tritt auch im ROBO-Modus und mit DBPSK-Modulation auf, da hier die Pakete aber segmentiert werden, tritt diese Latenz nur einmal beim letzten Segment auf und nicht wie bei DQPSK bei jedem Segment [Stu21].

Um den Durchsatz auf das theoretische Maximum anzuheben, könnte ein zweiter Sendepuffer eingesetzt werden, die dann kombiniert im Ping-Pong-Prinzip gefüllt werden. Alternativ kann versucht werden, mit Hilfe von *Inline-Assembler*-Programmierung den Programmcode so stark zu beschleunigen, dass dieser die erforderlichen 94,02 µs einhält. Dies würde allerdings zu einem wesentlich schlechter wartbaren Programmcode führen.

Da das Szenario einer unidirektionalen Kommunikation mit anhaltend maximaler Datenrate in einem realen Powerline-Kommunikationssystem in einem Bohrstrang extrem unwahrscheinlich ist, wurde auf die Durchsatzsteigerung in dieser Arbeit verzichtet.

Für die TCP-Messungen wurde die Fenstergröße (engl. *Window Size*) auf die Größe eines Paketes begrenzt, um das akkumulierte Senden von Paketen zu verhindern, die kleiner sind als die maximale Paketgröße (engl. *Maximum Transmission Unit*, MTU). Damit sendet das Protokoll erst ein neues Datenpaket aus, sobald das aktuelle erfolgreich am Empfänger angekommen ist (TCP-ACK). Damit soll die periodische Übertragung von Sensordaten simuliert und die dabei erreichte Durchsatzrate über die Paketgröße bestimmt werden. Aus dieser gemessenen Durchsatzrate und dem theoretischen Maximum lassen sich die Anzahl an Sensoren, die in einem Powerline-Netzwerk periodisch Daten versenden können, abschätzen.

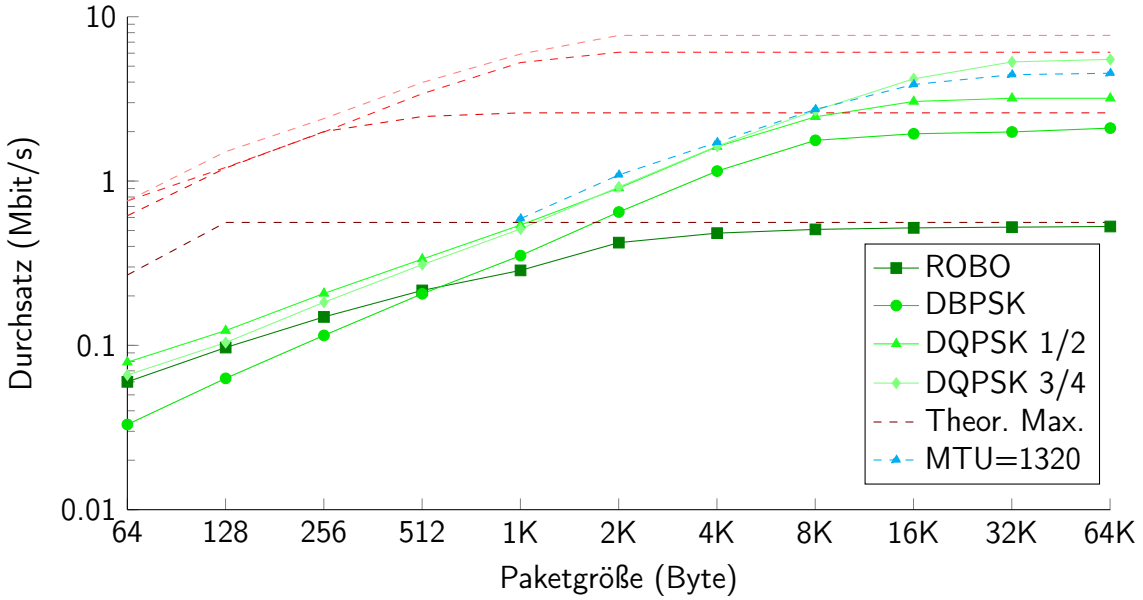

Abbildung 4.25: *Messung des TCP-Durchsatzes auf dem PXI-System für den ROBO-Modus und die drei verschiedenen Modulationsarten des Powerline-Systems. Die roten Kurven zeigen das jeweilige theoretische Maximum des Durchsatzes der MAC-Schicht für alle Messpunkte und Modulationsarten.*

Die Evaluation der TCP-Durchsatzmessung ist in Abbildung 4.25 dargestellt und besitzt nun eine logarithmische Ordinate zur besseren Differenzierung der Messkurven. Zu erkennen sind parallele exponentiell ansteigende Messkurven, die zu ihrem theoretischen Maximum hin abflachen. Dabei erreicht nur der ROBO-Modus für Pakete größer als 8 kByte das theoretische Maximum. Die DBPSK-Modulation erreicht bei maximaler Paketgröße nur 80 % der theoretisch maximalen Datenrate, was sich unter anderem durch den Mehraufwand des TCP-Protokolls erklären lässt. Bei den DQPSK-Modulationen überlagert sich der Mehraufwand mit dem oben beschriebenen Effekt der unzureichenden MAC-Sender-Geschwindigkeit konsekutiver Pakete und es werden nur 70–75 % der theoretisch maximalen Datenrate erreicht [Stu21].

Zu erkennen ist, dass sich die Datenraten für kleine Pakete bei allen Modulationsarten und dem ROBO-Modus sehr gering sind. Der ROBO-Modus hat bis zu einer Paketgröße von 256 Byte einen ähnlichen Durchsatz, wie die beiden DQPK-Modulationen und bis 512 Byte sogar einen höheren Durchsatz als die DBPSK-Modulation. Das heißt, dass für Paketgrößen kleiner als 1 kByte der ROBO-Modus bei Verwendung von TCP bevorzugt verwendet werden sollte, da dieser eine größere Robustheit gegenüber Störung aufweist. Einzig die in Abbildung 4.23 gezeigte Latenz für den ROBO-Modus ist hier zwei- bis dreimal so groß.

Oberhalb von 1 kByte nähert sich die Messkurve des ROBO-Modus asymptotisch dem theoretischen Maximum und die DBPSK- und DQPSK-Modulation steigen mit unveränderter Rate an. Dabei unterscheiden sich die beiden DQPSK-Modulationen erst ab einer Paketgröße von 8 kByte, da hier der Mehraufwand durch die nicht angepasste MTU die Datenrate senkt. Wie bei der UDP-Messung wurde für die DQPSK-Modulation ohne Punktierung die MTU auf 1320 reduziert und erneut vermessen. Auch hier zeigt sich eine deutliche Steigerung der Datenrate von 3,19 Mbit/s auf 4,53 Mbit/s für die maximale Paketgröße von 64 kByte und eine fast identische Datenrate zur DQPSK-Modulation mit Punktierung bis 32 kByte Paketgröße.

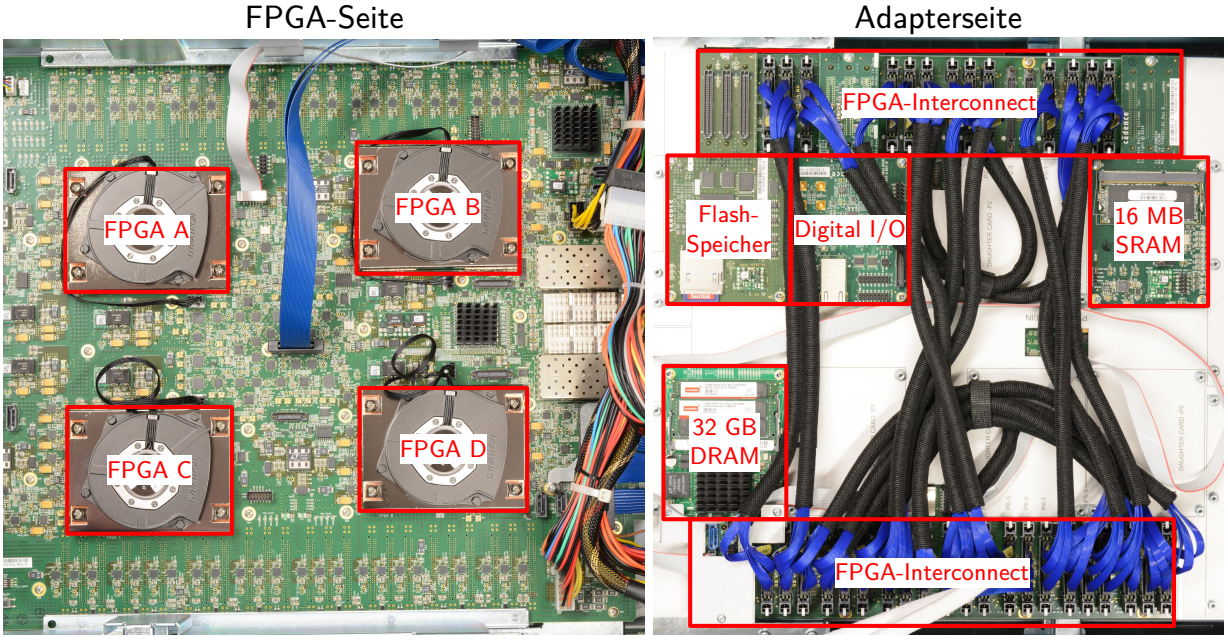

Abbildung 4.26: *Foto der beiden Platinenseiten des Protium-S1-Boards. Links die FPGA-Seite mit den vier FPGAs A – D und rechts die Adapterseite mit den aufgesteckten Erweiterungskarten und FPGA-Interconnects.*

Abschließend konnten für die Kommunikation mit TCP- und UDP-Kommunikation folgende Erkenntnisse gewonnen werden:

- Für die TCP-Kommunikation mit Paketgrößen kleiner als 1 kByte sollte immer der ROBO-Modus verwendet werden (außer die Pakete sind höchst latenzkritisch).
- Bei der Verwendung von UDP wird für Paketgrößen kleiner als 1 kByte bei allen Modulationsarten, außer DQPSK mit Punktierung, der theoretisch maximale Durchsatz erreicht.
- Für Paketgrößen ab 1 kByte kann sowohl mit TCP, als auch mit UDP eine hohe Datenrate erreicht werden und die Auswahl beschränkt sich im Wesentlichen auf den Abtausch zwischen der Sicherheit einer Verbindung und der Maximierung des Durchsatzes.
- Die MTU des Gesamtsystems sollte auf 1320 Byte angepasst werden, um den Durchsatz über alle Modulationsarten zu optimieren.

4.3.2. Evaluation der Skalierbarkeit auf dem Protium S1

Nach der Messung der maximalen Durchsatzraten und der Paketlatenzen bei bidirektionaler Kommunikation mit zwei HomePlug-1.0-Modems wird in diesem Abschnitt die Performance des Powerline-Netzes über eine variierende Anzahl an Modems evaluiert. Dabei soll der Paketverlust, die Latenz und die maximal Anzahl an Modems für verschiedene Netzkonfigurationen des Powerline-Netzes gemessen werden. Um diese Evaluation durchführen zu können, wird ein hochperformantes Emulationssystem benötigt, das möglichst viele HomePlug-1.0-Modems parallel emulieren kann.

Das in dieser Arbeit verwendete System ist die Cadence Protium-S1-Plattform, die in Abbildung 4.26 als Foto gezeigt ist. Diese vereint vier Xilinx Virtex UltraScale XU440 FPGAs auf einem Board

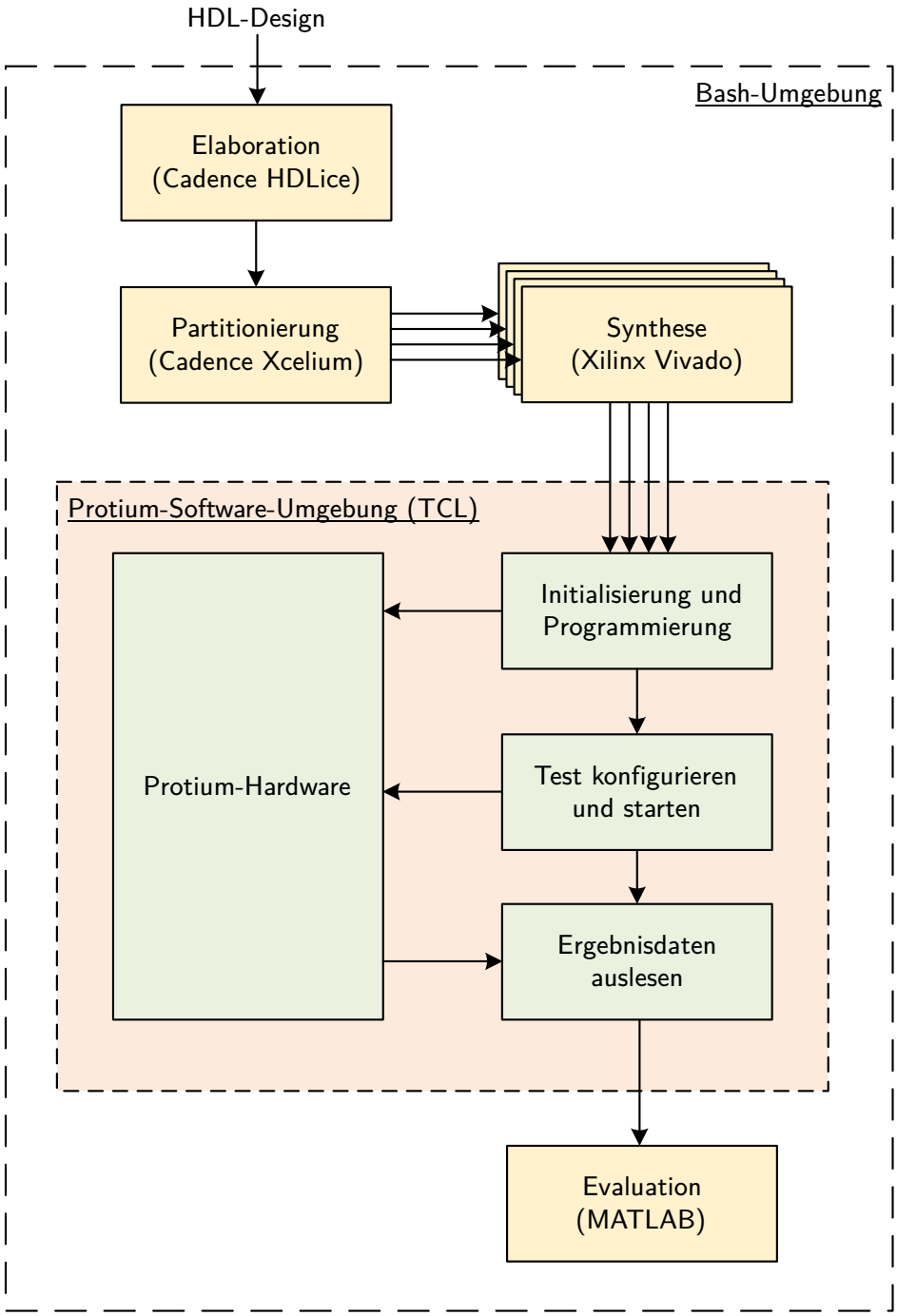

Abbildung 4.27: *Ablauf der Synthese, Ansteuerung und Evaluation für das Protium S1. Der gesamte Ablauf ist Konsolen-basiert und das Protium S1 selbst wird mit Hilfe von Tool-Command-Language-Skripten (TCL) gesteuert.*

(Abbildung 4.26 links) und hat damit insgesamt 22 Mio. Logikzellen, 44 MByte BlockRAM und 11400 DSPs. Zusätzlich lassen sich auf der anderen Board-Seite Erweiterungskarten einstecken (Abbildung 4.26 rechts), die das hier gezeigte System um Flash-Speicher, 128 MByte SRAM, 32 GByte DRAM und digitale Peripherie erweitern. Über die *FPGA-Interconnects* kann die Anzahl

der Verbindungsleitungen der FPGAs untereinander konfigurieren werden. Insgesamt ist damit das Protium-S1-System nicht nur größer und flexibler als fast alle kommerziell erhältlichen FPGA-Boards, es bietet auch eine Entwicklungsumgebung, die sich vollständig automatisieren lässt. Mit Hilfe von Konsolenbefehlen und der Programmiersprache TCL können Abläufe automatisiert und die Ansteuerung, das Debuggen oder die Evaluation des Protium-S1-Systems per Skript ausgeführt werden [Cad19].

Abbildung 4.27 zeigt den in dieser Arbeit verwendeten Ablauf vom HDL-Design über die Elaboration, die Portierung, die Synthese, die Ausführung auf dem Protium S1 bis zur Evaluation in MATLAB. Dabei ist bei jedem Arbeitsschritt (gelbe Blöcke) die Aufgabe und darunter in Klammern das verwendete Programm gezeigt. Zunächst wird das HDL-Design mit Hilfe von Cadence HDLice elaboriert, wobei die Syntax des Codes überprüft und Code-Beschreibungen auf Standardkomponenten, wie zum Beispiel Multiplexer oder Flip-Flips, abgebildet wird. Anschließend wird mit Hilfe von Cadence Xcelium eine Partitionierung des HDL-Designs für die vier FPGAs gesucht. Dabei verwendet Xcelium einen genetischen Algorithmus, der iterativ die Partitionierung mit höchster Taktfrequenz für die Emulation auf dem Protium-System ermittelt. Hierbei können bis zu 13 Iterationen durchgeführt werden. Das Ergebnis dieser Partitionierung sind vier Netzlisten, die anschließend in Xilinx Vivado parallel synthetisiert werden. Die aus der Synthese generierten *Bitstreams* werden dann der Protium-Software-Umgebung übergeben und zu Programmierung der FPGAs verwendet [Cad19].

Die Steuerung des Protium erfolgt mit Hilfe von TCL-Skripten, die zunächst das Board und alle dazugehörigen Hardware-Komponenten initialisieren und die einzelnen FPGAs programmieren. Anschließend kann die Protium-Hardware für den Test konfiguriert, die Emulation gestartet und abschließend die Ergebnisse ausgelesen werden. Die Ansteuerung der Hardware ist dabei eine direkte Zuweisung der *Top-Level*-Ports aus dem TCL-Skript. Die Ein- und Ausgabe der Testdateien erfolgt durch das Lesen und Schreiben von Speicherblöcken, welche ebenfalls direkt aus den TCL-Skripen adressierbar sind. Mit diesen Funktionen und der zusätzlichen Möglichkeit, Signale live aus dem Protium mitschneiden zu können, lassen sich Hardware-Systeme in Emulation echtzeitfähig verifizieren und zyklengenau testen [Cad19].

Das HDL-Design, das für diesen Test auf dem Protium ausgeführt wird, ist in Abbildung 4.28 gezeigt. Dieses besteht aus drei skalierbaren Teilen: einem Paketgenerator, N Powerline-Modems und einem Historgrammspeicher. Dabei ist N die ganzzahlige Anzahl der Station und hat mindestens die Größe zwei. Der Paketgenerator ist aus den TCL-Skripten steuerbar und erzeugt ein Paket mit gegebener Größe aus zufälligen Datenbytes von einem gegebenen Sender-Modem zu einem gegebenen Empfänger-Modem. Außerdem übernimmt der Paketgenerator die Ansteuerung des LLC-Interfaces und die Ablaufsteuerung für jede Station. Jedes Powerline-Modem besitzt einen eigenen *Timer*, der die Latenz einer Paketübertragung von der Übergabe an die Station bis zum Erhalt einer Antwort über das LLC-Interface bestimmt. Dabei wird der Timer vom Paketgenerator gestartet und gestoppt, wenn dieser eine LLC-Antwort erhält. Die gemessene Zeit wird von jeder Station als Latenz in Taktzyklen in einem Histogrammspeicher abgelegt. Dieser Speicher kann nach Beendigung des Testdurchlaufes ausgelesen und die mittlere Latenz und die Paketverlustrate bestimmt werden. Diese Auswertung erfolgt, wie in Abbildung 4.27 gezeigt, Skript-basiert in MATLAB.

Zuerst wurde die maximale Anzahl an Stationen auf dem Protium-S1-System bestimmt. Dafür wurden sukzessive Synthesen mit steigender Anzahl an Powerline-Stationen durchgeführt, bis das Maximum mit $N = 60$ Stationen erreicht wurde. Bei 60 Stationen ist das gesamte System

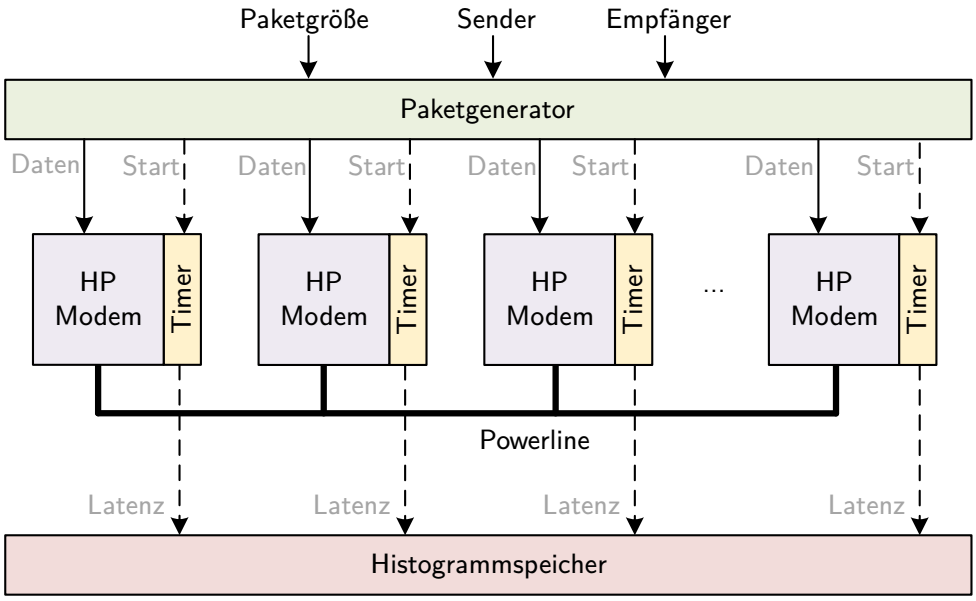

Abbildung 4.28: *Aufbau des Testsystems bestehend aus N Powerline-Stationen, einem Paketgenerator und einem Histogrammspeicher.*

Tabelle 4.7: *Verwendete FPGA-Ressourcen für 60 Powerline-Modems auf den vier XU440-FPGAs des Protium-S1-Systems.*

	FPGA A	FPGA B	FPGA C	FPGA D
LUTs	2.235.298 (89 %)	2.394.605 (95 %)	2.390.605 (95 %)	2.265.049 (90 %)
Flip-Flops	1.529.398 (31 %)	1.623.779 (33 %)	1.615.814 (32 %)	1.517.905 (30 %)
BRAM 36	255 (26 %)	255 (27 %)	272 (28 %)	238 (25 %)
BRAM 18	612 (26 %)	664 (27 %)	662 (28 %)	582 (25 %)
max. Freq.	64 MHz	68 MHz	68 MHz	57 MHz

zu 88 % ausgelastet und verbraucht die in Tabelle 4.7 aufgeführten FPGA-Ressourcen. Unter anderem sind die Logiktabellen (engl. *Look-Up Table*, LUT) zu 95 % belegt und das gesamte System lässt sich mit 57 MHz Systemtakt betreiben. Die Belegung der Flip-Flops und des Block-RAM (BRAM) sind mit 33 % und 28 % noch nicht maximal ausgelastet. Die Emulation des Testdesigns erfolgt dabei in mit einem Takt von 3,36 MHz, was um einen Faktor von 17 geringer ist als der Systemtakt. Damit hat das Protium 16 Takte pro Design-Takt mehr zur Verfügung, um zyklengenau Trigger-Signale auszuwerten, Design-Pins zu lesen oder schreiben oder Signalverläufe aufzuzeichnen.

Für die Evaluation der Latenz und des Paketverlustes über die Anzahl der Stationen im Netzwerk, wurde eine Busauslastung gewählt, die 50 bis 100 % der Kanalkapazität der jeweiligen Modulationsart entspricht. Tabelle 4.8 zeigt die Anzahl der 1500 Byte großen Pakete, die in einer Sekunde gesendet werden müssen, um die jeweilige Auslastung bei gegebener Modulationsart zu erreichen. Die maximal erreichbare Datenrate für eine gegebene Modulationsart, wurde dabei aus den Ergebnissen in Abbildung 4.24 extrapoliert. Um die Lastverteilung auf dem Kanal realistisch

Tabelle 4.8: *Anzahl der 1500 Byte großen Pakete, die pro Sekunde versendet werden, um die jeweilige prozentuale Busauslastung zu erreichen. Die Werte der maximalen Datenraten sind für eine Paketgröße von 1500 Byte aus Abbildung 4.24 extrapoliert.*

Modulationsart	max. Datenrate	Anzahl Pakete für Busauslastung von					
		50 %	60 %	70 %	80 %	90 %	100 %
ROBO	0,56 Mbit/s	24	29	34	39	44	49
DBPSK	2,50 Mbit/s	109	131	152	174	196	218
DQPSK 1/2	4,30 Mbit/s	187	225	262	300	360	375
DQPSK 3/4	5,17 Mbit/s	225	270	315	360	406	451

simulieren zu können, wurde eine zufällige Verteilung der Pakete innerhalb des Intervalls von einer Sekunde gewählt. Dafür wurde jedem zu sendenden Paket eine Zufallszahl zwischen 0 ms und 999 ms zugewiesen, die den Sendezeitpunkt des Paketes festlegt.

Der Sender eines jedes Paketes wurde ebenfalls zufällig gewählt und der Empfänger ist immer die Station mit der konsekutiven Stationsnummer. Station 1 sendet zum Beispiel an 2 oder Station 60 an 1. Der Grund für diese feste Verteilung ist die Aushandlung der *ToneMap*, die bei mehr als 16 Stationen nicht alle parallel gespeichert werden können. Wird eine siebzehnte *ToneMap* ausgehandelt, wird eine zufällige andere *ToneMap* verworfen. Um diesen, mit steigender Anzahl an Stationen zunehmenden Überhang an Kommunikation zu vermeiden, sind die Empfänger festgelegt und eine *ToneMap* wird vor Beginn einer jeden Messreihe ausgehandelt.

Jede in Tabelle 4.8 gezeigte Kombination aus Auslastung, Modulationsart und Anzahl Stationen wurde für $t_{\text{meas}} = 10\,\text{s}$ Kommunikationszeit auf dem Protium-S1-System evaluiert. Um die Anzahl der Stationen, die an der Kommunikation teilnehmen, künstlich zu reduzieren, wurden immer nur die Instruktionsspeicher der teilnehmenden Modems programmiert. Ohne die MAC- und PHY-Prozessoren werden weder Pakete gesendet noch empfangen. Die benötigte Messdauer auf dem Protium-S1-System ist dabei um einen Faktor $K_{\text{Protium}} = 3{,}35\,\text{MHz}/50\,\text{MHz} \approx 15$ größer als t_{meas}, da das Protium nur mit 3,35 MHz anstatt mit dem echtzeitfähigen Systemtakt von 50 MHz läuft. Die gesamte Ausführungszeit der in Tabelle 4.8 gezeigten Konfigurationen beträgt:

$$\begin{aligned} t_{\text{sim}} &= t_{\text{meas}}\, K_{\text{Protium}}\, N_{\text{mod}}\, N_{\text{util}}\, N_{\text{Station}} \\ &= 10\,\text{s} \cdot 15 \cdot 4 \cdot 6 \cdot 59 = 59\,\text{h} \end{aligned}$$

Dabei ist $N_{\text{mod}} = 4$ die Anzahl der verschiedenen Modulationsarten, $N_{\text{util}} = 6$ die Anzahl der evaluierten Auslastungen und $N_{\text{Station}} = 59$ die Anzahl der Netzwerkkonfigurationen von 2 bis 60 Stationen. Damit ist die Emulation auf dem Protium S1 zwar um den Faktor 15 langsamer als Echtzeit, aber auch um etwa einen Faktor von 3444 schneller als eine rein Software-basierte Simulation, die 203.222 Stunden ≈ 23 Jahre laufen würde.

Die Ergebnisse der Emulation auf dem Protium S1 sind in den Abbildungen 4.29 und 4.30 dargestellt. Diese zeigen die Latenz (in rot) und den Paketverlust (in grün) über die Busauslastung des Systems in Prozent und die Anzahl der Stationen. In Abbildung 4.29 ist die Modulation mit DBPSK (oben) und der ROBO-Modus (unten) gezeigt. Zu erkennen ist, dass bei der DBPSK-Modulation die Latenz mit der Auslastung des Powerline-Busses und der Anzahl der Stationen

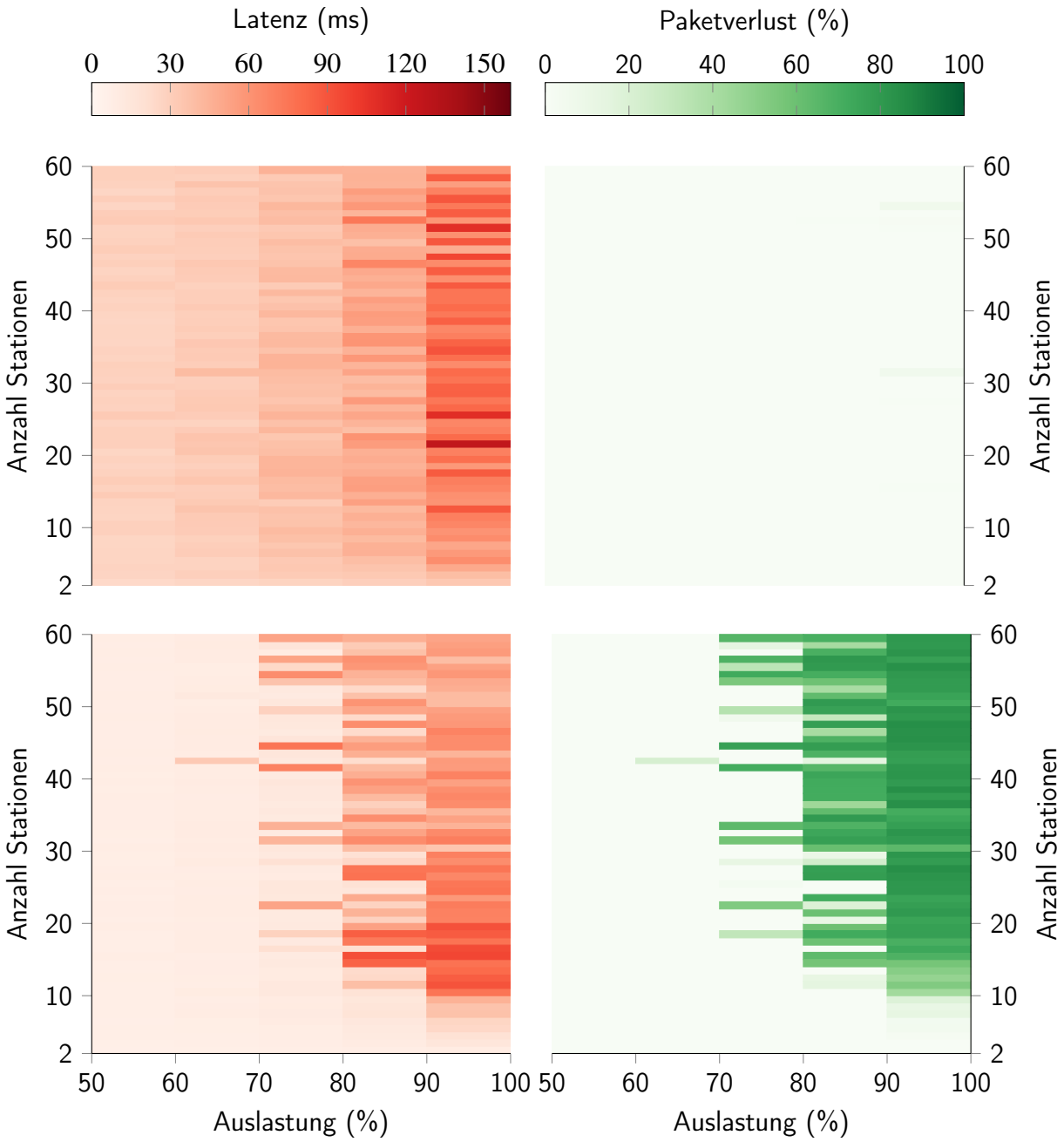

Abbildung 4.29: *Latenz (links) und Paketverlust (rechts) über die Busauslastung und Anzahl Stationen bei DBPSK-Modulation der Pakete (oben) und im ROBO-Modus (unten).*

zunimmt. Außerdem ist eine Korrelation zwischen Latenz und Paketverlust zu erkennen, wobei sich die Latenz erhöht, wenn die Anzahl Kollisionen im System steigt und wieder abfällt, wenn der Paketverlust so groß ist, dass kaum noch Pakete durch das Netzwerk kommen. Allgemein lassen sich drei Bereiche für die Evaluation dieser Kommunikation definieren:

Bereich 1: Kommunikation ohne signifikante Paketverluste und geringer Latenz.

