KARLSRUHE INSTITUTE OF TECHNOLOGY
Communications Engineering Lab
Prof. Dr.rer.nat. Friedrich K. Jondral

Empfang von LTE-Signalen in GNU Radio

Bachelorarbeit

Johannes Demel

Hauptreferent	:	Prof. Dr.rer.nat. Friedrich Jondral
Betreuer	:	Dipl.-Ing. Sebastian Koslowski

Beginn	:	1.5.2012
Abgabe	:	1.11.2012

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig und unter Beachtung der Satzung der Universität Karlsruhe (TH) zur Sicherung guter wissenschaftlicher Praxis in der aktuellen Fassung angefertigt habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und wörtlich oder inhaltlich übernommene Stellen als solche kenntlich gemacht.

Karlsruhe, den 31.11.2012

Johannes Demel

Zusammenfassung

In dieser Arbeit wird der Empfang von Long Term Evolution (LTE)-Signalen in GNU Radio vorgestellt. Dazu wird die Synchronisation und die Dekodierung der grundlegenden Systeminformationen einer LTE-Basisstation betrachtet.

LTE ist ein neuer Mobilfunkstandard, der in den nächsten Jahren flächendeckend in Deutschland und auch weltweit ausgebaut wird. Im Vergleich zu früheren Mobilfunkstandards wurde bei LTE, neben der Erhöhung der Datenrate, vor allem an der Optimierung des Gesamtsystems gearbeitet. Die Bandbreiteneffizienz konnte stark verbessert und die Latenzen reduziert werden. Während die Synchronisation bei früheren Mobilfunkstandards mehrere Sekunden benötigte, kann diese bei LTE innerhalb von 100 ms erfolgen. Außerdem können LTE Funkzellen eine viel höhere Anzahl Nutzer aufnehmen.

Da der LTE-Standard sehr umfangreich ist, wird in dieser Arbeit vornehmlich die Bitübertragungsschicht (PHY-Layer) betrachtet. Dazu werden zunächst die Grundlagen des LTE-Standards vorgestellt. LTE verwendet Orthogonal Frequency Division Multiplex (OFDM) zur Übertragung und es kommt Multiple Input Multiple Output (MIMO) zum Einsatz. Darauf aufbauend wird die Framestruktur, wie in Abbildung 0.1, betrachtet und damit die Notwendigkeit zur Synchronisation auf den Frametakt motiviert.

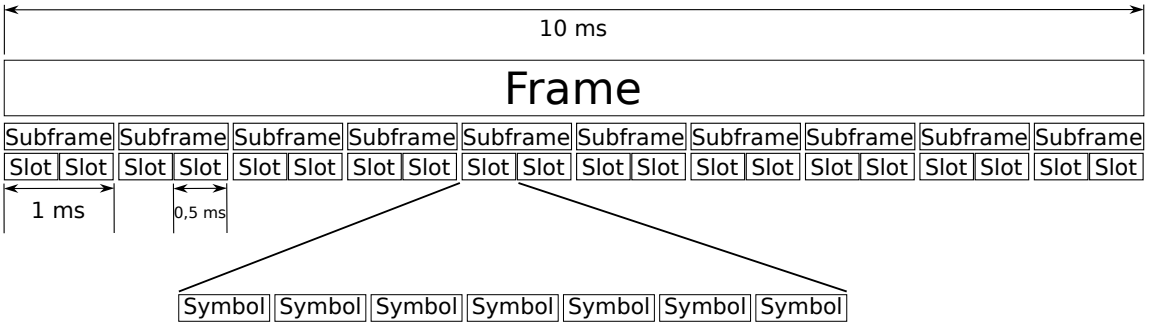


Abbildung 0.1.: Framestruktur in Zeitrichtung

Nach der Synchronisation auf den Frametakt ist die Dekodierung der grundlegenden Systeminformationen, die im Master Information Block (MIB) enthalten sind, möglich. Dieser wird durch die speziell dafür vorgesehenen logischen Kanäle – Broadcast Channel (BCH) und Physical Broadcast Channel (PBCH) – übertragen. In Abbildung 0.2 wird die Übertragung des MIB über diese logischen Kanäle schematisch dargestellt.

Auf die theoretischen Grundlagen und die Betrachtung des LTE-Standards folgt die Vorstellung der Implementierung des LTE-Empfängers in GNU Radio. GNU Radio ist ein Open Source Software Defined Radio (SDR) Framework, welches eine Vielzahl vorgefertigter Funktionen bietet und modular aufgebaut ist. Die Modularität hilft bei der schnel-

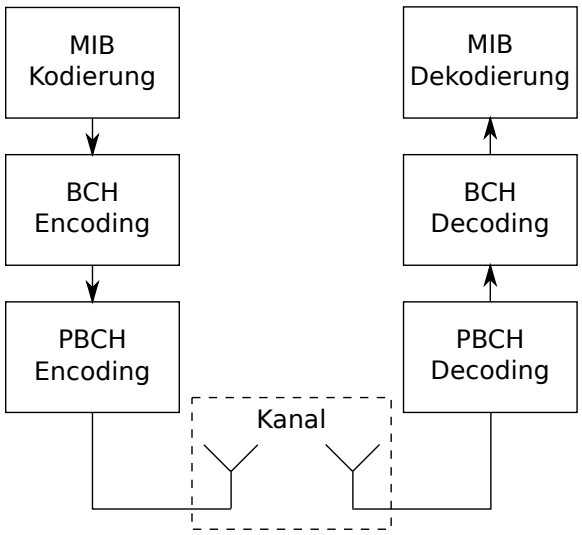


Abbildung 0.2.: Schematische Darstellung der MIB-Übertragung

leren Entwicklung neuer Anwendungen und der Anpassung bestehender Programme an neue Anforderungen. Bei der Implementierung des LTE-Empfängers wurde darauf geachtet, die Funktionen, die GNU Radio bereitstellt, nach Möglichkeit zu nutzen und neue Funktionen möglichst generisch zu halten, um diese wiederverwenden zu können. Der in GNU Radio enthaltene GNU Radio Companion (GRC) ist eine grafische Oberfläche, in der die einzelnen implementierten Blöcke visualisiert und zu einem Flowgraph verbunden werden. Der Flowgraph, der in dieser Arbeit implementiert wurde, ist in Abbildung 0.3 dargestellt.

Zur Verifikation des Flowgraphs wurde dieser mit dem Signal einer LTE-Basisstation getestet. Die Messung der LTE-Signale fand in Gondelsheim statt, da zum Zeitpunkt der Messung in Karlsruhe noch keine LTE-Basisstation verfügbar war. Zukünftige Messungen können auch in Karlsruhe durchgeführt werden, da zwischenzeitlich im Stadtgebiet LTE verfügbar ist. Der Messaufbau mit Laptop, Universal Software Radio Peripheral (USRP) und Spektrumanalysator ist in Abbildung 0.4 dargestellt. Die gemessenen Daten wurden gespeichert um sie im Verlauf der Entwicklung des LTE-Empfängers verwenden zu können und um die Leistungsfähigkeit der Implementierung zu testen.

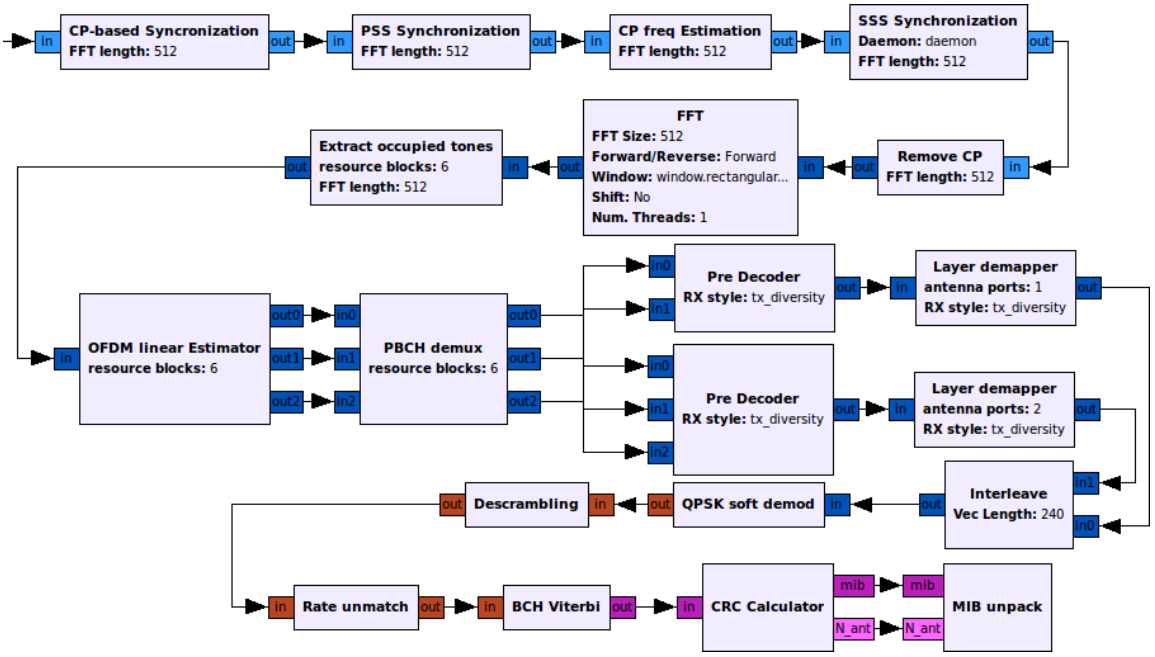


Abbildung 0.3.: Darstellung des kompletten GRC-Flowgraph

(a) **(b)**

Abbildung 0.4.: Aufnahmen des Messaufbaus

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AKF	Autokorrelationsfunktion
$N_{\mathbf{ant}}$	Anzahl der Antennenports
BCH	Broadcast Channel
BS	Basisstation
CAZAC	Constant Amplitude Zero Autocorrelation
CP	Cyclic Prefix
$CP_{\mathbf{0}}$	nulltes Cyclic Prefix
$N_{\mathbf{CP}}$	CP-Länge
$N_{\mathbf{CP,0}}$	CP_0-Länge
CRC	Cyclic Redundancy Check
FDD	Frequency Division Duplex
FFO	Fractional Frequency Offset
FFT	Fast Fourier Transform
$N_{\mathbf{FFT}}$	FFT-Länge
FSM	Finite State Machine
GRC	GNU Radio Companion
ICI	Inter Carrier Interference
IDFT	Inverse Diskrete Fourier Transformation
IFFT	Inverse Fast Fourier Transform
IFO	Integer Frequency Offset
ISI	Intersymbolinterferenz
KKF	Kreuzkorrelationsfunktion
LTE	Long Term Evolution

LUT	Look Up Table
MAC	Medium Access Control
MIB	Master Information Block
MIMO	Multiple Input Multiple Output
MS	Mobilstation
$N_{\mathbf{ID}}$	Cell ID
$N^{1}_{\mathbf{ID}}$	Cell Identity Group
$N^{2}_{\mathbf{ID}}$	Cell ID Number
$N^{\mathbf{DL}}_{\mathbf{RB}}$	Anzahl an Resource Blocks
NRZ	Non Return to Zero
OFDM	Orthogonal Frequency Division Multiplex
$N_{\mathbf{s}}$	OFDM Symbollänge
PBCH	Physical Broadcast Channel
PHICH	Physical Hybrid-ARQ Indicator Channel
PHY-Layer	Bitübertragungsschicht
PN	Pseudo-Noise
PSS	Primary Synchronization Symbol
QPSK	Quadrature Phase Shift Keying
RB	Resource Block
RE	Resource Element
SDR	Software Defined Radio
SFN	System Frame Number
$N_{\mathbf{Slot}}$	Slot Länge
SSS	Secondary Synchronization Symbol
STBC	Space Time Block Code
$T_{\mathbf{CP}}$	CP Dauer
$T_{\mathbf{CP,0}}$	nulltes Cyclic Prefix (CP_0) Dauer

$T_{\mathbf{CP,e}}$	extended CP Dauer
TDD	Time Division Duplex
$T_{\mathbf{F}}$	Framedauer
$T_{\mathbf{o}}$	OFDM Symboldauer
$T_{\mathbf{s}}$	Symboldauer
$T_{\mathbf{SF}}$	Subframedauer
USRP	Universal Software Radio Peripheral

1. Einleitung

Der limitierende Faktor bei mobiler Funkkommunikation ist nicht mehr die Datenverarbeitung in den Mobilgeräten, sondern die Datenrate. Dieses Limit wird durch immer effizientere Standards im Mobilfunk weiter angehoben. Daneben ist das elektromagnetische Spektrum, welches im Mobilfunk nutzbar ist, eine beschränkte Ressource. Die spektrale Effizienz noch weiter zu erhöhen, ist deshalb eines der zentralen Ziele zukünftiger Mobilfunkstandards.

Ältere Mobilfunkstandards erforderten komplexe Kanalschätzer, um eine mobile Datenübertragung überhaupt möglich zu machen. Der Einsatz von Orthogonal Frequency Division Multiplex (OFDM) in Mobilfunkstandards ermöglicht die Kanalschätzung und Entzerrung im Frequenzbereich durch einstufige Entzerrer. Darüber hinaus können, durch die Orthogonalität der Unterträger in OFDM-Systemen, die Schutzbänder entfallen und so die spektrale Effizienz gesteigert werden.

Long Term Evolution (LTE) ist der erste Mobilfunkstandard, der OFDM zur Übertragung nutzt. Im Vergleich zu vorherigen Mobilfunkstandards können zahlreiche Verbesserungen erzielt werden. Die spektrale Effizienz wird stark erhöht und die Flexibilität bei der Ressourcenzuweisung wird verbessert. Der LTE-Standard ist generell sehr flexibel aufgebaut, zum Beispiel sind verschiedene Systembandbreiten zwischen 1,4 MHz und 20 MHz möglich. Außerdem kann aus verschiedenen Multiple Input Multiple Output (MIMO)-Konfigurationen und unterschiedlichen Frequenzbändern je nach Bedarf und Verfügbarkeit ausgewählt werden. Die Anzahl Nutzer pro Zelle wird gesteigert, während die Komplexität des LTE-Gesamtsystems reduziert wird. Darüberhinaus wird die Reaktionszeit, sowie die Synchronisationszeit bis zur Anmeldung an der Basisstation (BS) stark reduziert.

GNU Radio ist ein Open Source Software Defined Radio (SDR) Framework. SDRs wurden in [Mit92] erstmals vorgeschlagen. Ziel ist es, die Signalverarbeitung komplett in Software zu realisieren, um maximale Flexibilität zu erhalten. Dadurch kann eine Wellenform schnell von einer Hardwareplattform auf eine andere portiert werden. Aber auch verschiedene Wellenformen auf der gleichen Hardwareplattform implementiert werden. Neue Funktionen können im Rapid Prototyping Prozess sofort evaluiert werden.