Bereich 2: Kommunikation mit niedrigen Paketverlusten und hohen Latenzen.

Bereich 3: Kommunikation mit hoher Paketverlustrate und Verbindungsabbrüchen.

Für die DBPSK-Modulation (Abbildung 4.29 oben) liegt bei Auslastungen von 50 % und 60 % für alle Anzahlen an Stationen die Latenz zwischen 5,6 ms und 8 ms und es gibt keine signifikanten Paketverluste (Bereich 1). Ab 70 % Auslastung ist die oben beschriebene Korrelation zwischen Latenz und Paketverlust erkennbar, wobei sich die Latenz erhöht, wenn die Anzahl Kollisionen im System steigt (Bereich 2). Bis einschließlich 20 Stationen gibt es keine signifikanten Paketverluste und nur wenige Kollisionen bei geringer Latenz (Bereich 1). Ab 21 Stationen ist die Latenz und Paketverlustrate relativ zufällig über die Anzahl Stationen verteilt und es ergibt sich eine Mischung aus Bereich 1 und 2 bis 60 Stationen. Zwischen 80 % und 90 % liegt der Bereich 1 unterhalb von 10 Stationen und zwischen 90 % bis 100 % Auslastung nur noch bei 6 Stationen. Ab diesen Anzahlen an Stationen steigt zuerst die Latenz auf Grund der steigenden Anzahl an Kollisionen und den damit verbundenen wiederholten Paketübertragungen, die die Busauslastung weiter erhöhen (Bereich 2). Ab ungefähr 20 Stationen und mehr als 80 % Auslastung oder 10 Stationen und mehr als 90 % Auslastung ist zu erkennen, dass die Latenz wieder abnimmt, die Paketverlustrate aber konstant hoch bleibt oder sogar weiter steigt (Bereich 3). Der Grund dafür ist die Limitierung der Wiederholungen auf maximal 16 Versuche pro Paketübertragung, nach denen ein Paket verworfen wird und somit nicht in der Latenzstatistik erscheint.

Im ROBO-Modus, in Abbildung 4.29 unten, ist zu erkennen, dass die Latenz generell höher ist als bei den anderen Modulationsarten, wie auch schon in Abschnitt 4.3.1 gezeigt. Diese Latenz ist über alle Anzahlen an Station gleich verteilt und steigt mit der Auslastung des Busses an. Die Paketverlustrate ist dabei aber konstant niedrig oder sogar null und erreicht im Maximum nur 9 % bei 100 % Busauslastung und 43 Stationen im Netzwerk. Diese geringe Paketverlustrate resultiert vor allem aus der geringen Paketanzahl, die zum Erreichen der maximalen Busauslastung notwendig ist. Selbst bei 100 % Busauslastung werden pro Sekunde nur 49 Pakete versendet (siehe Tabelle 4.8) und damit ist die Chance, dass zwei Pakete kollidieren, sehr gering. Da jedes Paket im ROBO-Modus aus 12 Segmenten besteht, ist der Bus trotzdem voll ausgelastet. Insgesamt befindet sich das Netzwerk für Busauslastungen zwischen 50 % und 80 % nur im Bereich 1, für Auslastungen von 80 % und mehr ab 3 Stationen im Bereich 2 und für die hier gemessenen Parameter niemals in Bereich 3.

Die in Abbildung 4.30 gezeigte Latenz und Paketverlustrate für DQPSK-Modulation ohne Punktierung (oben) zeigt, verglichen mit DBPSK-Modulation, einen früheren Anstieg beider Parameter. Selbst bei 50 % Auslastung gibt es bereits Paketverluste und Latenzanstiege, die vermutlich auf die ungünstige Segmentgröße von 1452 Byte zurückzuführen sind (siehe Abschnitt 4.3.1). Bei der DBPSK-Modulation werden für jedes Paket drei und bei DQPSK-Modulation ohne Punktierung zwei Segmente benötigt. Um die gleiche Busauslastung beider Modulationsarten bei 1500 Byte großen Paketen zu erreichen, werden für die DQPSK ohne Punktierung allerdings 70 % mehr Pakete benötigt, was die effektive Auslastung auf Segmentebene um etwa 15 % erhöht. Würde man die Anzahl Pakete reduzieren oder die Paketgröße auf die in Abschnitt 4.3.1 vorgeschlagenen 1320 Byte reduzieren, sollten sich die Latenz und Paketverlustrate deutlich reduzieren und unterhalb der DBPSK-Modulation befinden. Aus Gründen der Vergleichbarkeit aller Modulationsarten und der Generalisierung für nicht-angepasste Powerline-Systeme wurde dies in dieser Arbeit nicht weiter untersucht. Mit der hier verwendeten Konfiguration lässt sich bei einer Auslastung von 50 % und 60 % das Netzwerk mit 24 und 16 Stationen im Bereich 1 und ansonsten im Bereich 2 betreiben. Die Latenz für den Bereich 1 liegt zwischen 3 ms und 5,5 ms. Ab 70 % Auslastung können noch 11 Stationen ohne Paketverlust (Bereich 1), 20 Stationen mit wenigen Paketverlusten (Bereich 2) und danach nur mit hohen Paketverlustraten (Bereich

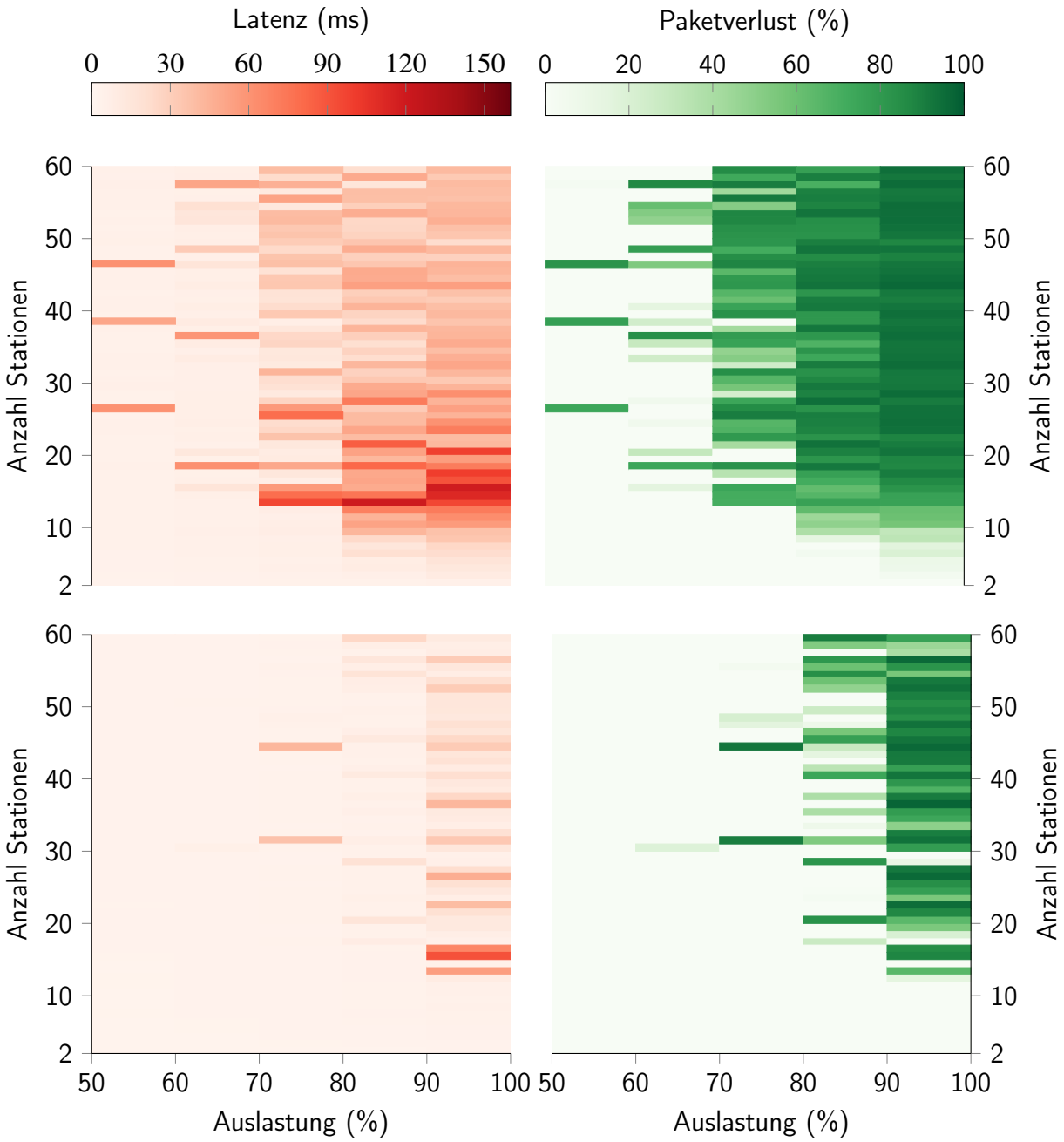

Abbildung 4.30: *Latenz (links) und Paketverlust (rechts) über die Busauslastung und Anzahl Stationen bei DQPSK-Modulation der Pakete ohne Punktierung (oben) und mit Punktierung (unten).*

3) betrieben werden. Für Busauslastungen ab 80 % sind die Anzahl Stationen der Bereiche nochmals um einen Faktor 2 oder mehr geringer.

Unter Verwendung von DQPSK mit Punktierung (siehe Abbildung 4.30 unten) können die geringsten Latenzen und Paketverlustraten erreicht werden. Für 50 % und 60 % sind die Latenzen mit 1,9 ms bis 2,5 ms am geringsten und es treten (fast) keine Paketkollisionen auf (Bereich 1). Erst zwischen 70 % und 80 % Auslastung gibt es erhöhte Latenzen und Paketverluste. Bei 70 % Auslastung kann ein Netzwerk mit 29 Stationen verlustfrei (Bereich 1) und ab da mit wenigen Verlusten (Bereich 2) betrieben werden. Bei 80 % Auslastung geht der Bereich 1 bis

Tabelle 4.9: *Anzahl der Stationen, die bei der jeweils gewählten Modulationsart und Busauslastung im Bereich ohne Paketverluste (Bereich 1) oder mit niedrigen Paketverlusten (Bereich 2) betrieben werden können.*

Modulationsart	Auslastung	Bereich 1		Bereich 2	
		#Stationen	Latenz	#Stationen	Latenz
ROBO	50–80 %	60	32,0 ms	60	32,0 ms
	80–100 %	3	36,0 ms	60	60,1 ms
DBPSK	50–70 %	60	7,6 ms	60	9,5 ms
	70–80 %	20	11,3 ms	60	22,6 ms
	80–90 %	10	12,8 ms	20	51,6 ms
	90–100 %	6	18,2 ms	10	56,2 ms
DQPSK 1/2	50–60 %	24	3,9 ms	60	9,0 ms
	60–70 %	16	5,4 ms	60	11,1 ms
	70–80 %	11	5,5 ms	20	36,0 ms
	80–90 %	4	6,1 ms	10	26,6 ms
	90–100 %	3	7,0 ms	8	21,7 ms
DQPSK 3/4	50–70 %	60	2,6 ms	60	2,6 ms
	70–80 %	29	2,9 ms	60	5,6 ms
	80–90 %	15	3,1 ms	50	5,5 ms
	90–100 %	8	3,6 ms	12	5,7 ms

15 Stationen und der Bereich 2 bis 50 Stationen im Netzwerk. Erst bei 100 % Auslastung ist der Anstieg und das anschließende Abfallen der Latenz wie bei den anderen Modulationsarten zu erkennen. Hier kann das System mit acht Stationen im Bereich 1, mit zwölf Stationen im Bereich 2 und danach nur noch mit hohen Paketverlustraten im Bereich 3 betrieben werden.

Tabelle 4.9 fasst die Latenzen und maximale Anzahl an Stationen für die Bereiche 1 und 2 über alle Modulationsarten und Auslastungen aus dem oben beschriebenen Abschnitt zusammen. Mit Hilfe dieser Ergebnisse kann ein Powerline-Kommunikationsnetz konzipiert und der Durchsatz, die Latenz und die Fehlerraten im System abgeschätzt werden. Wichtige Parameter für die Auswahl einer Konfiguration sind die Fehlertoleranz der kommunizierenden Systeme, die benötigte Datendurchsatzrate und die erwarteten Eigenschaften des Übertragungskanals, die die Modulationsart maßgeblich bestimmen.

Abschließend lassen sich folgende Skalierungseigenschaft der einzelnen Modulationsarten zusammenfassen:

- Die Latenz des ROBO-Modus ist am höchsten von allen Modulationsarten und skaliert linear mit der Busauslastung im System. Für bis zu 60 Stationen und 100 % Busauslastung wurde keine Paketverlustrate größer als 9 % gemessen.

- Für DBPSK wurden bis 70 % Busauslastung keine signifikanten Paketverluste gemessen und die durchschnittliche Latenz liegt bei 7,6 ms. Ab 70 % Busauslastung verkleinert sich der Bereich verlustfreier Kommunikation auf 20 Stationen und nimmt mit steigender Busauslastung linear ab.
- Die DQPSK-basierte Kommunikation ohne Punktierung hat die schlechteste Gesamtperformance. Bereits ab 50 % Auslastung ist eine fehlerfreie Kommunikation mit nur noch 24 Stationen möglich und diese Anzahl sinkt linear auf nur drei Stationen bei 100 % Busauslastung ab.
- Am besten skaliert die DQPSK-Modulation mit Punktierung mit einer durchschnittlichen Latenz von maximal 3,6 ms im Bereich 1. Ab einer Auslastung von 70 % kann mit immer noch mit 29 Stationen verlustfrei kommuniziert werden und bis 90 % sogar noch mit 50 Stationen bei niedrigen Paketverlusten. Wie auch bei den anderen Modulationsarten sinkt die Paketverlustrate bis 100 % Auslastung linear auf acht Stationen für den fehlerfreien Bereich 1 oder 12 Stationen mit niedriger Fehlerrate im Bereich 2.

4.4. Zusammenfassung

Das Ziel dieses Kapitels war die Entwurfsraumuntersuchung und Evaluation eines HomePlug-1.0-basierten Powerline-Modems auf verschiedenen Hardware-Plattformen. Dieses unterteilt sich in PHY- und MAC-Schicht, die zunächst getrennt implementiert und optimiert wurden.

Als erstes Design wurde der Algorithmus als C-Implementierung auf den CV32E40P-RISC-V portiert und evaluiert. Eine initiale Messung über die gesamte PULP-Plattform zeigte einen Durchsatz von 4,26 kbit/s bei einer Fläche von 10,33 mm^2 in der XFAB XT018-ASIC-Technologie. Durch Optimierung des Programmcodes und Integration eines Viterbi-Decoders und Galois-Feld-MAC-Einheit konnte der Durchsatz auf 45,33 kbit/s gesteigert werden, mit einer Gesamtfläche von 11,31 mm^2. Eine weitere Durchsatzsteigerung wäre mit großem Hardware-Aufwand verbunden, weshalb entschieden wurde, die Implementierung auf einer ASIP-Plattform weiter zu untersuchen.

Als ASIP-Plattform wurden verschiedene Cadence Tensilica LX7-Prozessoren gewählt, deren Leistungsfähigkeit von einem kleinen leistungsarmen Hörgeräteprozessor (Hifi3) bis zum großen Vektorprozessor für Videosignalverarbeitung (Fusion G) reichen. Nach der Portierung des RISC-V-optimierten Programmcodes auf die einzelnen Prozessoren wurde eine erste Abschätzung des Durchsatzes durchgeführt. Diese ergab minimal 10,92 kbit/s auf dem kleinsten Hifi3-Prozessor und maximal 17,02 kbit/s auf dem Fusion-G3-Vektorprozessor. Zur Steigerung des Durchsatzes wurden auch die beiden aufwändigsten Programmteile Viterbi-Decoder und Galois-Feld-MAC als Hardware-Komponenten umgesetzt und integriert. Damit konnte der Durchsatz auf minimal 180,85 kbit/s für den Hifi3-Prozessor und maximal 244,45 kbit/s für den G3-Vektorprozessor gesteigert werden. Das entspricht einer durchschnittlichen Steigerung um den Faktor 16 gegenüber der rein Software-basierten Implementierung und einem Faktor 5 gegenüber der RISC-V-Implementierung mit Hardware-Erweiterungen. Der Fusion G3-Prozessor bietet zwar den höchsten Durchsatz, ist aber mit einem Flächenbedarf von 16,89 mm^2 um den Faktor 1,4 größer als der Hifi3z mit lokalem Speicher, der einen ähnlichen Durchsatz von 243,92 kbit/s erreicht. Betrachtet man das Produkt aus Fläche, Verlustleistung und Durchsatz, hat dieser Hifi3z die beste Performance und wird für die Implementierung eines niederdatenratigen Powerline-Systems empfohlen.

Um die im Standard spezifizierte Datenrate von 14 Mbit/s zu erreichen, wurden alle Komponenten der PHY-Schicht als dedizierte Hardware umgesetzt. Alle Bestandteile des Encoders und des Decoders sind als Streaming-basierte Komponenten implementiert worden, um das Speichern großer Datenpakete zu vermeiden. Insgesamt erreicht diese Implementierung eine Datenrate von 14,48 Mbit/s bei einer Fläche von 8,13 mm^2 und ist damit kleiner und schneller als die PULP-Plattform und alle LX7-Prozessoren. Eine Untersuchung der Sende- und Empfangslatenzen zeigt eine zu hohe Verarbeitungszeit im Decoder von 28,06 µs, die durch Pipelining und parallele Decodierung auf 18,72 µs reduziert werden konnte. Damit kann die dedizierte PHY-Schicht-Implementierung für eine standardkonforme echtzeitfähige Powerline-Kommunikation ohne Einschränkungen eingesetzt werden.

Die MAC-Schicht wurde direkt als dedizierte Hardware-Architektur entworfen und umgesetzt, um echtzeitfähig zu kommunizieren. Nach der Verifikation der Implementierung wurde eine ASIC-Synthese in der XT018-Technologie durchgeführt, die eine Fläche von 6,40 mm^2 und einen maximalen Durchsatz von 10 Mbit/s ergibt. Eine Analyse der einzelnen Komponenten hat gezeigt, dass die Zustandsdiagramm-basierten Teile des Algorithmus sehr hohe Datenraten verarbeiten können, die vom HomePlug-1.0-Standard nicht benötigt werden. Außerdem wird etwa 83 % der Fläche durch Speicher belegt, die zum Speichern der Zwischenergebnisse dienen. Um den Flächenbedarf zu reduzieren und Speicher einzusparen, wurden zwei Prozessoren für die Umsetzung der Zustandsdiagramme eingesetzt und die restliche Implementierung als Streaming-Architektur aufgebaut. Damit konnte die Fläche um 43 % der Fläche und 59 % der geschätzten Verlustleistung eingespart werden. Zusätzlich hat diese Implementierung den Vorteil, dass Programmcode zur Laufzeit geändert oder angepasst werden kann.

Unter Verwendung der dedizierten PHY-Schicht- und Prozessor-basierten MAC-Schicht-Implementierung wurde ein gesamtes HomePlug-1.0-Powerline-Modem auf ein Xilinx Kintex-7-FPGA portiert und auf dem National Instruments PXIe-System echtzeitfähig getestet. Dabei wurden zwei proprietäre Maxim MAX2982-Evalboards mit je einem FPGA verschaltet, um auf Applikationsebene den Durchsatz und die Latenz des Systems zu bestimmen. Der ROBO-Modus hat von allen Modulationsarten die größte Latenz, was vor allem an hohen Redundanzen und der damit verbunden, geringen Segmentgröße liegt. DBPSK und DQPSK haben für Pakete bis 512 Byte eine geringe Latenz von maximal 2,2 ms, die danach monoton mit der Paketgröße ansteigt und nur noch von Segmentierung und Fragmentierung abhängt.

Die Messung der Durchsatzraten aller Konfigurationen zeigte, dass der ROBO-Modus, die DBPSK-Modulation und die DQPSK-Modulation ohne Punktierung schon ab kleinen UDP-Paketgrößen das theoretische Maximum erreichen. Für die DQPSK-Modulation ohne Punktierung wurde gezeigt, dass eine Reduzierung der MTU auf 1320 eine Steigerung der Datenrate um 25 % bringt. Die DQPSK-Modulation verfehlt das theoretische Maximum auf Grund zu hoher Paketverarbeitungslatenzen im Sender um 13 %, erreicht aber mit 6,71 Mbit/s die höchste gemessene Datendurchsatzrate des Powerline-Systems. Für die TCP-Kommunikation konnte gezeigt werden, dass für sporadische Kommunikation mit Paketgrößen kleiner als 1 kByte der ROBO-Modus am besten geeignet ist. Dieser bietet dabei die robusteste Paketübertragung und eine ebenso hohe Durchsatzrate, wie die DQPSK-Modulation ohne Punktierung.

Abschließend wurde die Skalierbarkeit eines Powerline-Netzwerkes auf dem Protium-S1-Emulationssystem evaluiert. Dazu wurden 60 Powerline-Modems mit Hilfe eines parametierbaren HDL-Designs für die vier FPGAs des Protium-Systems synthetisiert und vollständig automatisiert evaluiert. Die Messungen der Latenz und der Paketverlustrate haben gezeigt, dass der

ROBO-Modus mit 60 Stationen bis zu einer Busauslastung von 100 % verlustfrei kommunizieren kann, dabei aber hohe Latenzen von mindestens 25 ms aufweist. Die beste Gesamtperformance hat die DQPSK-Modulation mit Punktierung bei einer mittleren Latenz von nur 2,5 ms, einer verlustfreien Übertragung mit 60 Stationen bis 70 % Busauslatung und bis 90 % bei sporadischen Paketverlusten.

Zusammenfassend wurde in diesem Kapitel eine systematische Entwurfsraumexploration für das Powerline-Gesamtsystem gezeigt und ein echtzeitfähig kommunizierendes Powerline-Modem entworfen und evaluiert. Dieses erreicht einen hohen Durchsatz bei kleinen Paketen und dessen geringe Latenzen eignet sich gut für den Einsatz in Sensor-Aktor-basierten Powerline-Systemen. Zusätzlich skaliert das System mit bis zu 60 Kommunikationsknoten linear in der Latenz und lässt sich dabei mit geringen Paketverlusten durch Kollisionen betreiben. Dabei bietet die TCP-basierte Kommunikation im ROBO-Modus eine verlässliche Ende-zu-Ende-Kommunikation mit hoher Robustheit gegenüber Störungen, wie sie für den Einsatz in der Tiefbohrtechnik mit vielen Kommunikationsteilnehmer erforderlich ist.

KAPITEL 5

Integration der Transport- und Vermittlungsschicht

Nachdem im vorherigen Kapitel die echtzeitfähige Kommunikation des Powerline-Systems evaluiert wurde, wird in diesem Kapitel ein System zur TCP/IP-basierten Steuerung des Powerline-Systems implementiert. Dieses System besteht aus den OSI-Schichten 3 (Transport) und 4 (Vermittlung), wie sie in Abbildung 3.1 in Kapitel 3 vorgestellt wurden. Die Erweiterung um dieses System ermöglicht es Anwendungen eine zuverlässige Ende-zu-Ende-Kommunikation auf Basis von IP-Adressen aufzubauen. Die darüber liegenden Schichten 5 bis 7 werden üblicherweise von einem Betriebssystem ausgeführt und sind applikationsspezifisch. Daher sind diese Schichten nicht Teil wieder Arbeit. Um ein verlässliches Ende-zu-Ende-Hardware-Kommunikationssystem für den Betrieb unter schwierigen Umweltbedingungen bereitstellen zu können, wird eine hochtemperaturfeste Hardwareplattform benötigt. Diese Plattform muss die auf der Transport- und Vermittlungsschicht ausgeführten Protokolle mit mindestens 10 Mbit/s echtzeitfähig ausführen und dabei möglichst klein und verlustleistungsarm sein.

Die Protokolle, die in dieser Arbeit auf der Vermittlungsschicht ausgeführt werden sollen sind:

- ARP: *Address Resolution Protocol*
- ICMP: *Internet Control Message Protocol*
- IPv4: *Internet Protocol* Version 4
- IPv6: *Internet Protocol* Version 6

Auf der Transportschicht werden die folgenden Protokolle benötigt:

- TCP: *Transmission Control Protocol*
- UDP: *User Datagram Protocol*

Alle diese Protokolle sind kontrollflussorientiert, mit vielen und teils komplexen Einzelfunktionen, wie Routing, Verbindungsaufbau oder Zuverlässigkeit der Datenübertragung [Fal11]. Wie schon in Abschnitt 4.2 gezeigt, ist eine dedizierte Hardwareimplementierung eines Zustandsdiagramm-basierten Protokolls nicht optimal hinsichtlich Leistungsaufnahme und Fläche. Daher wird die

Tabelle 5.1: *Vergleich der Hardware-Eigenschaften der ARM Cortex-M-Prozessoren M0 bis M4. Im unteren Teil der Tabelle sind Syntheseergebnisse der Prozessoren in einer proprietären 180 nm-ARM-Technologie gegeben.*

Referenz	M0+/M0 [Lim09a]/[Lim09b]	M1 [Lim07b]	M3 [Lim10]	M4 [Lim09c]
CPU-Architektur	von-Neumann	von-Neumann	Harvard	Harvard
Pipeline-Stufen	2/3	3	3	3
Hw-Mult./Hw-Div.	Ja/Nein	Ja/Nein	Ja/Ja	Ja/Ja
MAC-Einheit	Nein	Nein	3 Takte	1 Takt
Gleitkomma	Nein	Nein	Nein	Einfach
SIMD	Nein	Nein	Nein	NEON(4x8 bit)
Technologie [nm]	180	–*	180	180
Fläche [mm^2]	0.098/0.11	–*	0.35	0.44
Leistung [µW/MHz]	47/66	–*	141	151

*Der Cortex-M1 ist ein Softcore-Prozessor für FPGA-Boards.

Implementierung auf möglichst kleine und verlustleistungsarme Prozessoren beschränkt, die sich durch Hardware-Erweiterungen beschleunigen lassen.

Nachfolgend werden in Abschnitt 5.1 die untersuchten ARM Cortex-M-Prozessoren und die Evaluationsplattform vorgestellt. In Abschnitt 5.2 werden die beiden verwendeten TCP/IP-Stacks Leightweight IP und Micro IP beschrieben und erste Software-Verbesserungen vorgestellt und ausgewertet. Anschließend werden in Abschnitt 5.3 Hardware-Erweiterung der Cortex-M-Prozessoren vorgestellt und deren Einfluss auf die Gesamtperformance evaluiert. Abschließend werden in Abschnitt 5.4 die Synthesen in der XFAB XT018-Technologie durchgeführt und die Ergebnisse in einem Hardware-Entwurfsraum ausgewertet.

5.1. ARM Cortex-M-Prozessoren

Für die oben gegebenen Randbedingungen sind die Prozessoren der ARM Cortex-M-Serie gut geeignet, da diese für Mikrocontroller-basierte Anwendungen entworfen wurden und sich durch Parameter Hardware-seitig an die Applikation anpassen lassen. Alle Prozessoren dieser Serie haben einen 32 bit-Instruktionssatz, besitzen keine Caches und lassen sich mit Hilfe der Programmiersprache C programmieren.

Eine Auswahl der kleinsten Prozessoren und mit ihren Hardware-Eigenschaften ist in Tabelle 5.1 dargestellt. Diese fünf Prozessoren der Cortex-M-Serie sind: M0(+), M1, M3 und M4. Die Cortex-M-Prozessoren M7, M23, M33, M35P, M55 und M85 sind nicht Teil der Auswahl, da diese durch die Verwendung von Caches, DSP-Coprozessoren oder doppelte Gleitkommazahlendarstellung überdimensioniert für die Ausführung von Zustandsdiagrammen sind. Der M0- und der M0+-Prozessor sind die kleinsten Prozessoren der Reihe, wobei der M0+ eine Hardware-seitig optimierte

Variante des M0 mit zwei statt drei Pipeline-Stufen ist. Der M1-Prozessor ist eine FPGA-optimierte Version des M0-Prozessors, der speziell als Softcore für die Ausführung auf FPGAs portiert wurde. Daher gibt es in der Tabelle 5.1 auch keine Angaben über dessen Größe als ASIC.

Die M0(+)- und M1-Prozessoren besitzen eine von-Neumann-Architektur und haben weder einen Dividierer noch eine MAC-Einheit. Die M3- und M4-Prozessoren haben eine Harvard-Architektur und unterscheiden sich im Wesentlichen in der Latenz der MAC-Einheit, der Verwendung von Gleitkommazahlen und der Ausführbarkeit von SIMD-Befehlen. Zusätzlich sind die Größe und Leistungsaufnahme existierender Chips in proprietärer ARM 180 nm-Technologie gegeben. Zu erkennen ist, dass die Prozessoren mit steigender aufsteigender Nummer in der Benennung größer, umfangreicher und durchsatzstärker werden. Da für die Ausführung von Zustandsdiagramm-basierten Programmen davon ausgegangen werden kann, dass SIMD, Gleitkommazahlen und Hardware-Dividierer keinen großen Performancegewinn bringen werden, wurde auf den M4-Prozessor verzichtet. Der M0+-Prozessor wurde nicht gewählt, da für diesen kein Zugriff auf eine HDL-Beschreibung bestand, um diesen Hardware-seitig zu adaptieren. Da aber sowohl der M1 als auch der M0+ vom M0 abgeleitete Prozessoren und alle drei Instruktionssatz-kompatibel sind, lässt sich aus der Evaluation des M0 auch eine Abschätzung der Performance für den M0+- und den M1-Prozessor geben.

Der Instruktionssatz des ARM Cortex-M0-Prozessor ist der ARMv6-M [Lim07a]. Dieser verwendet *Thumb*- und *Multiple*-Lade- und -Speicherinstruktionen, die den Instruktionscode stark komprimieren können und somit Hardware-seitig kleinere Speicher ermöglichen. *Thumb*-Instruktionen haben eine Länge von 16 bit und können nur eine Untermenge aller verfügbaren Register und Speicherbereiche adressieren, die im ARMv6-M-Instruktionssatz definiert sind. Dafür lassen sich zwei *Thumb*-Instruktionen in einem 32 bit-Wort speichern, was den Speicherbedarf bei ausschließlicher Verwendung dieser Befehle halbieren würde. Der Nachteil dieser Instruktionen ist, dass neben dem regulären ARM-Instruktionsdecoder ein zweiter *Thumb*-Decoder benötigt wird, zwischen denen je nach Instruktionstyp gewechselt werden muss. Für das Umschalten muss die Prozessor-Pipeline geleert und neu initialisiert werden, was bei einer gerade ausgeführten Division zu 12 Wartezyklen führen kann, da dies eine Multizykleninstruktion ist. Die *Multiple*-Lade- und -Speicherinstruktionen ermöglichen die Ausführung mehrerer Lade- oder Speicherbefehle auf konsekutive Register in nur einer Instruktion zu codieren. Diese Instruktion wird als Mehrzykleninstruktion verarbeitet und hilft die Menge des Instruktionscodes zu reduzieren.

Der Instruktionssatz des ARM Cortex-M3-Prozessors ist der ARMv7-M [Lim06b]. Dieser unterstützt alle Instruktionen des ARMv6-M-Instruktionssatzes und ist damit abwärts binär-kompatibel. Zusätzlich werden 32 bit-Thumb2-Instruktionen unterstützt, die das Decodieren und Ausführen von *Thumb*-Instruktionen über den gesamten Speicher- und Registeraddressraum erlaubt. Dies verhindert, dass zwischen dem regulären ARM- und dem *Thumb*-Decoder umgeschaltet werden muss, sobald mehr Register benötigt werden als sich durch die 16 bit-*Thumb*-Instruktionen codieren lassen.

Um Applikationenscode echtzeitfähig auf den beiden Prozessorkernen ausführen zu können, wird ein ARM MPS2+-FPGA-Entwicklungsboard verwendet. Dieses besitzt ein Intel CycloneV FPGA mit 300K Logikelementen, 8 MByte SRAM, Ethernet, SPI und viele weitere Anschlüsse für I/O und Debugging [Lim15]. Auf diesem CycloneV-FPGA lassen sich die Cortex-M-Prozessoren als *Softcore*-Prozessoren samt Busstruktur und Peripherieansteuerung abbilden.

Das gesamte auf dem FPGA abgebildete System samt Prozessoren ist in Abbildung 5.1 dargestellt. Der *Softcore*-Prozessorkern ist in ein Cortex-M-Subsystem eingebettet, das *Debug*-

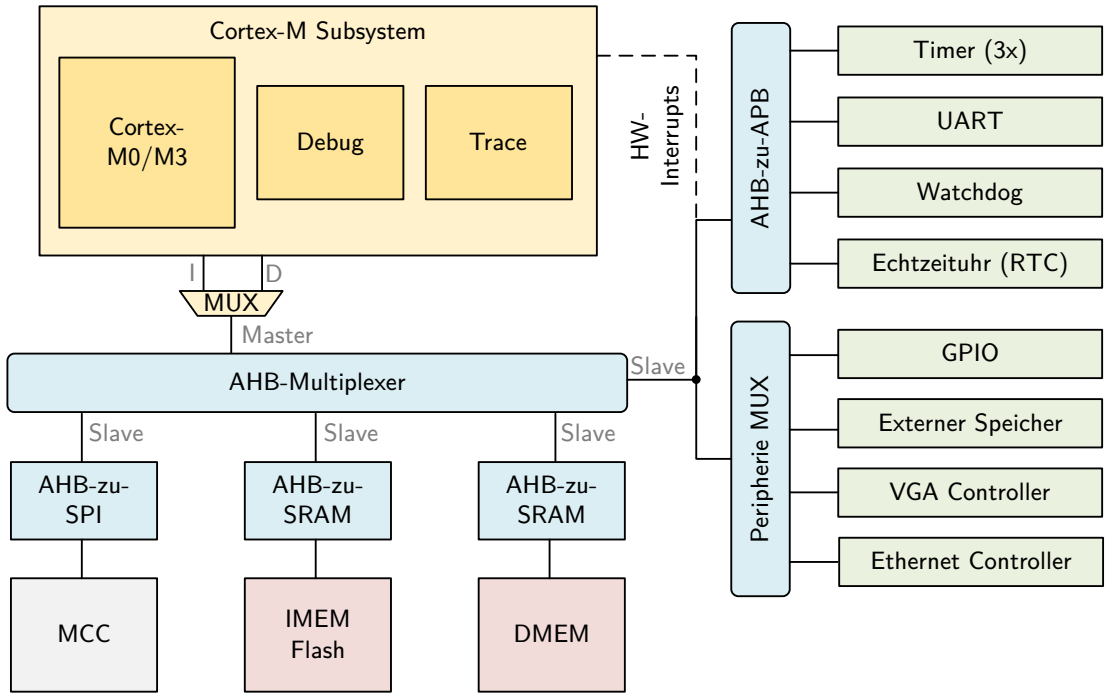

Abbildung 5.1: *Aufbau der DesignStart-Umgebung mit zentralem AHB-Lite und angeschlossenen Speichern und Komponenten. Der Prozessorkern selbst ist in ein Cortex-M-Subsystem mit Debugschnittstellen eingebettet [Lim15].*

und *Trace*-Schnittstellen bereitstellt, um Firmware zu debuggen. Peripherie und Speicher sind über einen *Advanced High-performance Bus* (AHB) angebunden. Dieser besitzt einen *Master*- und vier *Slave*-Ports, was ihn als *AHB Lite* klassifiziert [Lim06a]. Der Zugriff auf den Instruktions- (I) und Datenspeicher (D) erfolgt über einen Multiplexer, der als *Master*-Port an den AHB angeschlossen ist. Alle weiteren Komponenten des Busses sind *AHB-Slaves*.

Der *Motherboard Control Chip* (MCC) führt die *Boot*-Sequenz aus. Dazu wird zuerst das FPGA mit dem im Flash gespeicherten *Bitstream* konfiguriert und der Instruktionscode für den Prozessor in den IMEM geladen. Der Zugriff auf interne und externe Peripherie erfolgt über zwei separate AHB-Multiplexer. Der Peripherie-Multiplexer kombiniert mehrere externe Komponenten, so dass sie an nur einem *AHB-Slave* angeschlossen sind und immer nur Komponenten gleichzeitig kommunizieren kann. Dasselbe gilt für die per *Advanced Peripheral Bus* (APB) angeschlossenen Komponenten, mit der zusätzlichen Einschränkung, dass keine *Burst*-Übertragungen möglich sind und die Kommunikation mit einer höheren Latenz erfolgt. An diesem APB sind nur Komponenten angeschlossen, die weder hohe Datenraten noch geringe Latenzen benötigen.

Zur Programmierung des gesamten Systems liefert ARM mit *DesignStart* unter anderem eine Software-Umgebung mit Syntheseskripten, Intel Quartus-Projekten und GCC-Skripten, die die Kompilierung, die Synthese und das Ausführen des Applikationscodes auf dem MPS2+-Board automatisiert durchführen können [Lim22]. Die Skripte wurden in dieser Arbeit adaptiert und erweitert, um eine automatisierte Evaluation des emulierten Gesamtsystems durchführen zu können. Zur Messung der Performance der zu implementierenden Protokolle wurde ein *IPerf2-Server* auf Basis einer Implementierung von R. J. McMahon [McM04] adaptiert und auf dem Cortex-M-Prozessor ausgeführt [Jon22]. Ein *IPerf2-Client* wurden auf einem x86-basierten Linux-PC aufgesetzt und mit dem MPS2+-Board per Ethernet verbunden. Die maximale Datenrate ist

durch den dedizierten Ethernet-PHY-Chip auf dem MSP2+-Board begrenzt, der mit maximal 100 Mbit/s kommuniziert. Gemessen wird der Durchsatz und die Latenz einer TCP-Verbindung von *Server* zu *Client*.

5.2. Leightweight IP und Micro IP

Für die Implementierung der verschiedenen Protokolle auf den Cortex-M-Prozessoren wurden frei verfügbare Implementierungen von Protokollfamilien gesucht, die ohne Betriebssysteme lauffähig sind und mindestens ARP, ICMP, IPv4, UDP und TCP unterstützen. Basierend auf der heutzutage am weitesten verbreiteten BSD TCP/IP-Implementierung der *Computer Systems Research Group* der Berkeley Universität [Fal11] wurden von Adam Dunkels zwei Mikrocontroller-basierte Protokollfamilien implementiert [Dun01; Dun03]. Das Hauptaugenmerk dieser Implementierungen liegt auf der echtzeitfähigen Ausführung auf Mikrocontrollern ohne Betriebssystem und der Minimierung des benötigten Instruktions- und Datenspeichers.

Die erste Implementierung ist *Lightweight-IP* (lwIP), die fast den vollen Umfang der betriebssystembasierten BSD-Referenzimplementierung beibehält und den Fokus auf die Minimierung der benötigten Instruktions- und Datenspeicher legt. LwIP benötigt mit vollem Funktionsumfang nur 40 kByte im Instruktionsspeicher eines x86-Prozessors und unterstützt unter anderem die Protokolle TCP, IP, ICMP, ARP und UDP. Auch die Verwendung innerhalb eines Echtzeitbetriebssystems ist möglich [Dun01].

Als erster Schritt wurde die frei verfügbare lwIP-Implementierung in der Version 2.0.3 der *Free Software Foundation* auf die beiden ARM Cortex-M-Prozessoren portiert [Fou02]. Diese wurde so konfiguriert, dass zwei eingehende Pakete maximaler Größe (1522 Byte) gespeichert werden können. Sendeseitig können ebenfalls zwei Pakete der maximalen Ethernet-*Frame*-Größe (1460 Byte) gespeichert werden. Die resultierende Gesamtgröße des Instruktionscodes inklusive *IPerf2-Server*-Anwendung und Compiler-seitiger Optimierung auf Codegröße (-Os) ist 44,6 kByte für den Cortex-M0 und 41,1 kByte für den Cortex-M3. Davon sind etwa 86 % Instruktionscode des lwIP und 14 % die *IPerf2-Server*-Anwendung.

Die Messung der TCP/IP-Datenraten auf dem MPS2+-Boards ergibt 14,0 Mbit/s für den Cortex-M0 und 20,3 Mbit/s für den Cortex-M3 bei einer FPGA-Taktfrequenz von 25 MHz. Diese initialen Datenraten sind bereits schnell genug, um das Powerline-Modem aus Kapitel 4 anzusteuern. Das Ziel ist aber die Implementierung des Systems in der XFAB XT018-ASIC-Technologie, was in einer geringeren Taktrate als auf dem FPGA und einem hohen Flächenbedarf des Gesamtsystems resultiert. Da das Leistungsbudget im Bohrstrang limitiert ist und dieser Chip zusätzlich zum Powerline-Chip betrieben werden muss, wird im Folgenden eine maximale Erhöhung des Durchsatz angestrebt, um anschließend die Taktfrequenz so weit zu reduzieren, so dass die 10 Mbit/s erreicht werden. Dabei skaliert sowohl der Durchsatz als auch die Leistungsaufnahme linear mit der Taktfrequenz.

Die zweite Implementierung von Adam Dunkels ist *Micro IP* (uIP), die nur noch den minimalen Funktionsumfang der BSD-basierten Implementierung hat und die strikte Trennung der Transport- und Vermittlungsschicht aufweicht [Dun03]. Damit kann der Ressourcenbedarf des Instruktions- und Datenspeicher gegenüber lwIP noch einmal deutlich gesenkt werden. Entwickelt wurde das Protokoll mit dem Ziel, einen echtzeitfähigen TCP/IP-Stack für kleine Mikrocontroller mit

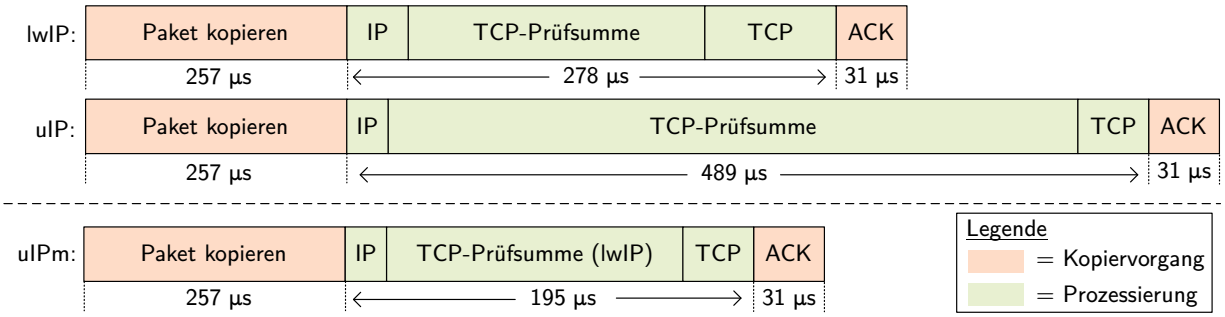

Abbildung 5.2: *Vergleich der Bearbeitungszeiten der Decodierung und Beantwortung eines eingehenden Paketes für lwIP und uIP. Unterhalb der gestrichelten Linie ist eine optimierte Version von uIP gezeigt (uIPm), die die Prüfsummenberechung von lwIP verwendet.*

16 bit- oder sogar nur 8 bit-Wortbreite bereitzustellen. Dieser unterstützt auch die Protokolle TCP, IP, ICMP, ARP und UDP, ist aber im Gegensatz zu lwIP auf ein einziges Netzwerkinterface beschränkt.

Auch der uIP-Stack wurde in der Version 1.0 auf die beiden Cortex-M-Prozessoren portiert [Dun13]. Die resultierende Größe des Instruktionscodes sind 9,2 kByte für den Cortex-M0 und 7,9 kByte für den Cortex-M3, was verglichen mit dem lwIP-Stack nur 20 % der Codegröße beträgt. Da der uIP-Stack nur einen einzigen 1522 Byte-großen bidirektionalen Pufferspeicher für Pakete verwendet, ist die resultierende Größe des Datenspeichers nur etwa die Hälfte des vom lwIP benötigten.

Die gemessene TCP/IP-Datenrate beträgt für den Cortex-M0 8,5 Mbit/s und für den Cortex-M3 10,7 Mbit/s und damit auf beiden Prozessoren etwa 80 % des Durchsatzes des lwIP-Stacks. Da der uIP-Stack kleiner und reduzierter gegenüber dem lwIP-Stack ist, war die Erwartung, dass der Durchsatz höher sein muss als beim lwIP-Stacks. Daher wurde ein *Profiling* beider Programmcodes für das Empfangen eines Paketes auf dem Cortex-M3 durchgeführt. Die Ergebnisse sind in Abbildung 5.2 gezeigt.

Zunächst wird das empfangene Paket über den Treiber der *Ethernet*-Schnittstelle kopiert, was für alle Implementierungen 257 µs dauert (rote Blöcke links in Abbildung 5.2). Als erstes wird der IP-*Header* prozessiert und dessen Prüfsumme ausgewertet. Danach wird die TCP-Prüfsumme über das gesamte Paket berechnet und verifiziert, was bei beiden Implementierungen den größten Teil der Paketverarbeitungszeit beansprucht. Anschließend folgt die restliche Prozessierung des TCP-Protokolls und die Generierung der Antwort (ACK), die in Abbildung 5.2 als „TCP“ zusammengefasst sind. Zum Schluss wird die Antwort zurück an das *Ethernet*-Interface kopiert (rote Blöcke rechts in Abbildung 5.2), was wieder für alle Implementierung 31 µs dauert.

Auffällig ist, dass die Berechnung der Prüfsumme bei uIP fast doppelt so lange dauert, wie bei lwIP, obwohl beide Routinen das identische Ergebnis berechnen und als C-Implementierung auf den Prozessoren laufen. Bei uIP wird eine einfache *while*-Schleife über jedes zu verrechnende Datenbyte verwendet. lwIP verwendet eine abgerollte Version dieser Schleife, was durch die *Multiple*-Lade- und -Speicherbefehle der Cortex-M-Prozessoren zu einer beschleunigten Ausführung führt. Zur Beschleunigung wurde die abgerollte Schleife auch in uIP verwendet, was im Folgenden als *Micro IP Modified* (uIPm) bezeichnet wird. Die resultierende Ausführungszeit (Abbildung 5.2 unten) ist durch die schnellere Berechnung des IP-Protokolls und der restlichen TCP-Anteile in der uIP-Implementierung schneller als lwIP. Eine erneute Messung des Durchsatzes ergab

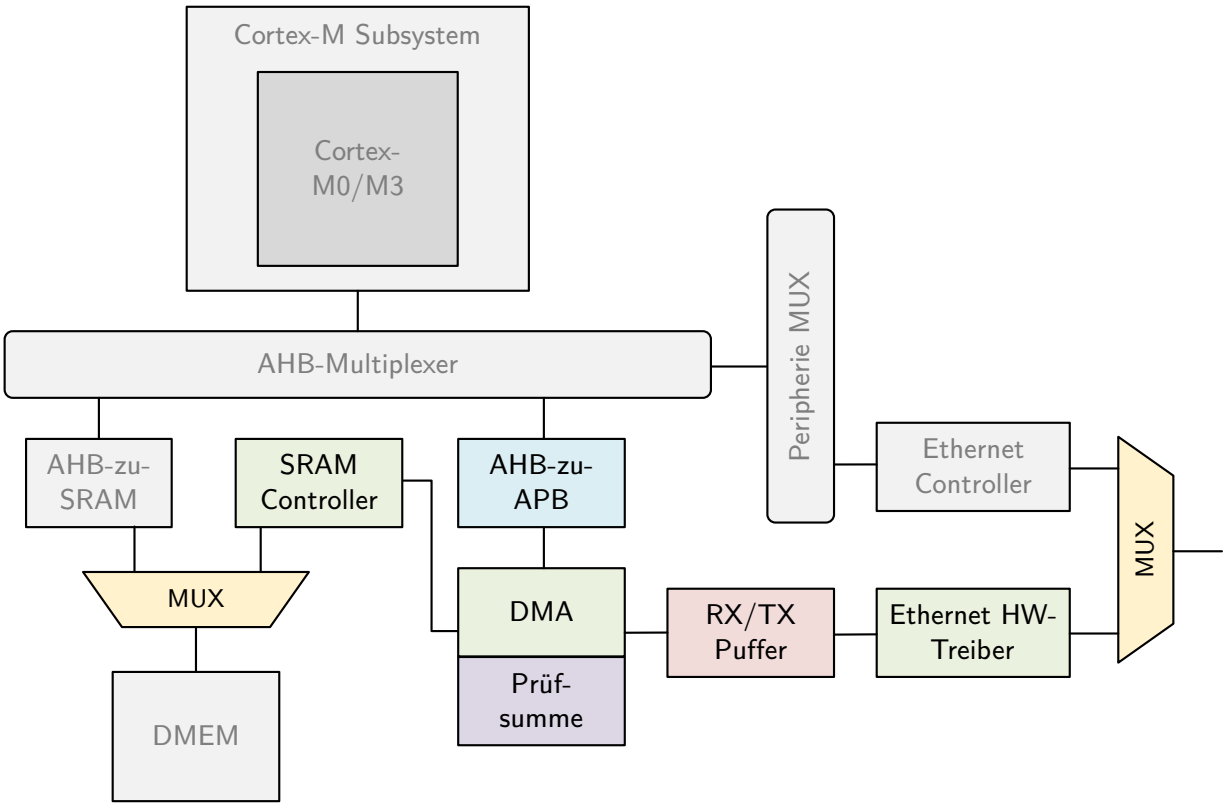

Abbildung 5.3: *Erweiterung des ARM DesignStart-Systems um einen DMA mit direktem Zugriff auf DMEM und den externen Ethernet-Controller und eine Prüfsummenberechung in Hardware. Alle neuen Komponenten sind farbig hervorgehoben.*

16,60 Mbit/s für den Cortex-M0 und 25,31 Mbit/s für den Cortex-M3, also einen Speedup von 1,9 für den Cortex-M0 und von 2,3 für den Cortex-M3 gegenüber der Basis-uIP-Implementierung.

Die verbleibenden signifikanten Anteile an der Ausführungszeit der Protokolle von LwIP und uIP sind das Kopieren der Pakete und die Berechnung der TCP-Prüfsumme. Um diese zu beschleunigen, wird im folgenden Abschnitt zuerst die Implementierung eines direkten Speicherzugriffs (DMA) vorgestellt und evaluiert. In einem zweiten Schritt wird die Prüfsummenlogik zur kontinuierlichen Berechnung während der Paketübertragung in das DMA-Modul integriert und evaluiert.

5.3. Hardware-Erweiterungen der Prozessoren

Für die Integration eines DMA in das bestehende System muss dieser einen direkten Zugriff auf den Datenspeicher (DMEM) und das *Ethernet*-Interface erhalten. Beide Komponenten sind per AHB an den Cortex-M-Prozessor angebunden, so dass der Zugriff gemultiplext werden muss. Abbildung 5.3 zeigt einen Ausschnitt aus Abbildung 5.1 mit den AHB-Verbindungen zu DMEM und *Ethernet*-Hardware. Alle grau dargestellten Blöcke sind Komponenten aus Abbildung 5.1 und die farbigen Blöcke zeigen neu implementierte Module, die zur Ansteuerung des DMA und dessen Zugriff auf die Speicher notwendig sind. Jeder dieser Blöcke wurde in VHDL implementiert, in Simulation verifiziert und für das MPS2+-Board synthetisiert.

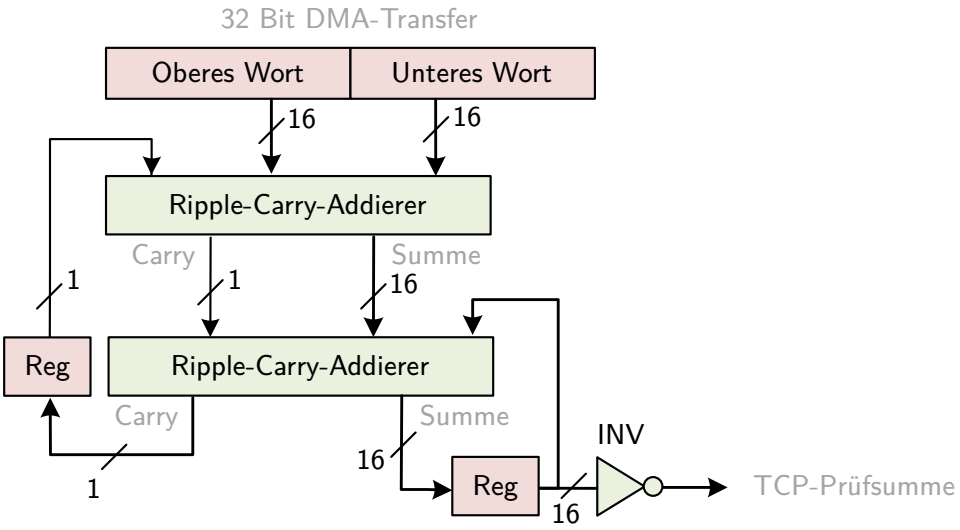

Abbildung 5.4: *Hardware-Einheit zur Berechnung der TCP-Prüfsumme nach Hsiao et al. [Hsi09]. Die Ausgabe ist das Einerkomplement durch Inversion des Inhalts des Akkumulationsregisters.*

Der DMA wird per AHB-zu-APB-Brücke an den AHB-Multiplexer angeschlossen und ist somit vom Cortex-M-Prozessor aus steuerbar. Der Zugriff auf den DMEM erfolgt mit Hilfe eines SRAM-Controllers und eines Multiplexers, der den Speicherzugriff zwischen DMA und Prozessor priorisiert regelt. Da nicht sichergestellt werden kann, dass der Prozessor während eines DMA-Transfers nicht auf den DMEM zugreift, wird das ausgeführte Programm für die Dauer des DMA-Transfers angehalten. Dies erlaubt zwar keine parallele Ausführung beider Einheiten, reduziert aber die Paketkopierzeit (inklusive der Antwort) bereits um den Faktor 18 auf 15,78 µs.

Der Anschluss an die *Ethernet*-Schnittstelle wird ebenfalls mit Hilfe eines Multiplexers realisiert, der je nach Anfrage den internen Controller oder den Hardware-Treiber durchschaltet. Zwischen DMA und *Ethernet*-Treiber wird ein Sende- und Empfangspuffer benötigt, der ein gesamtes *Ethernet*-Paket speichern kann. Beim Empfang eines Paketes schreibt der Treiber dieses in den Puffer und signalisiert dann dem DMA, dass ein Paket empfangen wurde. Sendeseitig geschieht dies in umgekehrter Reihenfolge, so dass DMA und *Ethernet*-Treiber unabhängig voneinander parallel arbeiten können. Dabei signalisieren DMA und Treiber das Schreiben des Puffers an die jeweils andere Einheit, um einen Schreibkonflikt zu verhindern.

Um die TCP-Prüfsummenberechnung zu beschleunigen, die den Großteil der Verarbeitungszeit ausmacht, wurde diese in den DMA integriert. Diese Berechnung kann schritthaltend während des Kopiervorgangs durchgeführt und die Prüfsumme am Ende ausgelesen werden. Für die Berechnung der TCP-Prüfsumme werden 16 bit-Worte ohne Wortbreitenzuwachs akkumuliert und das überlaufende *Carry*-Bit in der nächsten Addition als LSB addiert. Das Design der Hardware in Abbildung 5.4 basiert auf dem Ansatz von Chase et al. [Cha01] und wurde nach der Struktur von Hsiao et al. [Hsi09] implementiert. Dabei wird die Wortbreite des 32 bit-DMA-Transfers genutzt, um pro Transfer zwei Additionen durchzuführen und anschließend in einem Register zu speichern. Nachdem alle Datenbytes mittels DMA kopiert wurden, kann die TCP-Prüfsumme als Einerkomplement des Akkumulationsregisters über den AHB ausgelesen werden. Um ein korrektes Prüfsummenergebnis zu erhalten, muss beim Start jedes DMA-Transfers das Akkumulationsregister zurückgesetzt und am Ende das letzte *Carry*-Bit nachträglich addiert

Tabelle 5.2: *Vergleich des Durchsatzes der Software- und Hardware-Erweiterungen für die uIP- und lwIP-Implementierungen. Der Durchsatz wurde auf dem MPS2+-Board bei einer Taktfrequenz von 25 MHz gemessen.*

	Cortex-M0		Cortex-M3	
	[Mbit/s]	Speedup	[Mbit/s]	Speedup
uIP	12,76	-	16,96	-
uIPm	16,60	1,3	25,31	1,5
uIPm DMA	27,12	2,1	59,40	3,5
uIPm D&P	(123,3)	9,6	(228,62)	13,5
lwIP	14,00	-	20,29	-
lwIP DMA	22,20	1,6	41,46	2,0
lwIP D&P	(49,97)	3,6	(88,40)	4,4

werden. Beim Senden eines Paketes wird die TCP-Prüfsumme direkt in den TCP-*Header* im Sendepuffer geschrieben und das Paket anschließend versendet. Beim Paketempfang wird die Prüfsumme über den AHB gelesen und in Software mit der übertragenden Prüfsumme verglichen.

Nach der Integration aller Komponenten wurde erneut eine Evaluation der Durchsatzraten mit Hilfe des *IPerf2*-Programms durchgeführt. Die Ergebnisse sind in Tabelle 5.2 dargestellt. Da durch die Berechnung mit DMA und TCP-Prüfsumme in Hardware zusammen eine Durchsatzrate größer als 100 Mbit/s der verfügbaren *Ethernet*-Schnittstelle erreicht, wurde deren Durchsatz nach Gleichung 5.1 interpoliert.

$$R_{\text{D\&P}} = \frac{t_{\text{copy}} + t_{p,\text{old}}}{t_{\text{copy}} + t_{p,\text{new}}} R_{\text{DMA}} \tag{5.1}$$

Hierbei ist R die erreichte Durchsatzrate, t_{copy} die Kopierzeit des DMA mit 15,78 µs und t_p die alte und neue Prozessierungszeit eines Paketes in Software. Die Prozessierungzeiten wurden mit Hilfe eines Segger J-Link-Adapters und dessen proprietärer SystemView-Software gemessen. Damit kann zyklengenau und echtzeitfähig die Ausführungszeit zwischen zwei definierten Punkten in der Software gemessen werden. Die Ergebnisse dieser Interpolation sind in Tabelle 5.2 in Klammern gegeben. Dabei wurden auch die Durchsatzraten des lwIP approximiert, um die Ergebnisse der Berechnung durch die Hardware-Prüfsumme besser vergleichbar zu machen.

Bei Verwendung der DMA-Erweiterung konnte der Durchsatz beider Stacks auf dem Cortex-M0-Prozessor um etwa 60 % gesteigert werden. Für den Cortex-M3-Prozessor wurde eine Steigerung von 100 % für den lwIP-Stack und 135 % für den uIPm-Stack erreicht. Wird auch die Berechnung der TCP-Prüfsumme mit betrachtet (DMA und Prüfsumme, kurz D&P), kann für den Cortex-M0-Prozessor mit dem lwIP-Stack ein Durchsatz von 49,97 Mbit/s erreicht werden, was eine Steigerung um den Faktor 3,6 gegenüber der reinen Software-Berechnung entspricht. Der uIPm-Stack erreicht mit einem Durchsatz von 123,3 Mbit/s eine Steigerung um den Faktor 9,6. Auf dem Cortex-M3-Prozessor erzielt der lwIP-Stack mit 88,30 Mbit/s einen Speedup von 4,4 und der uIPm mit 228,62 Mbit/s einen Speedup von 13,5. Damit ist die uIPm-Implementierung

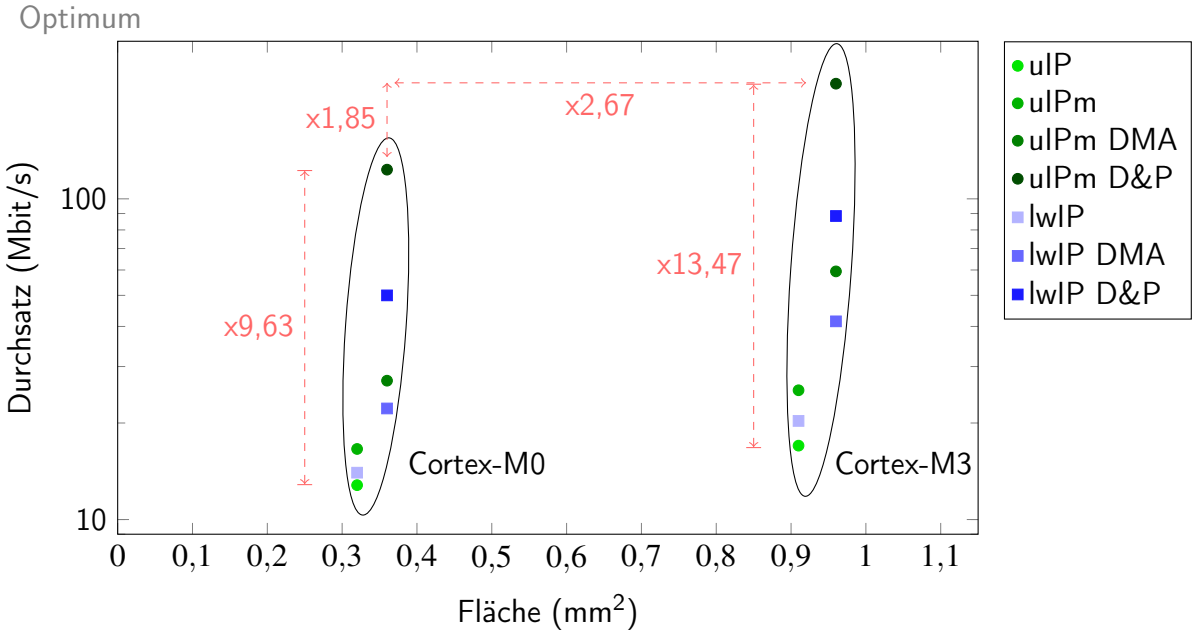

Abbildung 5.5: *Entwurfsraumevaluation des Cortex-M0 und Cortex-M3 über die benötigte Siliziumfläche gegenüber dem erreichten Durchsatz des lwIP- und uIP-Stacks.*

auf dem Cortex-M3 fast doppelt so schnell wie auf dem Cortex-M0 und auf beiden Prozessoren schneller als der lwIP-Stack.

5.4. Ergebnisse der ASIC-Synthesen

Um feststellen zu können, welchen Einfluss die Hardware-Erweiterungen auf den Flächenzuwachs und die Leistungsaufnahme des Gesamtsystems haben, wurden ASIC-Synthesen aller DesignStart-Netzlisten jeweils mit dem Cortex-M0- und dem Cortex-M3-Prozessor durchgeführt. Abbildung 5.5 zeigt die Siliziumfläche der Prozessorkerne und ihren Hardware-Erweiterungen gegenüber dem erreichten Durchsatz für beide Stacks auf den jeweiligen Implementierungen.