Mit GNU Radio ist ein leistungsfähiges Framework verfügbar, dass schon eine große Anzahl an Funktionen bietet und dem neue Funktionen schnell hinzugefügt werden können. Es ist bereits eine gut definierte Struktur von Schnittstellen vorhanden und zahlreiche Hilfsmittel zur Entwicklung werden bereitgestellt. Bei der Entwicklung des Frameworks wurde auf größtmögliche Modularität geachtet, um einzelne Module für verschiedene Anwendungsszenarien einsetzen zu können. Auf diese Weise kann sich ein Entwickler auf die Implementierung neuer Module konzentrieren ohne sich um die Details von de-

ren Steuerung zu kümmern.

Im Rahmen dieser Arbeit soll eine Implementierung eines Empfängers der LTE-Luftschnittstelle in GNU Radio entwickelt werden. Dazu wird die Synchronisation des Systems, die Kanalschätzung und die Dekodierung der grundlegenden LTE-Systemparameter in GNU Radio implementiert. Diese Systemparameter werden im Master Information Block (MIB) kodiert und durch den Broadcast Channel (BCH) und Physical Broadcast Channel (PBCH) übertragen.

Zunächst werden die theoretischen Grundlagen betrachtet, auf denen LTE basiert. Darauf aufbauend werden die entsprechenden Teile des Standards ausgiebig betrachtet. Hier wird auf die Synchronisationssymbole, die Pilotsymbole und die Übertragungskanäle – BCH und PBCH – eingegangen.

Danach wird die Implementierung in GNU Radio vorgestellt. Dazu werden die implementierten Teile des Standards vorgestellt und deren spezifische Besonderheiten in GNU Radio. Bei der Implementierung wird darauf geachtet nach Möglichkeit GNU Radio Funktionen zu verwenden und diese nur zu parametrisieren.

Abschließend wird die Funktionalität der entwickelten Anwendung vorgestellt und auf die verschiedenen Herausforderungen während der Entwicklung eingegangen. Dazu werden die Ergebnisse aus Simulation und Tests mit Messdaten vorgestellt. Außerdem wird ein Ausblick auf zukünftige Möglichkeiten zur Erweiterung der Anwendung gegeben.

2. Die LTE Luftschnittstelle

In diesem Kapitel werden die Grundlagen und der Aufbau der Long Term Evolution (LTE) Luftschnittstelle dargestellt. Es wird die Framestruktur, das Cyclic Prefix (CP) und die Synchronisation mithilfe von Synchronisationssymbolen eingeführt. Außerdem wird auf die Pilotsymbole und die grundlegenden Multiple Input Multiple Output (MIMO) Eigenschaften eingegangen. Abschließend werden die grundlegenden Systemparameter und die dafür relevanten Übertragungskanäle, der Physical Broadcast Channel (PBCH) und der Broadcast Channel (BCH), vorgestellt.

2.1. Grundlagen

LTE verwendet Orthogonal Frequency Division Multiplex (OFDM) für die Übertragung. Die angestrebten hohen Datenraten führen bei Einträgerübertragung zu einer sehr kurzen Symboldauer (T_s), wodurch die Intersymbolinterferenz (ISI), die durch Mehrwegeausbreitung entsteht, einen besonders starken negativen Einfluss hat. Der Vorteil von OFDM ist, dass hochratige Übertragungen in N_t niederratige parallele Datenströme, mit einer OFDM Symboldauer (T_o) von $N_t T_\mathrm{s}$ aufgeteilt werden.

Im Folgenden wird zunächst OFDM, wie in [Sch11], eingeführt. Die einzelnen OFDM Unterträger werden durch

$$\Psi_p = \exp\left(j2\pi f_p t\right) \frac{1}{\sqrt{T_o}} \, rect\left(\frac{t}{T_o}\right) , \quad p = 0, \ldots, N_t - 1 \tag{2.1}$$

dargestellt. Für diese Basisfunktionen wird die Einhaltung der Orthogonalitätsbedingung

$$\int_{-\infty}^{\infty} \Psi_p(t) \cdot \Psi_q^*(t) dt = \int_{0}^{T_o} \Psi_p(t) \cdot \Psi_q^*(t) dt = \begin{cases} 1: & p = q \\ 0: & p \neq q \end{cases} \tag{2.2}$$

gefordert. Daraus folgt auch der Minimale Trägerabstand Δf und die Unterträgerbandbreite B_t zu

$$\Delta f = \frac{1}{T_o} \quad \Rightarrow \quad \Delta f = B_t \tag{2.3}$$

Durch die Orthogonalität der einzelnen Unterträger ist zwischen den Unterträgern kein Schutzband, welches Inter Carrier Interference (ICI) verhindert, notwendig. Die resultierenden Trägerfrequenzen f_p sind durch

$$f_p = f_{min} + p \cdot \Delta f \quad p = 0, \ldots, N_t - 1 \tag{2.4}$$

gegeben. Außerdem wird mit

$$B_o = N_t \cdot \Delta f = N_t \cdot \frac{1}{T_o} = N_t \cdot \frac{1}{N_t T_s} = \frac{1}{T_s} = B_s \tag{2.5}$$

deutlich, dass die Systembandbreite eines OFDM-Systems B_o unverändert bleibt im Vergleich zur Bandbreite B_s des Einträgerübertragungssystems. Das OFDM Sendesignal wird im Zeitbereich durch

$$u(t) = \sum_{\mu=-\infty}^{\infty} \sum_{p=0}^{N_t-1} a_{k,l}(\mu N_t + p) \cdot \Psi_p(t - \mu T_o) \tag{2.6}$$

beziehungsweise

$$u(t) = \sum_{p=0}^{N_t-1} a_{k,l}(p) \cdot \exp\left(j2\pi f_p \cdot t\right), \quad 0 \leq t \leq T_o \tag{2.7}$$

für $\mu = 0$ repräsentiert. Die Folge $a_{k,l}$ stellt die diskreten Modulationssymbole, die übertragen werden sollen, dar.

Für die digitale Signalverarbeitung ist eine zeitdiskrete Darstellung des Signals notwendig. Vergleicht man die zeitdiskrete Darstellung des Signals

$$u(q) = u(t = q \cdot \frac{T_o}{N_t}) = \sum_{p=0}^{N_t-1} a_{k,l}(p) \cdot \exp\left(j\frac{2\pi}{N_t} \cdot p \cdot q\right), \quad q = 0, \ldots, N_t - 1 \tag{2.8}$$

mit der Inverse Diskrete Fourier Transformation (IDFT)

$$x(q) = \frac{1}{N} \sum_{p=0}^{N-1} X(p) \cdot \exp\left(j\frac{2\pi}{N} \cdot p \cdot q\right) = IDFT\{X(p)\}, \quad q = 0, \ldots, N - 1 \tag{2.9}$$

wird deutlich, dass die IDFT und das Sendesignal bis auf einen Vorfaktor identisch sind.

$$u(q) = N_t \cdot IDFT\{a_{k,l}(p)\}, \quad q = 0, \ldots, N_t - 1 \tag{2.10}$$

Die Berechnung eines diskreten OFDM Sendesignals ist also durch die IDFT durchführbar. Im Folgenden wird meist der Begriff Inverse Fast Fourier Transform (IFFT) genutzt, da diese als effizientere Implementierung der IDFT oft praktisch eingesetzt wird.

2.1.1. Cyclic Prefix

Als Schutzinterval in einem OFDM-System dient das Cyclic Prefix (CP). Dazu wird das Ende des OFDM-Symbols, wie in Abbildung 2.1 dargestellt, zyklisch wiederholt. Die Länge des CP wird so gewählt, dass möglichst alle Störungen, die durch Mehrwegeausbreitung entstehen, eine kleinere Zeitverzögerung gegenüber dem direkten Signal haben, als die Zeitdauer des CP. Bei Zeitsynchronisationsfehlern eines OFDM-Systems führt eine zum Symbol zu frühe Synchronisation, falls noch Platz ist, so nur zu einer Phasendrehung und nicht zu Intersymbolinterferenz (ISI). Dies ist erforderlich, weil in OFDM-Systemen, die Symbole durch die Fast Fourier Transform (FFT) implizit periodisch fortgesetzt

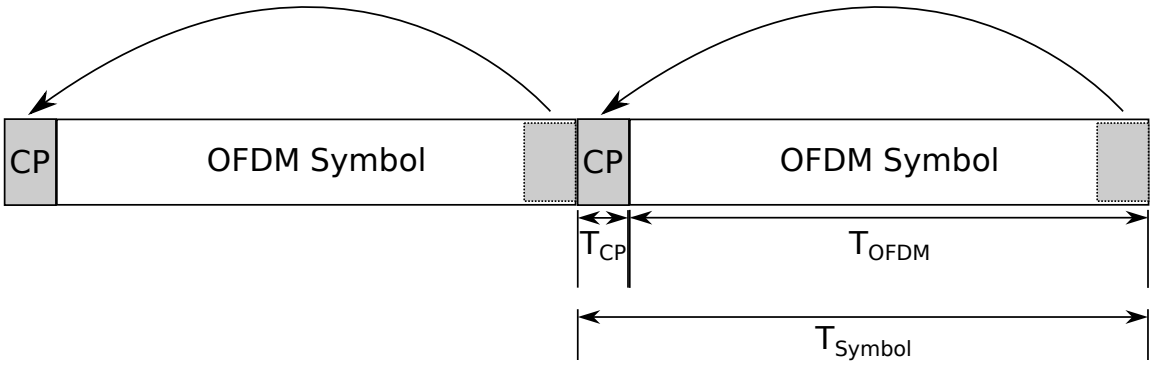


Abbildung 2.1.: CP Schema in OFDM-Systemen

werden. ISI zerstört die Orthogonalität der einzelnen OFDM-Symbole und somit die Möglichkeit durch eine IFFT die Trennung der Träger durchzuführen.

Im LTE-Standard ist eine normale und eine extended CP-Länge (N_{CP}) spezifiziert. Bei einer Konfiguration mit extended CP, hat jedes CP eine extended CP Dauer ($T_{\mathrm{CP,e}}$) von 16,896 µs. In einem Slot werden dann sechs OFDM-Symbole übertragen. Im Fall normaler CP-Länge hat das erste OFDM-Symbol in einem Slot eine längere Dauer als die folgenden OFDM-Symbole im Slot und es werden sieben OFDM-Symbole übertragen. Das nulltes Cyclic Prefix (CP_0) hat dann eine CP_0 Dauer ($T_{\mathrm{CP,0}}$) von 5,28 µs und die anderen eine CP Dauer (T_{CP}) von 4,752 µs. Auf die Slot- und damit Framestruktur wird in Abschnitt 2.2 eingegangen.

2.1.2. OFDM Sender- und Empfängerstruktur

In Abbildung 2.2 ist die OFDM Senderstruktur dargestellt. Es werden Binärdaten auf komplexe Modulationssymbole $a_{k,l}$ abgebildet und seriell-parallel gewandelt. In der OFDM-Operation werden diese auf die N_t Unterträger um den DC-Carrier herum angeordnet. Der DC-Carrier selbst wird zu null gesetzt. Die so entstandenen Vektoren werden invers fouriertransformiert, parallel-seriell gewandelt und ein CP eingefügt. Nach der OFDM-Operation werden die digitalen Abtastwerte digital-analog gewandelt und Tiefpass-gefiltert. Am Ausgang liegt ein komplexes Sendesignal $u(t)$ im Basisband vor. Analog zur

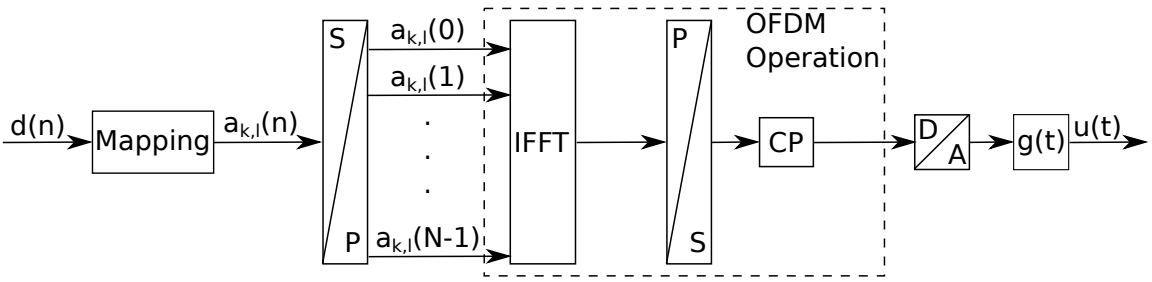


Abbildung 2.2.: OFDM Senderstruktur mit anschließendem Digital-Analog-Wandler und Tiefpassfilter $g(t)$

Senderstruktur ist in Abbildung 2.3 die Empfängerstruktur dargestellt. Die IFFT, die im Sender durchgeführt wurde, wird im Empfänger durch eine FFT rückgängig gemacht.

Auf die Funktion der Blöcke *CP* und *rm CP* wird in 2.1.1 eingegangen. Die Blöcke zwischen den Blöcken *FFT* und *parallel-seriell Wandler* im Empfänger werden in 2.3 genauer betrachtet. Werden im Empfänger nicht die Abtastwerte die einer Periode entsprechen gemeinsam fouriertransformiert, sondern es sind Anteile des vorherigen oder nachfolgenden OFDM-Symbol enthalten, dann wird die Orthogonalitätsbedingung verletzt. Dies führt zu ISI.

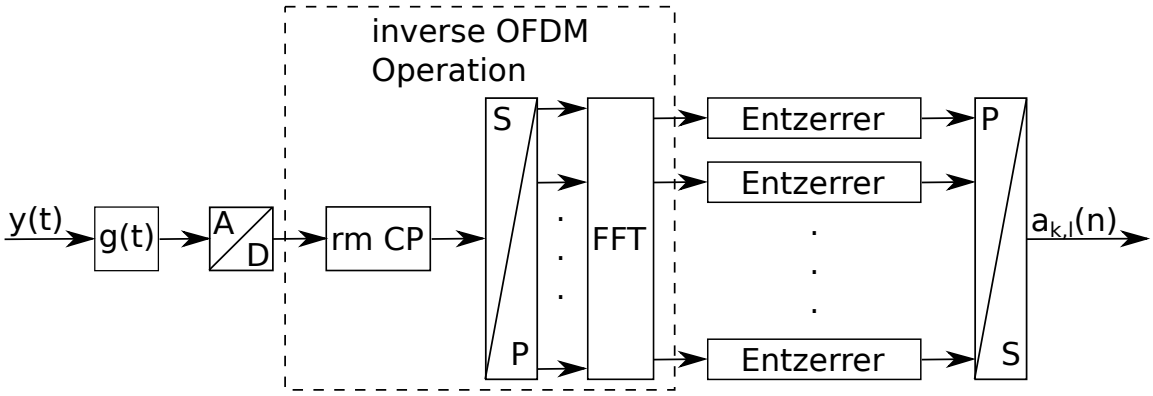


Abbildung 2.3.: OFDM Empfängerstruktur mit analogem Eingangssignal $y(t)$ und empfangenen Modulationssymbolen $\tilde{a}_{k,l}(n)$

2.1.3. Systembandbreite

Die Systembandbreite B_s ist durch die Anzahl der Resource Blocks (RBs) bestimmt. Ein RB repräsentiert jeweils 12 Unterträger mit je $\Delta f = 15\,\text{kHz}$ Bandbreite, sodass die in Tabelle 2.1 angegebenen Beziehungen zwischen den Werten bestehen (vgl. [3GP12] Tabelle 5.6-1). Die Werte für die Bandbreite enthalten die Schutzbänder, die dem System hinzugefügt werden. In Abschnitt 2.2 wird auf die genaue Framestruktur eingegangen und die weitere Bedeutung des RB erläutert.

Bandbreite B_s	Anzahl RBs	N_t Unterträger
1,4 MHz	6	72
3 MHz	15	180
5 MHz	25	300
10 MHz	50	600
15 MHz	75	900
20 MHz	100	1200

Tabelle 2.1.: Zuordnung der Bandbreite zur Anzahl der RBs

2.2. Framestruktur

Um eine effiziente Ressourcenzuteilung realisieren zu können, werden OFDM-Symbole in größere Einheiten, den Frames, gruppiert. Der LTE-Standard definiert zwei Modi, Frequency Division Duplex (FDD) und Time Division Duplex (TDD). Bei TDD senden und empfangen Basisstation (BS) und Mobilstation (MS) abwechselnd auf derselben Frequenz. Im FDD-Modus senden BS und die MS kontinuierlich auf verschiedenen Frequenzen. Im Folgenden wird nur auf den FDD-Modus eingegangen. In Abbildung 2.4 ist die zeitliche Struktur eines LTE Frames dargestellt. Frames mit einer Framedauer T_F =10 ms sind unterteilt in zehn Subframes mit einer Subframedauer T_{SF} =1 ms. Jeder Subframe ist nochmals in zwei Slots unterteilt, mit jeweils sieben OFDM Symbolen. Jeder Subframe enthält somit vierzehn OFDM-Symbole und jeder Frame besteht aus insgesamt 140 OFDM-Symbolen. Im Modus mit extended CP besteht ein Slot nur aus sechs OFDM-Symbolen (vlg. 2.1.1).

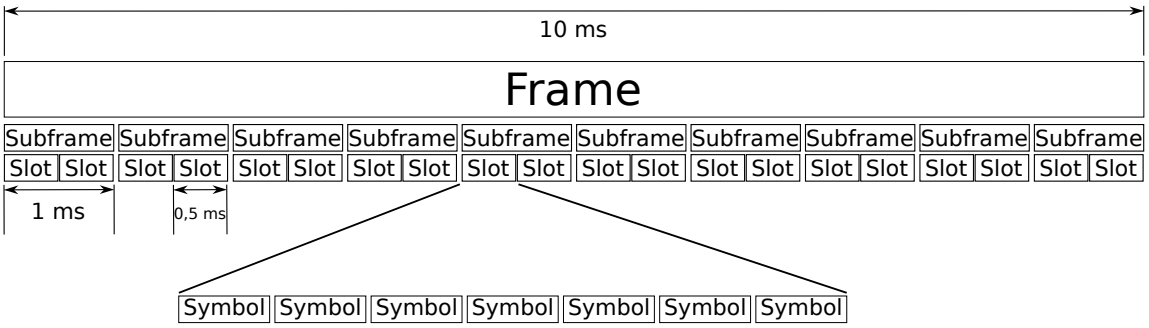


Abbildung 2.4.: Framestruktur in Zeitrichtung

Ein Modulationssymbol wird Resource Element (RE) genannt und kann bis zu sechs Bit Daten enthalten. Für effektives Scheduling werden die Resource Elements in Resource Blocks (RBs) gruppiert. Diese bestehen aus zwölf REs in Frequenzrichtung und den sieben REs eines Slots in Zeitrichtung. Abbildung 2.5 stellt einen RB in der Zeitfrequenzebene dar. Es wird je nach MIMO-Konfiguration für jeden Antennenport eine extra Ebene genutzt.

2.3. Synchronisation

Im LTE-Empfänger liegen die Empfangsdaten zunächst weder Zeit- noch Frequenzsynchronisiert vor. Aus diesem Grund muss eine Synchronisation durchgeführt werden. Diese wird in vier Schritte unterteilt, der Symboltakt-, Halbframetakt-, Frametakt- und Frequenzsynchronisation. Die CP-basierte Methode synchronisiert auf den Symboltakt. Für die Halbframetaktsynchronisation wird das Primary Synchronization Symbol (PSS) eingesetzt. Die Frametaktsynchronisation wird mithilfe des Secondary Synchronization Symbol (SSS) durchgeführt.

PSS und SSS sind Synchronisationssymbole, die in den letzten beiden OFDM-Symbolen

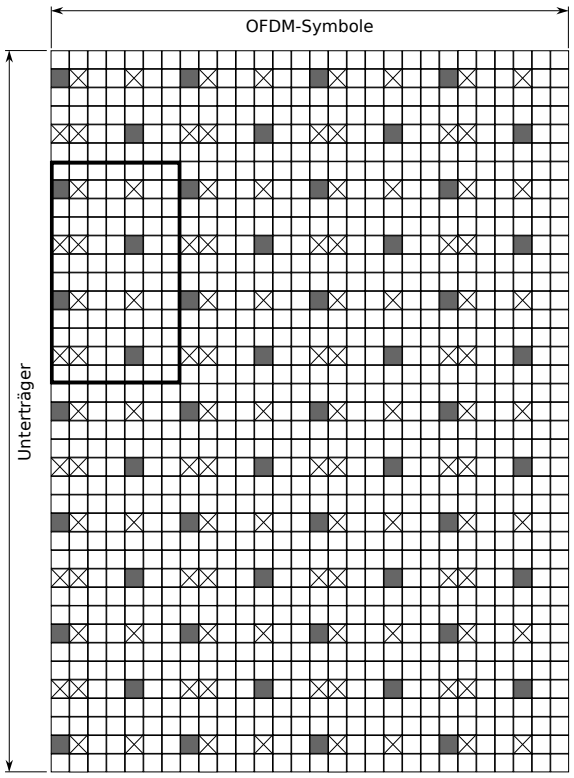


Abbildung 2.5.: Darstellung der Zeitfrequenzebene, roter Rahmen markiert einen RB

des nullten und zehnten Slots – Slot_0 und Slot_{10} – in einem Frame übertragen werden. Sie werden jeweils auf den 62 Unterträgern direkt um den DC-Carrier herum übertragen.

Die Frequenzoffset ist zweigeteilt in Integer Frequency Offset (IFO) und Fractional Frequency Offset (FFO). Der IFO stellt den Frequenzoffset um Vielfache des Unterträgerabsands dar und kann bei der Detektion des PSS geschätzt werden. Der FFO beschreibt den Frequenzoffset innerhalb eines Unterträgers. Die FFO Schätzung erfolgt mithilfe von CPs.

2.3.1. Symboltaktsynchronisation

Der erste Synchronisationsschritt in einem LTE-System ist die Symboltaktsynchronisation des Empfängers. Dazu wird das CP genutzt. Zur Detektion des Symbolbeginns wird eine Korrelation

$$\gamma(n) = \sum_{m=n}^{n+N_{\text{CP}}-1} r(m)\, r^*(m - N_{\text{FFT}}) \tag{2.11}$$

mit festem Versatz FFT-Länge (N_{FFT}) genutzt. Das CP und der letzte Teil des OFDM-Symbols sind korreliert, ansonsten sind diese unkorreliert, wie in Abbildung 2.6 dargestellt. Zur Synchronisation wird nun eine Fensterung vorgenommen, die über die OFDM-

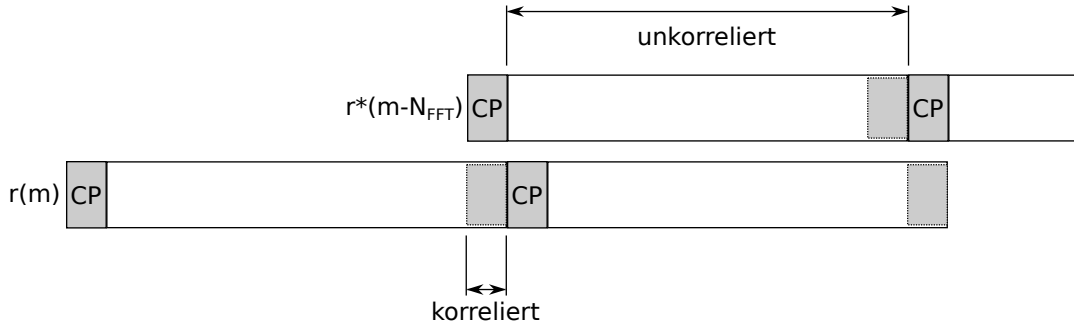


Abbildung 2.6.: Cyclic Prefix Schema mit korreliertem und unkorreliertem Teil

Symbole geschoben wird. $\gamma(n)$ wird dann maximal, wenn im Korrelationsfenster gerade das CP und seine verschobene Version zu finden sind. Durch

$$n_{sym,a} = \arg\max_n |\gamma(n)| \tag{2.12}$$

kann so der Symbolbeginn $n_{sym,a}$ detektiert werden, [MEJ+09]. Für jedes OFDM-Symbol resultiert ein Maximum. Über diese Maxima gemittelt wird der OFDM-Symbolbeginn bestimmt. Dazu ist in Abbildung 2.7 der entstehende Verlauf von $\gamma(n)$ dargestellt.

Die Berechnung der Autokorrelationsfunktion (AKF) eines CP kann auch zur Schätzung des FFO genutzt werden. In Gleichung 2.11 ist zu erkennen, dass bei exakter Gleichheit von CP und seiner zeitlich verschobenen Version, der Wert von $\gamma(n)$ rein reell sein muss. Ein FFO führt aber zu einer Phasendrehung zwischen CP und seinem Original. Dieser Frequenzoffset kann durch

$$f_{fo} = \frac{f_s}{2\pi N_{FFT}} \angle\gamma(n_{sym,a}) \tag{2.13}$$

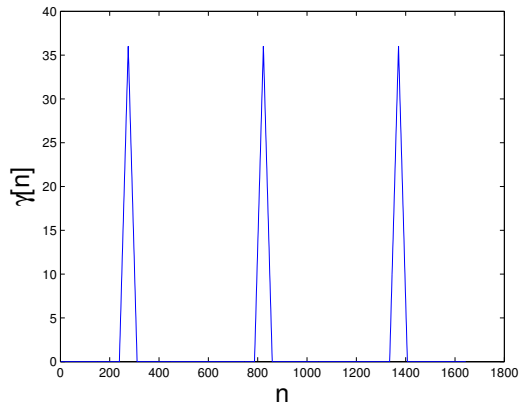


Abbildung 2.7.: Korrelation und Fensterung nach $\gamma(n)$

berechnet und zur Frequenzkorrektur genutzt werden.

2.3.2. Cell ID

Die Cell ID (N_{ID}) charakterisiert jede Basisstation (BS), wobei insgesamt 504 verschiedene N_{ID}s möglich sind. Die N_{ID} setzt sich aus der Cell Identity Group (N^1_{ID}) und Cell ID Number (N^2_{ID}) durch

$$N_{\mathrm{ID}} = 3 \cdot N^1_{\mathrm{ID}} + N^2_{\mathrm{ID}} \tag{2.14}$$

zusammen. N^1_{ID} und N^2_{ID} sind definiert durch:

$$\begin{array}{rcl} N^1_{\mathrm{ID}} & = & 0, \ldots, 167 \\ N^2_{\mathrm{ID}} & = & 0, 1, 2 \end{array} \tag{2.15}$$

Die N_{ID} wird an verschiedenen Stellen im LTE-Standard genutzt um eine zellspezifische Charakterisierung zu erhalten, wie beispielsweise bei der Berechnung der Scramblingsequenz.

2.3.3. Primary Synchronization Symbol

Das Primary Synchronization Symbol (PSS) wird zur Synchronisation auf den Halbframetakt und zur Detektion der Cell ID Number (N^2_{ID}) verwendet.Dazu wird eine Zadoff-Chu Sequenz der Länge $N_{\mathrm{ZC}} = 63$ generiert, [Chu72]. Die Berechnung der Sequenz ist durch

$$d_u(n) = \exp\left(-j\,\frac{\pi\, u\, n\,(n+1)}{N_{\mathrm{ZC}}}\right) \quad n = 0, 1, 2, ..., N_{\mathrm{ZC}} - 1 \tag{2.16}$$

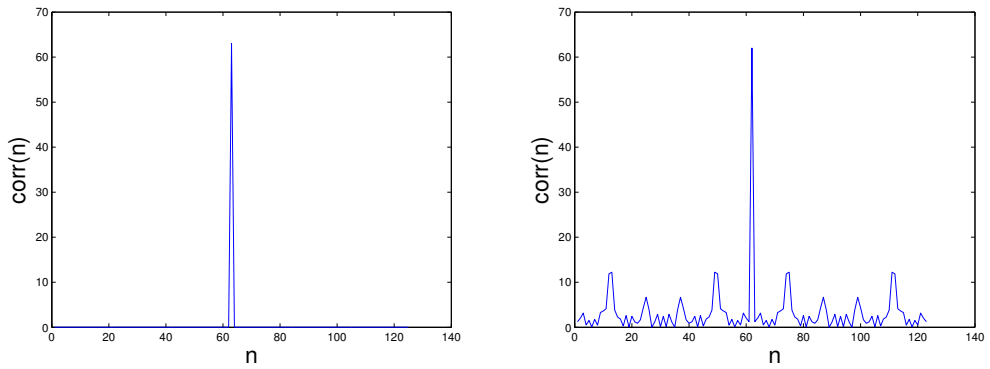
gegeben. Der Parameter u ist bei LTE abhängig von der N^2_{ID} und kann die Werte 25, 29 oder 34 annehmen. Das 32. Element der Sequenz wird zu null gesetzt, da dieses der Position des DC-Carriers entspricht, [3GP11b]. Die Zadoff-Chu Sequenzen gehören zur Klasse der Constant Amplitude Zero Autocorrelation (CAZAC) Sequenzen, dass heißt ihre komplexe

Einhüllende ist konstant, wodurch beispielsweise günstigere Verstärker, mit kleinerem linearen Bereich, zum Einsatz kommen können. Eine Zadoff-Chu Sequenz ist orthogonal zu allen zyklischen Verschiebungen dieser Sequenz und der Wert der Kreuzkorrelationsfunktion (KKF) zu Zadoff-Chu Sequenzen gleicher Länge mit anderen u ist konstant gegeben durch $\frac{1}{\sqrt{N_{ZC}}}$. Für Zadoff-Chu Sequenzen wird weiter gefordert das u und N_{ZC} teilerfremd sind.