Der Cortex-M0-Prozessorkern hat mit 0,32 mm^2 etwa ein Drittel der Fläche eines Cortex-M3-Prozessorkerns mit 0,91 mm^2. Die Größe der DMA-Erweiterung, die alle neu integrierten Komponenten außer der Prüfsumme aus Abbildung 5.3 umfasst, hat mit 0,04 mm^2 nur 12,5 % der Fläche eines Cortex-M0-Prozessorkerns. Die Erweiterung um die Prüfsumme ist mit 0,02 mm^2 noch kleiner und etwa die Hälfte der DMA-Hardware-Erweiterung. Diese beiden Erweiterungen werden in Abbildung 5.5 mit dem Suffix DMA für die DMA-Erweiterung ohne Prüfsummenberechung und D&P für die DMA-Erweiterung mit Prüfsummenberechung bezeichnet. Der Zuwachs der Fläche ist für beide Prozessoren identisch und ist entlang der Abszisse mit steigendem Durchsatz zu sehen. Das Optimum der Implementierungen ist in der unten rechten Ecke zu finden, wo die Fläche minimal und der Durchsatz maximal sind. Mit jeder der vorgestellten Hardware- und Software-Optimierungen nähert sich die Implementierung diesem an und steigt überproportional im Durchsatz gegenüber dem Flächenzuwachs. Dabei zeigt sich, dass der uIPm-Stack dem lwIP-Stack vom Durchsatz in allen Implementierungen auf beiden Prozessoren überlegen ist. Der Unterschied zwischen dem Cortex-M0- und dem Cortex-M3-Prozessor

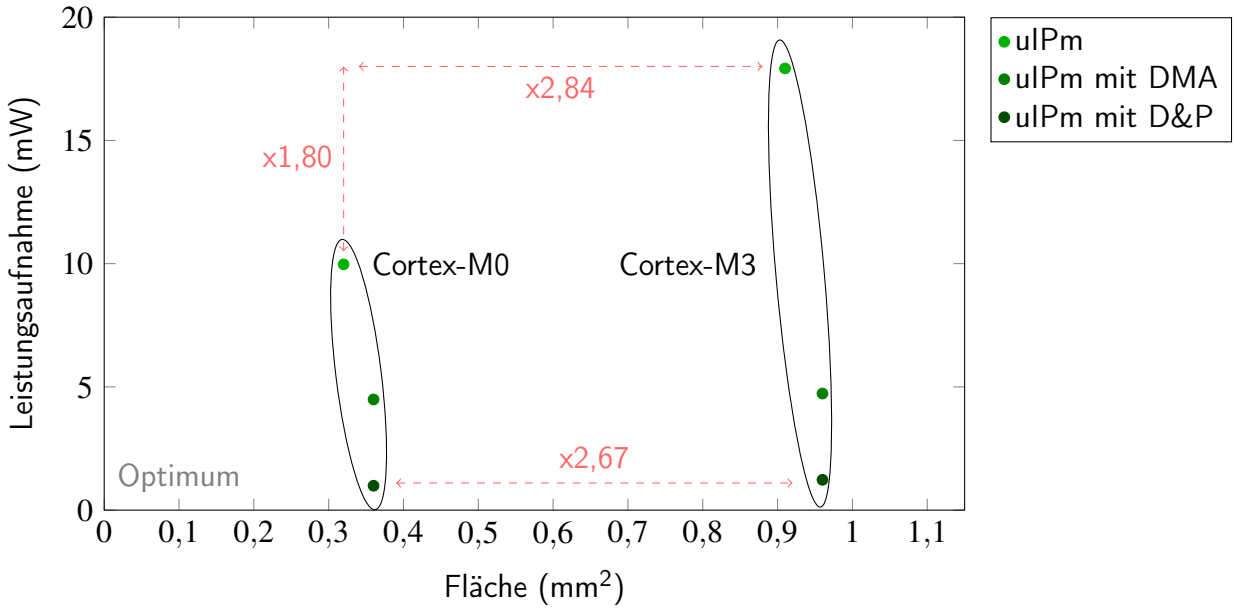

Abbildung 5.6: *Vergleich des Cortex-M0- und Cortex-M3-Prozessors bezogen auf deren Flächenbedarf und Leistungsaufnahme unter Verwendung des uIPm-Stacks. Die Taktfrequenz wurde adaptiert, so dass ein Durchsatz von nur noch 10 Mbit/s für die Steuerung des Powerline-Systems erreicht wird.*

mit allen Hardware-Erweiterungen und dem uIPm-Stack ist ein Abtausch der Steigerung des Durchsatzes um den Faktor 1,85 gegenüber einem Zuwachs der Fläche um den Faktor 2,67. Insgesamt wurde der uIP-Stack auf dem Cortex-M0-Prozessor um den Faktor 9,63 und auf dem Cortex-M3-Prozessor um den Faktor 13,47 gesteigert.

Abschließend wird noch die Adaption für die Steuerung des in Kapitel 4 beschriebenen Powerline-Systems vorgestellt. Dafür wird nur noch der uIPm-Stack verwendet, da dieser auf beiden Prozessoren schneller und kleiner ist als der lwIP-Stack. Das Powerline-System hat eine Durchsatzrate von 10 Mbit/s am Eingang der MAC-Schicht und ist damit langsamer als alle vorgestellten uIPm-Versionen. Daher wird der Systemtakt jeder Hardware-Version so weit reduziert, dass diese ebenfalls 10 Mbit/s erreicht. Anschließend wird die Leistungsaufnahme des Systems mittels statischer Analyse durch das Synthesewerkzeug Cadence Genus ermittelt. Dieses nimmt für alle Signale eine Schaltaktivität von 50 % an. Das ist zwar für reale Anwendungen deutlich zu hoch, liefert aber eine obere Grenze der Leistungsaufnahme und eine gute Vergleichbarkeit der Implementierungen untereinander. Die Ergebnisse dieser Evaluation sind in Abbildung 5.6 gegenüber der Siliziumfläche aufgetragen.

Das Optimum der Implementierungen für die beiden Parameter liegt im Ursprung des Graphen, wo Leistungsaufnahme und Fläche minimal sind. Die rein Software-basierte Berechnung der beiden Stacks unterscheidet sich zwischen vom Cortex-M0- zum Cortex-M3-Prozessor in einem Faktor 1,8 höherer Leistungsaufnahme und Faktor 2,84 mehr an Fläche. Letzterer Faktor ist größer als in der Evaluation in Abbildung 5.5, da durch die verringerte Taktrate das Synthesewerkzeug eine Reduzierung der Fläche erreicht. Diese Reduzierung fällt, verglichen mit den Synthesen für eine Zieltaktfrequenz von 25 MHz, beim Cortex-M0 stärker aus, als beim Cortex-M3.

Beim Vergleich der Implementierungen mit Hardware-Erweiterungen, führt die auf Grund des steigenden Durchsatzes noch stärker reduzierte Taktrate zu einer fast identischen Leistungsaufnah-

me der beiden Prozessoren. Da der Cortex-M0 um den Faktor 2,67 kleiner ist als der Cortex-M3, ist für die in diesem Abschnitt vorgestellte Steuerung des Powerline-Systems der Cortex-M0 zu bevorzugen. Der Hauptgrund für dieses schlechte Abschneiden des Cortex-M3-Prozessors in dieser Untersuchung ist das uIPm-Protokoll, das die zusätzlichen Hardware-Einheiten, die den Größenunterschied der beiden Prozessoren ausmachen, nicht ausnutzen kann.

Abschließend können folgende Erkenntnisse für die Integration eines TCP/IP-basierten Stacks auf ARM Cortex-M-Prozessoren in der XFAB XT018-Technologie zusammengefasst werden:

- Der gezeigte Entwurfsraum zeigt einen Abtausch von Durchsatz und Verlustleistung gegenüber der Chip-Fläche auf.
- Der höchste Durchsatz von 228,62 Mbit/s wird mit dem ARM Cortex-M3-Prozessor und dem modifizierten uIP-Stack erreicht.
- Der Durchsatz des uIPm-Stack ist in allen Implementierungen größer als der des lwIP.
- Die Pareto-optimalen Punkte des Entwurfsraumes sind die Implementierungen des uIPm-Stacks auf dem Cortex-M0-Prozessor. Dabei ist die Version mit DMA und Prüfsummenberechnung in Hardware mit einer Fläche von 0,36 mm^2 und einer Leistungsaufnahme von 0,99 mW optimal.

KAPITEL 6

Design und Evaluation des Powerline-ASICs

Auf Grundlage des in Abschnitt 4.3 verifizierten und charakterisierten FPGA-Designs, wird in diesem Kapitel zunächst die Überführung des Designs in eine ASIC-Netzliste vorgestellt. Dazu wird die XT018-Technologie der Firma XFAB verwendet, die den Chip auf Basis der Netzliste nach Platzierung und Verdrahtung (engl. *Place and Route*) fertigt. Da der Fertigungsprozess inklusive Chip-Verifikation mehrere Monate dauert und die Kosten der Fertigung sehr hoch sind, wurde nur das Powerline-System und nicht der in Kapitel 5 vorgestellte ARM-Prozessor, gefertigt.

Zunächst muss das vorliegende FPGA-Design des Powerline-Kommunikationssystems für die XT018-Technologie adaptiert werden. Unter anderem müssen die Speicherblöcke im Design durch vorkonfigurierte Blöcke des Technologieherstellers XFAB ersetzt und getestet werden. Auch die im FPGA vorhandenen DSP-Blöcke müssen durch dedizierte Multiplikations- und MAC-Einheiten ersetzt werden. Startpunkt für die Adaptierung ist die VHDL-Beschreibung, die in Abschnitt 4.1 evaluiert wurde.

Zunächst wird in Abschnitt 6.1 die Adaptierung des verifizierten FPGA-Designs für die ASIC-Fertigung gezeigt. Anschließend wird in Abschnitt 6.2 das Design des Chips gezeigt und die finale Netzliste zur Fertigung des ASIC evaluiert. Hierbei ergeben sich erste Abschätzungen zur Leistungsaufnahme, Fläche und Laufzeit des Chips. Die Evaluation des gefertigten ASICs erfolgt in Abschnitt 6.3 durch Messungen der Selbsterwärmung und Hochtemperaturmessungen im Klimaschrank. Abschließend werden die Ergebnisse dieser Messungen in Abschnitt 6.4 diskutiert und Empfehlungen für das Design zukünftiger Kommunikationssysteme in extremen Umweltbedingungen gegeben.

6.1. Adaptierung des FPGA-Designs

Bei der Synthese eines HDL-Designs für ein FPGA bildet das Synthesewerkzeug die beschriebenen Strukturen auf dedizierte Hardware-Blöcke ab, zum Beispiel werden Multiplikationen auf DSP-

Tabelle 6.1: *Anzahl und Größe der verwendeten SRAM-Speicherbausteine in der XT018-Technologie. Zusätzlich ist angegeben, ob die Speicher bitweisen Zugriff unterstützen oder nur die volle Wortbreite.*

Anzahl	Kapazität	Tiefe	Breite	bitweiser Zugriff?
31	0,5 kByte	256	16 bit	Ja
9	2 kByte	1024	16 bit	Ja
1	8 kByte	2048	32 bit	Nein
1	32 kByte	16384	16 bit	Ja

Blöcken berechnet oder Speicher als Block-RAM implementiert. Die dedizierten Strukturen müssen für eine ASIC-Synthese per Hand optimiert und eingefügt werden, da sonst das Synthese-Tool unter anderem Speicherblöcke als Flip-Flops synthetisiert, was zu extrem großen Chipgrößen führen würde.

Für dieses ASIC-Design müssen die folgenden Änderungen vorgenommen werden:

- Partitionierung und Hierarchisierung der Speicherstruktur
- Analyse des kritischen Pfades mit neuen Speicherblöcken
- Implementierung von Konfigurationsregistern und *Bootloader*

Zunächst wurde das Design des Powerline-Modems auf Speicherblöcke untersucht, die mehr als 32 Byte an Daten zwischenspeichern. Insgesamt wurden 16 Module mit Speicherblöcken gefunden, davon 8 in der PHY-Schicht und 8 in der MAC-Schicht. Für den NEO430-Prozessoren in der MAC-Schicht ist dabei wichtig, dass der Speicherbaustein für den Datenspeicher byteweise adressierbar ist. Da sich fast alle Speichergrößen der Module unterscheiden, müsste für jedes Modul ein eigens generierter Speicher generiert werden. Jedes dieser Speichermodule muss separat erstellt, getestet und evaluiert werden. Daher wurden größere Speicherblöcke teilweise aus kleinen bereits verifizierten Speichern zusammenzusetzen.

Außerdem gab es zum Zeitpunkt der ASIC-Adaptierung in die XFAB XT018-Technologie nur Speicherbausteine als *Single-Port*-Speicher, die auch für Temperaturen bis 175 °C zertifiziert sind (Stand 01.04.2020). Daher mussten alle als *Dual-Port*-Speicher ausgelegten Speicherblöcke im FPGA-Design aus zwei *Single-Port*-Speichern zusammengesetzt werden, die den identischen Speicherinhalt verwalten. Alternativ könnte auch ein Speicher mit der doppelten Taktrate betrieben werden. Dies funktioniert nur für die Teile des Designs, die nach der Basisband-Konvertierung (vgl. Abschnitt 3.2.1) mit der halben Taktrate von 25 MHz betrieben werden können, da kein verfügbarer Speicher eine Taktrate von 100 MHz erreicht.

Tabelle 6.1 zeigt die vier SRAM-Speicherbausteine, die für das ASIC-Design erstellt wurden. Der kleinste gemeinsame Speicher ist ein 256x16 bit-Speicher, der hauptsächlich die Pufferspeicher in der PHY-Schicht ersetzt. Nur das Empfangs-FIFO im PHY-Schicht-Decoder wurde mit zwei Bausteinen der Größe 2 kbit ersetzt. Zwei weitere dieser Speicherblöcke wurden als 4 kByte großer Datenspeicher des NEO430 zusammengefügt. Da diese Speicherblöcke bitweise adressierbar sind, wurden immer acht Bitleitungen zu einer Byteleitung zusammengeführt, um byteweise aus dem NEO430-Prozessor den Speicher lesen und schreiben zu können. Die restlichen fünf Speicherblöcke wurden für die lokalen Speicher im Sende- und Empfangsstream der MAC-Schicht verwendet. Die 8 kByte und 32 kByte großen Speicherblöcke sind die Instruktionsspeicher

des PauloBlaze- und des NEO430-Prozessors. Diese wurden nicht aus kleineren Speichern zusammengesetzt, da durch die komplexere Adressierung der kritische Pfad länger als 20 ns werden würde.

Nachdem die Speicher im VHDL-Design eingesetzt und getestet wurden, wurde erneut eine Synthese in der XT018-Technologie durchgeführt und dabei die entworfenen Speicher als Bibliothek eingebunden. Die Timinganalyse zeigt, dass kein Speicher im kritischen Pfad liegt und dieser weiterhin mit 14,85 ns vom DES-Modul bestimmt wird.

Zuletzt wurden noch Konfigurationsregister auf oberster Ebene des Designs eingefügt, die über das *Bootloader*-Interface adressiert werden können (siehe Abschnitt 4.3). Die Register haben eine Breite von 16 bit und belegen die obersten 4096 Adressen der 15 bit-Adressraum des *Bootloaders* (0x7000 - 0x7FFF). In der in dieser Arbeit vorgestellten Chipversion werden davon nur 14 Register verwendet. Diese Konfigurationsregister enthalten Variablen, wie den Korrelationsschwellwert oder die *ToneMask*, die sich so zur Laufzeit des Systems adaptieren lassen. Sollte bei den späteren Hochtemperaturmessungen zum Beispiel festgestellt werden, dass der Übertragungskanal immer eine starke Tiefpasscharakteristik aufweist, lassen sich die hochfrequenten Träger in der *ToneMask* abschalten und so die Kommunikationsstabilität verbessern. Folgende Konfigurationsregister gibt es:

MACAddr [3 Worte] Adresse: 0x7000 – 0x7002

Die MAC-Adresse ist die einzigartige Adresse jedes Modems, anhand derer sie von anderen Stationen adressiert wird. Diese sollte in der Regel für eine Station immer identisch sein.

SoftBits [1 Wort, Wertebereich: 1 bis 3] Adresse: 0x7003, Default: 3

Die Anzahl der *Soft-Decision*-Bits gibt die Quantisierung der Decodierinformationen im Demodulator an (siehe Abschnitt 3.3).

CorrThresh [1 Wort, Wertebereich: 0 bis 2047] Adresse: 0x7004, Default: 128

Der Schwellwert der Präambeldetektion in der Zeitsynchronisierung S_{th} kann angepasst werden (siehe Abschnitt 3.2.1). Dies kann bei stark verrauschten Kanälen genutzt werden, um falsch-positive Detektionen im Rauschen zu vermeiden.

SyncPnt [1 Wort, Wertebereich: 0 bis 85] Adresse: 0x7005, Default: 60

Der Synchronisationspunkt gibt an, ab welcher Stelle im Guard-Intervall ein OFDM-Symbol dekodiert werden soll. Je nach statischen Eigenschaften des Kanals kann es sinnvoll sein, diesen Punkt zu adaptieren. Zum Beispiel für Kanäle mit langen Impulsantworten sollte dieser Punkt später im Symbol sein.

ToneMask [6 Worte, davon: 84 Bit] Adresse: 0x7006 – 0x700B, Default: Standard-Maske

Die *ToneMask* ist länderspezifisch reguliert und sollte zur Laufzeit angepasst werden können. Außerdem kann es sinnvoll sein, bestimme Frequenzbereiche von der Übertragung auszunehmen, falls diese zum Beispiel diese durch starke Schwankungen nur selten eine stabile Kommunikation ermöglichen.

FreezeMode [1 Wort, Wertebereich: 0 bis 1] Adresse: 0x700C, Default: 1

Der Sperrmodus für die AGC gibt an, ob diese sofort den aktuellen Verstärkungswert hält (FreezeMode = 0) oder sich für weitere 8 µs adaptiert und erst dann hält (FreezeMode = 1).

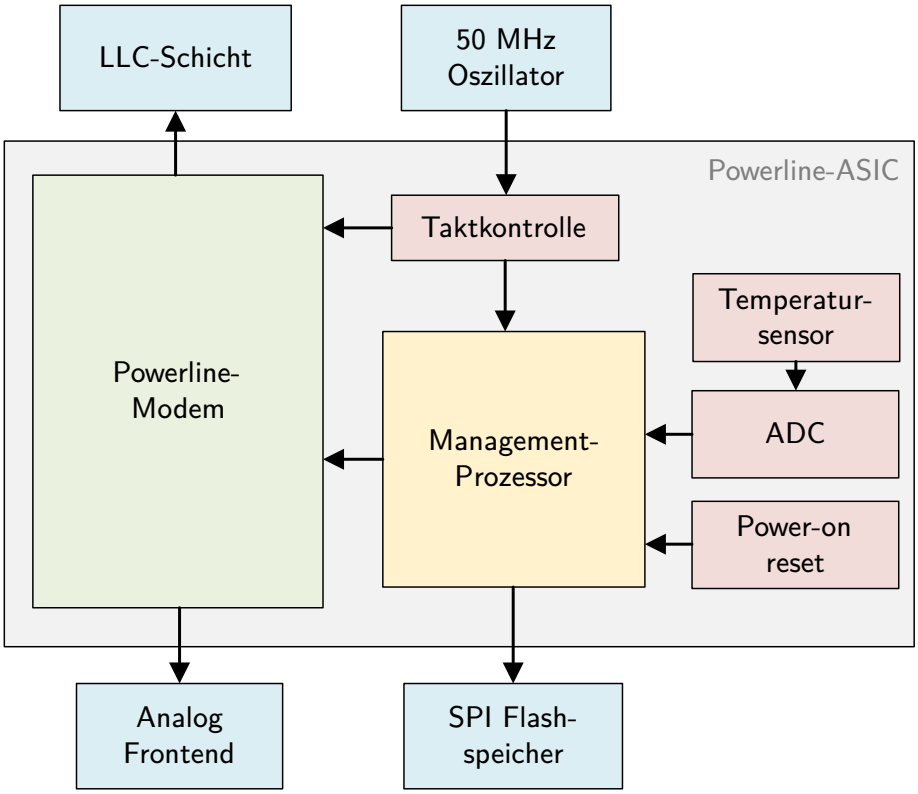

Abbildung 6.1: *Systemdesign des Powerline ASIC mit analogen Komponenten in rot, externen Komponenten in blau, dem Powerline-Modem in grün und einem Management-Prozessor in gelb. Die externen Komponenten sind nicht Teil des ASIC und dienen als Referenz für die Interfaces des Chips.*

Abschließend wurden das Gesamtsystem nochmals synthetisiert und verifiziert, um auszuschließen, dass der *Bootloader* und die Konfigurationsregister einen Einfluss auf den kritischen Pfad haben. Damit ist die Adaption des FPGA-Designs des Powerline-Modems vollständig und muss nun um die Ansteuerung externer Komponenten, eine Steuerung des Bootprozesses und externe Interfaces ergänzt werden. Dieses Gesamtdesign des Chips wird im folgenden Abschnitt vorgestellt.

6.2. Aufbau und Ergebnisse des Chip-Entwurfs

Das Powerline-Modem benötigt für dessen Ansteuerung und zum Booten unter anderem analoge Komponenten auf dem Chip und externe Komponenten, deren Ansteuerung ebenfalls im Powerline ASIC implementiert werden müssen. Abbildung 6.1 zeigt das Gesamtdesign des Powerline ASIC inklusive analoger Komponenten auf dem Chip (rot) und Anschlüssen zu externen Komponenten (blau). Zusätzlich wird ein Management Prozessor (MMP) verwendet, der die Bootsequenz steuert, die analogen Chipkomponenten abfragt und mit den externen Komponenten kommuniziert. Für den *Boot*-Prozess lädt der MMC den Programmcode aus dem externen *Flash*-Speichers über das *Bootloader*-Interface in die Instruktionsspeicher der Powerline-Modem-Prozessoren. Anschließend werden die Register konfiguriert und der Reset des Gesamtsystems deaktiviert. Ein *Power-on-Reset*-Modul erlaubt hierbei das deaktivieren des Resets erst, wenn alle benötigten Spannungsversorgungen stabil am ASIC anliegen. Zur Messung der Temperatur und eventueller

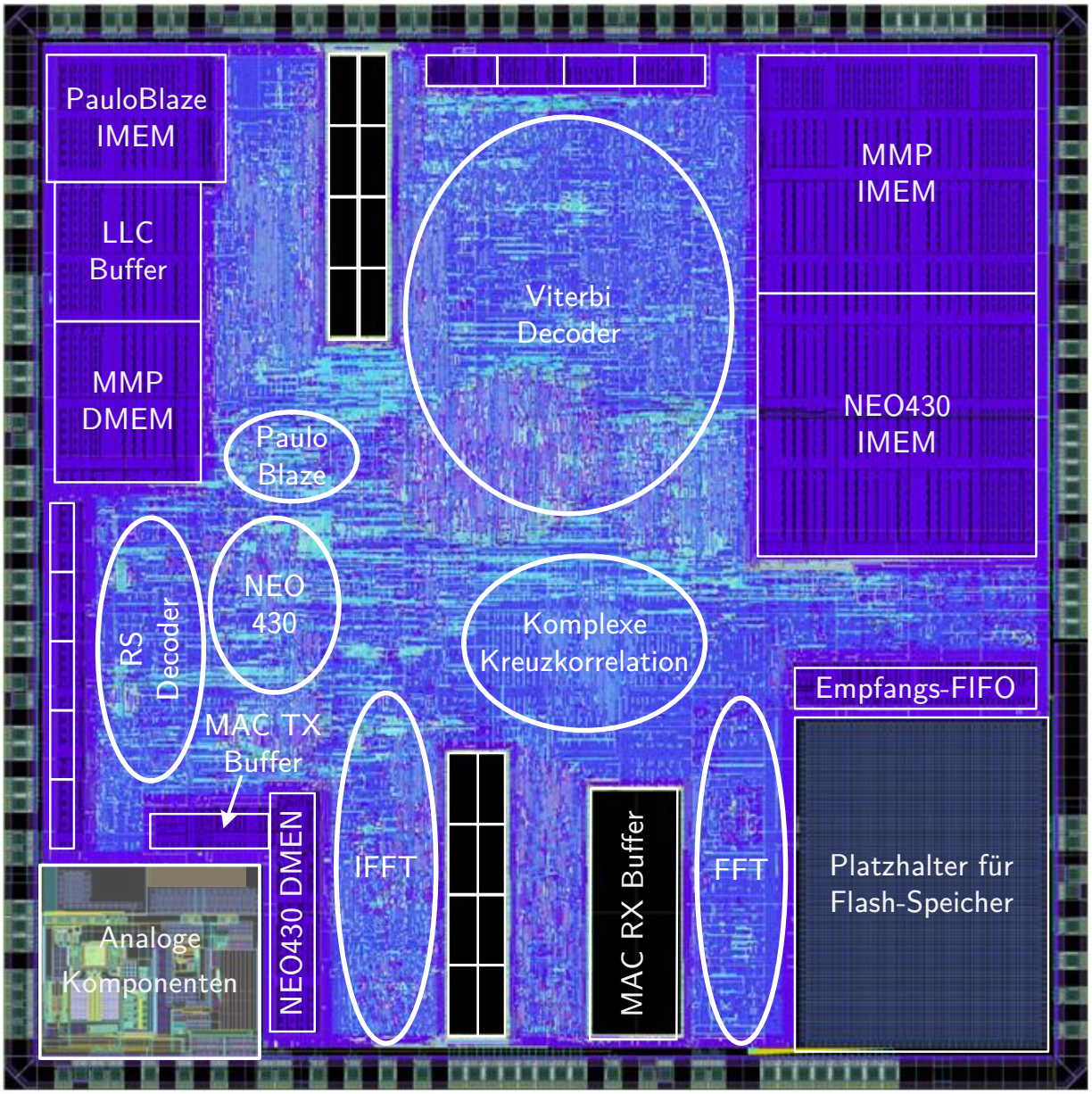

Abbildung 6.2: *Partitionierte Netzliste des gesamten ASICs nach Platzierung und Verdrahtung. Speicherblöcke sind als weiße Rechtecke und Logikblöcke als weiße Kreise in ungefährer Position und Größe angegeben. Unten links sind die analogen Komponenten aus Abbildung 6.1 gezeigt.*

Abschaltung des ASIC gibt es auf dem Chip einen Temperatursensor inklusive ADC, der vom Management Prozessor ausgelesen werden kann. Ein externer 50 MHz-Oszillator wird über ein Taktkontrollmodul angesprochen, das die zusätzlich benötigten Systemtakte von 25 MHz und 6,25 MHz abgeleitet und verteilt. Die beiden *Interfaces* zum LLC und AFE sind direkt an I/O-Pins des ASIC angeschlossen und werden später auf dem PCB an die entsprechenden ICs verdrahtet.

Das so entstandene Gesamtsystem des Powerline-ASIC wurde erneut verifiziert und synthetisiert und anschließend an eine externe Firma für das *Backend*-Design übergeben. Die daraus resultierende Netzliste nach Platzierung und Routing, die für die Fertigung des Chips verwendet wurde, ist in Abbildung 6.2 gezeigt. In den weißen Rechtecken sind die Speicherbausteine dargestellt, die als Platzhalter für die Integration im Fertigungsprozess ausgelegt sind. Diese

Tabelle 6.2: *Leistungsaufnahme des Powerline ASIC simuliert an der Netzliste nach Partitionierung und Routing für alle verfügbaren Bibliotheken der XFAB XT018-Technologie.*

	1,80 V	1,62 V					1,98 V	
	25 °C	−40 °C	85 °C	125 °C	150 °C	175 °C	−40 °C	0 °C
Boot (mW)	174	159	156	155	158	158	192	187
Idle (mW)	179	153	157	156	158	159	193	186
Full (mW)	237	208	211	211	214	215	263	258

sind überwiegend am Rand des Chips verteilt, um die Logik kompakt in der Mitte des Chips zu platzieren. Unten links sind die analogen Komponenten aus Abbildung 6.1 zu erkennen und unten rechts ein Platzhalter für die Integration eines Flash-Speichers. Dieser war zum Zeitpunkt der ASIC-Fertigung noch nicht für eine Temperatur von 175 °C verfügbar und kann in einer späteren Revision des Chips bei Verfügbarkeit eingebettet werden.

Die weißen Kreise zeigen die größten Logikblöcke der PHY-Schicht und die Prozessoren der MAC-Schicht. Die Größe der Kreise gibt hier nur das Verhältnis an und die Position die ungefähre Platzierung des Blocks. Zu erkennen ist, dass der Viterbi-Decoder der mit Abstand größte Block im Design ist, gefolgt von der komplexen Kreuzkorrelation im Decoder, dem RS-Decoder und der (I)FFT. Die Logik der Prozessoren ist deutlich kleiner als ihre Instruktions- und Datenspeicher.

Der Flächenbedarf des Chips ist mit einer quadratischen Grundfläche von $5{,}25\,\text{mm} \times 5{,}25\,\text{mm} = 27{,}25\,\text{mm}^2$. Die Fläche ohne I/O-Pads unterteilt sich in 48 % kombinatorische Logik, 47 % Speicher und 5 % Buffer für die Signalauffrischung. Der Chip hat 134 I/O-Pins und eine äquivalente Gatteranzahl von 1,52 Mio. NAND2-Gattern. Insgesamt ergibt sich eine Gatterdichte der Logikanteile von $72{,}7\,\text{kGE/mm}^2$.

Um die Leistungsaufnahme des gefertigten Chips abschätzen zu können, wurden Simulationen mit Hilfe von Cadence INCISIVE 15.2 auf der Netzliste aus Abbildung 6.2 durchgeführt. Hierfür wurde das Zeitverhalten und die Stromaufnahme der einzelnen Logikblöcke aus den XFAB-Bibliotheken zurück annotiert und die folgenden drei typischen Anwendungsfälle simuliert:

Boot: Startvorgang des gesamten Systems inklusive Laden des Instruktionscodes und Initialisieren der Prozessoren.

Idle: Kontinuierliches suchen von Präambeln auf dem Powerline-Kanal ohne senden und empfangen von Paketen.

Full: Maximale Auslastung des Modems bei kontinuierlichem Senden und Empfangen von Paketen.

Insgesamt werden von XFAB Bibliotheken mit drei verschiedenen Versorgungsspannungen bereitgestellt: die nominelle Spannung von 1,80 V und jeweils ±10 % Abweichung, also 1,62 V und 1,98 V. Für die nominelle Spannung existiert nur eine Bibliothek für eine Raumtemperatur von 25 °C als Referenz. Für die erhöhte Versorgungsspannung wird nur der spezifizierte niedrige Temperaturbereich von −40 °C und 0 °C bereitgestellt. Hierbei wird der Grenzfall der schnellsten Schaltgeschwindigkeit mit der höchsten Leistungsaufnahme abgebildet. Den größten Temperaturbereich von −40 °C bis 175 °C deckt die Versorgungsspannung von 1,62 V ab. Dabei

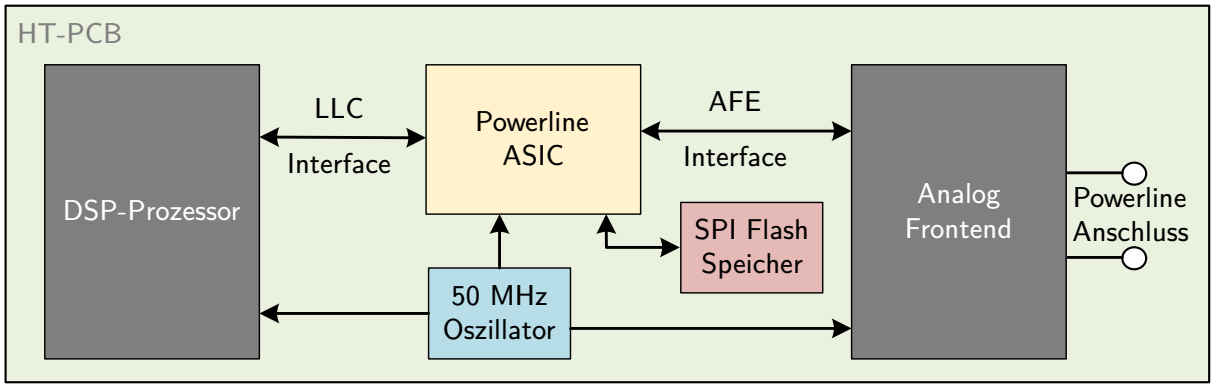

Abbildung 6.3: *Aufbau des hochtemperaturfesten PCB mit dem gefertigten Powerline-ASIC und seinen externen Komponenten. Rechts davon ist das AFE mit den Anschlüssen zur physikalischen Powerline gezeigt. Links davon ein DSP-Prozessor, der das LLC-Interface ansteuert und die Testapplikation ausführt.*

bildet eine Temperatur von 175 °C den langsamsten spezifizierten Grenzfall ab, da hier sowohl die Transistoren durch die geringere Versorgungsspannung langsamer schalten, als auch die Elektronenbeweglichkeit durch die hohe Temperatur am geringsten ist. An der nur minimalen Schwankung der Leistungsaufnahme über der Temperatur kann man, den in Kapitel 2 beschriebenen, Effekt der Kompensation vom Absinken der Schwellenspannung und der Reduktion der Ladungsträgerbeweglichkeit erkennen [Vee17].

Die Ergebnisse der Simulationen sind in Tabelle 6.2 gezeigt. Die geringste Leistungsaufnahme wird, wie zu erwarten bei der Versorgungsspannung von 1,62 V für den „Idle"-Zustand erreicht und beträgt 153 mW. Über den Temperaturbereich von −40 °C bis 175 °C beträgt die maximale Abweichung 6 mW bzw. 3,7 % und ist damit vernachlässigbar gering. Für den „Boot"-Vorgang werden vergleichbare Leistungswerte erreicht. Bei maximaler Auslastung des Modems im „Full"-Modus liegt die Leistungsaufnahme durchschnittlich 55 mW über dem „Idle"-Modus, verhält sich aber vergleichbar stabil über den spezifizierten Temperaturbereich. Mit steigender Versorgungsspannung nimmt die Leistungsaufnahme mit etwa 86,8 mW/V für alle drei Betriebsmodi zu. Das Maximum der Leistungsaufnahme liegt mit 263 mW bei 1,98 V und −40 °C im „Full"-Modus [Stu21].