Mit der Extraktion der genutzten Unterträger aus den OFDM-Symbolen, kann durch Berechnung der KKF und der Entscheidung für das Maximum auf die genutzte Zadoff-Chu Sequenz und die N_{ID}^2 entschieden werden. Der entfernte Wert, der dem DC-Carrier entspricht, wird insofern beachtet, dass auch in der Vergleichssequenz im Empfänger, dieser Wert aus der Sequenz entfernt wird. Dadurch wird die Länge der Zadoff-Chu Sequenz auf $N_{ZC} - 1$ verringert. Die KKF wird durch

$$Z(n) = \sum_{k=0}^{N_{ZC}-1-1} z(k)\, r^*(k+n-61) \quad n = 0, 1, 2, ..., 2(N_{ZC} - 1 - 1) \tag{2.17}$$

berechnet. $z(k)$ ist dabei die Referenzsequenz und $r^*(k+n-61)$ die um $n-61$ verschobene komplex konjugierte Empfangssequenz. Der IFO kann durch die Abweichung des Maximums der KKF von der Position $N_{ZC} - 1 - 1$ bestimmt werden. In Abbildung 2.8 ist die zyklische KKF einer Zadoff-Chu Sequenz dargestellt. Mit dem Korrelationsmaximum wird die N_{ID}^2 ermittelt und mit der Verschiebung des Maximums von der Position $n = N_{ZC} - 1 - 1$ der berechneten Werte, kann der IFO ermittelt werden.



(a) ideal **(b)** DC-Carrier entfernt

Abbildung 2.8.: Zyklische KKF einer Zadoff-Chu Sequenz, wie sie bei $N_{ID}^2 = 1$ eingesetzt wird.

Nach der Synchronisation auf den PSS Takt, ist der Empfänger auf Basis eines halben Frames – eines 5ms Rasters – synchronisiert.

2.3.4. Secondary Synchronization Symbol

Das Secondary Synchronization Symbol (SSS) wird zur Synchronisation auf den Frametakt und zur Detektion der Cell Identity Group (N_{ID}^1) genutzt. Es besteht aus zwei, je nach Slot unterschiedlich, verschachtelten m-Sequenzen. Diese Unterscheidung nach Position

des Slots im Frame ermöglicht die Frametaktsynchronisation. Das SSS wird, wie das PSS, auf 62 Unterträger direkt um den DC-Carrier herum, im vorletzten OFDM-Symbol in Slot_0 und Slot_{10}, gesendet. Es trägt die Sequenz $d(n)$, wie in [3GP11b] Abschnitt 6.11.2 definiert. Die grundlegenden Sequenzen sind im folgenden dargestellt. Eine ausführliche Darstellung ist im Anhang A zu finden.

$$\begin{aligned} d(2n) &= \begin{cases} s_0^{(m_0)}(n)c_0(n) & \text{in Subframe 0 / slot 0} \\ s_1^{(m_1)}(n)c_0(n) & \text{in Subframe 5 / slot 10} \end{cases} \\ d(2n+1) &= \begin{cases} s_1^{(m_1)}(n)c_1(n)z_1^{m_0}(n) & \text{in Subframe 0 / slot 0} \\ s_0^{(m_0)}(n)c_0(n)z_1^{m_1}(n) & \text{in Subframe 5 / slot 10} \end{cases} \end{aligned} \tag{2.18}$$

Die Indizes m_0 und m_1 werden mit der N_{ID}^1 wie folgt berechnet:

$$\begin{aligned} m_0 &= m' \mod 31 \\ m_1 &= (m_0 + \lfloor m'/31 \rfloor + 1) \mod 31 \\ m' &= N_{\text{ID}}^1 + q(q+1)/2 \\ q &= \lfloor \tfrac{N_{\text{ID}}^1 + q'(q'+1)/2}{30} \rfloor \\ q' &= \lfloor N_{\text{ID}}^1/30 \rfloor \end{aligned} \tag{2.19}$$

2.4. Pilotsymbole

Zur Entzerrung des OFDM Empfangssignals werden in jedem Slot, in jedem OFDM-Symbol_0 und OFDM-Symbol_4, Pilotsymbole gesendet. Zwischen dem OFDM-Symbol_0 und OFDM-Symbol_4 sind drei und zwischen OFDM-Symbol_4 und OFDM-Symbol_0 des nächsten Slots sind zwei OFDM-Symbole ohne Piloten. In [3GP11b] Abschnitt 6.10.1 werden Zellen-spezifische Pilotsymbole beschrieben, deren Aufbau im Folgenden dargestellt werden soll. Die komplexen Pilotsymbole werden durch

$$r_{l,n_s}(m) = \frac{1}{\sqrt{2}}\{(1-2\,c(2m)) + j(1-2\,c(2m+1))\} \quad m = 0, 1, \ldots, 2N_{\text{RB}}^{\max,\text{DL}} - 1 \tag{2.20}$$

Non Return to Zero (NRZ) kodiert. Durch n_s wird die Slotnummer im Frame und durch l die Symbolnummer im Slot bezeichnet. Der Standard definiert die Konstante $N_{\text{RB}}^{\max,\text{DL}} = 110$. Die PN-Folge $c(n)$ wird in [3GP11b] Abschnitt 7.2 definiert (vgl. Anhang B) und mit

$$\begin{aligned} c_{init} &= 2^{10}(7(n_s+1)+l+1)(2N_{\text{ID}}+1) + 2N_{\text{ID}} + N_{\text{CP}} \\ \text{mit} \quad N_{\text{CP}} &= \begin{cases} 1 & \text{für normales CP} \\ 0 & \text{für extended CP} \end{cases} \end{aligned} \tag{2.21}$$

für jedes OFDM-Symbol neu initialisiert. Die Referenzsymbole r_{l,n_s} werden abhängig vom Antennenport p, der Symbolnummer l und der N_{ID} auf die REs abgebildet. Dazu werden

$$v_{shift} = N_{\text{ID}} \mod 6 \tag{2.22}$$

$$v = \begin{cases} 0 & \text{für } p = 0 \text{ und } l = 0 \\ 3 & \text{für } p = 0 \text{ und } l \neq 0 \\ 3 & \text{für } p = 1 \text{ und } l = 0 \\ 0 & \text{für } p = 1 \text{ und } l \neq 0 \\ 3(n_s \mod 2) & \text{für } p = 2 \\ 3 + 3(n_s \mod 2) & \text{für } p = 3 \end{cases} \tag{2.23}$$

und die Indizes

$$\begin{aligned} k &= 6m + (v + v_shift) \mod 6 \\ l &= \begin{cases} 0, N_{\text{symb}}^{\text{DL}} - 3 & \text{für } p = 0, 1 \\ 1 & \text{für } p = 2, 3 \end{cases} \\ m &= 0, 1, \ldots, 2N_{\text{RB}}^{\text{DL}} - 1 \\ m' &= m + N_{\text{RB}}^{\text{max,DL}} - N_{\text{RB}}^{\text{DL}} \end{aligned} \tag{2.24}$$

definiert. Die Pilotsymbole werden nun auf die komplexen Modulationssymbole

$$a_{k,l}^{(p)} = r_{l,n_s}(m') \tag{2.25}$$

abgebildet. In der Zeitfrequenzebene ergibt sich die in Abbildung 2.9 dargestellte Anordnung mit den gefüllten Kästen, die die REs mit Pilotsymbolen repräsentieren. Um in einem MIMO-System jede der empfangenen Zeitfrequenzebenen entzerren zu können müssen die Positionen mit Pilotsymbolen in anderen Zeit/Frequenzebenen in der betrachteten Zeitfrequenzebene zu null gesetzt werden. Die Kästen mit Kreuzen repräsentieren diese REs die zu null gesetzt werden. Beispielsweise für den Empfang des PBCH wird immer eine 4x4 MIMO Antennenkonfiguration angenommen, da in diesem Fall die Antennenkonfiguration noch unbekannt ist. Daraus ergibt sich die in Abbildung 2.9 abgebildete Anordnung von Pilotsymbolen und Resource Elements, die zu null gesetzt werden. Bei einer zwei Antennenkonfiguration können auf den Positionen mit Kreuz in OFDM-Symbol$_1$ Nutzdaten übertragen werden.

2.5. MIMO

Zur Verbesserung der Diversity und oder der Datenrate wird in LTE MIMO eingesetzt. Dies führt zum Einsatz von Space Time Block Codes (STBCs) – oder Alamouti Codes – für eine bessere Diversity. Das in [Ala98] beschriebene Alamouti-Schema

$$\begin{array}{c} \\ \text{timeslot}_0 \\ \text{timeslot}_1 \end{array} \begin{array}{c} \begin{array}{cc} \text{ant}_0 & \text{ant}_1 \end{array} \\ \begin{pmatrix} x^{(0)} & x^{(1)} \\ -x^{(1)*} & x^{(0)*} \end{pmatrix} \end{array} \tag{2.26}$$

wird auf zwei Antennen übertragen. Dabei stellen $x^{(0)}$ und $x^{(1)}$ die Zuweisung der einzelnen Modulationssymbole zu den Antennen dar. Im Empfänger wird

$$\begin{pmatrix} r_0 \\ r_1 \end{pmatrix} = \begin{pmatrix} x^{(0)} & x^{(1)} \\ -x^{(1)*} & x^{(0)*} \end{pmatrix} \tag{2.27}$$

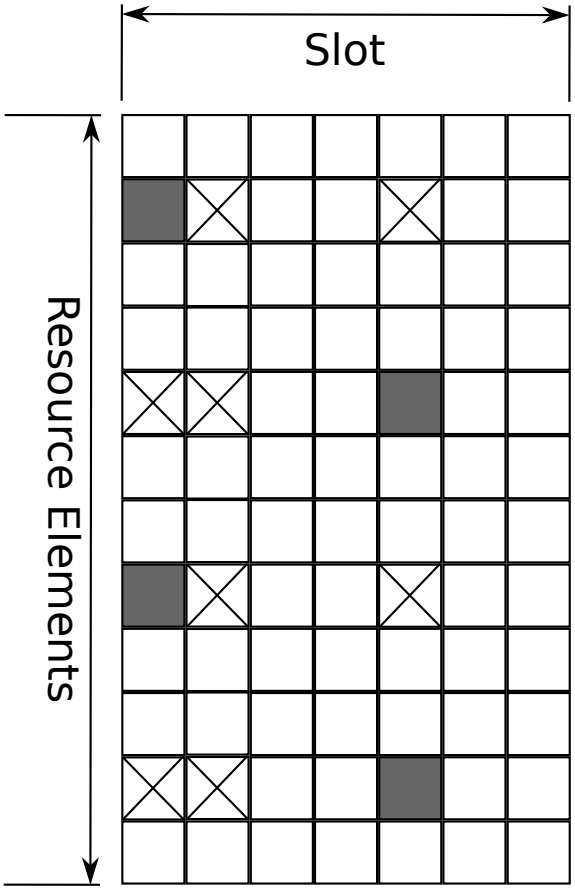


Abbildung 2.9.: Pilotsymbole in der Zeitfrequenzebene

verarbeitet, indem das Alamouti-Schema umgekehrt wird und die Empfangssymbole geschätzt werden.

$$\begin{pmatrix} \bar{x}^{(0)} \\ \bar{x}^{(1)} \end{pmatrix} = \begin{pmatrix} r_0 & r_1^* \\ r_0 & -r_1^* \end{pmatrix} \tag{2.28}$$

Mit 2.27 eingesetzt in 2.28 ergibt sich

$$\begin{array}{rcccl} \bar{x}^{(0)} & = & x^{(0)} + x^{(1)} + x^{(0)} - x^{(1)} & = & 2x^{(0)} \\ \bar{x}^{(1)} & = & x^{(0)} + x^{(1)} - x^{(0)} + x^{(1)} & = & 2x^{(1)} \end{array} \tag{2.29}$$

In Gleichung 2.29 wird deutlich, dass durch das Alamouti-Schema die Signalenergie der einzelnen Modulationssymbole im Empfänger verdoppelt werden kann. Weitere Vorteile sind, dass Kanaleinflüsse, wie schnelles Fading, einen geringeren Einfluss haben, da die Übertragungskanäle der einzelnen Antennen voneinander unabhängig sind.

Im LTE-Standard wird MIMO eingesetzt um die Diversity oder die Datenrate zu erhöhen. Die genauen Parameter sind jedoch abhängig vom logischen Kanal und von den aktuellen Übertragungsbedingungen. Der PBCH beispielsweise nutzt immer MIMO zur Verbesserung der Diversity. Desweiteren wird für jede Antenne ein eigener Frame berechnet.

2.5.1. Antennenkonfiguration

Die Antennenkonfiguration bei LTE wird durch die Anzahl der Antennenports (N_{ant}) definiert. Durch diesen Systemparameter ist die MIMO Konfiguration – 1x1, 2x2 oder 4x4 MIMO– in LTE festgelegt.

2.6. Downlink Kanäle

LTE ist durch zahlreiche Werte parametrisiert, die die BS periodisch senden muss, um den MSs die Synchronisation und Anmeldung am System zu ermöglichen. Im Rahmen dieser Arbeit wird auf den Empfang und die Dekodierung der grundlegenden LTE-Systeminformationen eingegangen, die im Master Information Block (MIB) enthalten sind. Auf der Bitübertragungsschicht (PHY-Layer) sind die physikalischen Kanäle, mit der Modulation und Zuordnung in der Zeitfrequenzebene, definiert. Die Medium Access Control (MAC) Schicht stellt logische Kanäle bereit und definiert die Sicherungsverfahren, wie beispielsweise eine Faltungscodierung. Den logischen Kanälen sind direkt physikalische Kanäle zugeordnet. In diesem Abschnitt wird der BCH, ein logischer Kanal, und der PBCH, der zum BCH gehörige physikalische Kanal, betrachtet, [GZMA10].

In Abbildung 2.10 sind die einzelnen Teile der Übertragung dargestellt, die im Folgenden näher betrachtet werden. Zunächst wird der Datenfluss im BCH und PBCH betrachtet. Abschließend wird auf die Bedeutung der einzelnen Teile des MIB eingegangen. Teile der im MIB übertragenen Systemdaten stehen in direktem Zusammenhang mit im BCH und PBCH kodierten Daten.

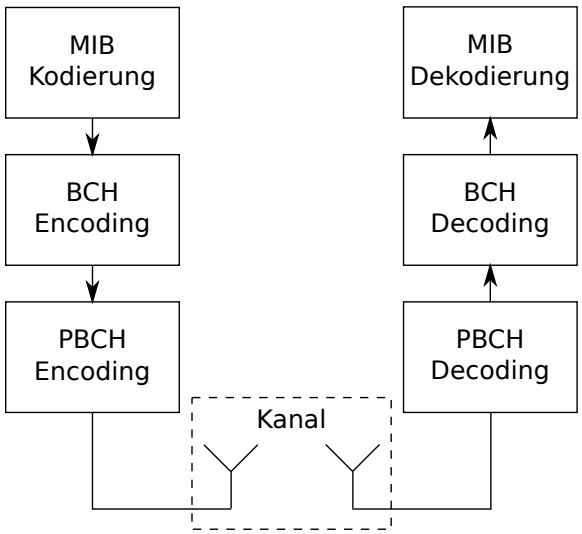


Abbildung 2.10.: Schematische Darstellung der MIB-Übertragung

2.6.1. Datenfluss im BCH

Der Datenfluss im BCH ist in Abbildung 2.11 dargestellt und in [3GP11a] beschrieben.