Mit diesen Simulationsergebnissen kann die minimale und maximale Leistungsaufnahme des ASICs für das Design eines PCBs abgeschätzt werden. Zusammen mit den externen Komponenten, den Informationen über Größe und I/O-Pinpositionen und einer entsprechend dimensionierten Spannungsquelle, konnte das Test-PCB schon während der etwa vier- bis fünfmonatigen Chip-Fertigungsphase erstellt werden. Das Design dieses PCB und die anschließenden Hochtemperaturmessungen im Klimaschrank nach Erhalt und Kontaktierung des Chips auf dem PCB, werden im folgenden Abschnitt präsentiert.

6.3. Evaluation des ASICs

Der Aufbau eines hochtemperaturfesten PCB-Designs und die anschließende Evaluation des gefertigten ASICs werden in diesem Abschnitt beschrieben. Zunächst muss ein PCB erstellt werden, das mindestens den spezifizierten Temperaturen der XFAB XT018-Technologie, also −40 °C bis 175 °C, standhält. Auf diesem PCB sind der in Abschnitt 6.2 vorgestellte Chip, der

50 MHz-Oszillator, der *Flash*-Speicher, das Analog-Frontend und ein hochtemperaturfester DSP-Prozessor. Dieser DSP dient zur Ansteuerung des Powerline-Chips über das LLC-Interface und zur Ausführung der Messapplikation. Dieser Aufbau ist in Abbildung 6.3 schematisch dargestellt.

Als Trägermaterial für das PCB wird Polyimid verwendet, das je nach Ausprägung Temperaturen bis zu 300 °C standhalten kann [Gon20]. Der DSP-Prozessor steuert die Kommunikation mit dem Powerline-ASIC über das LLC-Interface und verwendet einen lwIP-Stack (siehe Abschnitt 5.2) zur Ausführung der TCP/IP-Kommunikation. Als Applikation wird das Programm *Ping*, das das *Internet Control Message Protocol* (ICMP) ausführt und die Latenz von Paketen und den Paketverlust messen kann [Rig14]. Das analoge Frontend ist digital direkt an das AFE-Interface des Powerline ASIC und analogseitig direkt an die Spannungsversorgungsleitung angeschlossen. Der 50 MHz-Oszillator versorgt alle drei Komponenten mit einem Referenztakt, der intern von den Komponenten geteilt oder vervielfacht wird. Der SPI Flashspeicher hat eine Größe von 2 MByte und ist direkt an das *Bootloader*-Interface des Management-Prozessors im ASIC angeschlossen. Alle Komponenten außer der Powerline-ASIC selbst wurden vorab für die Verwendung unter Hochtemperaturbedingungen im spezifizierten Bereich der XT018-Technologie qualifiziert. Außerdem sind alle Komponenten außer der Powerline-ASIC mit Gehäuse aufgebracht, so dass die Pins gelötet wurden. Der Powerline ASIC wurde direkt auf das PCB geklebt und mit Bonddrähten auf Pads des PCB gebondet.

Dieses PCB wird in den folgenden Abschnitten zur Evaluation des Powerline ASIC verwendet. Dazu wird zunächst in Abschnitt 6.3.1 die Selbsterwärmung des Chips bei Raumtemperatur gemessen und anschließend in Abschnitt 6.3.2 das aufgebaute PCB im Klimaschrank getestet.

6.3.1. Messung der Selbsterwärmung

Vor der Charakterisierung des Powerline-ASIC im Klimaschrank, wurde dessen Selbsterhitzung bei Raumtemperatur ermittelt. Dazu wurde eine FLIR T660 Wärmebildkamera mit einem Makroobjektiv mit 50 µs Pixelgröße in der resultierenden Wärmebildaufnahme verwendet. Bei einer Chipgröße von 5,25 mm × 5,25 mm ergibt sich eine Bildauflösung von 105×105 Pixel für den Chip selbst. Diese Messung soll Aufschluss über die maximale Selbsterwärmung und Temperaturverteilung des Chips geben. Der Maximalwert ist relevant, da das Anbringen eines Temperatursensors auf einem Chip ohne Gehäuse nur schwer möglich ist und somit nur die Umgebungstemperatur nahe dem Chip in den Klimaschrankmessungen ermittelt werden kann [Stu22].

Das linke Foto in Abbildung 6.4 zeigt die Temperaturverteilung des Chips im „Idle"-Zustand, bei dem das Powerline Modem kontinuierlich nach Präambeln von Paketen auf dem Übertragungskanal sucht. Die maximale Temperatur beträgt 43,9 °C im Bereich der komplexen Kreuzkorrelation, die die bekannte Präambel mit dem Empfangssignal korreliert. Die minimale Temperatur des Chips ist im Bereich der I/O-Pads zu finden und beträgt 43,2 °C. Damit weist der Chip eine sehr gleichmäßige Wärmeverteilung mit drei leicht wärmeren Regionen auf. Anhand der Netzliste aus Abbildung 6.2 lassen sich diese als der Viterbi-Decoder, die komplexe Kreuzkorrelation und die FFT identifizieren. Alle drei Komponenten sind Teil des PHY-Schicht-Decoders und die größten Komponenten im Empfangspfad. Insgesamt liegt die maximale Temperatur 22,9 °C und die minimale Temperatur 21,2 °C über der Raumtemperatur von 21 °C.

Der rechte Teil in Abbildung 6.4 zeigt den gleichen Bildausschnitt wie im „Idle"-Modus, hier aber unter maximaler Auslastung des Chips. Die Steigerung der Temperatur ist mit 0,7 °C sehr

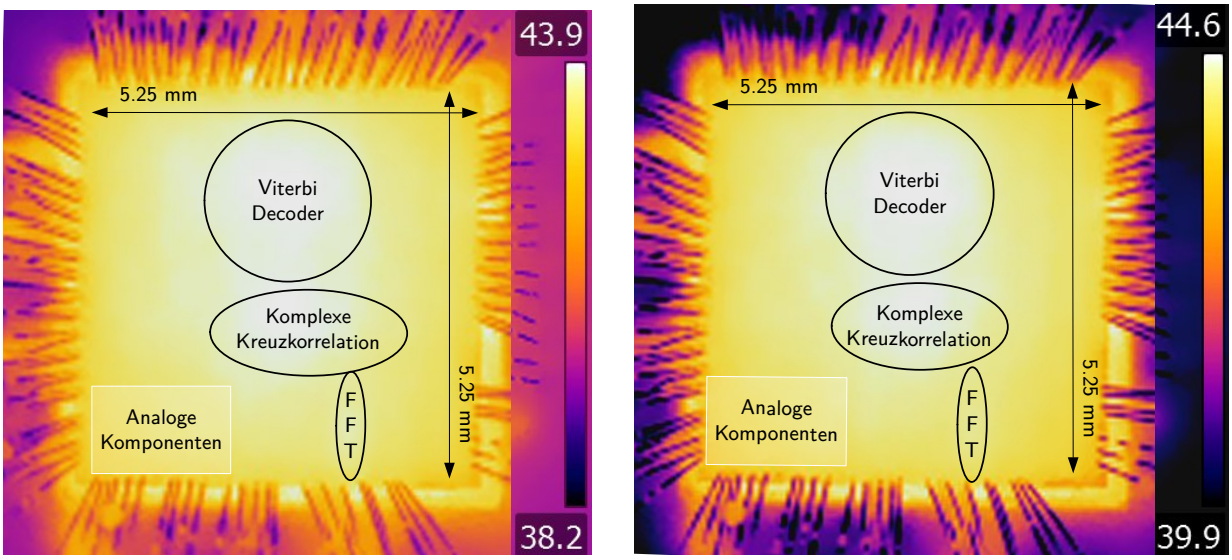

Abbildung 6.4: *Wärmebildaufnahmen des Powerline-ASIC mit einer FLIR T660 Wärmebildkamera mit Makroobjektiv. Links im „Idle"-Modus (kontinuierliche Suche nach Paketen auf dem Kanal) und rechts bei maximaler Auslastung des Chips. Die in Kreisen abgebildeten Verarbeitungsblöcke zeigen die Positionen größter Selbsterhitzung. Die analogen Komponenten dienen als Referenz der Orientierung des Chips verglichen mit der Netzliste in Abbildung 6.2.*

gering. Zusätzlich gibt es eine Verschiebung des maximalen Temperaturpunktes zum Viterbi Decoder. Wie auch im „Idle"-Modus sind die drei heißesten Bereichen der Viterbi Decoder, die komplexe Kreuzkorrelation und die FFT. Die minimale Temperatur des Chips ist weiterhin 43,2 °C. Auch hier ist die Verteilung der Temperatur, bis auf kleinere Schwankungen, sehr gleichmäßig über den gesamten Chip. In einer nächsten Revision des Chips kann durch die Verwendung von *Clock-* und *Power-Gating*, vor allem in den Bereichen der maximalen Temperatur, die Selbsterhitzung reduziert werden.

$$R_{\mathrm{J/A}} = \frac{T_J - T_A}{P_D} = \frac{43{,}9\,\mathrm{K} - 21\,\mathrm{K}}{200\,\mathrm{mW}} = 114{,}5\,\mathrm{K/W} \tag{6.1}$$

Aus der Messung im „Idle"-Zustand lässt sich der thermische Widerstand nach Gleichung 6.1 berechnen [Pau66]. Dabei ist T_J die Sperrschichttemperatur des ASIC, T_A die Umgebungstemperatur und P_D die Leistungsaufnahme des Chips.

$$\Delta T = T_J - T_A = R_{\mathrm{J/A}} P_D \tag{6.2}$$

Dieser Widerstandswert kann bei den nachfolgenden Messungen im Klimaschrank verwendet werden, um die Selbsterhitzung des Chips über die Umgebungstemperatur nach Gleichung 6.2 zu bestimmen. Hier ist ΔT die Selbsterhitzung des Chips und $R_{\mathrm{J/A}} = 114{,}5\,\mathrm{K/W}$ der thermische Widerstand.

6.3.2. Hochtemperaturmessungen im Klimaschrank

Für die Durchführung der Messungen wird ein Hochtemperatur-PCB innerhalb des Klimaschranks platziert und ein Maxim MAX2982 EVKIT (vorgestellt in Abschnitt 4.3.1) außerhalb, das von

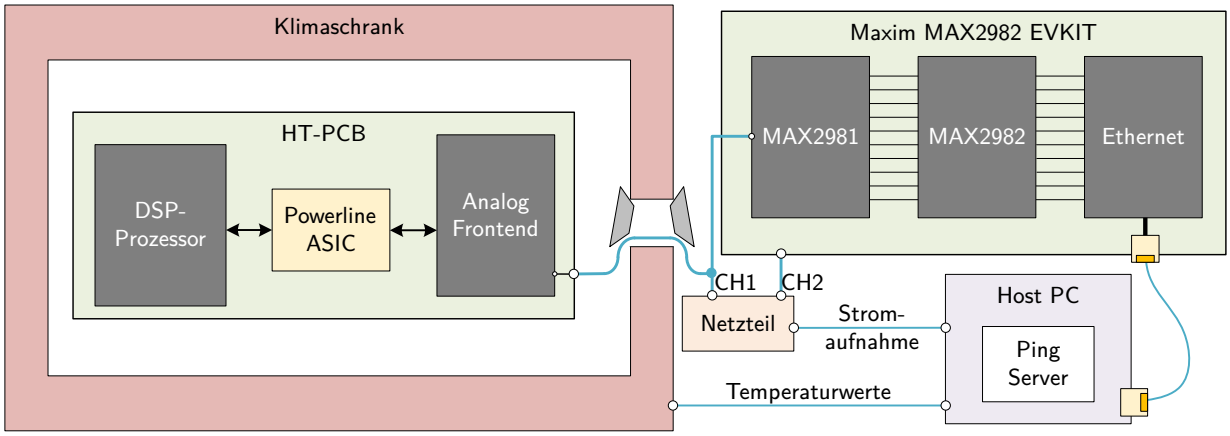

Abbildung 6.5: *Messaufbau der Klimaschrankmessung mit einem PCB innerhalb und einem MAX2982-Evalboard außerhalb der Klimakammer. Zusätzlich wird ein 2-Kanal-Netzteil und ein Host-PC zur Steuerung und Messung verwendet.*

einem Host-PC per *Ethernet*-Schnittstelle gesteuert wird. Dieser Aufbau ist schematisch in Abbildung 6.5 dargestellt. Auf dem Host-PC läuft ein Ping-Client, der mit dem Ping-Server auf dem DSP-Prozessor kommuniziert. Das PCB im Klimaschrank wird durch ein externes Netzteil mit Strom versorgt, das auch gleichzeitig die Kommunikationsleitung darstellt. Das Maxim MAX2982 EVKIT wird von einem galvanisch getrennten, zweiten Kanal desselben Netzteils versorgt, um die Stromaufnahme des zu testenden PCBs über den ersten Kanal isoliert bestimmen zu können. Zu dem Host-PC werden über eine serielle Schnittstelle des Netzteils auch die aktuellen Strom- und Spannungswerte übertragen.

Für die Messungen wurde die Temperatur im Bereich von −30 °C bis 170 °C in 10 °C-Schritten erhöht und ab 170 °C in 5 °C-Schritten bis zum Ausfall der Kommunikation. Zusätzlich wurde bei jeder Temperatur die Spannungsversorgung um ±10 % variiert, so dass mit den Spannungswerten 1,62 V, 1,80 V und 1,98 V gemessen wurde. An jedem Temperatur- und Spannungspunkt wurde die Stromaufnahme im „Idle“-Modus gemessen (vgl. Abschnitt 6.3.1) und anschließend 30 Messungen bei maximaler Auslastung der Powerline-Kommunikation durchgeführt, der im Folgenden als „Max“-Modus bezeichnet wird. Jede Messung im „Max“-Modus wird während der Ping-Kommunikation mit einer Paketgröße von 20 kByte ohne Wartezeit zwischen den Paketen durchgeführt. Damit wird sichergestellt, dass an den beiden Modems kontinuierlich Pakete zum Senden und Empfangen bereitstehen [Stu22].

Die Ergebnisse dieser Messungen sind in Abbildung 6.6 gezeigt. Aus Tabelle 6.2 sind die simulierten Leistungsaufnahmen als Kreuze und Kreise ergänzt, jeweils farblich passend codiert zu den Spannungswerten. Die gestrichelte Linie zeigt die Messwerte im „Idle“-Modus und die durchgehende Linie im „Max“-Modus, wo jeweils das Maximum aus den 30 Messungen gewählt wurde. Da der verwendete Klimaschrank nur Temperaturwerte bis −30 °C stabil halten kann, existieren keine Messungen unterhalb dieses Wertes. Die rechte Achse zeigt die geschätzte Selbsterhitzung nach Gleichung 6.2 unter der Annahme, dass der thermische Widerstand $R_{\mathrm{J/A}}$ sich über den gemessenen Temperaturbereich nicht ändert. Zusätzlich ist in grau der von der Technologie spezifizierte Temperaturbereich eingetragen. Die Differenz zwischen „Max“- und „Idle“-Modus ist für alle drei Betriebsspannungen gleichbleibend über die Temperatur und beträgt zwischen 15 mW bis 20 mW. Der Abstand zwischen den Messkurven verschiedener

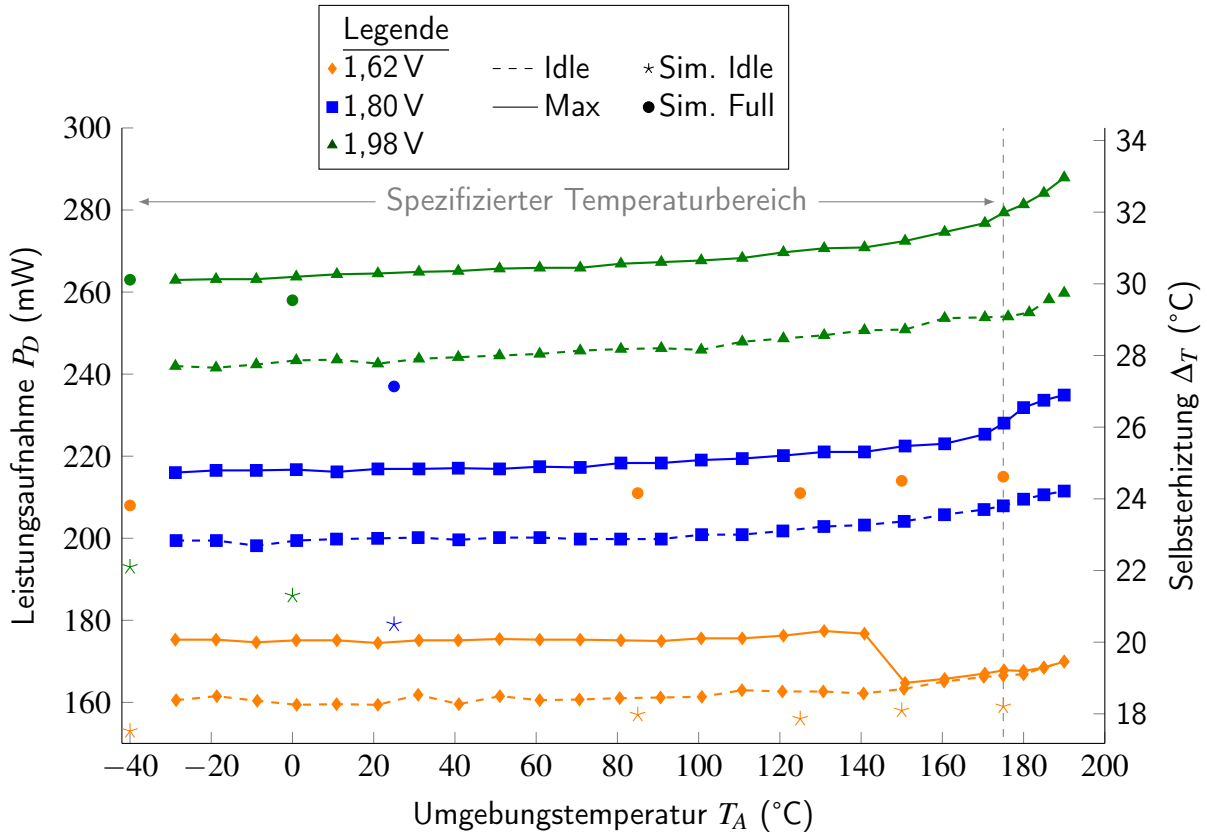

Abbildung 6.6: *Messergebnisse der Leistungsaufnahme des Powerline ASICs über die Umgebungstemperatur. Die orange Werte zeigen eine Spannungsversorgung von 1,62 V, die blauen Werte 1,80 V und die grünen Werte 1,98 V. Durchgezogene und gestrichelte Linien sind die Ergebnisse der Klimaschrankmessungen und Kreise und Sterne zeigen Simulationsergebnisse anhand der Netzliste nach Tabelle 6.2. Die rechte Ordinate zeigt die nach Gleichung 6.2 abgeschätzte Selbsterwärmung über den gesamten Temperaturbereich. Dabei beziehen sich die Messkurven gleichermaßen auf beide Ordinaten.*

Betriebsspannungen ist ebenfalls relativ konstant und beträgt 40 mW zwischen 1,62 V und 1,80 V und 45 mW zwischen 1,80 V und 1,98 V.

Durch eine Exponentialfunktion der Form $y(x) = a + be^{cx}$ kann der gesamte Verlauf der Kurven approximiert werden. Für die Kurve mit einer Versorgungsspannung von 1,98 V ergibt sich zum Beispiel $P_{max}(T) = 264\,\text{mW} + 0.32e^{0.02T}\,\text{mW}$ mit einem RMS-Fehler von nur 0,82 K. Für 1,62 V Betriebsspannung beträgt der Fehler 0,33 K für und 0,76 K für 1,80 V Betriebsspannung. Dieser exponentielle Anstieg der Leistungsaufnahme lässt sich unter anderem mit der Verdopplung der Leckströme bei Steigerung der Temperatur um 10 °C erklären [Sed09; Roy03].

Die maximale Leistungsaufnahme des Chips bis 175 °C beträgt 279,38 mW bei einer Selbsterhitzung von 32 °C. Bei der Spannungsversorgung 1,62 V bricht die Kommunikation bei maximaler Auslastung oberhalb von 150 °C ein und fällt auf die Werte des „Idle"-Modus. Eine mögliche Erklärung ist, dass die hohe Umgebungstemperatur zu einem höheren Spannungsabfall entlang der Stromversorgungsbahnen auf dem Chip führt. Vermutlich ist dieser ab 150 °C so groß, dass die verbleibende Betriebsspannung nicht mehr ausreicht, um die *Setup*-Zeiten der Register und Speicher am Ende dieser Bahnen einzuhalten. Eine mögliche Lösung für dieses Problem ist eine Erweiterung der Stromversorgung um mehr Bahnen, damit die Tiefe dieser Bahnen sinkt und

die Spannungsversorgung insgesamt stabiler ist. Der Nachteil dieser Korrektur ist ein erhöhter Flächenbedarf des Chips [Stu22].

Die Abschätzungen der Leistungsaufnahmen aus der Simulation unterschätzen den „Idle"-Modus und überschätzen den „Max"-Modus. Für den schlechtesten Fall von 1,62 V liegt die Abschätzung im „Idle"-Modus (orange Sterne) nur 6 mW unter den gemessenen Werten, für den „Max"-Modus (orange Kreise) aber 34 mW darüber. Bei nomineller Spannung von 1,80 V werden für Raumtemperatur (blauer Kreis und Stern) die Werte um jeweils 20 mW unter- und überschätzt. Bei erhöhter Spannung von 1,98 V liegen die simulierten „Idle"-Werte (grüne Sterne) sogar um 53 mW unter den Messwerten, aber die „Max"-Werte (grüne Kreise) stimmen fast mit den gemessenen überein. Eine Erklärung dieser Abweichungen sind die Ungenauigkeiten der realen Messungen, bei denen, anders als in der Simulation, nicht genau die Leistungsaufnahme einer Paketübertragung gemessen werden kann, sondern ein Mittelwert über mehrere hundert Millisekunden während einer Paketübertragung mitgeschnitten wird. Dennoch liefern die Werte der Simulation eine Ober- und eine Untergrenze der Leistungsaufnahme, die sich aber für die Dimensionierung einer Spannungsversorgungseinheit während der ASIC-Fertigung gut eignen.

Die maximale Temperatur, bei der eine Kommunikation ohne Verbindungsabbrüche zwischen den beiden Modems aufrechterhalten werden konnte, war 190 °C. Bei 195 °C brach die Kommunikation ab, konnte aber nach einer Abkühlphase und einem erneuten Heizen auf 190 °C wiederhergestellt werden. Auch die Verwendung von kleineren Paketen oder geringeren Datenraten hat keine Kommunikation bei 195 °C ermöglicht. Bei genauerer Untersuchung wurde festgestellt, dass die Kommunikation zwischen Powerline-Modem und DSP-Prozessor abbrach und das Modem keine Pakete mehr weitergab. Eine bestimmte Komponente, die diesen Ausfall verursacht, konnte mit dem bestehenden Messsystem nicht identifiziert werden [Stu22].

Neben der Umgebungstemperatur wurde auch die Paketfehlerrate der Ping-Kommunikation ermittelt. Abbildung 6.7 zeigt die relative Paketfehlerrate über die drei Betriebsspannungen aller Messdatenpunkte der Klimaschrankmessungen. Zur besseren Darstellung der Messdaten wurden alle Nullwerte durch einen Wert von 10^{-4} ersetzt, da diese in einem logarithmischen Plot ansonsten nicht dargestellt werden können. Zusätzlich ist die Anzahl der Messdatenpunkte pro Temperaturwert für alle drei Betriebsspannungen als graue gestrichelte Linien gegeben. Pro Temperaturschritt existieren bis zu 4100 Messwerte, was zur Bestimmung von Paketfehlerraten von minimal 10^{-2} ausreichend ist [Guy98].

Zu erkennen ist, dass mit steigender Temperatur die Anzahl übertragener Pakete abnimmt, da die Verbindung der Kommunikationsstrecke zeitweise einbricht. Geschieht dies, wird zunächst versucht mittels ARP-Paketen die Verbindung wieder aufzubauen, bevor das Ping-Protokoll weitere Pakete versendet. In dieser Zeit werden also keine Ping-Pakete übertragen und somit ist die Gesamtzahl der übertragenen Pakete für diesen Messpunkt geringer, was zu einer ungenaueren relativen Paketverlustrate führt. Dieser Effekt führt zum Beispiel dazu, dass eine relative Paketverlustrate von 50 % berechnet wird, wenn ein Paket erfolgreich und ein Paket nicht erfolgreich übertragen wird, die Verbindung aber ansonsten die ganze Zeit unterbrochen ist. Daher hat die relative Paketfehlerrate ab einer Paketanzahl von weniger als 1000 Paketen pro Messdatenpunkt nur eine begrenzte Aussagekraft. Dies bedeutet aber auch, dass nur noch ein Viertel aller möglichen Übertragungen in diesem Zeitpunkt stattfinden kann und das Kommunikationssystem möglichst nicht mit diesen Parametern betrieben werden sollte. Der Punkt des initialen Anstiegs der Paketverlustrate und die Unterschiede zwischen den

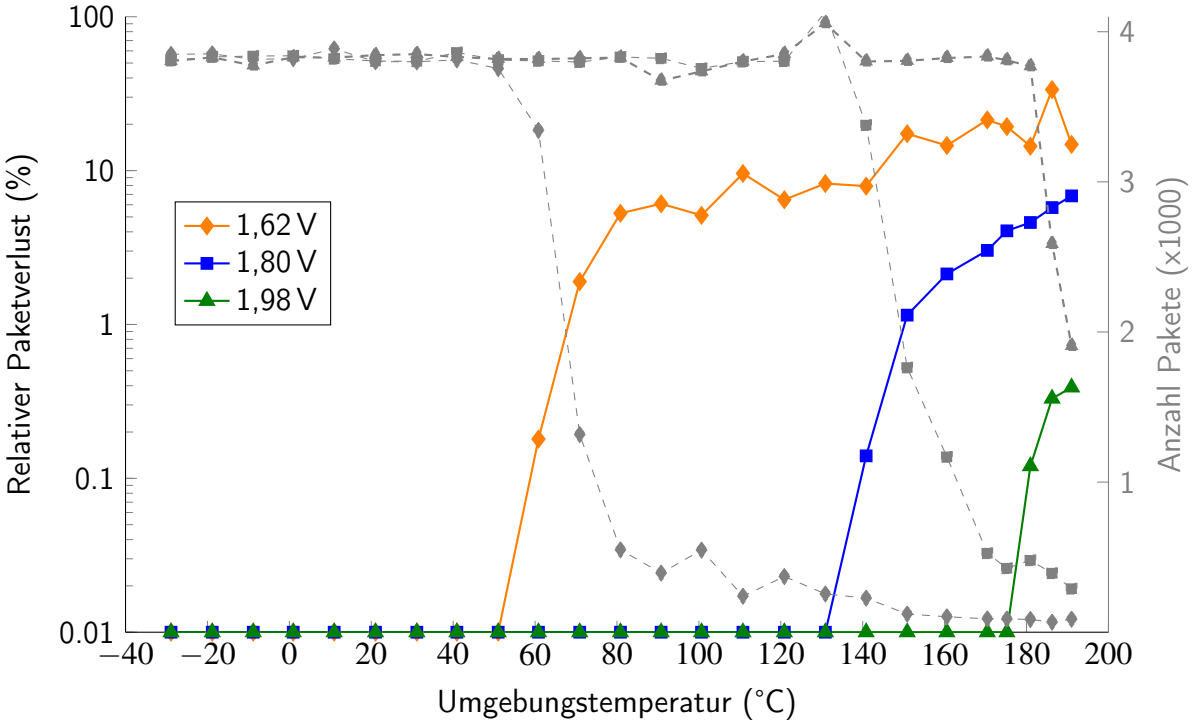

Abbildung 6.7: *Messung der Paketverlustrate über die Umgebungstemperatur. Alle Messpunkte bei denen keine Paketfehler auftraten wurden mit 10^{-4} ersetzt, um diese auf der logarithmischen Skala darstellen zu können. Die gestrichelten grauen Linien beziehen sich auf die rechte Ordinate, die die Anzahl versendeter Ping-Pakete pro Messdatenpunkt zeigt.*

Betriebsspannungen sind davon aber nicht betroffen und können anhand dieser Grafik gut verglichen werden.

Zu erkennen ist, dass bei einer reduzierten Betriebsspannung von 1,62 V die Paketfehlerrate bereits ab einer Temperatur von 60 °C ansteigt und ab 110 °C größer als 10 % ist. Dabei ist die Aussagekraft der Kurve oberhalb von 150 °C stark eingeschränkt, da dort die Verbindung einbricht (siehe Abbildung 6.6). Mit einer nominellen Spannung von 1,80 V steigt die Paketfehlerrate erst ab 140 °C an und beträgt 5 % bei 175 °C. Bei einer Betriebsspannung von 1,98 V sind innerhalb des spezifizierten Temperaturbereiches keine Pakete verloren gegangen und eine messbare Paketverlustrate tritt erst ab 180 °C mit einer Rate von 0,1 % auf. Bei der maximal erreichbaren Temperatur von 190 °C liegt dieser Wert bei 0,4 %.

Die Temperaturdifferenz der Kurven zwischen 1,80 V und 1,98 V beträgt 45 °C und zwischen 1,80 V und 1,62 V 80 °C. Unter der Annahme einer linearen Steigung der Paketverlustrate über die Erhöhung der Betriebsspannung, lässt sich für eine Temperaturerhöhung von 1 °C eine Spannungserhöhung von 4–7 mV ableiten, um eine identische Paketverlustrate zu erhalten.

6.4. Diskussion der Ergebnisse

Aus der Messung der Leistungsaufnahme in Abbildung 6.6 lassen sich zwei wesentliche Erkenntnisse für das Design eines Powerline-Kommunikationssystems für die Tiefbohrtechnik ableiten.

Zum einen kann bei einem gegebenen Leistungsbudget, das durch den Generator begrenzt wird, die maximale Anzahl an Kommunikationsknoten berechnet werden. Dabei gibt es einen Abtausch der maximalen Anzahl an Kommunikationsteilnehmern, die versorgt werden können und der Paketverlustrate (vgl. Abbildung 6.7). Wird eine reduzierte Versorgungsspannung von 1,62 V verwendet, kann mit einer Leistungsaufnahme von maximal 180 mW pro Knoten gerechnet werden. Dabei sollte die Umgebungstemperatur einen Wert von 80 °C nicht überschreiten, da sonst die Kommunikation sehr instabil wird und ab 140 °C sogar komplett abbricht. Durch die oben erwähnte Erweiterung der Stromversorgungsbahnen in einer nächsten Revision des Chips, könnte sich die maximale Betriebstemperatur noch deutlich vergrößern und die Paketverluste erst bei höheren Umgebungstemperaturen ansteigen.

Für eine Versorgungsspannung von 1,80 V kann pro Knoten eine maximale Leistungsaufnahme von maximal 230 mW angenommen werden und eine stabile Kommunikation bis zu einer Umgebungstemperatur von 175 °C, wo die Paketverlustrate auf 5 % ansteigt. Für einen Betrieb des Systems bei 1,98 V Versorgungsspannung kann das System bis 190 °C stabil kommunizieren und erreicht dabei eine maximale Leistungsaufnahme von 290 mW bei 0,4 % Paketverlust. Zusammenfassend lassen sich die folgenden Temperaturgrenzen und Leistungsaufnahmen für eine stabile Kommunikation bei den jeweiligen Betriebsspannungen ableiten:

- 1,62 V: Maximal 80 °C Umgebungstemperatur und 180 mW Leistungsaufnahme.
- 1,80 V: Maximal 175 °C Umgebungstemperatur und 230 mW Leistungsaufnahme.
- 1,98 V: Maximal 190 °C Umgebungstemperatur und 290 mW Leistungsaufnahme.

Mit diesen Erkenntnissen lässt sich ein Kommunikationssystem mit einem Leistungsbudget von 1 W für die Powerline-Kommunikation folgendermaßen entwerfen: entweder mit fünf Knoten bei 1,62 V und maximal 80 °C oder mit drei Knoten bei 1,98 V und bis 190 °C (bei Vernachlässigung des AFE und der externen Komponenten).