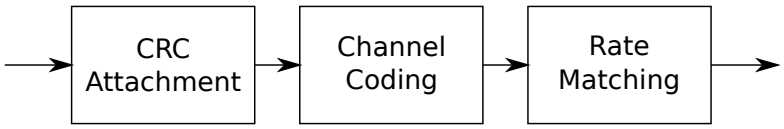


Abbildung 2.11.: Datenfluss des BCH

CRC-Berechnung

An den 24 Bit langen MIB wird ein 16 Bit Cyclic Redundancy Check (CRC) Prüfwort angehängt, dass durch das Generatorpolynom,

$$g_{CRC16}(D) = [D^{16} + D^{12} + D^5 + 1] \tag{2.30}$$

berechnet wird. Für jede mögliche Antennenkonfiguration wird danach das Prüfwort mit einer spezifischen Scramblingsequenz, wie in Tabelle 2.2 angegeben, belegt. Im Empfänger wird durch eine erfolgreiche CRC Prüfung in Verbindung mit der verwendeten Scramblingsequenz auf N_{ant} entschieden.

Channel Coding

Das Channel Coding wird für die 40 Bit Sequenz nach der CRC-Berechnung durchgeführt. In diesem Block wird eine *tail-biting* Faltungskodierung mit einer Coderate von ein drittel und einer Einflusslänge von sieben, wie in Abbildung 2.12, durchgeführt. Die *tail-biting*

N_{ant}	Scramblingsequenz
1	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2	1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
4	0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1

Tabelle 2.2.: Scramblingsequenzen für die Kodierung der N_{ant} im CRC-Prüfwort

Eigenschaft bedeutet, dass der Initialzustand des Faltungskodierers dem Endzustand entspricht. Durch das Channel Coding erhöht sich die Anzahl der Bits auf 120. Im Empfänger

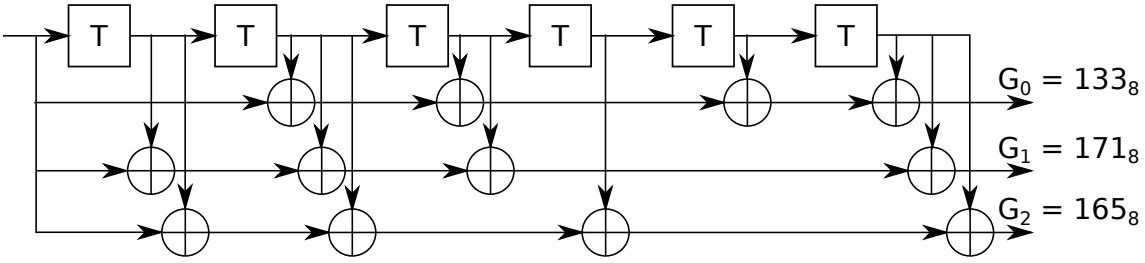


Abbildung 2.12.: Darstellung des Faltungsencoders

werden durch eine Viterbiberechnung die kodierten Bits wieder gewonnen. Auf Grund der *tail-biting* Eigenschaft ist der Anfangs- und Endzustand des Faltungskodierers unbekannt. Daher wird im Empfänger durch doppelte Viterbiberechnung sichergestellt, dass am Anfang des zweiten Berechnungsdurchlaufs, der Dekodierer im richtigen Zustand ist.

Rate Matching

Das Rate Matching, wie in Abbildung 2.11 dargestellt, beginnt mit dem Subblock Interleaving, einem Matrix-Interleaving. Es wird für jeden der drei Sequenzen – G_0, G_1, G_2 – aus Abbildung 2.12 getrennt durchgeführt. Im Folgenden wird beispielhaft G_0 verwendet, dies ist aber auf die anderen Sequenzen übertragbar. Das Interleaving wird durch Spaltenpermutation einer Matrix erreicht. Für den BCH wird eine Matrix mit 32 Spalten und zwei Zeilen, also insgesamt 64 Elementen, definiert. Die Sequenz G_0 besteht zunächst aus 40 Elementen, deshalb werden diesem 24 NULL-Elemente vorangestellt, so dass nun 64 Elemente vorhanden sind und somit alle Zellen der Matrix gefüllt werden können. Diese Elemente werden nun zeilenweise, wie in Abbildung 2.13 zu sehen, in die Matrix geschrieben und die Spalten, wie in Tabelle 2.3 angegeben, permutiert. Anschließend wird die Matrix spaltenweise, wie in Abbildung 2.14 dargestellt, ausgelesen.

Zielspalte	1, 17, 9, 25, 5, 21, 13, 29, 3, 19, 11, 27, 7, 23, 15, 31,
	0, 16, 8, 24, 4, 20, 12, 28, 2, 18, 10, 26, 6, 22, 14, 30

Tabelle 2.3.: Permutationsvorschrift für die Interleaving-Matrix

Nach dem Interleaving werden die NULL Elemente entfernt, sodass jede Sequenz wie-

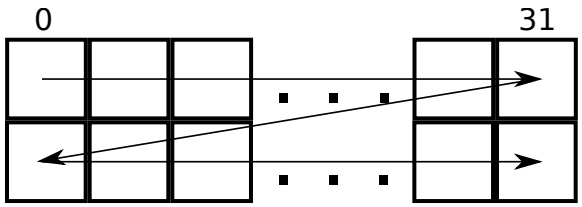


Abbildung 2.13.: Zeilenweises Beschreiben der Interleaving-Matrix

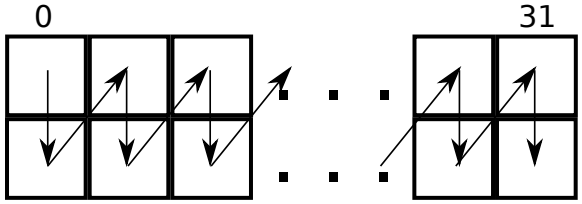


Abbildung 2.14.: Spaltenweises Auslesen der Interleaving-Matrix

der 40 Elemente enthält. Die drei Sequenzen G_0, G_1, G_2 werden danach vereinigt zu einer Sequenz $[G_0, G_1, G_2]$. Diese wird wiederum 16 mal wiederholt, sodass insgesamt eine Sequenz mit 1920 Elementen entsteht. Jedes sechzehntel dieser Sequenz mit 1920 Elementen enthält die gleichen Informationen.

Der Empfänger muss die Bits, der so bearbeiteten Sequenz, durch Deinterleaving wieder in ihre ursprüngliche Reihenfolge bringen. Außerdem kann durch Soft-Combining die Qualität der Sequenz im Empfänger verbessert werden.

2.6.2. Datenfluss im PBCH

Aus der Signalverarbeitung im BCH wird eine Sequenz mit 1920 Elementen an den Physical Broadcast Channel (PBCH) übergeben. Der Datenfluss im PBCH durchläuft die in Abbildung 2.15 dargestellten Verarbeitungsschritte, die nun vorgestellt werden, [3GP11b].

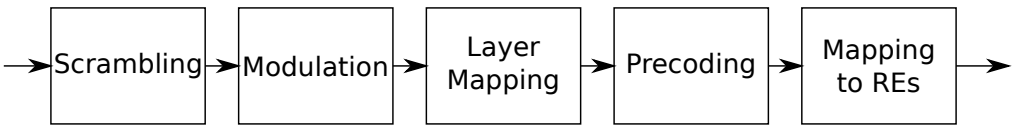


Abbildung 2.15.: Datenfluss im PBCH

Scrambling

Das Scrambling wird, mithilfe der N_{ID}, zellspezifisch durchgeführt. Dazu wird die Scramblingsequenz $c(i)$ mit dem in Anhang B definierten Pseudo-Noise (PN)-Sequenzgenerator mit $c_{init} = N_{\mathrm{ID}}$ und $M_{PN} = 1920$ erzeugt. Die Berechnung der Ausgangssequenz des

Scramblings erfolgt nach

$$\tilde{b}(i) = \big(b(i) + c(i)\big) \mod 2 \quad \text{mit } i = 0, \ldots, 1920 - 1 \tag{2.31}$$

wobei $b(i)$ die Eingangs- und $\tilde{b}(i)$ die Ausgangssequenz repräsentieren. Das Descrambling im Empfänger erfolgt durch erneutes berechnen des Scramblings.

Modulation

Für die Modulation der Bits $\tilde{b}(i)$ auf die entsprechenden komplexen Modulationssymbole $d(i)$ wird im PBCH Quadrature Phase Shift Keying (QPSK) eingesetzt. Dabei werden, wie in Abbildung 2.16 dargestellt, jeweils zwei Bit einem Modulationssymbol zugeordnet. Dadurch werden aus den 1920 Bit insgesamt 960 Modulationssymbole.

Der Empfänger kann weich oder hart demodulieren. Weiche Demodulation bedeutet, dass beim Empfang die Werte für Real- und Imaginärteil genutzt werden um für jedes demodulierte Bit einen Wert für dessen Zuverlässigkeit zu erhalten. Bei der Anwendung des Viterbialgorithmus können diese Zuverlässigkeitswerte als Güte mit einbezogen werden. Harte Demodulation hingegen bedeutet, dass auf Grund der Lage, der Modulationssymbole, in einem der vier Quadranten auf deren Bits entschieden wird.

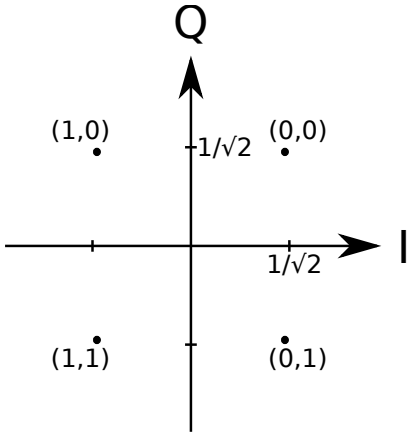


Abbildung 2.16.: QPSK-Konstellationsdiagramm

Layer Mapping

Das Layer Mapping dient der Vorbereitung für das darauf folgende Precoding. Die Anzahl der genutzten Layer v ist dabei gleich N_{ant}. Die Modulationssymbole $d(n)$ mit $n = 0, \ldots, M_{symb} - 1$ werden den Sequenzen $x(i) = [x^{(0)}(i) \ \ldots \ x^{(v-1)}(i)]^T$ auf den jeweiligen Layern zugewiesen. In Tabelle 2.4 ist die Zuordnung in Abhängigkeit der Anzahl an Layern angegeben. Für $v = 2$ erfüllt die Anzahl Modulationssymbole M_{symb} immer die Bedingung $M_{symb} \mod 2 = 0$. Für $v = 4$ kann der Fall $M_{symb} \mod 4 \neq 0$ eintreten. In

diesem Fall werden der Sequenz der Modulationssymbole $d(n)$ zwei Modulationssymbole mit null Bits angehängt. Der Empfänger muss aus den Modulationssymbolen der Layer, wieder die Sequenz der Modulationssymbole herstellen.

v	Zuweisung	mit $i = 0, \ldots, M_{symb}^{layer} - 1$
1	$x^{(0)}(i) = d(i)$	mit $M_{symb}^{layer} = M_{symb}$
2	$x^{(0)}(i) = d(2i)$ $x^{(1)}(i) = d(2i+1)$	mit $M_{symb}^{layer} = M_{symb}/2$
4	$x^{(0)}(i) = d(4i)$ $x^{(1)}(i) = d(4i+1)$ $x^{(2)}(i) = d(4i+2)$ $x^{(3)}(i) = d(4i+3)$	mit $M_{symb}^{layer} = \begin{cases} M_{symb}/4 & \text{für } M_{symb} \mod 4 = 0 \\ (M_{symb}+2)/4 & \text{für } M_{symb} \mod 4 \neq 0 \end{cases}$

Tabelle 2.4.: Zuweisung der Modulationssymbole auf die jeweiligen Layer

Precoding

Der Precoding Block nutzt die Alamouti-Kodierung um je nach Anzahl an Antennenports, die Diversity zu erhöhen. Dazu werden die in Abschnitt 2.5 beschriebenen MIMO-Grundlagen eingesetzt. Abhängig von der Anzahl der Antennenports (N_{ant}) wird für jeden verfügbaren Antennenport p eine Ausgangssequenz $y^{(p)}(i)$, wie in Tabelle 2.5 angegeben, berechnet.

N_{ant}	Berechnung
1	$y^{(0)}(i) = x^{(0)}(i)$
2	$\begin{pmatrix} y^{(0)}(i) \\ y^{(1)}(i) \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} x^{(0)}(i) & x^{(1)}(i) \\ -x^{(1)*}(i) & x^{(0)*}(i) \end{pmatrix}$
4	$\begin{pmatrix} y^{(0)}(i) \\ y^{(1)}(i) \\ y^{(2)}(i) \\ y^{(3)}(i) \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} x^{(0)}(i) & x^{(1)}(i) & 0 & 0 \\ 0 & 0 & x^{(2)}(i) & x^{(3)}(i) \\ -x^{(1)*}(i) & x^{(0)*}(i) & 0 & 0 \\ 0 & 0 & -x^{(2)*}(i) & x^{(3)*}(i) \end{pmatrix}$

Tabelle 2.5.: Precoding Matrizen für verschiedene Antennenkonfigurationen

Bei genauerer Betrachtung des Falles $p = 4$ fällt auf, dass dieser dem Fall $p = 2$ sehr ähnlich ist. Die angegebene Matrix kann in zwei Teile zerlegt werden, sodass

$$\begin{pmatrix} y^{(0)}(i) \\ y^{(2)}(i) \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} x^{(0)}(i) & x^{(1)}(i) \\ -x^{(1)*}(i) & x^{(0)*}(i) \end{pmatrix} \tag{2.32}$$

und

$$\begin{pmatrix} y^{(1)}(i) \\ y^{(3)}(i) \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} x^{(2)}(i) & x^{(3)}(i) \\ -x^{(3)*}(i) & x^{(2)*}(i) \end{pmatrix} \tag{2.33}$$

gilt. Wird nun noch der Fall $v = 4$ aus Tabelle 2.4 berücksichtigt, wird ersichtlich, dass das Precoding für $p = 4$ bis auf die Zuordnung zu den Antennenports genau der Übertragung mit $p = 2$ entspricht.

In Abschnitt 2.5 wurde der Empfang von MIMO Signalen beschrieben. Mithilfe der Werte aus einem Kanalschätzer können im Empfänger die Werte für die einzelnen Layer geschätzt werden.

Mapping der Modulationssymbole

Das Mapping des PBCH auf die REs ist für jeden Frame gleich. Der PBCH wird in Slot_1 auf den OFDM-Symbolen null bis drei gesendet. Es ist dabei zu beachten, dass alle REs, die in Abbildung 2.9 als mögliche Position für Pilotsymbole markiert sind, nicht verwendet werden können, da die Antennenkonfiguration beim Empfang des PBCH nocht nicht bekannt ist.

In jedem Frame werden so 240 Modulationssymbole übertragen, also erstreckt sich die Übertragung des PBCH über vier Frames. Deshalb wird auch die Scramblingsequenz alle vier Frames reinitialisiert. Für die Dekodierung des MIB wird nur eine der vier Teile benötigt, es kann aber Softcombining zur Verbesserung der Signalqualität genutzt werden.

2.6.3. Master Information Block

Der Master Information Block (MIB) enthält die grundlegenden LTE-Systeminformationen einer LTE-Funkzelle. Er wird durch den BCH und den PBCH übertragen. In Abbildung 2.17 ist die Struktur des MIB und die Position der übertragenen Daten dargestellt und in Tabelle 2.6 deren Kodierung erläutert.