Zum anderen wird aus den Messungen in Abbildung 6.6 ersichtlich, dass die Zunahme der Leistungsaufnahme über den spezifizierten Temperaturbereich nur 5 % beträgt und maximal 10 % für den gesamten Messbereich bis 190 °C. Dies entspricht einer Steigung von 63 µW/K unterhalb von 150 °C und einem maximalen Anstieg von etwa 394 µW/K bei 190 °C über alle Versorgungsspannungen. Damit sind die Schwankungen über den Temperaturbereich zwar gering, die Einsparung von nur 11 % der Verlustleistungsaufnahme im „Idle“-Modus aber auch. Hier könnte in einer nächsten Version des Chips mit Hilfe von *Clock-Gating* für alle nicht aktiven Schaltungsteile die Leistungsaufnahme deutlich reduziert werden. Zum Beispiel könnten während des Suchens nach Paketen alle Encoderteile und die Datendecoder-Komponenten vom Takt abgeschaltet und letztere nur während der Paketdecodierung aktiviert werden. Dazu wäre ein zusätzlicher *Clock-Gating*-Controller notwendig, der die Latenzen jedes einzelnen Blockes kennt und die Schaltungsteile anhand der Zustände des Modems an- und abschaltet.

Weiterhin führt die konstant hohe Leistungsaufnahme zu einer hohen Selbsterhitzung des Chips, die bei nomineller Versorgungsspannung von 1,80 V im „Idle“-Modus bereits 23 °C beträgt. Betrachtet man die Schätzung der Selbsterhitzung nach Gleichung 6.2 bei der maximal erreichten Temperatur ergibt sich eine Sperrschichttemperatur von 223 °C. Damit erreicht der Chip eine um 48 °C höhere Maximaltemperatur als von der XFAB XT018-Technologie spezifiziert.

Die Selbsterhitzung von bis zu 33 °C resultiert vor allem aus dem hohen thermischen Widerstandswert von 114,5 K/W (vgl. Abschnitt 6.3.1). Dieser kann durch die Verwendung von Wärmeableitung, wie zum Beispiel mit einem Kühlkörper oder der Verwendung von PBC-Materialien mit

geringeren thermischen Widerständen, reduzieren werden [Guo13]. Wenn man den Widerstandswert beispielsweise auf 50 K/W absenken könnte, würde die Selbstherhitzung nur noch 14,4 °C bei 190 °C Umgebungstemperatur und 1,98 V Versorgungsspannung betragen. Bezogen auf die maximal erreichte Sperrschichttemperatur von 223 °C, wäre eine Kommunikation in einer Umgebungstemperatur von bis zu 208 °C möglich. Damit könnte das Powerline-Kommunikationssystem auch in Bohrtiefen mit über 200 °C eingesetzt werden, was deutlich über dem in dieser Arbeit gesetztem Ziel liegt.

In dieser Arbeit wurde der Chip mit dem Einsatzziel in der Tiefenbohrtechnik entworfen und evaluiert. Darüber hinaus lässt sich dieser aber auch in anderen Bereichen mit hoher thermischer Belastung einsetzen, wie zum Beispiel im Flugzeugen in der Nähe von Turbinen oder bei der Steuerung von Batterien in Elektrofahrzeugen. Weiterhin ist es möglich den Flächenbedarf des Chips zu reduzieren, indem für die Zielanwendung nicht relevante Teile entfernt werden. So kann zum Beispiel in der Tiefenbohrtechnik auf die Verschlüsselung verzichtet und dafür etwa 1 mm^2 an Fläche eingespart werden. Zusätzlich würde sich hierdurch der Chip schneller takten lassen, da sich der kritische Pfad in der Verschlüsslung befindet. Sollten die Umweltbedingungen dauerhaft extrem schlecht sein, ist es außerdem möglich das Powerline-Kommunikationssystem nur auf den ROBO-Modus zu reduzieren, was etwa ein Drittel der Fläche (und der Verlustleistung) einsparen würde.

Zusammenfassend könnten folgende Erkenntnisse gewonnen werden:

- Die maximal Selbsterhitzung des Chips beträgt 33 °C bei 1,98 V Versorgungsspannung und 190 °C Umgebungstemperatur.
- Die Leistungsaufnahme lässt sich mit Hilfe der Formel $P_{max}(T) = 264\,\text{mW} + 0.32e^{0.02T}\,\text{mW}$ mit hoher Genauigkeit approximieren.
- Die Paketverlustrate ist mit 0,4 % bei 1,98 V Versorgungsspannung und 190 °C Umgebungstemperatur sehr gering.
- Zur Kompensation steigender Paketverluste kann die Versorgungsspannung um 7 mW/°C erhöht werden.

KAPITEL 7

Zusammenfassung

Der Einsatz von elektronischen Bauteilen in der Tiefbohrtechnik bis zu 6 km tief unter der Erdoberfläche mit Temperaturen über 150 °C stellt erhöhte Anforderungen an deren Zuverlässigkeit, Stabilität und Langlebigkeit. In der Tiefbohrtechnik treten neben diesen umweltbedingten Einflüssen auch mechanische Schocks und Vibrationen durch das Zerspanen von Gestein auf. Die elektronischen Komponenten befinden sich entlang der unteren 100 m des Bohrstrangs und müssen autark Daten sammeln und auswerten, um Entscheidungen des weiteren Bohrverlaufs treffen zu können.

Das Ziel dieser Arbeit war es, die existierende Stromversorgungsleitung dieses Systems zu verwenden, um eine robuste Powerline-basierte Kommunikation zwischen den Komponenten zu entwerfen, zu implementieren und zu evaluieren. Dafür wurden zwei zentrale Forschungsfragen definiert. Die erste Forschungsfrage war nach der Durchführung einer Exploration verschiedener Hardware-Architekturen für die Integration eines Powerline-Standards in einem Tiefbohrsystem mit den oben genannten Umweltbedingungen. Der Fokus lag dabei auf der echtzeitfähigen Kommunikation des Systems mit möglichst geringem Flächenbedarf, einer geringen Leistungsaufnahme und einer hohen Nutzdatenrate. Die zweite Forschungsfrage befasste sich mit der Integration und Charakterisierung der performantesten Architektur aus der Hardware-Exploration als ASIC mit anschließender Hochtemperaturevaluation in einer Klimakammer. In dieser Klimakammer sollten die Umwelteinflüsse simuliert und parallel die Kommunikationsparameter des Systems evaluiert werden. Die Ergebnisse dieser beiden Forschungsfragen sollten Aufschluss über die Kommunikation unter extremen Umweltbedingungen geben und Erkenntnisse für das Design zukünftiger Kommunikationssysteme liefern.

Zuerst wurde ein Überblick über den aktuellen Stand der Forschung zu Powerline-Kommunikationsstandards erstellt und kommerziell erhältliche Hochtemperaturdesigns für den Einsatz in der Tiefbohrtechnik gesucht. Der Standard, der in dieser Arbeit verwendet wurde, ist der HomePlug 1.0-Standard der HomePlug Powerline Alliance mit einer Bruttodatenrate von 14 Mbit/s. Dieser verwendet unter anderem den robusten OFDM-Modus (ROBO), um mit Hilfe von extra Redundanzen eine Kommunikation bei stark gestörten Übertragungskanälen zu ermöglichen. Bei der

Definition der hochtemperaturfesten Hardware-Plattform für die echtzeitfähige Ausführung und Adaption dieses Standards wurde eine Literatur- und Marktanalyse existierender Plattformen durchgeführt. Da keines der existierenden Designs alle Anforderungen gleichzeitig erfüllen konnte, wurde entschieden eine eigene Plattform in einer Hochtemperatur-ASIC-Technologie zu entwerfen, zu verifizieren und fertigen zu lassen. Dafür wurde der aktuelle Stand der Technik dieser Technologien untersucht und die XFAB XT-018-Technologie als geeignetste ASIC-Technologie für diese Arbeit ausgewählt. Anschließend wurde der HomePlug 1.0-Standard detailliert vorgestellt und die nicht im Standard spezifizierte Hardware-Struktur des Decoders entworfen. Das Hauptaugenmerk lag hierbei auf dem Synchronisationssystem des Empfängers, das nach aktuellem Stand der Technik aufgebaut und verifiziert wurde.

Für die Entwurfsraumexploration wurde zunächst eine Referenzimplementierung des HomePlug 1.0-Standards erstellt. Dabei wurde die PHY-Schicht in MATLAB und die MAC-Schicht als C-Referenz implementiert. Mit Hilfe der MATLAB-Implementierung wurden zunächst die algorithmischen Freiheitsgrade der Modulationsart und der Quantisierung der Informationsbits über verschiedene Übertragungskanäle untersucht. Dabei wurden die unteren SNR-Grenzwerte der Modulationenarten bestimmt und die *Soft-Decision*-Demodulation mit 3 bit Genauigkeit als performanteste Möglichkeit der Fehlerkorrektur evaluiert. Danach wurde der Algorithmus der PHY-Schicht auf die RISC-V-basierte PULP-Plattform und verschiedene Cadence Tensilica LX7-ASIPs portiert. Der Programmcode wurde zunächst Software-seitig für die einzelnen Prozessoren adaptiert und anschließend mit Hardware-Erweiterungen beschleunigt. Als Pareto-optimale Punkte unter den Prozessor-basierten Designs wurden folgende Prozessoren ermittelt: der LX7 Fusion F1 ohne Hardware-Erweiterungen mit 14,10 kbit/s und 9,86 mm^2, die PULP-Plattform mit Viterbi-Decoder und Galois-Feld-MAC in Hardware mit 45,33 kbit/s und 11,31 mm^2 und der LX7 Hifi3z ASIP mit Viterbi-Decoder und Galois-Feld-MAC in Hardware mit 244 kbit/s und 12,23 mm^2. Da ab diesem Punkt eine weitere Erhöhung der Datenrate nur mit überproportional großem Hardware-Aufwand hätte erreicht werden können, wurde entschieden, die gesamte PHY-Schicht als dedizierte Hardware zu implementieren. Diese erreicht eine Datenrate von 14,5 Mbit/s, ist mit 8,13 mm^2 das kleinste Design in der XT018-Technologie und im Entwurfsraum über alle Architekturen das Optimum der evaluierten Architekturen. Alle genannten optimalen Designs lassen sich Standard-konform mit ihrer jeweiligen Datenrate betreiben und bilden im gewählten Entwurfsraum einen Abtausch der flexiblen Anpassung des Designs, der erreichbaren Datenrate, der Chipgröße und der Verlustleistung.

Die MAC-Schicht wurde auf Basis dieser Erkenntnisse der PHY-Schichtexploration direkt aus der C-Referenzimplementierung als dedizierte Hardware umgesetzt. Diese erreicht zwar die für die MAC-Schicht erforderlichen 10 Mbit/s, ist aber mit 6,40 mm^2 sehr groß und mit bis zu 420 Mbit/s in einigen Submodulen viel zu schnell. Daher wurden zwei Mikrocontroller als Steuerprozessoren für die Zustandsdiagramm-basierten Prozesse eingesetzt, um die Größe dieser Module und deren Zwischenspeicher zu reduzieren. Die restlichen Module wurden als Streaming-Architektur ohne Speicher für Pakete entworfen. Die resultierende MAC-Schicht-Implementierung hat eine Größe von 3,61 mm^2 und ist damit um 43 % kleiner als die direkte dedizierte Implementierung. Auch mit dieser Implementierung wurde die erforderliche Datenrate von 10 Mbit/s erreicht.

Basierend auf den Ergebnissen der MAC- und PHY-Schicht-Exploration wurde ein Gesamtkommunikationssystem erstellt, das in Emulation auf FPGAs evaluiert wurde. Hierfür wurde das National Instruments PXIe-System verwendet, um mit Hilfe der FPGA- und Peripherie-Module

eine Ende-zu-Ende-Kommunikation über zwei proprietären Homeplug-1.0-Modems aufzubauen. Die Messungen der Latenz eines Modems zeigten einen linearen Anstieg von 14,8 µs/Byte im ROBO-Modus und nur 0,4 µs/Byte mit DQPSK-Modualation und Punktierung. Die minimale Latenz lag bei 2,2 ms für den ROBO-Modus und bei 1,0 ms für die DQPSK-Modulation. Damit ergab sich ein Abtausch der Robustheit der Datencodierung gegenüber steigender Latenz durch die Wahl der Modulationsart. Die Messung des maximalen Durchsatzes einer UDP-basiert Kommunikation zeigen, dass für den ROBO-Modus und Modulation mit DBSPK der jeweils theoretisch maximale Durchsatz mit 0,56 Mbit/s und 2,60 Mbit/s erreicht wird. Für die DQPSK-Modulation ohne Punktierung wurde eine Datenrate von 4,51 Mbit/s erreicht, also nur 74 % des theoretischen Maximums. Die Reduzierung der MTU für die DQPSK-Modulation ohne Punktierung auf 1320 Byte erreichte eine Beschleunigung von 24 % auf 5,58 Mbit/s. Die DQPSK-Modulation mit Punktierung erreichte den höchsten Durchsatz mit 6,71 Mbit/s. Bei der TCP-Kommunikation konnte gezeigt werden, dass für Sensor-basierte periodische Paketübertragungen mit Paketgrößen kleiner als 1 kByte der ROBO-Modus am besten geeignet ist. Dieser erzielt hierbei vergleichbare Datenraten wie die DQPSK-Modulation und hat gleichzeitig die höchste Robustheit gegenüber Störungen auf dem Übertragungskanal. Abschließend wurde das Skalierungsverhalten von Powerline-Kommunikationsnetzen mit bis zu 60 Stationen in Emulation evaluiert. Dazu wurde auf dem Protium-S1-System die Latenz und der Paketverlust über die Anzahl der Kommunikationsteilnehmer und die Busauslastung gemessen. Das beste Skalierungsverhalten zeigte hierbei der ROBO-Modus, der zwar Latenzen von mindestens 25 ms aufweist, aber mit 60 Stationen bis zu einer Busauslastung von 100 % verlustfrei kommunizieren kann. Die beste Gesamtperformance hat die DQPSK-Modulation mit Punktierung bei einer mittleren Latenz von nur 2,5 ms. Diese kann ohne Paketverluste mit 60 Stationen bis 70 % Busauslatung und bis 90 % mit sporadischen Paketverlusten kommunizieren.

Nach der echtzeitfähigen Charakterisierung des Powerline-Gesamtsystems wurde dessen Steuerung durch ARM Cortex-M-Prozessoren mit einem Software-basierten TCP/IP-Stack (OSI-Schicht 3 und 4) untersucht. Das Ziel war, die Prozessoren ebenfalls in der XFAB XT018-Techologie zu synthetisieren, um auch eine anwendungsorientierte Ende-zu-Ende-Kommunikation über TCP oder UDP in einem Bohrstrangsystem zu realisieren. Dazu wurden die kleinsten Prozessoren der Corte-M-Reihe gewählt: der Cortex-M0 und der Cortex-M3. Für diese wurden jeweils der Leightweight-IP- und der Micro-IP-Stack kompiliert und die Funktionsweise auf einem FPGA verifiziert. Eine initiale Messung der Performance mit IPerf2 ergab eine maximale Datenrate von 20,29 Mbit/s für die Kombination aus Leightweight IP Stack und Cortex-M3-Prozessor. Durch Softwareoptimierungen im Micro-IP-Stack konnte dessen Durchsatz auf 25,31 Mbit/s auf dem Cortex-M3 gesteigert werden. Durch die Integration eines DMAs für das Kopieren der Sende- und Empfangspakete mit integrierter CRC-Berechnung in Hardware konnte die Ausführung beider Stacks auf bis zu 228,62 Mbit/s für den Micro-IP-Stack auf dem Cortex-M3 gesteigert werden. Abschließend wurde untersucht, welche Prozessorkonfiguration für die Steuerung des Powerline-Modems das geringste Produkt aus Fläche und Verlustleistung in der XT018-Technologie aufweist. Dafür wurde bei Verwendung des Micro-IP-Stacks die Taktfrequenz jeder Konfiguration so weit reduziert, dass diese die für die Powerline-Kommunikation notwendigen 10 Mbit/s erreicht. Hierbei zeigte sich, dass sich der Cortex-M0-Prozessor mit allen Hardware-Erweiterung am besten für diese Aufgabe eignet und dabei eine Fläche von 3,60 mm^2 und eine Leistungsaufnahme von nur 0,99 mW aufweist.

Zur Beantwortung der zweiten Forschungsfrage wurde das auf dem FPGA verifizierte Powerline-Kommunikationssystem als ASIC in der XT018-Technologie gefertigt und evaluiert. Dazu wurde

das Powerline-Modem um einen Steuerprozessor und analoge Komponenten erweitert und nach dessen Fertigung auf einer hochtemperaturfesten Leiterplatte aufgebaut. Zuerst wurde die Selbsterwärmung des Chips im „Idle“-Zustand und unter maximaler Auslastung gemessen. Diese betrug maximal 44,6 °C bei einer Raumtemperatur von 21 °C, womit sich ein thermischer Widerstand von 114,5 K/W ergab. Bei den anschließenden Messungen im Klimaschrank konnte dieser Wert genutzt werden, um eine Selbsterwärmung von maximal 33 °C bei der höchsten Leistungsaufnahme zu berechnen.

Für den von der Technologie spezifizierten Temperaturbereich von −40 °C bis 175 °C und nomineller Spannungsversorgung von 1,80 V wurde eine Leistungsaufnahme von 200 mW im „Idle“-Zustand und 222 mW bei maximaler Auslastung gemessen. Diese Schwankung über den gesamten gemessenen Temperaturbereich beträgt dabei nur 5 %. Die Differenz der Leistungsaufnahme zwischen „Idle“-Zustand und voller Auslastung beträgt nur etwa 10 %, was eine hohe Ruhestromaufnahme des Chips ergibt. Bei Variation der Versorgungsspannung um ±10 % ist eine Verschiebung der Leistungsaufnahme um ±45 mW für alle Temperaturwerte gleichermaßen zu erkennen. Für eine Versorgungsspannung von 1,62 V bricht die Kommunikation ab 150 °C ein, was wahrscheinlich an einer Spannungsunterversorgung der Komponenten am Ende der Stromversorgungsbahnen liegt. Dies kann in einer nächsten Revision des Chips korrigiert werden, indem die Tiefe der Bahnen reduziert wird.

Anschließend wurde auch die Paketverlustrate der Powerline-Kommunikation mit den gleichen Versorgungsspannungen und bis zu einer maximalen Temperatur von 190 °C bestimmt. Hierbei zeigte sich eine geringe Paketverlustrate für die nominelle Spannung von 1,80 V bis 140 °C. Ab dieser Temperatur steigt diese sukzessive auf 7 % bei 190 °C an. Bei einer erhöhten Spannungsversorgung von 1,98 V treten Paketverluste erst außerhalb des spezifizierten Temperaturbereichs auf und betragen nur 0,4 % bei 190 °C. Damit ergibt sich ein Abtausch der Kommunikationsstabilität gegenüber der Leistungsaufnahme pro Powerline-Modem. Dieser Abtausch bestimmt durch das festgelegte Energiebudget des Generators maßgeblich die Größe des realisierbaren Powerline-Kommunikationsnetzes.

Als letzte Auswertungen wurden auch die Leistungsaufnahmen oberhalb des spezifizierten Temperaturbereichs bis 190 °C bestimmt. Diese steigen exponentiell an, was unter anderem durch die exponentiell steigenden Leckströme über die Temperaturzunahme zu erklären ist. Die maximale Leistungsaufnahme bei voller Auslastung, 190 °C Umgebungstemperatur und 1,98 V Versorgungsspannung beträgt 288 mW. Durch Kombination der hochgerechneten Selbsterhitzung von 33 °C mit einer Umgebungstemperatur von 190 °C kann eine Sperrschichttemperatur von 223 °C geschätzt werden. Diese ist 48 °C über der maximal spezifizierten Temperatur und erlaubt einen Betrieb des Chips in eine Tiefe von 5,6 km in der Tiefbohrtechnik. Mit Hilfe eines Kühlkörpers ließe sich diese Sperrschichttemperatur absenken und das Powerline-Modem bei noch größeren Bohrtiefen einsetzen.

Mit Abschluss dieser Arbeit steht ein Entwurfsraum für die Implementierung von Powerline-Kommunikation für Hochtemperaturanwendungen zur Verfügung, der den aktuellen Stand der Forschung um Pareto-optimale Designs für den Einsatz unter extremen Umweltbedingungen erweitert. Außerdem existiert ein echtzeitfähiger, temperaturstabiler Forschungs-Chip, der für den Einsatz unter extremen Umweltbedingungen in der Tiefbohrtechnik verifiziert, charakterisiert und evaluiert wurde und ebenfalls den aktuellen Stand der Forschung in der Hochtemperaturelektronik erweitert.

Literaturverzeichnis

[Ahm14] I. Ahmad u. a. „Reliable Technology for Drilling Operations in a High-Pressure/High-Temperature Environment“. In: IADC/SPE Drilling Conference and Exhibition. OnePetro, 4. März 2014. DOI: 10.2118/167972-MS. URL: https://onepetro.org/SPEDC/proceedings/14DC/All-14DC/SPE-167972-MS/212764 (besucht am 27. 07. 2022).

[Al-03] Omar Al-Askary. „Iterative Decoding of Product Codes“. Doktorarbeit. Stockholm, Schweden: Kungl Tekniska Högskolan, Apr. 2003. 150 S. URL: http://kth.diva-portal.org/smash/get/diva2:7543/FULLTEXT01.pdf (besucht am 26. 04. 2021).

[And05] John B. Anderson. *Digital Transmission Engineering, 2nd Edition | Wiley*. IEEE Series on Digital & Mobile Communication. Wiley-IEEE Press, Juli 2005. 472 S. ISBN: 978-0-471-69464-9. URL: https://www.wiley.com/en-us/Digital+Transmission+Engineering%2C+2nd+Edition-p-9780471694649 (besucht am 24. 08. 2021).

[And12] André Neubauer. *DFT - Diskrete Fourier-Transformation*. 1. Aufl. Vieweg+Teubner Verlag Wiesbaden, 2012. 164 S. ISBN: 978-3-8348-1997-0. URL: https://doi.org/10.1007/978-3-8348-1997-0 (besucht am 20. 07. 2022).

[Bel94] Mihir Bellare, Joe Kilian und Phillip Rogaway. „The Security of Cipher Block Chaining“. In: *Advances in Cryptology — CRYPTO '94*. Hrsg. von Yvo G. Desmedt. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1994, S. 341–358. ISBN: 978-3-540-48658-9. DOI: 10.1007/3-540-48658-5_32.

[Ber13] Lars Torsten Berger, Andreas Schwager und J. Joaquín Escudero-Garzás. „Power Line Communications for Smart Grid Applications“. In: *Journal of Electrical and Computer Engineering* 2013 (28. März 2013). Publisher: Hindawi, e712376. ISSN: 2090-0147. DOI: 10.1155/2013/712376. URL: https://www.hindawi.com/journals/jece/2013/712376/ (besucht am 30. 04. 2021).

[Ber15] Lars T. Berger u. a. „MIMO Power Line Communications“. In: *IEEE Communications Surveys Tutorials* 17.1 (2015). Conference Name: IEEE Communications Surveys Tutorials, S. 106–124. ISSN: 1553-877X. DOI: 10.1109/COMST.2014.2339276.

[Ber19] Mouhammed Jandal Berro und Matthias Reich. „Laboratory investigations of a hybrid mud pulse telemetry (HMPT) – A new approach for speeding up the transmitting of MWD/LWD data in deep boreholes“. In: *Journal of Petroleum Science and Engineering* 183 (1. Dez. 2019), S. 106374. ISSN: 0920-4105. DOI: 10.1016/j.petrol.2019.106374. URL: https://www.sciencedirect.com/science/article/pii/S0920410519307958 (besucht am 27. 07. 2022).

[Ber72] E. R. Berlekamp. „A Survey of Coding Theory“. In: *Journal of the Royal Statistical Society. Series A (General)* 135.1 (1972). Publisher: [Royal Statistical Society, Wiley], S. 44–73. ISSN: 0035-9238. DOI: 10.2307/2345039. URL: https://www.jstor.org/stable/2345039 (besucht am 28. 06. 2021).

[Bio19] Valerio Bioglio, Carlo Condo und Ingmar Land. „Construction and Decoding of Product Codes with Non-Systematic Polar Codes“. In: *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. 2019 IEEE Wireless Communications and Networking Conference (WCNC). ISSN: 1558-2612. Apr. 2019, S. 1–6. DOI: 10.1109/WCNC.2019.8886100.

[Blu05] H. Blume, H. T. Feldkaemper und T. G. Noll. „Model-Based Exploration of the Design Space for Heterogeneous Systems on Chip“. In: *Journal of VLSI signal processing systems for signal, image and video technology* 40.1 (1. Mai 2005), S. 19–34. ISSN: 0922-5773. DOI: 10.1007/s11265-005-4936-4. URL: https://doi.org/10.1007/s11265-005-4936-4 (besucht am 25. 11. 2022).

[Blu08] Holger Blume. „Exploration des Entwurfsraumes für heterogene Architekturen zur digitalen Videosignalverarbeitung“. Habilitation. Aachen, Deutschland: RWTH Aachen, Feb. 2008.

[Bol21] Bollerott, Michael. *Hochtemperatur SOI-CMOS Technologie*. Frauenhofer IMS. 2021. URL: https://www.ims.fraunhofer.de/de/Geschaeftsfelder/High-Temperature-Electronics/Technologien/HT-SOI-CMOS.html (besucht am 26. 07. 2021).

[Bre20] A. Breitkopf. *Weltweiter Energiemix nach Energieträger 2018*. Statista. 11. Apr. 2020. URL: https://de.statista.com/statistik/daten/studie/167998/umfrage/weltweiter-energiemix-nach-energietraeger/ (besucht am 28. 04. 2021).

[Buc15] G Bucci u. a. „Power Line Communication, Overview of Standards and Applications“. In: *XXI IMEKO World Congress* (2015), S. 6.

[Bur01] William Burr. „Data Encryption Standard“. In: *Data Encryption Standard* (2001). Publisher: National Institute of Standards and Technology, S. 250–253. URL: https://csrc.nist.gov/publications/detail/book/2001/data-encryption-standard (besucht am 20. 08. 2021).

[Cad19] Cadence Design System. *Protium S1 FPGA-Based Prototyping Platform: The fastest way to prototype your ASIC*. 14. Aug. 2019. URL: https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/system-design-verification/protium-s1-fpga-based-prototyping-platform-ds.pdf (besucht am 01. 09. 2022).

[Can16] Cristina Cano u. a. „State of the Art in Power Line Communications: From the Applications to the Medium“. In: *IEEE Journal on Selected Areas in Communications* 34.7 (Juli 2016). Conference Name: IEEE Journal on Selected Areas in Communications, S. 1935–1952. ISSN: 1558-0008. DOI: 10.1109/JSAC.2016.2566018.

[Cha01] J.S. Chase, A.J. Gallatin und K.G. Yocum. „End system optimizations for high-speed TCP“. In: *IEEE Communications Magazine* 39.4 (Apr. 2001). Conference Name: IEEE Communications Magazine, S. 68–74. ISSN: 1558-1896. DOI: 10.1109/35.917506.

[Chi12] Tzi-Dar Chiueh, Pei-Yun Tsai und I-Wei Lai. *Baseband Receiver Design for Wireless MIMO-OFDM Communications: Chiueh/Baseband Receiver Design for Wireless MIMO-OFDM Communications*. Singapore: John Wiley & Sons Singapore Pte. Ltd., 25. Apr. 2012. ISBN: 978-1-118-18819-4. DOI: 10.1002/9781118188194. URL: http://doi.wiley.com/10.1002/9781118188194 (besucht am 08.07.2021).

[Chi18] Wilson Chin. *Measurement While Drilling: Signal Analysis, Optimization and Design, 2nd Edition | Wiley*. John Wiley & Sons, Ltd, Aug. 2018. 522 S. ISBN: 978-1-119-47930-7. URL: https://www.wiley.com/en-us/Measurement+While+Drilling%3A+Signal+Analysis%2C+Optimization+and+Design%2C+2nd+Edition-p-9781119479154 (besucht am 30.08.2021).

[Chi64] R. Chien. „Cyclic decoding procedures for Bose- Chaudhuri-Hocquenghem codes“. In: *IEEE Transactions on Information Theory* 10.4 (Okt. 1964). Conference Name: IEEE Transactions on Information Theory, S. 357–363. ISSN: 1557-9654. DOI: 10.1109/TIT.1964.1053699.

[Cho98] Philemon John Chose. „A VLSI synthesis of a Reed-Solomon processor for digital communication systems“. masters. Memorial University of Newfoundland, 1998. URL: https://research.library.mun.ca/1439/ (besucht am 12.10.2021).

[Cob64] R. S. C. Cobbold und F. N. Trofimenkoff. „Theory and application of the field-effect transistor. Part 1: Theory and d.c. characteristics“. In: *Proceedings of the Institution of Electrical Engineers* 111.12 (1. Dez. 1964). Publisher: IET Digital Library, S. 1981–1992. ISSN: 2053-7891. DOI: 10.1049/piee.1964.0324. URL: https://digital-library.theiet.org/content/journals/10.1049/piee.1964.0324 (besucht am 29.08.2022).

[Col89] O. Collins und F. Pollara. „Memory management in traceback Viterbi decoders“. In: *The Telecommunications and Data Acquisition Report* (15. Nov. 1989). NTRS Author Affiliations: Johns Hopkins Univ., Jet Propulsion Lab., California Inst. of Tech. NTRS Document ID: 19900010127 NTRS Research Center: Legacy CDMS (CDMS). URL: https://ntrs.nasa.gov/citations/19900010127 (besucht am 30.06.2021).

[Con16] Francesco Conti u. a. „PULP: A Ultra-Low Power Parallel Accelerator for Energy-Efficient and Flexible Embedded Vision“. In: *Journal of Signal Processing Systems* 84.3 (1. Sep. 2016), S. 339–354. ISSN: 1939-8115. DOI: 10.1007/s11265-015-1070-9. URL: https://doi.org/10.1007/s11265-015-1070-9 (besucht am 16.09.2021).

[Con99] L. Condra u. a. „Terminology for use of parts outside manufacturer-specified temperature ranges“. In: *IEEE Transactions on Components and Packaging Technologies* 22.3 (Sep. 1999). Conference Name: IEEE Transactions on Components and Packaging Technologies, S. 355–356. ISSN: 1557-9972. DOI: 10.1109/6144.796532.

[Das10] Apurba Das. „Digital Modulation Techniques“. In: *Digital Communication: Principles and System Modelling*. Hrsg. von Apurba Das. Signals and Communication Technology. Berlin, Heidelberg: Springer, 2010, S. 111–141. ISBN: 978-3-642-12743-4. DOI: 10.1007/978-3-642-12743-4_6. URL: https://doi.org/10.1007/978-3-642-12743-4_6 (besucht am 24. 08. 2021).

[Des09] Jean-Pierre Deschamps. *Hardware Implementation of Finite-Field Arithmetic*. 1. Aufl. USA: McGraw-Hill, Inc., 2009. 360 S. ISBN: 978-0-07-154581-5.

[Dos97] Dostert, Klaus M. *Telecommunications over the Power Distribution Grid - Possibilities and Limitations*. 1997. URL: http://rhythmetadyne.nutesla.com/wp-content/uploads/2016/07/Telecommunications-over-the-Power-Distribution-Grid-0563_001.pdf (besucht am 29. 04. 2021).

[Dun01] Adam Dunkels. *Design and Implementation of the lwIP TCP/IP Stack*. Swedish Institute of Computer Science, 20. Feb. 2001, S. 46. (Besucht am 16. 05. 2022).

[Dun03] Adam Dunkels. „Full TCP/IP for 8-bit architectures“. In: *Proceedings of the 1st international conference on Mobile systems, applications and services*. MobiSys '03. New York, NY, USA: Association for Computing Machinery, 5. Mai 2003, S. 85–98. ISBN: 978-1-4503-7797-3. DOI: 10.1145/1066116.1066118. URL: https://doi.org/10.1145/1066116.1066118 (besucht am 16. 05. 2022).

[Dun13] Adam Dunkels. *GitHub uIP-Project*. original-date: 2013-03-27T20:24:58Z. GitHub, 27. März 2013. URL: https://github.com/adamdunkels/uip (besucht am 30. 05. 2022).

[Ech07] Echelon Corporation. *PL 3120/ PL 3150 Power LineSmart Transceiver Data Book*. Manual 2. Echelon Corporation, 27. Sep. 2007. URL: https://web.archive.org/web/20070927005459/http://www.echelon.com/support/documentation/manuals/transceivers/005-0154-01D.pdf (besucht am 30. 04. 2021).