Die Anzahl an Resource Blocks ($N_{\text{RB}}^{\text{DL}}$) gibt, nach 2.1, die Systembandbreite an. Die Werte PHICH Dauer und Ressourcen werden benötigt um die Position des Physical Hybrid-ARQ Indicator Channel (PHICH) zu identifizieren. Dieser ist Teil des in LTE genutzten Acknowledge Prozesses. Im MIB werden nur die acht höchstwertigen Bits des System Frame Number (SFN) übertragen (vgl. 2.6.3). Die zwei niedrigwertigsten Bits werden durch die Position des erfolgreich dekodierten MIB in der Sequenz nach dem Descrambling detektiert.

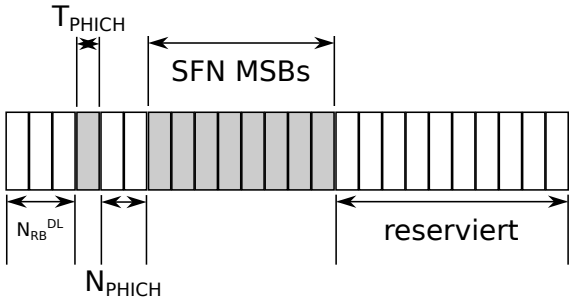


Abbildung 2.17.: Illustration der 24 Bit des MIB und deren Bedeutung

MIB Abschnitt	Kodierung
$N_{\mathrm{RB}}^{\mathrm{DL}}$	$0 \simeq 6 \quad 1 \simeq 15 \quad 2 \simeq 25 \quad 2 \simeq 50 \quad 4 \simeq 75 \quad 5 \simeq 100$
PHICH Duration	$0 \simeq$ normal $\quad 1 \simeq$ extended
PHICH Resources	$0 \simeq {}^{1}/_{6} \quad 1 \simeq {}^{1}/_{2} \quad 2 \simeq 1 \quad 3 \simeq 2$
SFN MSBs	8 MSB der SFN
reserviert	immer 0, für zukünftige Anwendung reserviert

Tabelle 2.6.: Dezimale Repräsentation der Kodierung der einzelnen Teile des MIB und deren Bedeutung

System Frame Number

Mit der System Frame Number (SFN) werden die einzelnen Frames durchnummeriert um über mehrere Frames hinweg Ressourcen zuweisen zu können. Für die möglichen Werte der SFN gilt $0 \leq \mathrm{SFN} < 1023$. Daraus folgt, dass für die Kodierung der SFN zehn Bit benötigt werden und bei einer Framedauer (T_{F}) von 10 ms, die SFN alle 10,24 s wiederholt wird. Die acht höchstwertigen Bits der SFN werden im MIB übertragen. Die beiden niedrigwertigsten Bits der SFN werden im PBCH kodiert. Dies geschieht dadurch, dass der PBCH über vier Frames verteilt übertragen wird. Jeder Frame enthält somit ein Viertel des PBCH, welches bereits alle Informationen des MIB enthält. Durch das duplizieren und berechnen der Sequenz nach dem Descrambling, erhält man eine Sequenz, die nur in einem Viertel gültige Daten enthält, während in den anderen Teilen nur zufällige Daten enthalten sind. Die Position der gültigen Daten kodiert dann die niedrigwertigsten Bits der SFN.

3. Implementierung

In diesem Kapitel wird die Implementierung und Struktur des GNU Radio Flowgraphs vorgestellt. Zunächst wird ein Überblick über GNU Radio selbst und den implementierten Flowgraph gegeben und danach werden die Einzelteile vorgestellt. Dazu wird detailliert auf die Synchronisation, die Orthogonal Frequency Division Multiplex (OFDM) Demodulation, die Entzerrung, die Extraktion des Physical Broadcast Channel (PBCH) und die Decodierung des Master Information Block (MIB) eingegangen. Zur besseren Visualisierung des Flowgraphs wurde die Blockstruktur in den GNU Radio Companion (GRC) integriert.

3.1. GNU Radio

GNU Radio ist ein Open Source Software Defined Radio (SDR) Framework, [GNU12]. Es stellt eine Vielzahl sogenannter Blöcke bereit, die jeweils grundlegende Funktionen, wie beispielsweise die Multiplikation oder Addition der Eingangsdatenströme, implementieren. Eine GNU Radio Anwendung lässt sich als gerichteter azyklischer Datenflussgraph, dem sogenannten Flowgraph, darstellen. Blöcke entsprechen dabei den Knoten und die gerichteten Kanten stellen den Datenfluss zwischen den Blöcken dar.

Die GNU Radio Blöcke werden vorwiegend in der Programmiersprache C++ implementiert und der Flowgraph wird mithilfe der Skriptsprache Python erstellt. Für die Erstellung des Flowgraphs steht auch eine grafische Oberfläche, der GNU Radio Companion (GRC), zur Verfügung.

Neben den einfachen Blöcken gibt es noch Quellen- und Senkenblöcke, sowie hierarchische Blöcke. Quellen- und Senkenblöcke können Schnittstellen zu Hardware, Dateien, Netzwerk und vielem mehr bereitstellen. Hierarchische Blöcke dienen der Gruppierung der Blöcke zu komplexeren Einheiten und erhöhen so die Übersichtlichkeit des Flowgraph und die Wiederverwendbarkeit der Blöcke.

Zu GNU Radio können jederzeit eigene Blöcke hinzugefügt und genutzt werden. Dabei stehen für die Ein- und Ausgangsdatenformate fünf verschiedene Datenformate – `complex`, `float`, `integer`, `short`, `byte` – zur Verfügung. Das Framework selbst kümmert sich bei allen Blöcken um das Scheduling und das weiterreichen von Daten.

Zur Entwicklung der Blöcke des Long Term Evolution (LTE) Empfängers wurde zunächst vor allem C++ und Python genutzt und später die Unterstützung für GRC hinzugefügt.

3.1.1. GNU Radio Tags

GNU Radio Tags sind Zusatzinformationen, die einzelnen Abtastwerten fest zugeordnet werden. Ein Block kann auf beliebige Abtastwerte ein Tag setzen und Folgeblöcke können diese lesen und mit diesen Informationen weiterarbeiten. Jedes Tag enthält dabei vier Datenfelder, die in Tabelle 3.1 beschrieben sind. Zusätzlich kann für jeden Block eine *Tag Propagation Policy* definiert werden. Dabei gibt es drei verschiedene Möglichkeiten, die in Tabelle 3.2 beschrieben sind.

Abkürzung	Beschreibung
srcid	eindeutige Identifikation des Quellenblocks (Source ID)
key	Schlüssel zur Identifikation der Art der Information
value	ein Integerwert, den das Tag beinhaltet
offset	absoluter Offset des zugehörigen Abtastwerts zum ersten Abtastwert

Tabelle 3.1.: Informationen, die in Tags gespeichert sind

Policy	Beschreibung
TPP_DONT	zum Block propagierte Tags können genutzt werden, werden aber nicht weiterpropagiert
TPP_ONE_TO_ONE	Tags werden von Eingang null zu Ausgang null propagiert, entsprechend für alle Ein- und Ausgänge
TPP_ALL_TO_ALL	Tags von einem Eingang werden an alle Ausgänge weiterpropagiert (Standardeinstellung)

Tabelle 3.2.: Tag Propagation Policies

3.2. Überblick

Um eine gute Übersicht zu erhalten wurde der GRC-Flowgraph des LTE-Emfängers zunächst in große hierarchische Blöcke zusammengefasst. Die hierarchische Darstellung im GRC ist in Abbildung 3.2 zu sehen. Im Flowgraph ist keine Quelle dargestellt, jedoch kann hier je nach Situation beispielsweise ein Dateiblock oder ein Hardware Block genutzt werden. Eine flachere Struktur des Flowgraphs ist in Abbildung 3.3 zu finden. Vor dem Ausführen des Flowgraphs muss die FFT-Länge (N_{FFT}) festgelegt werden, denn von ihr werden, wie in Abbildung 3.1 dargestellt, mehrere Parameter abgeleitet.

Der hierarchische Block *LTE Synchronization* enthält alle Blöcke die zur Synchronisation des LTE-Systems notwendig sind. Die Eingangsdaten für diesen Block sind komplexe Abtastwerte, auf denen die Synchronisation auf die Framestruktur durchgeführt wird. Neben einer Frequenzoffsetkorrektur, sind die Abtastwerte am Ausgang mit Tags für den Beginn und die Nummer des Slots markiert. Folgende Blöcke können so mit bekannter Framestruktur, wie in 2.2 beschrieben, arbeiten.

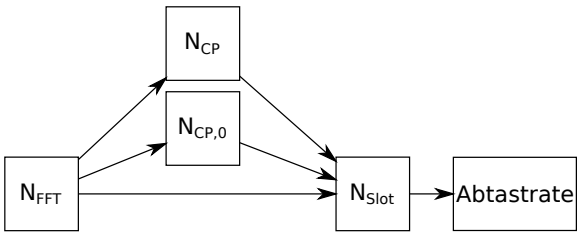


Abbildung 3.1.: Parameter Abhängigkeiten

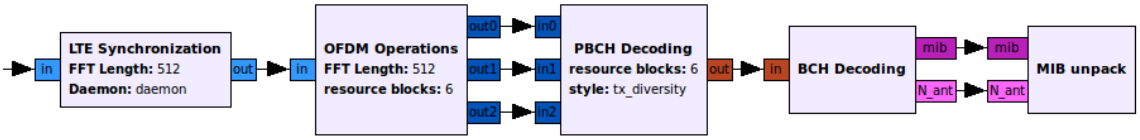


Abbildung 3.2.: hierarchische Darstellung des GRC-Flowgraph

Nach der Synchronisation werden die Eingangsdaten im Block *OFDM Operations* weiter verarbeitet. Es wird das Cyclic Prefix (CP) entfernt, eine Fast Fourier Transform (FFT) sorgt für den Übergang in den Frequenzbereich, die genutzten Unterträger werden extrahiert und eine Kanalschätzung wird durchgeführt. Am Ausgang dieses Blocks liegen die Daten im Frequenzbereich als Vektoren für jedes OFDM-Symbol vor. Außerdem werden die Korrekturwerte für Ein- und Zweiantennenkonfigurationen bereitgestellt. Die Antennenkonfiguration ist in diesem Block noch unbekannt, da diese erst nach dem erfolgreichen Dekodieren des MIB am Ende des Flowgraphs zur Verfügung steht.

Bis zum hierarchischen Block *PBCH Decoding* werden alle Abtastwerte, die dem Flowgraph übergeben werden, verarbeitet. Im Block *PBCH Decoding* werden nur die Abtastwerte, die die Informationen über den PBCH tragen, wie in 2.6.2 beschrieben, extrahiert. Das Pre Decoding – ein inverser Alamouti Code, wie in 2.5 vorgestellt –, das Layer Demapping, die Quadrature Phase Shift Keying (QPSK) Softdemodulation und das Descrambling werden durchgeführt. Die Ausgangsdaten des Blocks *PBCH Decoding* sind Vektoren mit empfangenen Bits inklusive Informationen über deren Zuverlässigkeit, die im Block *BCH Decoding* weiterverarbeitet werden.

Im Block *BCH Decoding* werden das Rate Unmatching mit Deinterleaving, das Viterbi Decoding und die Cyclic Redundancy Check (CRC) Berechnung durchgeführt. Durch die verschiedenen Scramblingsequenzen für die CRC-Prüfsumme kann bei korrekt berechneter CRC-Prüfsumme die Antennenkonfiguration bestimmt werden. Am Ende des Flowgraphs steht der Block *MIB Unpack*, in dem die grundlegenden Systeminformationen ausgelesen werden.

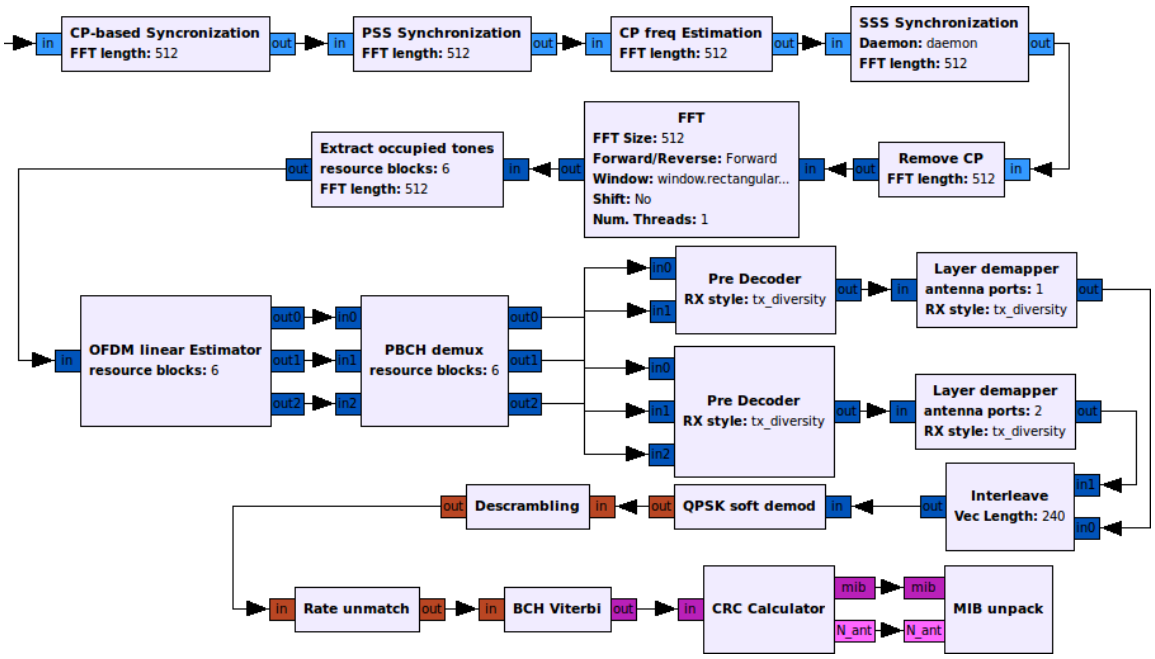


Abbildung 3.3.: Darstellung des kompletten GRC-Flowgraph

3.3. Synchronisation

Die Synchronisation ist in vier Schritte unterteilt, der CP-basierten Synchronisation, die auf den Symboltakt synchronisiert, der Primary Synchronization Symbol (PSS) Synchronisation, die auf den halben Frametakt synchronisiert, einer Fractional Frequency Offset (FFO) Synchronisation und der abschließenden Secondary Synchronization Symbol (SSS) Synchronisation, die den Frametakt synchronisiert und die Cell ID (N_{ID}) schätzt.

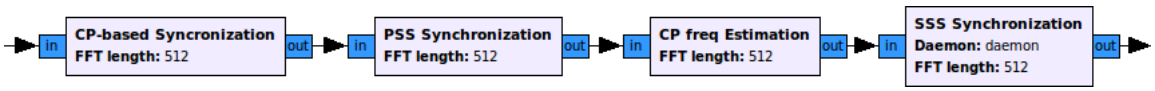


Abbildung 3.4.: Darstellung der Synchronisationsblöcke

3.3.1. Cyclic Prefix Synchronisation

Es wird nun die Implementierung des Blocks *CP-based Synchronization* vorgestellt, dessen Grundlagen in Abschnitt 2.3.1 zu finden sind. Der Block *CP-based Synchronization* wird von `gr_sync_block` abgeleitet. Als Parameter wird diesem Block die FFT-Länge (N_{FFT}) übergeben und die Systemparameter CP-Länge (N_{CP}), CP_0-Länge ($N_{\mathrm{CP},0}$) und Slot Länge (N_{Slot}) berechnet:

$$\begin{aligned} N_{\mathrm{CP}} &= N_{\mathrm{FFT}} \cdot 144/2048 \\ N_{\mathrm{CP},0} &= N_{\mathrm{FFT}} \cdot 160/2048 \\ N_{\mathrm{Slot}} &= 7\,N_{\mathrm{FFT}} + 6\,N_{\mathrm{CP}} + N_{\mathrm{CP},0} \end{aligned} \tag{3.1}$$

Um die Anzahl der Korrelationsberechnungen nach Gleichung 2.11 zu reduzieren, wird

die CP-Synchronisation zweigeteilt. Im ersten Teil wird eine Schrittweite $s = N_{\mathrm{CP},0}/4$ eingeführt. Es wird somit nicht für jedes n (vgl. Gl. 2.11) eine Korrelation berechnet sondern nur für $n = sk,\ k \in \mathbb{N}$ und daraus das

$$n_{max,1} = \arg\max_{n} \gamma(n) \tag{3.2}$$

Dadurch reduziert sich der Rechenaufwand um den Faktor s. Im zweiten Teil wird für $n_{max,1} \pm s/2$ für jedes n eine Korrelation berechnet und so mit

$$n_{sym,a} = \arg\max_{n} \gamma(n) \tag{3.3}$$

der absolute OFDM-Symbolbeginn $n_{sym,a}$ detektiert. Der maximale Absolutwert der Korrelationsberechnungen ist ein Attribut des Blocks. Nur wenn ein neues Maximum gefunden wurde, wird auch die Position des Symbolbeginns aktualisiert. Der Symbolbeginn $n_{sym,m}$ wird berechnet durch

$$n_{sym,m} \quad = \quad (n_{sym,a}) \mod (N_{\mathrm{Slot}}) \tag{3.4}$$

An dieser Stelle wird genutzt, dass jedem GNU Radio Block die absolute Anzahl der bisher bearbeiteten und produzierten Abtastwerte bekannt ist. Der Symbolbeginn $n_{sym,m}$ stellt somit den Offset zwischen dem ersten Abtastwert und dem Beginn des ersten empfangenen Symbols dar. Es wird nicht mit $n_{sym,a} \mod N_{\mathrm{s}}$ gerechnet, da dies auf Grund der unterschiedlichen Längen von Cyclic Prefix (CP) und CP_0 zu Fehlern in der Berechnung führt.

Ein- und Ausgangsdaten bleiben bei diesem Block unverändert, es wird jedoch periodisch ein Tag gesetzt, das die zuletzt ermittelte Position des Symbolbeginns $n_{sym,m}$ trägt. Dieses Tag nutzen nachfolgende Blöcke für weitere Synchronisationsschritte.

3.3.2. PSS-Synchronisation

In diesem Abschnitt wird die Implementierung der Halbframesynchronisation durch das Primary Synchronization Symbol (PSS) vorgestellt. Die Struktur und Position des PSS im Frame wurde bereits in Punkt 2.3.3 beschrieben. Die innere Struktur des Blocks *PSS Synchronization* ist in Abbildung 3.5 dargestellt.

Die Aufgabe des Blocks *PSS Selector* ist es, auf Basis der Informationen aus dem Block *CP-based Synchronization*, die Abtastwerte für die einzelnen Symbole zu extrahieren und als Vektoren weiterzugeben. Dabei wird auch ein Tag mit der genauen Anfangsposition des extrahierten Symbols hinzugefügt. Sobald vom Block *PSS Calculator* der vermutete Offset für das PSS gesetzt wurde, wird für $n_{pss} \pm N$ Abtastwerte um diese Position herum, jeweils ein Symbol extrahiert. Dadurch kann im Block *PSS Calculator* die Zeitsynchronisation präzisiert werden, während der zusätzliche Berechnungsaufwand nicht um N erhöht wird. Die Blöcke *FFT* und *Extract occupied tones* werden in 3.4.1 und 3.4.2 beschrieben.

Der größte Rechenaufwand entsteht im Block *PSS Calculator*, da dieser für jedes OFDM-Symbol drei Kreuzkorrelationsfunktionen – eine für jede mögliche Zadoff-Chu Sequenz

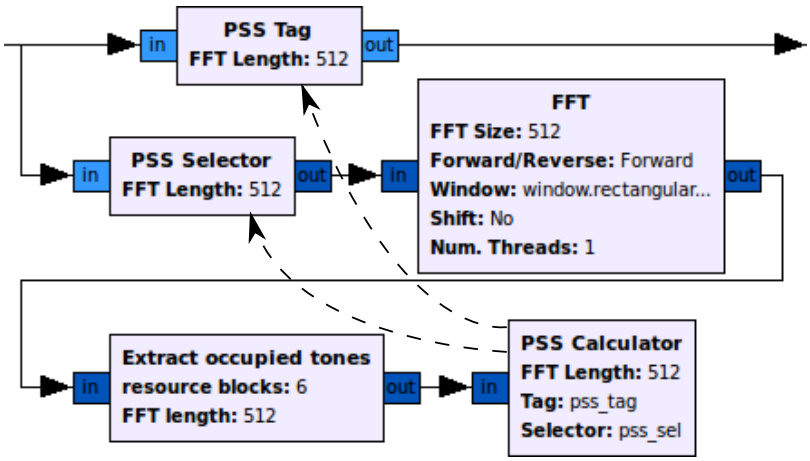


Abbildung 3.5.: Innere Struktur des Blocks *PSS Synchronization*

– berechnen muss. Wie in Abschnitt 2.3.3 dargestellt, wird aus den berechneten Werten die Cell ID Number (N_{ID}^2) und der Integer Frequency Offset (IFO) berechnet. Die vom Block *PSS Selector* hinzugefügten Tags mit dem Beginn des Symbols, werden zur Berechnung des Halbframeoffsets genutzt und somit auch für den genauen Beginn der einzelnen OFDM-Symbole. Die PSS-Synchronisation ist nur zu Beginn notwendig, deshalb wird der Block *PSS Calculator* deaktiviert, sobald über 200 aufeinanderfolgende OFDM-Symbole keine Änderung der Cell ID Number (N_{ID}^2) mehr bewirkt haben. Dies geschieht indem der Block *PSS Selector* angewiesen wird keine Vektoren mehr weiterzugeben.

Es ist in GNU Radio nicht möglich Rückkopplungen zu realisieren, deshalb benötigt der Block *PSS Calculator* Referenzen auf die Blöcke *PSS Selector* und *PSS Tag*, um diese zu steuern. Dies wird in Abbildung 3.5 durch gestrichelte Pfeile dargestellt. Beide Blöcke benötigen Informationen über die Position des PSS. Der Block *PSS Tag* benötigt außerdem den Wert der Cell ID Number (N_{ID}^2). Es hat sich als unpraktikabel erwiesen, diese Informationen über einen Datenfluss zum Block *PSS Tag* weiterzugeben, da dieser sonst nicht als synchroner Block implementiert werden kann.

Der Block *PSS Tag* entfernt, die Tags des Blocks *CP-based Synchronization* und fügt am Beginn jedes Slots einen Tag mit der Slotnummer in einem halben Frame hinzu (vgl. 2.3.3). Sobald das Attribut für die N_{ID}^2 einen gültigen Wert hat, wird zu Beginn jedes Halbframes ein Tag mit der N_{ID}^2 gesetzt, da es zur SSS Berechnung benötigt wird.

3.3.3. Cyclic Prefix Frequenz Korrektur

Auf Grund der verschiedenen Längen von CP_0 und der anderen CPs kann nicht direkt nach der Symbolsynchronisation eine FFO-Korrektur stattfinden, sondern es muss zunächst noch mindestens auf den Slottakt synchronisiert werden, dies geschieht im oben beschriebenen Block *PSS Synchronization*. In Abbildung 3.6 ist die innere Struktur des Blocks *CP freq Estimation* dargestellt. Die gestrichelte Linie repräsentiert die Rückkopplung vom Block *CP Offset Calculator* zum Block *Signal Source*.

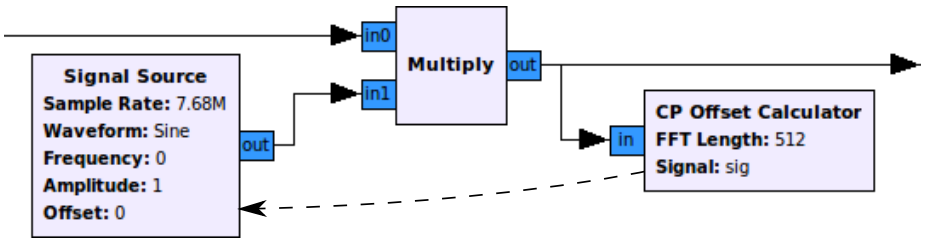


Abbildung 3.6.: Innere Struktur des Blocks *CP freq Estimation*

Die Eingangsdaten werden mit einem komplexen Sinus der Frequenz $-f_{fo}$ multipliziert, ein neuer Korrekturwert wird berechnet und die Daten werden an nachfolgende Blöcke weitergegeben. Dabei gilt

$$f_{fo\ neu} = f_{fo\ alt} + \alpha f_{est} \tag{3.5}$$

wobei α die Gewichtung der neuen Schätzung und f_{est} der Wert der neuesten Schätzung ist. Der Schätzwert f_{est} wird für jeden Slot nach Gleichung 2.13 mit der Autokorrelationsfunktion (AKF) von nulltes Cyclic Prefix (CP_0) und der sechs CPs in jedem Slot berechnet. f_{est} ist der Mittelwert der sieben auf diese Weise berechneten Werte. Im Block *CP Offset Calculator* ist ein Buffer realisiert, der die Werte eines Slots zwischenspeichert, bis alle Werte des nächsten Slots vorhanden sind. Dadurch wird verhindert, dass die Funktion zur Berechnung von f_{est} nur für einzelne OFDM-Symbole aufgerufen wird und es müssen weniger Operationen zum Kopieren von Speicherbereichen aufgerufen werden. Dies liegt an den Voraussetzungen für den Einsatz, der von GNU Radio bereitgestellten, *VOLK Bibliothek*. Die *VOLK*-Operationen stellen Forderungen an das Binding der Werte an bestimmte Speicherbereiche, die so erfüllt werden.

3.3.4. SSS-Synchronisation

Der letzte Synchronisationsschritt wird im Block *SSS Synchronization* durchgeführt. Danach ist das System auf den LTE-Frametakt synchronisiert und die Cell ID (N_{ID}) ist bekannt. Die N_{ID} wird dann an nachfolgende Blöcke, durch den Block *Cell ID Daemon* weitergegeben. Die entsprechenden Verbindungen, die nicht direkt im Flowgraph zu sehen sind, sind in Abbildung 3.7 durch gestrichelte Pfeile dargestellt. Dem Block *Cell ID Daemon* wird die N_{ID} und dem Block *SSS Tag* wird der Framestartoffset übergeben. Der Block *SSS Tag* erhält den Framestartoffset und fügt dem ersten Abtastwert jedes Slots ein Tag hinzu, dessen Wert die Slotnummer im Frame repräsentiert. Da Tags, die bis zu diesem Block propagiert wurden, nicht mehr weiter benötigt werden, werden sie auch nicht mehr weiterpropagiert.

Der Block *SSS Selector* ist dem Block *PSS Selector* ähnlich, mit dem Unterschied, dass das OFDM-Symbol mit dem SSS selektiert und weitergegeben wird. Die Blöcke *FFT* und *Extract occupied tones* sind genauso parametrisiert, wie in 3.3.2 beschrieben.

In 2.3.4 wurden die Gleichungen zur Berechnung des SSS eingeführt. Ziel ist es, die Indizes m_0 und m_1 zu ermitteln und zusammen mit der N_{ID}^2 aus dem Block *PSS Synchro-*

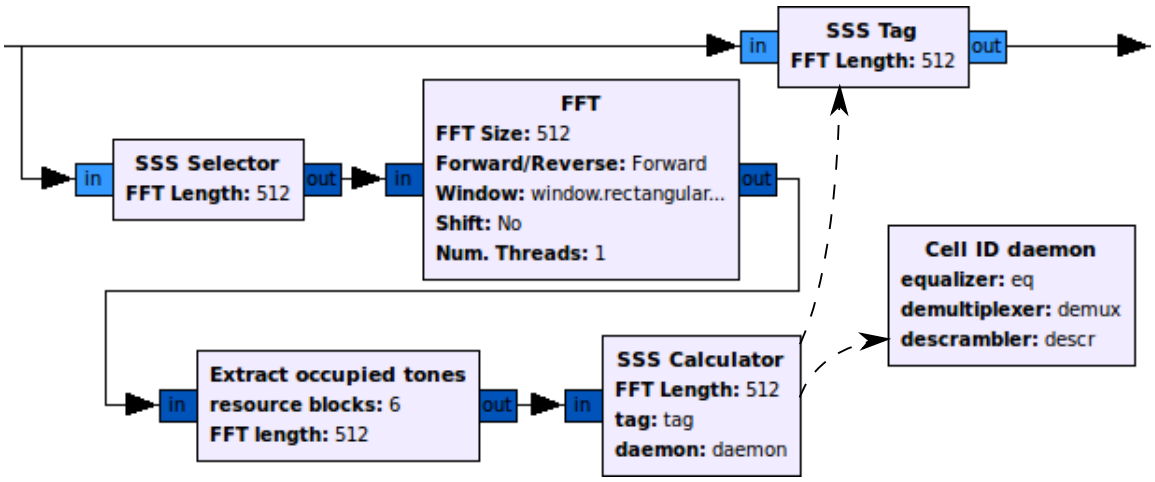


Abbildung 3.7.: Innere Struktur des Blocks *SSS Synchronization*

nization die Cell ID (N_{ID}) zu ermitteln. Außerdem kann aus der Folge $d(n)$, die Position im Frame ermittelt werden. Dies entspricht der Synchronisation auf den Frametakt. Um Redundanzen zu vermeiden, werden alle Folgen, die nicht von zu ermittelnden Systemparametern abhängen, nur einmal bei der Initialisierung berechnet (vgl. Gln. A.4, A.5, A.7, A.8, A.10, A.11). Die Ermittlung der Cell Identity Group (N^1_{ID}) aus den Indizes m_0 und m_1 wird ebenfalls über eine vorberechnete Look Up Table (LUT) realisiert. Die Berechnung der N^2_{ID} aus dem SSS wird in mehrere Schritte unterteilt, Quelle: [MEJ+09].