[Eli54] P. Elias. „Error-free Coding“. In: *Transactions of the IRE Professional Group on Information Theory* 4.4 (Sep. 1954). Conference Name: Transactions of the IRE Professional Group on Information Theory, S. 29–37. ISSN: 2168-2704. DOI: 10.1109/TIT.1954.1057464.

[Fal11] Kevin R. Fall und W. Richard Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*. 2. Aufl. Bd. 1. Upper Saddle River, NJ: Addison Wesley, 15. Nov. 2011. 1056 S. ISBN: 978-0-321-33631-6.

[Fet90] G. Fettweis und H. Meyr. „Cascaded feedforward architectures for parallel Viterbi decoding“. In: *1990 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1990 IEEE International Symposium on Circuits and Systems (ISCAS). Mai 1990, 978–981 vol.2. DOI: 10.1109/ISCAS.1990.112266.

[Fet91] G. Fettweis und H. Meyr. „High-speed parallel Viterbi decoding: algorithm and VLSI-architecture“. In: *IEEE Communications Magazine* 29.5 (Mai 1991). Conference Name: IEEE Communications Magazine, S. 46–55. ISSN: 1558-1896. DOI: 10.1109/35.79382.

[Flo05] A. Flocke, H. Blume und T. G. Noll. „Implementation and modeling of parametrizable high-speed Reed Solomon decoders on FPGAs“. In: *Advances in Radio Science* 3 (12. Mai 2005), S. 271–276. ISSN: 1684-9973. DOI: 10.5194/ars-3-271-2005. URL: https://ars.copernicus.org/articles/3/271/2005/ (besucht am 25. 06. 2021).

[Fly70] M.J. Flynn. „On Division by Functional Iteration“. In: *IEEE Transactions on Computers* C-19.8 (Aug. 1970). Conference Name: IEEE Transactions on Computers, S. 702–706. ISSN: 1557-9956. DOI: 10.1109/T-C.1970.223019.

[For65] G. Forney. „On decoding BCH codes“. In: *IEEE Transactions on Information Theory* 11.4 (Okt. 1965). Conference Name: IEEE Transactions on Information Theory, S. 549–557. ISSN: 1557-9654. DOI: 10.1109/TIT.1965.1053825.

[For73] G.D. Forney. „The viterbi algorithm“. In: *Proceedings of the IEEE* 61.3 (März 1973). Conference Name: Proceedings of the IEEE, S. 268–278. ISSN: 1558-2256. DOI: 10.1109/PROC.1973.9030.

[Fou02] Free Software Foundation. *lwIP - A Lightweight TCP/IP stack*. 17. Okt. 2002. URL: https://savannah.nongnu.org/projects/lwip/ (besucht am 25. 05. 2022).

[Fra14] Nicholas G. Franconi u. a. „Wireless Communication in Oil and Gas Wells“. In: *Energy Technology* 2.12 (2014), S. 996–1005. ISSN: 2194-4296. DOI: 10.1002/ente.201402067. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/ente.201402067 (besucht am 30. 08. 2021).

[Gau17] Michael Gautschi u. a. „Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices“. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25.10 (Okt. 2017). Conference Name: IEEE Transactions on Very Large Scale Integration (VLSI) Systems, S. 2700–2713. ISSN: 1557-9999. DOI: 10.1109/TVLSI.2017.2654506.

[Gem01] T. Gemmeke, V.S. Gierenz und T.G. Noll. „Scalable, power and area efficient high throughput Viterbi decoder implementations“. In: *Proceedings of the 27th European Solid-State Circuits Conference*. Proceedings of the 27th European Solid-State Circuits Conference. Sep. 2001, S. 474–477.

[Gen21] Paul R. Genssler. *pauloBlaze*. original-date: 2015-07-28T15:00:09Z. 10. Dez. 2021. URL: https://github.com/krabo0om/pauloBlaze (besucht am 12. 05. 2022).

[Ges19] Sven Gesper u. a. „Evaluation of Different Processor Architecture Organizations for On-site Electronics in Harsh Environments“. In: *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Hrsg. von Dionisios N. Pnevmatikatos, Maxime Pelcat und Matthias Jung. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, S. 3–17. ISBN: 978-3-030-27562-4. DOI: 10.1007/978-3-030-27562-4_1.

[Ges21] Sven Gesper u. a. „Evaluation of Different Processor Architecture Organizations for On-Site Electronics in Harsh Environments“. In: *International Journal of Parallel Programming* 49.4 (1. Aug. 2021), S. 541–569. ISSN: 1573-7640. DOI: 10.1007/s10766-020-00686-8. URL: https://doi.org/10.1007/s10766-020-00686-8 (besucht am 24. 07. 2022).

[Gi 94] Byeong Gi Lee und Seok Chang Kim. „Fundamentals of Scrambling Techniques“. In: *Scrambling Techniques for Digital Transmission*. Hrsg. von Byeong Gi Lee und Seok Chang Kim. Telecommunication Networks and Computer Systems. London: Springer, 1994, S. 13–24. ISBN: 978-1-4471-3231-8. DOI: 10.1007/978-1-4471-3231-8_2. URL: https://doi.org/10.1007/978-1-4471-3231-8_2 (besucht am 20. 07. 2022).

[Gon18] Benito González, Raúl Rodríguez und Antonio Lázaro. „Thermal Resistance Characterization for Multifinger SOI-MOSFETs“. In: *IEEE Transactions on Electron Devices* 65.9 (Sep. 2018). Conference Name: IEEE Transactions on Electron Devices, S. 3626–3632. ISSN: 1557-9646. DOI: 10.1109/TED.2018.2853799.

[Gon20] Peter Gong. *What Is Polyimide PCB and Polyimide PCB Buying Guide*. FX PCB. 31. Aug. 2020. URL: https://sfxpcb.com/what-is-polyimide-pcb/ (besucht am 02. 06. 2022).

[Goo19] Chinthaka P. Gooneratne u. a. „Downhole Communication and Power Supplies to Instruments and Communication Modules“. In: *Instruments, Measurement Principles and Communication Technologies for Downhole Drilling Environments*. Hrsg. von Chinthaka P. Gooneratne u. a. Smart Sensors, Measurement and Instrumentation. Cham: Springer International Publishing, 2019, S. 97–109. ISBN: 978-3-030-04900-3. DOI: 10.1007/978-3-030-04900-3_5. URL: https://doi.org/10.1007/978-3-030-04900-3_5 (besucht am 27. 07. 2022).

[Got04] M. Gotz, M. Rapp und K. Dostert. „Power line channel characteristics and their effect on communication system design“. In: *IEEE Communications Magazine* 42.4 (Apr. 2004). Conference Name: IEEE Communications Magazine, S. 78–86. ISSN: 1558-1896. DOI: 10.1109/MCOM.2004.1284933.

[Guo13] Guosheng Jiang, Liyong Diao und Ken Kuang. *Advanced Thermal Management Materials*. 1. Aufl. Springer New York, NY, 2013. 156 S. ISBN: 978-1-4614-1962-4. URL: https://link.springer.com/book/10.1007/978-1-4614-1963-1 (besucht am 05. 07. 2022).

[Guy98] I. Guyon u. a. „What size test set gives good error rate estimates?“ In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.1 (Jan. 1998). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, S. 52–64. ISSN: 1939-3539. DOI: 10.1109/34.655649.

[Hab10] Irfan Habib, Özgün Paker und Sergei Sawitzki. „Design Space Exploration of Hard-Decision Viterbi Decoding: Algorithm and VLSI Implementation“. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 18.5 (Mai 2010). Conference Name: IEEE Transactions on Very Large Scale Integration (VLSI) Systems, S. 794–807. ISSN: 1557-9999. DOI: 10.1109/TVLSI.2009.2017024.

[Hae07] Thomas Haenselmann u. a. „Scriptable Sensor Network Based Home-Automation“. In: *Emerging Directions in Embedded and Ubiquitous Computing*. Hrsg. von Mieso K. Denko u. a. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2007, S. 579–591. ISBN: 978-3-540-77090-9. DOI: 10.1007/978-3-540-77090-9_54.

[Hag89] J. Hagenauer und P. Hoeher. „A Viterbi algorithm with soft-decision outputs and its applications“. In: *1989 IEEE Global Telecommunications Conference and Exhibition 'Communications Technology for the 1990s and Beyond'*. 1989 IEEE Global Telecommunications Conference and Exhibition 'Communications Technology for the 1990s and Beyond'. Nov. 1989, 1680–1686 vol.3. DOI: 10.1109/GLOCOM.1989.64230.

[Ham50] R. W. Hamming. „Error detecting and error correcting codes“. In: *The Bell System Technical Journal* 29.2 (Apr. 1950). Conference Name: The Bell System Technical Journal, S. 147–160. ISSN: 0005-8580. DOI: 10.1002/j.1538-7305.1950.tb00463.x.

[Hao15] Y. Hao u. a. „A 0.18μm SOI BCD technology for automotive application“. In: *2015 IEEE 27th International Symposium on Power Semiconductor Devices IC's (ISPSD)*. 2015 IEEE 27th International Symposium on Power Semiconductor Devices IC's (ISPSD). ISSN: 1946-0201. Mai 2015, S. 177–180. DOI: 10.1109/ISPSD.2015.7123418.

[He96] Shousheng He und M. Torkelson. „A new approach to pipeline FFT processor“. In: *Proceedings of International Conference on Parallel Processing*. Proceedings of International Conference on Parallel Processing. Apr. 1996, S. 766–770. DOI: 10.1109/IPPS.1996.508145.

[Hel71] J. Heller und I. Jacobs. „Viterbi Decoding for Satellite and Space Communication“. In: *IEEE Transactions on Communication Technology* 19.5 (Okt. 1971). Conference Name: IEEE Transactions on Communication Technology, S. 835–848. ISSN: 2162-2175. DOI: 10.1109/TCOM.1971.1090711.

[Hoc11] Martin Hoch. „Comparison of PLC G3 and PRIME“. In: *2011 IEEE International Symposium on Power Line Communications and Its Applications*. 2011 IEEE International Symposium on Power Line Communications and Its Applications. Apr. 2011, S. 165–169. DOI: 10.1109/ISPLC.2011.5764384.

[Hoh20] M. Hohmann. *Weltweite Energienachfrage: Entwicklung bis 2050*. Statista. 15. Sep. 2020. URL: https://de.statista.com/statistik/daten/studie/258258/umfrage/entwicklung-der-weltweiten-energienachfrage/ (besucht am 28. 04. 2021).

[Hom08] HomePlug Alliance. *TIA-1113 : Medium-Speed (up to 14 Mbps) Power Line Communications (PLC) Modems using Windowed OFDM*. 1. Mai 2008. URL: https://global.ihs.com/doc_detail.cfm?document_name=TIA-1113 (besucht am 05. 05. 2021).

[Hon21] Honeywell. *Extreme Design: Developing integrated circuits for -55 degC to +250 degC*. White Paper. Honeywell, 15. Juni 2021, S. 1–7. URL: https://aerospace.honeywell.com/content/dam/aerobt/en/documents/learn/products/microelectronics/technical-articles/ExtremeDesignDevelopingIntegratedCircuitsFor55DegCTo250DegC.pdf (besucht am 27. 07. 2021).

[Hsi09] Yi-Mao Hsiao u. a. „High speed UDP/IP ASIC design“. In: *2009 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*. 2009 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS). Jan. 2009, S. 405–408. DOI: 10.1109/ISPACS.2009.5383815.

[Hsi98] Meng-Han Hsieh und Che-Ho Wei. „Channel estimation for OFDM systems based on comb-type pilot arrangement in frequency selective fading channels“. In: *IEEE Transactions on Consumer Electronics* 44.1 (Feb. 1998). Conference Name: IEEE Transactions on Consumer Electronics, S. 217–225. ISSN: 1558-4127. DOI: 10.1109/30.663750.

[Jim13] Jim LeClare. *An Overview, History, and Formation of IEEE P1901.2 for Narrowband OFDM PLC*. Application Note 5676. 7. Feb. 2013, S. 7. URL: https://www.maximintegrated.com/en/design/technical-documents/app-notes/5/5676.html (besucht am 12. 07. 2022).

[Joe93] O.J. Joeressen, M. Vaupel und H. Meyr. „Soft-output Viterbi decoding: VLSI implementation issues“. In: *IEEE 43rd Vehicular Technology Conference*. IEEE 43rd Vehicular Technology Conference. ISSN: 1090-3038. Mai 1993, S. 941–944. DOI: 10.1109/VETEC.1993.510966.

[Jon22] Jon Dugan u. a. *iPerf - iPerf3 and iPerf2 user documentation*. iPerf - The ultimate speed test tool for TCP, UDP and SCTP. 25. Mai 2022. URL: https://iperf.fr/iperf-doc.php (besucht am 25. 05. 2022).

[Kal14] Amit A. Kale u. a. *A Probabilistic Approach for Reliability and Life Prediction of Electronics in Drilling and Evaluation Tools*. Section: Technical Reports. Baker Hughes Incorporated Houston United States, 2. Okt. 2014. URL: https://apps.dtic.mil/sti/citations/AD1002788 (besucht am 30. 08. 2021).

[Kel01] T. Keller u. a. „Orthogonal frequency division multiplex synchronization techniques for frequency-selective fading channels“. In: *IEEE Journal on Selected Areas in Communications* 19.6 (Juni 2001). Conference Name: IEEE Journal on Selected Areas in Communications, S. 999–1008. ISSN: 1558-0008. DOI: 10.1109/49.926356.

[Kir20] Lucas Kirschbaum u. a. „AI-Driven Maintenance Support for Downhole Tools and Electronics Operated in Dynamic Drilling Environments“. In: *IEEE Access* 8 (2020). Conference Name: IEEE Access, S. 78683–78701. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2990152.

[Kri96] A.V. Krishnamoorthy und D.A.B. Miller. „Scaling optoelectronic-VLSI circuits into the 21st century: a technology roadmap“. In: *IEEE Journal of Selected Topics in Quantum Electronics* 2.1 (Apr. 1996). Conference Name: IEEE Journal of Selected Topics in Quantum Electronics, S. 55–76. ISSN: 1558-4542. DOI: 10.1109/2944.541875.

[Kum05] A. Kumar und S. Sawitzki. „High-Throughput and Low-Power Architectures for Reed Solomon Decoder“. In: *Conference Record of the Thirty-Ninth Asilomar Conference onSignals, Systems and Computers, 2005.* Conference Record of the Thirty-Ninth Asilomar Conference onSignals, Systems and Computers, 2005. Pacific Grove, California: IEEE, 2005, S. 990–994. ISBN: 978-1-4244-0131-4. DOI: 10.1109/ACSSC.2005.1599906. URL: http://ieeexplore.ieee.org/document/1599906/ (besucht am 25. 06. 2021).

[Lee03] M. K. Lee u. a. „HomePlug 1.0 powerline communication LANs—protocol description and performance results“. In: *International Journal of Communication Systems* 16.5 (2003). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/dac.601, S. 447–473. ISSN: 1099-1131. DOI: https://doi.org/10.1002/dac.601. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.601 (besucht am 29. 04. 2021).

[Lee04] Chanho Lee. „A Viterbi Decoder with Efficient Memory Management“. In: *ETRI Journal* 26.1 (2004), S. 21–26. ISSN: 2233-7326. DOI: 10.4218/etrij.04.0303.0009. URL: https://onlinelibrary.wiley.com/doi/abs/10.4218/etrij.04.0303.0009 (besucht am 21. 02. 2022).

[Lim06a] ARM Limited. *AMBA 3 AHB-Lite Protocol Specification*. 6. Juni 2006. URL: https://developer.arm.com/documentation/ihi0033/a/Introduction/About-the-protocol (besucht am 13. 06. 2022).

[Lim06b] ARM Limited. *ARMv7-M Architecture Reference Manual*. Juni 2006. URL: https://developer.arm.com/documentation/ddi0403/ee?lang=en (besucht am 17. 05. 2022).

[Lim07a] ARM Limited. *ARMv6-M Architecture Reference Manual*. März 2007. URL: https://developer.arm.com/documentation/ddi0419/c/ (besucht am 17. 05. 2022).

[Lim07b] ARM Limited. *Cortex-M1*. Cortex-M1. 2007. URL: https://developer.arm.com/Processors/Cortex-M1 (besucht am 16. 05. 2022).

[Lim09a] ARM Limited. *Cortex-M0*. Cortex-M0. 2009. URL: https://developer.arm.com/Processors/Cortex-M0#Technical-Specifications (besucht am 16. 05. 2022).

[Lim09b] ARM Limited. *Cortex-M0+*. Cortex-M0+. 2009. URL: https://developer.arm.com/Processors/Cortex-M0-Plus (besucht am 16. 05. 2022).

[Lim09c] ARM Limited. *Cortex-M4*. Cortex-M4. 2009. URL: https://developer.arm.com/Processors/Cortex-M4 (besucht am 16. 05. 2022).

[Lim10] ARM Limited. *Cortex-M3*. Cortex-M3. 2010. URL: https://developer.arm.com/Processors/Cortex-M3 (besucht am 16. 05. 2022).

[Lim15] ARM Limited. *MPS2+ FPGA Prototyping Board*. 6. Okt. 2015. URL: https://developer.arm.com/Tools%20and%20Software/MPS2%20Plus%20FPGA%20Prototyping%20Board (besucht am 25. 05. 2022).

[Lim22] ARM Limited. *Arm Academic Access*. Arm | The Architecture for the Digital World. 25. Mai 2022. URL: https://www.arm.com/resources/research/enablement/academic-access (besucht am 25. 05. 2022).

[Liu17] Yanyan Liu u. a. „Area-Efficient Reed–Solomon Decoder Using Recursive Berlekamp–Massey Architecture for Optical Communication Systems“. In: *IEEE Communications Letters* 21.11 (Nov. 2017). Conference Name: IEEE Communications Letters, S. 2348–2351. ISSN: 1558-2558. DOI: 10.1109/LCOMM.2017.2734884.

[Löh00] Michael Löhning, Tim Hentschel und Gerhard Fettweis. „Digital down conversion in software radio terminals“. In: *2000 10th European Signal Processing Conference*. 2000 10th European Signal Processing Conference. Sep. 2000, S. 1–4.

[Mas69] J. Massey. „Shift-register synthesis and BCH decoding“. In: *IEEE Transactions on Information Theory* 15.1 (Jan. 1969). Conference Name: IEEE Transactions on Information Theory, S. 122–127. ISSN: 1557-9654. DOI: 10.1109/TIT.1969.1054260.

[McC19] F. Patrick McCluskey, Richard Grzybowski und Thomas Podlesak, Hrsg. *High Temperature Electronics*. Boca Raton: CRC Press, 20. Dez. 2019. 352 S. ISBN: 978-0-203-75197-8. DOI: 10.1201/9780203751978.

[McM04] Robert Joseph McMahon. *Iperf v2.0.0, a tool for measuring Internet bandwidth performance*. Version 2.0. GitHub, 2004. URL: https://github.com/daynix/iperf2 (besucht am 22.07.2022).

[Mly15] P. Mlynek u. a. „Narrowband Power Line Communication for Smart Metering and Street Lighting Control“. In: *IFAC-PapersOnLine*. 13th IFAC and IEEE Conference on Programmable Devices and Embedded Systems 48.4 (1. Jan. 2015), S. 215–219. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2015.07.035. URL: https://www.sciencedirect.com/science/article/pii/S2405896315008101 (besucht am 18.08.2021).

[Mon18] Arijit Mondal u. a. „Efficient Coding Architectures for Reed–Solomon and Low-Density Parity-Check Decoders for Magnetic and Other Data Storage Systems“. In: *IEEE Transactions on Magnetics* 54.2 (Feb. 2018). Conference Name: IEEE Transactions on Magnetics, S. 1–15. ISSN: 1941-0069. DOI: 10.1109/TMAG.2017.2778053.

[Moo06] Gordon E. Moore. „Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff.“ In: *IEEE Solid-State Circuits Society Newsletter* 11.3 (Sep. 2006). Conference Name: IEEE Solid-State Circuits Society Newsletter, S. 33–35. ISSN: 1098-4232. DOI: 10.1109/N-SSC.2006.4785860.

[Mor99] M. Morelli und U. Mengali. „An improved frequency offset estimator for OFDM applications“. In: *IEEE Communications Letters* 3.3 (März 1999). Conference Name: IEEE Communications Letters, S. 75–77. ISSN: 1558-2558. DOI: 10.1109/4234.752907.

[Nat15] Instruments National. *NI PXIe-1085 Series User Manual*. Feb. 2015. URL: https://www.amplicon.com/actions/viewDoc.cfm?doc=NI_PXIe-1085_manual.pdf (besucht am 20.06.2022).

[Nol22] Stefan Nolting. *The NEO430 Processor*. original-date: 2017-02-07T01:09:02Z. 30. März 2022. URL: https://github.com/stnolting/neo430 (besucht am 11.05.2022).

[Oei19] Pao-Yu Oei u. a. *Klimaschutz und Kohleausstieg: Politische Strategien und Maßnahmen bis 2030 und darüber hinaus*. 27. Umweltbundesamt, Deutschland, Juli 2019, S. 313.

[Oks11] Vladimir Oksman und Jin Zhang. „G.HNEM: the new ITU-T standard on narrowband PLC technology“. In: *IEEE Communications Magazine* 49.12 (Dez. 2011), S. 36–44. ISSN: 1558-1896. DOI: 10.1109/MCOM.2011.6094004.

[Pap13] Theofilos A. Papadopoulos u. a. „Application of Narrowband Power-Line Communication in Medium-Voltage Smart Distribution Grids“. In: *IEEE Transactions on Power Delivery* 28.2 (Apr. 2013). Conference Name: IEEE Transactions on Power Delivery, S. 981–988. ISSN: 1937-4208. DOI: 10.1109/TPWRD.2012.2230344.

[Pau66] Reinhold Paul. *Transistormeßtechnik*. 1. Aufl. Vieweg+Teubner Verlag Wiesbaden, 1966. 344 S. ISBN: 978-3-663-00827-9. URL: https://link.springer.com/book/10.1007/978-3-663-02740-9 (besucht am 04. 07. 2022).

[Pér11] Juan Pablo Alegre Pérez, Santiago Celma und Belén Calvo López. *Automatic Gain Control: Techniques and Architectures for RF Receivers*. Analog Circuits and Signal Processing. New York: Springer-Verlag, 2011. ISBN: 978-1-4614-0166-7. DOI: 10.1007/978-1-4614-0167-4. URL: https://www.springer.com/gp/book/9781461401667 (besucht am 25. 08. 2021).

[Per19] Gabriele Perrone u. a. „Reed–Solomon Decoder Based on a Modified ePIBMA for Low-Latency 100 Gbps Communication Systems“. In: *Circuits, Systems, and Signal Processing* 38.4 (1. Apr. 2019), S. 1793–1810. ISSN: 1531-5878. DOI: 10.1007/s00034-018-0938-x. URL: https://doi.org/10.1007/s00034-018-0938-x (besucht am 25. 06. 2021).

[Pet17] Konstantin O. Petrosyants u. a. „Electrical characterization and reliability of submicron SOI CMOS technology in the extended temperature range (to 300°C)“. In: *Microelectronics Reliability* 79 (1. Dez. 2017), S. 416–425. ISSN: 0026-2714. DOI: 10.1016/j.microrel.2017.05.018. URL: https://www.sciencedirect.com/science/article/pii/S002627141730149X (besucht am 11. 07. 2022).

[Pol94] T. Pollet, P. Spruyt und M. Moeneclaey. „The BER performance of OFDM systems using non-synchronized sampling“. In: *1994 IEEE GLOBECOM. Communications: The Global Bridge*. 1994 IEEE GLOBECOM. Communications: The Global Bridge. Nov. 1994, 253–257 vol.1. DOI: 10.1109/GLOCOM.1994.513417.

[Pol95] T. Pollet, M. Van Bladel und M. Moeneclaey. „BER sensitivity of OFDM systems to carrier frequency offset and Wiener phase noise“. In: *IEEE Transactions on Communications* 43.2 (Feb. 1995). Conference Name: IEEE Transactions on Communications, S. 191–193. ISSN: 1558-0857. DOI: 10.1109/26.380034.

[Pro83] John G. Proakis. *Digital communications*. McGraw-Hill series in electrical engineering. Communications and information theory. New York: McGraw-Hill, 1983. 608 S. ISBN: 978-0-07-066490-6.

[Pus07] Henri Puska und Harri Saarnisaari. „Matched Filter Time and Frequency Synchronization Method for OFDM Systems using PN-sequence Preambles“. In: *2007 IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications*. 2007 IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications. ISSN: 2166-9589. Sep. 2007, S. 1–5. DOI: 10.1109/PIMRC.2007.4394602.

[Rah11] Md. Mustafizur Rahman u. a. „Medium access control for power line communications: an overview of the IEEE 1901 and ITU-T G.hn standards“. In: *IEEE Communications Magazine* 49.6 (Juni 2011). Conference Name: IEEE Communications Magazine, S. 183–191. ISSN: 1558-1896. DOI: 10.1109/MCOM.2011.5784004.

[Ren09] Guangliang Ren, Huining Zhang und Yilin Chang. „SNR estimation algorithm based on the preamble for OFDM systems in frequency selective channels“. In: *IEEE Transactions on Communications* 57.8 (Aug. 2009). Conference Name: IEEE Transactions on Communications, S. 2230–2234. ISSN: 1558-0857. DOI: 10.1109/TCOMM.2008.08.060406.

[Rig14] Wolfgang Riggert. *Rechnernetze: Grundlagen - Ethernet - Internet*. Hrsg. von Christian Märtin und Michael Lutz. 5., aktualisierte Auflage. Lehrbücher zur Informatik. Series Title: Lehrbücher zur Informatik. München: Hanser, 2014. 281 S. ISBN: 978-3-446-44096-8. URL: http://dx.doi.org/10.3139/9783446440968 (besucht am 26. 07. 2022).

[Ros15] Davide Rossi u. a. „PULP: A parallel ultra low power platform for next generation IoT applications“. In: *2015 IEEE Hot Chips 27 Symposium (HCS)*. 2015 IEEE Hot Chips 27 Symposium (HCS). Aug. 2015, S. 1–39. DOI: 10.1109/HOTCHIPS.2015.7477325.

[Rot18] Niklas Rother u. a. „A Case Study on Multi-Softcore Aided Hardware Architectures for Powerline MAC-Layer“. In: *ICT. OPEN 2018* (2018).

[Roy03] K. Roy, S. Mukhopadhyay und H. Mahmoodi-Meimand. „Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits“. In: *Proceedings of the IEEE* 91.2 (Feb. 2003). Conference Name: Proceedings of the IEEE, S. 305–327. ISSN: 1558-2256. DOI: 10.1109/JPROC.2002.808156.

[Sar01] D.V. Sarwate und N.R. Shanbhag. „High-speed architectures for Reed-Solomon decoders“. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 9.5 (Okt. 2001). Conference Name: IEEE Transactions on Very Large Scale Integration (VLSI) Systems, S. 641–655. ISSN: 1557-9999. DOI: 10.1109/92.953498.

[Sch12] Dagmar Schönfeld, Herbert Klimant und Rudi Piotraschke. „Kanalkodierung“. In: *Informations- und Kodierungstheorie*. Hrsg. von Dagmar Schönfeld, Herbert Klimant und Rudi Piotraschke. Wiesbaden: Vieweg+Teubner Verlag, 2012, S. 125–265. ISBN: 978-3-8348-8218-9. DOI: 10.1007/978-3-8348-8218-9_8. URL: https://doi.org/10.1007/978-3-8348-8218-9_8 (besucht am 26. 08. 2021).

[Sch18] Pasquale Davide Schiavone u. a. „Quentin: an Ultra-Low-Power PULPissimo SoC in 22nm FDX“. In: *2018 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*. 2018 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S). ISSN: 2573-5926. Okt. 2018, S. 1–3. DOI: 10.1109/S3S.2018.8640145.

[Sch97] T.M. Schmidl und D.C. Cox. „Robust frequency and timing synchronization for OFDM“. In: *IEEE Transactions on Communications* 45.12 (Dez. 1997). Conference Name: IEEE Transactions on Communications, S. 1613–1621. ISSN: 1558-0857. DOI: 10.1109/26.650240.

[Sed09] Adel Sedra und Kenneth C. Smith. *Microelectronic Circuits*. 6. Aufl. New York: Oxford Univ Pr, 2009. 1386 S. ISBN: 978-0-19-532303-0.

[Sha99] G.G. Shahidi u. a. „Partially-depleted SOI technology for digital logic“. In: *1999 IEEE International Solid-State Circuits Conference. Digest of Technical Papers. ISSCC. First Edition (Cat. No.99CH36278)*. 1999 IEEE International Solid-State Circuits Conference. Digest of Technical Papers. ISSCC. First Edition (Cat. No.99CH36278). ISSN: 0193-6530. Feb. 1999, S. 426–427. DOI: 10.1109/ISSCC.1999.759337.

[Shi04] Y.Q. Shi u. a. „Interleaving for combating bursts of errors“. In: *IEEE Circuits and Systems Magazine* 4.1 (2004). Conference Name: IEEE Circuits and Systems Magazine, S. 29–42. ISSN: 1558-0830. DOI: 10.1109/MCAS.2004.1286985.

[Sil99] Joseph H. Silverman. „Fast Multiplication in Finite Fields GF(2n)“. In: *Cryptographic Hardware and Embedded Systems*. Hrsg. von Çetin K. Koç und Christof Paar. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1999, S. 122–134. ISBN: 978-3-540-48059-4. DOI: 10.1007/3-540-48059-5_12.

[Sto97] William C. Stone. *Electromagnetic Signal Attenuation in Construction Materials*. 3. Last Modified: 2018-11-10T10:11-05:00 Publisher: William C. Stone. NIST Construction Automation Program, 1. Okt. 1997, S. 200. URL: https://www.nist.gov/publications/electromagnetic-signal-attenuation-construction-materials (besucht am 11. 07. 2022).

[Stu21] Tobias Stuckenberg u. a. „Powerline Communication System-on-Chip in 180 nm Harsh Environment SOI Technology“. In: *2021 IEEE Nordic Circuits and Systems Conference (NorCAS)*. 2021 IEEE Nordic Circuits and Systems Conference (NorCAS). Okt. 2021, S. 1–5. DOI: 10.1109/NorCAS53631.2021.9599855.

[Stu22] Tobias Stuckenberg u. a. „Design and Evaluation of a 180nm Powerline Communication ASIC for Harsh Environment“. In: *(eingereicht)*. Microprocessors and Microsystems (30. Apr. 2022), S. 12.

[Tec99] National Institute of Standards and Technology. *Data Encryption Standard (DES)*. Federal Information Processing Standard (FIPS) 46-3 (Withdrawn). U.S. Department of Commerce, 25. Okt. 1999. URL: https://csrc.nist.gov/publications/detail/fips/46/3/archive/1999-10-25 (besucht am 05. 05. 2022).

[Tei09] Frederik Teichert. „Digital-Down-Converter für Field-Programmable-Gate-Arrays in VHDL“. Accepted: 2020-09-29T16:08:10Z. Thesis. Hochschule für angewandte Wissenschaften Hamburg, 13. Mai 2009. URL: https://reposit.haw-hamburg.de/handle/20.500.12738/9670 (besucht am 26. 08. 2021).

[Tek19] Tekmos. *High Temperature ASICs, 250C and 200C*. 11. Aug. 2019. URL: https://www.tekmos.com/products/high-temperature-semiconductors/high-temperature-asics-250-c-and-200-c (besucht am 26. 07. 2021).

[Teo21] Catalin Teodoriu und Opeyemi Bello. „An Outlook of Drilling Technologies and Innovations: Present Status and Future Trends“. In: *Energies* 14.15 (Jan. 2021). Number: 15 Publisher: Multidisciplinary Digital Publishing Institute, S. 4499. ISSN: 1996-1073. DOI: 10.3390/en14154499. URL: https://www.mdpi.com/1996-1073/14/15/4499 (besucht am 27. 07. 2022).

[Tur60] G. Turin. „An introduction to matched filters“. In: *IRE Transactions on Information Theory* 6.3 (Juni 1960). Conference Name: IRE Transactions on Information Theory, S. 311–329. ISSN: 2168-2712. DOI: 10.1109/TIT.1960.1057571.

[Ul 13] Ul Haq und Syed Wahaj. *Differential QPSK Based Communication System von Ul Haq, Syed Wahaj*. LAP LAMBERT Academic Publishing, 23. Jan. 2013. 68 S. ISBN: 978-3-659-32009-5. URL: https://www.medimops.de/ul-haq-syed-wahaj-differential-qpsk-based-communication-system-taschenbuch-M03659320099.html (besucht am 21. 07. 2022).

[US 20] US Energy Information Administration. *Average Depth of Crude Oil and Natural Gas Wells*. Independent Statistics & Analysis. 10. Jan. 2020. URL: https://www.eia.gov/dnav/pet/pet_crd_welldep_s1_a.htm (besucht am 22. 07. 2021).

[Van04] T. Vanhoucke, H.M.J. Boots und W.D. van Noort. „Revised method for extraction of the thermal resistance applied to bulk and SOI SiGe HBTs". In: *IEEE Electron Device Letters* 25.3 (März 2004). Conference Name: IEEE Electron Device Letters, S. 150–152. ISSN: 1558-0563. DOI: 10.1109/LED.2004.824242.

[Vee17] Harry J. M. Veendrick. *Nanometer CMOS ICs: From Basics to ASICs*. Google-Books-ID: 3Gb0vgEACAAJ. Springer International Publishing, 2017. book. ISBN: 978-3-319-47596-7.

[Vit67] A. Viterbi. „Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". In: *IEEE Transactions on Information Theory* 13.2 (Apr. 1967). Conference Name: IEEE Transactions on Information Theory, S. 260–269. ISSN: 1557-9654. DOI: 10.1109/TIT.1967.1054010.

[Vit98] A.J. Viterbi. „An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes". In: *IEEE Journal on Selected Areas in Communications* 16.2 (Feb. 1998). Conference Name: IEEE Journal on Selected Areas in Communications, S. 260–264. ISSN: 1558-0008. DOI: 10.1109/49.661114.

[VOR16] VORAGO Technologies. *Using HARDSIL to minimize the impact of extreme temperature on CMOS integrated circuits*. White Paper. Austin, Texas: VORAGO Technologies, 15. Dez. 2016, S. 1–11. URL: https://static1.squarespace.com/static/5d920c8760259e0ec548338d/t/5dd242601a63bd7bd1b91c02/1574060642624/High+Temperature+White+Paper.pdf (besucht am 27. 07. 2021).

[Wan17] Yang Wang u. a. „An Improved Concatenation Scheme of Polar Codes With Reed–Solomon Codes". In: *IEEE Communications Letters* 21.3 (März 2017). Conference Name: IEEE Communications Letters, S. 468–471. ISSN: 1558-2558. DOI: 10.1109/LCOMM.2016.2639482.

[Wat15] Jeff Watson und Gustavo Castro. „A review of high-temperature electronics technology and applications". In: *Journal of Materials Science: Materials in Electronics* 26.12 (1. Dez. 2015), S. 9226–9235. ISSN: 1573-482X. DOI: 10.1007/s10854-015-3459-4. URL: https://doi.org/10.1007/s10854-015-3459-4 (besucht am 30. 08. 2021).

[Wer08] Martin Werner. *Signale und Systeme: Lehr- und Arbeitsbuch mit MATLAB®-Übungen und Lösungen*. 3., vollst. überarb. und erw. Aufl. 2008 Edition. Wiesbaden: Vieweg+Teubner Verlag, 15. Juli 2008. 404 S. ISBN: 978-3-8348-0233-0.

[Wic95] Stephen B. Wicker. *Error Control Systems for Digital Communication and Storage*. Prentice Hall, 1995. 536 S. ISBN: 978-0-13-200809-9.

[Wol84] Wold und Despain. „Pipeline and Parallel-Pipeline FFT Processors for VLSI Implementations“. In: *IEEE Transactions on Computers* C-33.5 (Mai 1984). Conference Name: IEEE Transactions on Computers, S. 414–426. ISSN: 1557-9956. DOI: 10.1109/TC.1984.1676458.

[Wri97] M.B. Wright, D. Humphrey und F.P. McCluskey. „Uprating electronic components for use outside their temperature specification limits“. In: *IEEE Transactions on Components, Packaging, and Manufacturing Technology: Part A* 20.2 (Juni 1997). Conference Name: IEEE Transactions on Components, Packaging, and Manufacturing Technology: Part A, S. 252–256. ISSN: 1558-3678. DOI: 10.1109/95.588581.

[Wu04] Hsiao-Chun Wu und Xiaozhou Huang. „Joint phase/amplitude estimation and symbol detection for wireless ICI self-cancellation coded OFDM systems“. In: *IEEE Transactions on Broadcasting* 50.1 (März 2004). Conference Name: IEEE Transactions on Broadcasting, S. 49–55. ISSN: 1557-9611. DOI: 10.1109/TBC.2004.824746.

[Wu15] Yingquan Wu. „New Scalable Decoder Architectures for Reed–Solomon Codes“. In: *IEEE Transactions on Communications* 63.8 (Aug. 2015). Conference Name: IEEE Transactions on Communications, S. 2741–2761. ISSN: 1558-0857. DOI: 10.1109/TCOMM.2015.2445759.

[Yon13] Larry Yonge u. a. „An Overview of the HomePlug AV2 Technology“. In: *Journal of Electrical and Computer Engineering* 2013 (19. März 2013). Publisher: Hindawi, e892628. ISSN: 2090-0147. DOI: 10.1155/2013/892628. URL: https://www.hindawi.com/journals/jece/2013/892628/ (besucht am 30. 04. 2021).

[Zai18] Ali Zaidi u. a. „Chapter 5 - Multicarrier Waveforms“. In: *5G Physical Layer*. Hrsg. von Ali Zaidi u. a. Academic Press, 1. Jan. 2018, S. 119–158. ISBN: 978-0-12-814578-4. DOI: 10.1016/B978-0-12-814578-4.00010-2. URL: https://www.sciencedirect.com/science/article/pii/B9780128145784000102 (besucht am 02. 07. 2021).

[Zei56] Werner Zeil. *Brinkmanns Abriss der Geologie. 1. Band: Allgemeine Geologie*. 9. Stuttgard: Ferdinand Enke, 1956. 280 S. URL: https://www.zvab.com/Brinkmanns-Abriss-Geologie-1-Band-Allgemeine/30706641906/bd (besucht am 22. 07. 2021).

[Zim02] M. Zimmermann und K. Dostert. „A multipath model for the powerline channel“. In: *IEEE Transactions on Communications* 50.4 (Apr. 2002). Conference Name: IEEE Transactions on Communications, S. 553–559. ISSN: 1558-0857. DOI: 10.1109/26.996069.

[Zim80] H. Zimmermann. „OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection“. In: *IEEE Transactions on Communications* 28.4 (Apr. 1980). Conference Name: IEEE Transactions on Communications, S. 425–432. ISSN: 1558-0857. DOI: 10.1109/TCOM.1980.1094702.

[Zyr11] Jim Zyren. „The HomePlug Green PHY specification the in-home Smart Grid“. In: *2011 IEEE International Conference on Consumer Electronics (ICCE)*. 2011 IEEE International Conference on Consumer Electronics (ICCE). ISSN: 2158-4001. Jan. 2011, S. 241–242. DOI: 10.1109/ICCE.2011.5722562.

Appendix

ANHANG A

Berechnung der Größe eines Segmentes

Die Größe eines Segmentes, das von der PHY-Schicht übertragen werden kann, hängt von den geschätzten Kanalparametern ab (vgl. Abschnitt 3.2.1). Um die Segmentierung eines zu übertragenden Frames durchführen zu können, müssen die Kanalinformationen (ToneMaps) des Paketempfängers der MAC-Schicht bekannt sein. Ansonsten erfolgt immer eine Kommunikation im ROBO-Modus. Zur Bestimmung einer *ToneMap* speichert die PHY-Schicht den geschätzten Kanal im Kanal-Charakteristik-Speicher (vgl. Abbildung 3.2 in Abschnitt 3.1.1). MAC-Schicht berechnet daraus und anhand der Schwellwerte aus Abschnitt 3.3 eine aktuelle *ToneMap*. Diese *ToneMap* wird im *ToneMap*-Speicher abgelegt und für die Segmentierung von der MAC-Schicht gelesen. Die folgenden Informationen sind in der *ToneMap* enthalten:

- **Frequenzträger:** Liste aller Frequenzträger, mit Einsen für zu verwendende und Nullen für nicht zu verwendende Träger.
- **Modulationsart:** Verwendung von ROBO-Modus oder reguläre Modulation als BSPK, QPSK oder punktierte QPSK.
- **Bridge-MACs:** Bekannte MAC-Adressen der gebrückten Station des *ToneMap*-Senders.

Diese Informationen werden auf Anfrage einer Station an dieser übertragen und anschließend für den Paketaustausch verwendet. Dabei hat jede Station eigene Sende- und Empfangs-*ToneMap* mit individuellen Segmentgrößen.

Zur Berechnung der maximalen Anzahl an Bytes für ein Segment, ermittelt die MAC-Schicht nach Gleichung A.1 zuerst die maximale Anzahl an Bytes, die in einem 20er- oder 40er-OFDM-Symbolblock codiert werden können.

$$\text{Byte}_{\text{RS, Max}} = \left\lfloor \frac{1}{8} * n_{\text{Träger}} * n_{\text{Bit/Träger}} * n_{\text{Block}} * b_{\text{Punkt}} - n_{\text{Bit, Tail}} \right\rfloor \tag{A.1}$$

Dabei ist $n_{\text{Träger}}$ die Anzahl der zu verwendeten Frequenzträger (mindestens 11, maximal 84), die aus dem geschätzten Kanal bestimmt werden. $n_{\text{Bit/Träger}}$ ist die Anzahl der Informationsbits,

die pro Träger moduliert werden (1 für BPSK oder 2 QPSK) können und n_{Block} die Größe des zu berechnenden OFDM-Symbolblocks (20 oder 40 Symbole). b_{Punkt} ist das Verhältnis zwischen der Anzahl an Eingangsbits am Faltungsencoder und der Anzahl an Ausgangsbits nach der Punktierung. Dieser Wert kann entweder 0.5 ohne Punktierung und 0.75 mit Punktierung sein. $n_{\text{Bit, Tail}}$ ist die Anzahl der Nullen, um den Faltungsencoder wieder in den initialen Zustand zurückzuführen (hier immer 6).

Anschließend kann mit Hilfe von Gleichung A.2 die minimale Anzahl an Reed-Solomon-Iterationen pro OFDM-Symbolblock berechnet werden. Die dabei verwendete Größe einer Reed-Solomon-Iteration wird nach Gleichung A.3 bestimmt.

$$n_{\text{RSBlock}} = \left\lfloor \frac{\text{Byte}_{\text{RS,Max}} + 239}{255} \right\rfloor \tag{A.2}$$

$$\text{Byte}_{\text{RSBlock}} = min\left(254, \left\lfloor \frac{\text{Byte}_{\text{RS,Max}}}{n_{\text{RSBlock}}} \right\rfloor\right) \tag{A.3}$$

Die Anzahl Bytes am Eingang der PHY-Schicht für einen OFDM-Symbolblock kann abschließend mit den Gleichungen A.4 und A.5 ermittelt werden. Diese unterscheiden, ob der ROBO-Modus $\text{Byte}_{\text{ROBO}}$ oder eine reguläre Modulation $\text{Byte}_{\text{Regulär}}$ verwendet wird, da hierbei unterschiedliche Reed-Solomon-Codes verwendet werden.

$$\text{Byte}_{\text{Regulär}} = (\text{Byte}_{\text{RSBlock}} - 16) * n_{\text{RSBlock}} \tag{A.4}$$

$$\text{Byte}_{\text{ROBO}} = (\text{Byte}_{\text{RSBlock}} - 8) * n_{\text{RSBlock}} \tag{A.5}$$

Tabelle A.1 zeigt die Anzahl der kodierbaren Bytes pro 20er- und 40er-Symbolblock am Eingang der PHY-Schicht für alle Parameterkonfigurationen und drei Anzahlen von Trägern: minimal (31), die Hälfte (42) und maximal (84). Für den ROBO-Modus ist eine Blockgröße von 40 Symbolen, eine Faltungsencoderrate von 0.5 und BPSK-Modulation vorgeschrieben. Eine Punktierung kann nur in Kombination mit QPSK-Modulation gewählt werden.

Um die Größe eines Segmentes bestimmen zu können, werden die Ergebnisse für die aktuelle Konfiguration des 20er- und 40er-Blockes verwendet und die Framelänge der PHY-Schicht bestimmt. Die Framelänge kann zwischen 20 und 160 Symbolen lang sein und lässt sich in 20er-Schritten erhöhen. Zum Beispiel kann bei Verwendung des ROBO-Modus mit allen 84 Trägern ein Frame mit 160 Symbolen insgesamt 172 Byte enthalten, also vier 40er-Symbolblöcke mit 43 Byte. Für eine reguläre BPSK-Modulation und 84 Trägern lassen folgende Paketgrößen berechnen: Damit kann in dieser Konfiguration ein Segment 836 Byte enthalten. Ist das zu sendende Paket größer als diese maximale Schwelle, muss dieses segmentiert werden. Für den Fall eines 1500 Byte großen *Ethernet*-Paketes, wäre das zweite Segment 1500 Byte − 836 Byte = 664 Byte groß und kann mit einer Framelänge von 140 Symbolen codiert werden.

Framelänge [Symbole]	Größe [Byte]
20	104
40	209
60	313
80	418
100	522
120	627
140	731
160	836

Tabelle A.1: *Anzahl der Nutzdaten-Bytes, die in einem 20er- oder 40er-Symbolblock enthalten sein können. Der ROBO-Modus ist dabei auf 40er-Block mit BPSK ohne Punktierung beschränkt.*

$n_{\text{Träger}}$	$n_{\text{Bit/Träger}}$	n_{Block}	b_{Punkt}	$\text{Byte}_{\text{ROBO}}$	$\text{Byte}_{\text{Regulär}}$
31	1	20	0.5	–	38
		40	0.5	10	76
	2	20	0.5	–	76
			0.75	–	115
		40	0.5	–	154
			0.75	–	231
42	1	20	0.5	–	51
		40	0.5	17	104
	2	20	0.5	–	104
			0.75	–	156
		40	0.5	–	209
			0.75	–	314
84	1	20	0.5	–	104
		40	0.5	43	209
	2	20	0.5	–	209
			0.75	–	314
		40	0.5	–	419
			0.75	–	629

ANHANG B

Abrollen der Scrambler-Iterationen

Das Abrollen der Scrambler-Gleichung dient der Beschleunigung des Descramblers in der Implementierung als dedizierte Hardware in Abschnitt 4.1.3. Insgesamt werden acht Iterationen der Gleichung B.1 abgerollt, um die bitweise zu einer byteweise parallelen Verarbeitung zu beschleunigen.

$$S(x) = x^7 + x^4 + 1 \tag{B.1}$$

Die Gedächtnislänge des Sramblers ist sieben, was gleichzeitig die höchste Potenz in Gleichung B.1 ist. Um diese Gleichung abzurollen, kann die Eigenschaft des linearen Feedbacks des LFSR aus Abbildung B.1 genutzt, dass nur die Potenzen dekrementiert werden. Durch die Schieberegistersruktur des Scramblers wird nur der Wert in x^1 jeden Takt neu geschrieben und die Werte x^7 bis x^2 sind die alten Werte x^6 bis x^1. Für den neuen Wert x^1 wird die Rückkopplung der alten Werte $x^7 + x^4$ (*Feeback*) an diese Stelle eingesetzt. Die abgerollten Gleichungen für die Ausgangsbits $S_{0...7}$ in Abhängigkeit der Eingangsbits $d_{0...7}$ sind wie folgt:

$$\begin{aligned}
S_0(x) &= x^7 + x^4 + d_0 \\
S_1(x) &= x^6 + x^3 + d_1 \\
S_2(x) &= x^5 + x^2 + d_2 \\
S_3(x) &= x^4 + x^1 + d_3 \\
S_4(x) &= x^3 + x^7 + x^4 + d_4 \\
S_5(x) &= x^2 + x^6 + x^3 + d_5 \\
S_6(x) &= x^1 + x^5 + x^2 + d_6 \\
S_7(x) &= x^7 + x^4 + x^4 + x^1 + d_7 = x^7 + x^1 + d_7
\end{aligned}$$

Dabei repräsentiert jede Gleichung für $S_n(x)$ einen Takt in der seriellen Ausführung, so dass nach $S_7(x)$ das LFSR den gleichen Zustand erreicht hat, wie in der iterativen Version nach acht Takten. Bei der achten Iteration des Abrollens $S_7(x)$ entstehen zwei Terme mit x^4, die sich durch die XOR-Berechnung innerhalb des LFSR gegenseitig auslöschen. Damit hat jede

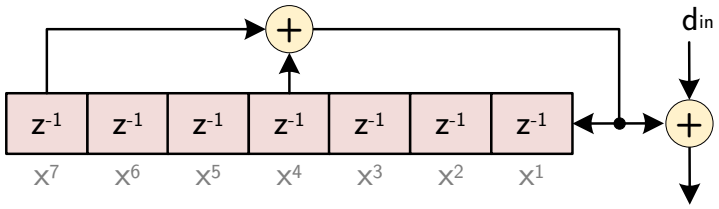

Abbildung B.1: *Hardware-Struktur des Scramblers nach dem HomePlug-1.0-Standard [Hom08].*

Gleichung für $S_i(x)$ maximal drei Terme, die mit dem Eingangsbits verrechnet werden müssen. Alle Gleichungen lassen sich parallel berechnen und haben einen maximalen kritischen Pfad von drei XOR-Gattern.

ANHANG C

MAC-Schicht Zustandsdiagramme

Die detaillierten Zustandsdiagramme des MAC-Schicht-Senders und des MAC-Schicht-Empfängers sind in Abbildung C.1 und Abbildung C.2 gezeigt. Diese Zustandsdiagramme sind detaillierte Ansichten der in Abbildung 3.2 aus Abschnitt 3.1.1 gezeigten Blöcke und nach dem Standard erstellt [Hom08].

Abbildung C.1 a) zeigt den TX-MAC-Service-Block, der die Anfragen der LLC-Schicht bearbeitet, beziehungsweise empfangene Daten an diese weitergibt. Dabei beantwortet der Block Statistik-Anfragen direkt, ohne diese an die unteren Blöcke weiterzugeben. Alle anderen Anfragen können, je nach Zieladresse, direkt beantwortet oder als Paket an eine Verschlüsselung weitergegeben werden.

Das Zustandsdiagramm in Abbildung C.1 b) zeigt die Verschlüsselung eines Segmentes mit den 16 Iterationen der Subschlüssel. Dieser Block wartet bei der initialen Permutation auf eingehende Daten und arbeitet dann das Zustandsdiagramm linear ab.

Abbildung C.1 c) zeigt die Segmentierung, die in der Initialisierung die verfügbaren Framegrößen berechnet (siehe Anhang C), Segmente an die Verschlüsselung gibt und anschließend mit den weiteren Segmenten wartet, bis das aktuelle erfolgreich übertragen wurde.

Das abschließende Senden eines Frames ist in Abbildung C.1 d) dargestellt und folgt mit seinen Zuständen dem Protokoll auf dem Übertragungskanal. Erscheint ein neues Frame an dessen Eingang, wird dieses initialisiert und entsprechend dem Protokoll auf eine Prioritätsaushandlung gewartet. Hat die Station die höchste Priorität, wird das Frame gesendet und auf eine eventuelle Antwort gewartet. Ist die Antwort eine FAIL-Nachricht vom Empfänger wird das Frame zurückgestellt und 10 ms gewartet. War die Übertragung erfolgreich, wird entweder direkt das nächste Frame initialisiert oder auf ein neues Frame gewartet. Findet keine erfolgreiche Übertragung innerhalb der Paketlebensdauer statt, wird dieses verworfen.

Der Empfänger verarbeitet die Blöcke in umgekehrter Reihenfolge und beginnt mit dem Empfangen eines Frames in Abbildung C.2 d). Ähnlich wie der Frame-Senden-Block des Senders

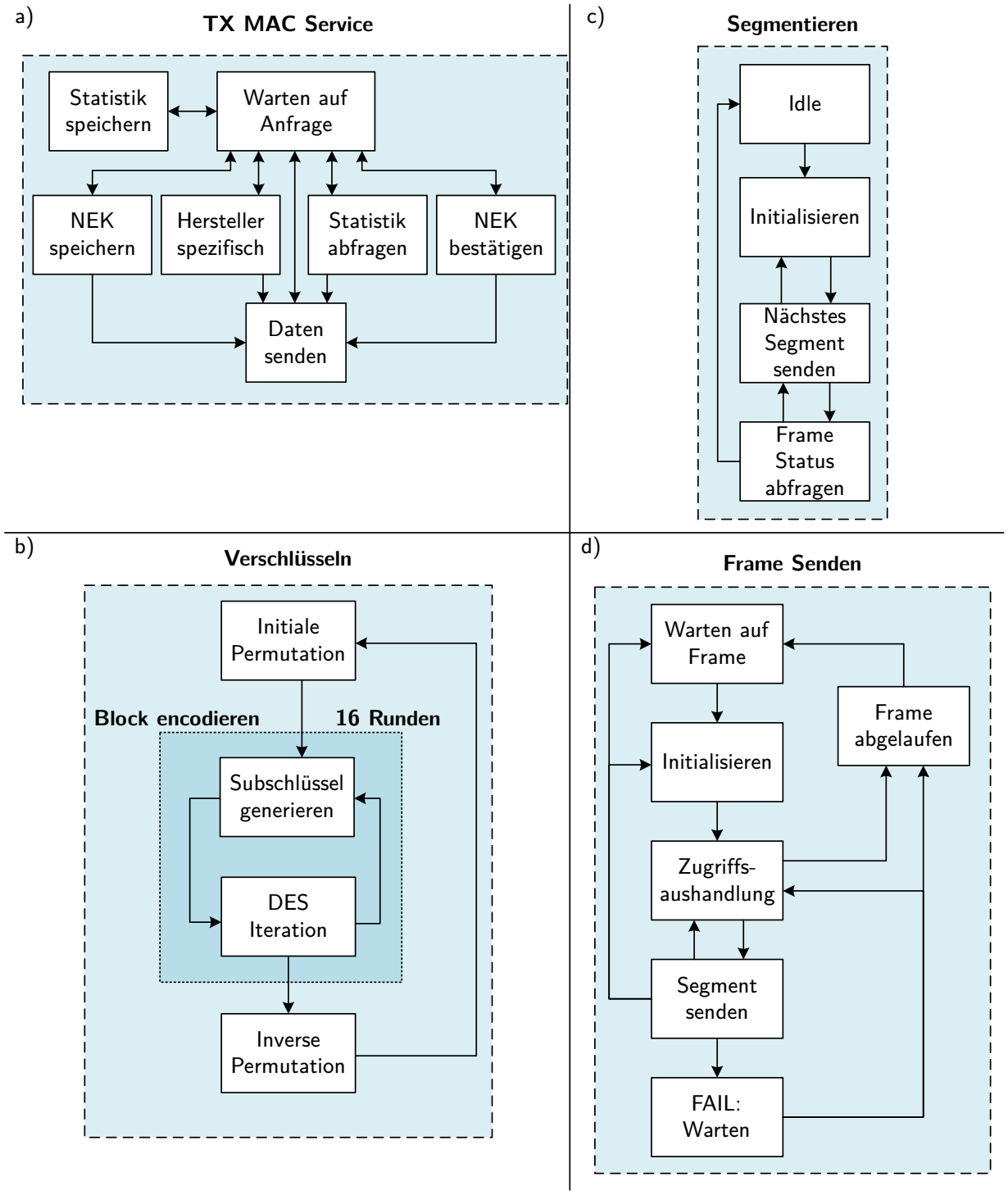

Abbildung C.1: *Zustandsdiagramme des Sendepfades der MAC-Schicht: In a) der TX-MAC-Serviceblock, in b) die Verschlüsselung, in c) die Segmentierung und in d) das Senden eines Frames.*

folgt dieser dem Protokoll auf dem Übertragungskanal und die empfangenen Frames entgegen. Dazu wird zunächst nach einer Präambel auf dem Kanal gesucht und der anschließend der Delimiter verarbeitet. Ist dies der Start eines Frames, wird gewartet bis der Rest empfangen wurde und im Falle eines Delimiters am Ende der Übertragung wird überprüft, ob eine Antwort der eigenen Station angefragt wurde. Eine Antwort wird gesendet, wenn dies der Fall ist und

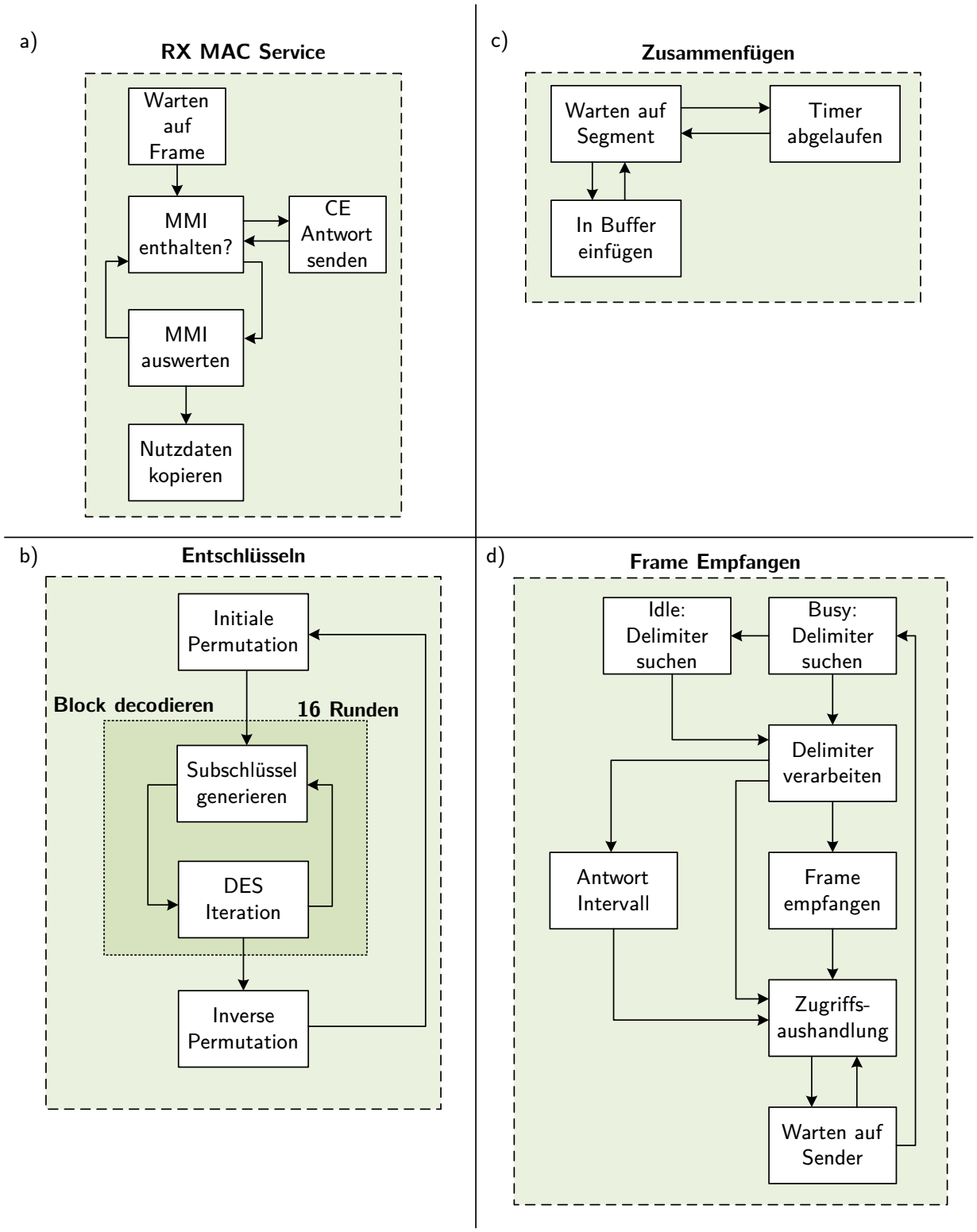

Abbildung C.2: *Zustandsdiagramme des Empfanspfades der MAC-Schicht: In a) der RX-MAC-Serviceblock, in b) die Entschlüsselung, in c) das Zusammensetzen und in d) der Empfang eines Frames.*

das Paket an die Zusammensetzung weitergegeben. Ist der gefundene Delimiter eine Antwort einer anderen Station, wird diese decodiert, um sich mit dem Protokoll zu synchronisieren.

Nach dem Empfang werden in Abbildung C.2 c) die Segmente zu einem Paket zusammengesetzt. Werden alle Segmente innerhalb des dafür vorgesehenen Timeouts empfangen, werden diese an die Entschlüsselung weitergeben. Ist dies nicht der Fall und der Paket-Timer läuft ab, bevor das letzte Segment eintrifft, wird das Paket verworfen. Abbildung C.2 b) zeigt die Entschlüsselung, die analog zur Verschlüsselung die 16 Iterationen auf den 16 Subschlüsseln (in umgekehrter Reihenfolge) durchführt.

Abschließend werden in Abbildung C.2 a) MAC-Management-Informationen aus dem Paket extrahiert und eventuell beantwortet oder weitergegeben. Die enthaltenen Nutzdaten werden dann dem LLC-Interface übergeben.

Curriculum Vitae

Zur Person

Tobias Stuckenberg
geboren am 12.11.1990 in Göttingen (Niedersachsen)

Ausbildung

10/2013 - 02/2016 **Masterstudium der Technischen Informatik**
Leibniz Universität Hannover
Vertiefungsrichtung Mikroelektronik
Abschluss *Master of Science (M.Sc.)*

10/2010 - 11/2013 **Bachelorstudium der Technischen Informatik**
Leibniz Universität Hannover
Vertiefungsrichtung Mikroelektronik
Abschluss *Bachelor of Science (B.Sc.)*

2001 - 2010 **Geschwister-Scholl-Gesamtschule**
Göttingen
Abschluss *Abitur*

Beruflicher Werdegang

seit 04/2016 **Wissenschaftlicher Mitarbeiter**
Institut für Mikroelektronische Systeme
Fachgebiet Architekturen und Systeme
Leibniz Universität Hannover

11/2014 - 04/2015 **Fachpraktikum**
Sennheiser
Department of Research and Development
San Francisco, USA

08/2012 - 10/2013 **Studentische Hilfskraft**
Institut für Mikroelektronische Systeme
Leibniz Universität Hannover

Wissenschaftliche Veröffentlichungen

2022

Design and Evaluation of a 180nm Powerline Communication ASIC for Harsh Environment
T. Stuckenberg; M. Rücker; M. Gottschlich; N. Rother; R. Nowosielski; F. Wiese; H. Blume
Journal of Microprocessors and Microsystems 2022, Status: akzeptiert

Real-Time Evaluation of 60 Powerline Communication Modems using the Protium S1 FPGA Platform
T. Stuckenberg; H. Blume
CadenceLIVE 2022, München, Deutschland

2021

Powerline Communication System-on-Chip in 180 nm Harsh Environment SOI Technology
T. Stuckenberg; M. Rücker; N. Rother; R. Nowosielski; F. Wiese; H. Blume
Proceedings of the 2021 IEEE Nordic Circuits and Systems Conference (NORCAS). 2021

2020

Evaluation of Different Processor Architecture Organizations for On-Site Electronics in Harsh Environments
S. Gesper; M. Weissbrich; T. Stuckenberg; P. Jaaskelainen; H. Blume; G. Payá Vayá
International Journal of Parallel Programming, 2020

2019

Design and Optimization of an ARM Cortex-M-based SoC for TCP/IP Communication in High Temperature Applications
T. Stuckenberg; M. Gottschlich; S. Nolting; H. Blume
19th International Conference, 2019, SAMOS, Griechenland

Evaluation of Different Processor Architecture Organizations for On-Site Electronics in Harsh Environments
S. Gesper; M. Weißbrich; S. Nolting; T. Stuckenberg; P. Jääskeläinen; H. Blume; G. Payá-Vayá
19th International Conference, 2019, SAMOS, Griechenland

2018

A Case Study on Multi-Softcore Aided Hardware Architectures for Powerline MAC-Layer

N. Rother; T. Stuckenberg; S. Nolting; C. Uhlemann; H. Blume

ICT.OPEN 2018, Amersfoort, Niederlande

Processor Architecture Tradeoffs for On-Site Electronics in Harsh Environment

S. Nolting; S. Gesper; A. Schmider; M. Weißbrich; T. Stuckenberg; G. Payá Vayá; H. Blume

Cadence User Conference 2018, München, Deutschland

2017

A Hardware Efficient Preamble Detection Algorithm for Powerline Communication

T. Stuckenberg; H. Blume

10th International Conference on Computer Science and Information Technology (ICCSIT) 2017, Florenz, Italien