1. Ermittle die Folgenwerte $d(2n)$ und $d(2n+1)$ aus dem SSS
2. Berechne die Folgen c_0 und c_1 mit N^2_{ID} (Gl. A.6)
3. Berechne die Folge $s_0^{(m_0)} = d(2n)/c_0$
4. Berechne den Index m_0
5. Berechne die Folge $z_1^{(m_0)}$ mit berechnetem m_0 nach Gl. A.9
6. Berechne die Folge $s_1^{(m_1)} = d(2n+1)/c_1$
7. Berechne Index m_1
8. Falls $m_0 > m_1$ gilt, tausche ihre Werte und setzte die SSS Position auf Slot_{10}.
9. Ermittelte die N^1_{ID} aus der vorberechneten LUT für m_0 und m_1
10. Berechne die $N_{\mathrm{ID}} = 3\,N^1_{\mathrm{ID}} + N^2_{\mathrm{ID}}$

Die Berechnung der Indizes m_0 und m_1 werden durch die Kreuzkorrelationsfunktion (KKF) der Folgen $s_0^{(m_0)}$ und $s_1^{(m_1)}$ mit $s_{ref} = \{s_0^{(0)}; s_0^{(0)}\}$ ermittelt. In Abbildung 3.8 wird das Ergebnis einer solchen KKF gezeigt. Daraus wird

$$n_{max} = \arg\max_n \ \mathrm{KKF}(s_1^{(m_1)}, s_{ref}) \tag{3.6}$$

berechnet und aus der Position von n_{max} mit $m_1 = n_{max} - 62$ dann der Index errechnet. Dies geschieht in gleicher Weise für den Index m_0.

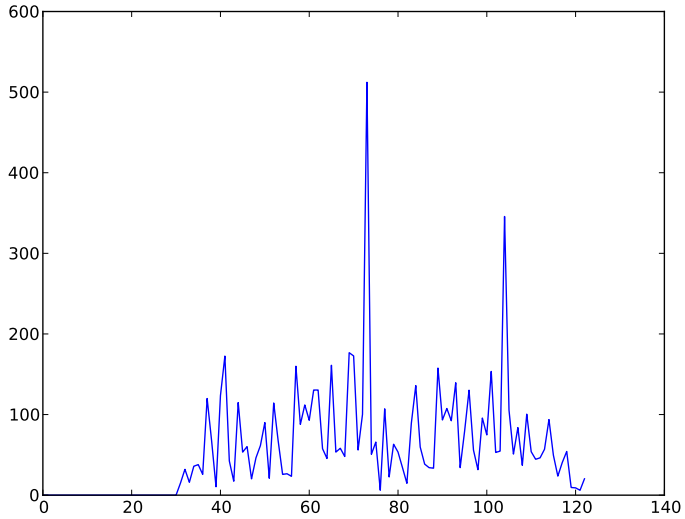


Abbildung 3.8.: Plot der KKF von $s_1^{(m_1)}$ und s_{ref} mit gemessenen Daten.

Cell ID Daemon

Nach der Berechnung der N_{ID} benötigen mehrere Blöcke diesen Wert. Dem Block *SSS Synchronization* wird dafür eine Referenz auf den Block *Cell ID Daemon* übergeben. Dieser Block hat keine Ein- und Ausgänge, er wird genutzt um die N_{ID} an alle Blöcke die diese benötigen weiterzugeben. Dazu werden dem Block *Cell ID Daemon* Referenzen auf die Blöcke, die die N_{ID} benötigen übergeben. Dies trägt zu einer klareren Struktur des Flowgraph bei.

3.4. OFDM Operationen

Für den Empfang von LTE-Signalen müssen die OFDM-Operationen, die in jedem OFDM-System vorhanden sind, durchgeführt werden. In diesem Abschnitt wird die in Abbildung 3.9 dargestellte LTE-spezifische Struktur vorgestellt. In Abbildung 2.3 wurde dazu bereits eine prinzipielle OFDM-Empfängerstruktur eingeführt. Im ersten Schritt wird das CP entfernt, danach wird der Übergang in den Frequenzbereich mittels einer FFT durchgeführt und die Modulationsymbole auf den benötigten Unterträgern extrahiert. Im Block *OFDM linear Estimator* wird mittels der in Abschnitt 2.4 beschrieben Pilotsymbole die Kanalmatrix geschätzt.

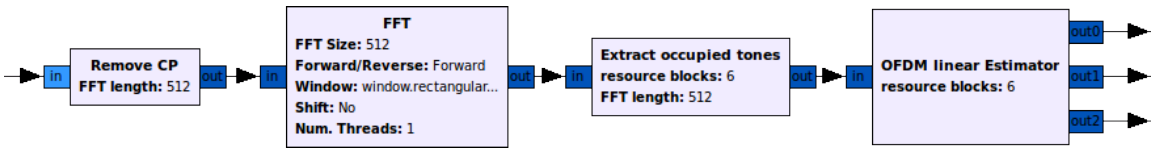


Abbildung 3.9.: Innere Struktur des Blocks *OFDM Operations*

3.4.1. Cyclic Prefix und FFT Parametrisierung

Bevor in einem LTE-System der Übergang vom Zeit- in den Frequenzbereich durchgeführt werden kann, müssen die Abtastwerte der einzelnen OFDM-Symbole aus dem Datenfluss extrahiert werden. Dabei werden auch die CPs entfernt. Der Block *remove CP* extrahiert die OFDM-Symbole und gibt diese als Vektoren weiter. Als Parameter benötigt er die FFT-Länge (N_{FFT}) um die CP-Länge (N_{CP}) und die Größe des Ausgangsvektors zu berechnen.

Solange noch nicht auf den Frametakt synchronisiert wurde, ist die Position der einzelnen physikalischen Kanäle und der Pilotsymbole unbekannt. Abtastwerte, die den Block vor dem Abschluss der Synchronisation erreichen, können nicht weiter für den Empfang verwendet werden und werden deshalb vom Block *remove CP* nicht weitergegeben. Sobald Der Block den ersten Tag, der den Beginn eines Frames markiert, empfängt, werden Vektoren mit OFDM-Symbolen an folgende Blöcke weitergegeben. Jedem Vektor wird ein Tag angehängt, welches die OFDM-Symbolnummer im Frame als Wert enthält, um diese im Frame identifizieren zu können.

Die extrahierten OFDM-Symbolvektoren werden an den Block *FFT* weitergegeben, der von GNU Radio bereitgestellt wird und deshalb nur noch für den LTE Empfänger parametrisiert werden muss. Die entsprechenden Parameter und ihre Bedeutung werden in Tabelle 3.3 erläutert.

Parameter	Wert	Erläuterung
FFT Size	N_{FFT}	Parameter ist für alle Blöcke im System gleich
Forward/Reverse	Forward	FFT Richtung (Forward $\simeq$ in Frequenzbereich)
Window	Rectangular	Vektor zur Fensterung des Spektrums
Shift	No	Führe FFT Shift durch
Num. Threads	1	Anzahl paralleler Berechnungsprozesse

Tabelle 3.3.: Parametrisierung des GNU Radio Block *FFT*

3.4.2. Extrahierung der Nutzsymbole

Nach dem Übergang in den Frequenzbereich müssen die Abtastwerte, die die genutzten Unterträger repräsentieren, extrahiert werden. Dazu werden dem Block *Extract occupied tones* Block die Parameter Anzahl an Resource Blocks ($N_{\text{RB}}^{\text{DL}}$) aus Abschnitt 2.2 und N_{FFT} übergeben. Es ist auch darauf zu achten, dass der DC-Carrier bei LTE nicht genutzt wird und deshalb entfernt wird. Der Block dezimiert also die Eingangsvektoren der Länge N_{FFT}

auf die Ausgangsvektoren der Länge $12 \cdot N_{\mathrm{RB}}^{\mathrm{DL}}$.

3.4.3. Linearer OFDM Kanalschätzer

Bevor im Block *Pre Decoder* die Entzerrung durchgeführt werden kann, muss zunächst eine Kanalschätzung anhand der Pilotsymbole stattfinden. Die für den Empfang benötigten Kanalschätzwerte, werden im Block *OFDM linear Estimator* berechnet. Dazu erhält dieser vom Block *Extract occupied tones* OFDM-Symbolvektoren, die die Modulationsymbole der genutzten Unterträger enthalten. Die Vektorgröße wird dem Block durch den Parameter Anzahl an Resource Blocks ($N_{\mathrm{RB}}^{\mathrm{DL}}$) übergeben. Der Block *OFDM linear Estimator* ist der erste, der die vom Cell ID Daemon bereitgestellte N_{ID} benötigt, da von diesem die Werte und Position der Pilotsymbole abhängen (vgl. 2.4).

Für die Kanalschätzung werden zunächst die Abweichung der Amplitude und Phase zwischen den empfangenen Pilotsymbolen und den Referenzen berechnet. Darauf folgt eine lineare Interpolation der Schätzwerte in Freqenzrichtung und anhand dieser eine lineare Interpolation in Zeitrichtung zwischen den OFDM-Symbolen mit Pilotsymbolen. Dies muss prinzipiell zunächst für jeden möglichen Antennenport durchgeführt werden. Nachdem der MIB dekodiert wurde und somit die Antennenkonfiguration bekannt ist, kann die Berechnung auf die tatsächlich vorhandenen Antennenports beschränkt werden.

Die Kanalschätzwerte, wie auch die Empfangswerte, werden für jeden Antennenport an einem Ausgang weitergereicht. Nachfolgende Blöcke können so diese Werte weiterverarbeiten. Da 4x4 MIMO noch nicht genutzt wird, wird dies auch noch nicht unterstützt. Deshalb ist für Antennenport drei und vier keine Schätzung implementiert, da dies sehr rechenintensiv ist.

3.5. PBCH-Dekodierung

Nach der Synchronisation und Kanalschätzung beginnt die Signalverarbeitung im Physical Broadcast Channel (PBCH), die in diesem Abschnitt vorgestellt wird. Der Block *OFDM Operations* stellt dafür die synchronisierten OFDM-Symbolvektoren mit den dazugehörigen Kanalschätzwerten bereit, aus denen der PBCH extrahiert wird.

In Abschnitt 2.6.2 wurde der PBCH vorgestellt. Der GRC-Flowgraph zu seiner Dekodierimplementierung ist in Abbildung 3.10 zu finden. Zusätzlich zu den fünf bereits beschriebenen Teilen, kommt der Block *Interleave* hinzu, der die beiden Eingänge abwechselnd auf den Ausgang weiterleitet, so dass die Vektoren beider Datenflüsse weitergeleitet werden. Dies ist notwendig, da die Antennenkonfiguration der Basisstation (BS) unbekannt ist und aus diesem Grund das Predecoding und das Layer Demapping für Konfigurationen mit einem und zwei Antennenports durchgeführt wird.

3.5.1. PBCH Demultiplexer

Im Block *PBCH demux* wird der PBCH und die Kanalschätzwerte aus den jeweiligen Datenflüssen extrahiert. Auf diese Weise werden am Ausgang dieses Blocks nur Daten, die

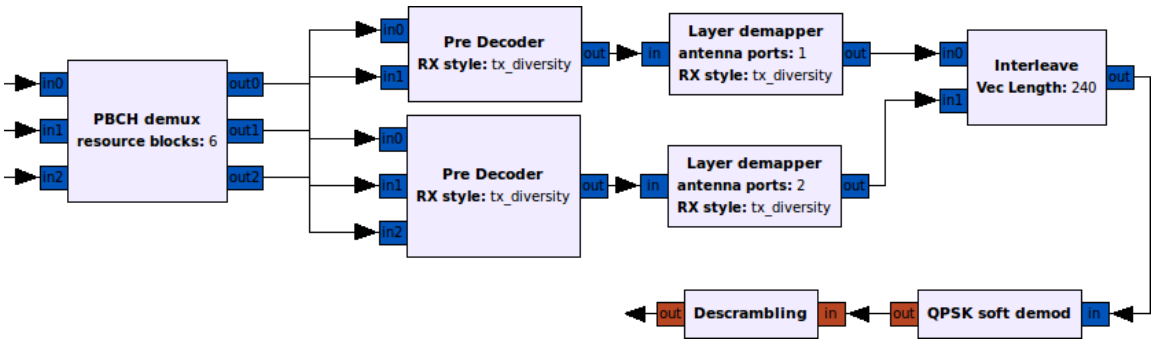


Abbildung 3.10.: Innere Struktur des Blocks *PBCH Decoding*

den PBCH und seine Kanalschätzwerte enthalten, weitergegeben. Dies sind Vektoren mit jeweils 240 Elementen. An dieser Stelle sind zunächst Vektoren mit 960 Elementen, wie in Abschnitt 2.6.2 beschrieben, zu erwarten. Der PBCH ist jedoch über vier Frames verteilt. Vektoren die aus einem Frame gewonnen werden enthalten alle Daten des MIB, deshalb wird für die Abtastwerte in jedem Frame ein Dekodierversuch durchgeführt. Dies ist auch im Standard so gewollt, um bei günstigen Übertragungskanaleigenschaften möglichst schnell Verbindung mit der BS aufnehmen zu können.

3.5.2. Pre Decoder

Nach dem Block *PBCH demux* folgt der Block *Pre Decoder* in dem die Entzerrung durchgeführt wird. In Abbildung 3.10 sind, wegen der unbekannten Antennenkonfiguration, zwei Blöcke *Pre Decoder* mit unterschiedlicher Parametrisierung vorhanden. Die in diesen Blöcken durchgeführten Berechnungen wurden bereits in Abschnitt 2.6.2 vorgestellt. Der Block *Pre Decoder* benötigt zwei Parameter, die Antennenkonfiguration und die Übertragungsart. Durch die Angabe der Anzahl der Antennenports (N_{ant}) ergibt sich direkt die Anzahl der Eingänge, da für jeden Antennenport ein weiterer Eingang für Kanalschätzwerte benötigt wird. Der Parameter Übertragungsart kann die Werte *tx_diversity* und *spatial_multiplexing* annehmen, da aber für den PBCH nur *tx_diversity* benötigt wird, ist *spatial_multiplexing* derzeit nicht implementiert. Am Ausgang des Blocks *Pre Decoder* liegen die korrigierten Abtastwerte für alle Layer in einem Vektor vor. Dies ist möglich, da die Gesamzanzahl der Abtastwerte unabhängig von der Anzahl der Layer immer gleich ist, nur deren Ordnung unterscheidet sich.

3.5.3. Layer Demapper

Dem Block *Layer Demapper* wird ein Vektor mit den Abtastwerten für alle Layer übergeben um die Vereinigung der Layer in einen Datenfluss durchzuführen. Wie schon dem Block *Pre Decoder* werden auch hier die N_{ant} und die Übertragungsart als Parameter übergeben. Bei einer Parametrisierung mit nur einem Antennenport wird der Eingangsvektor lediglich an den Ausgang weitergegeben und somit könnte der Block entfernt werden.

3.5.4. QPSK Softdemodulation

Die QPSK Demodulation wird im Block *QPSK soft demod* durchgeführt. Wie in Abschnitt 2.6.2 beschrieben kann eine weiche Demodulation durchgeführt werden. Die komplexen Modulationsymbole $x(i)$ werden dabei nach Tabelle 3.4 in Gleitkommazahlen $\tilde{b}(i)$ für die entsprechenden Bits demoduliert. Dabei kommt Non Return to Zero (NRZ) Kodierung der einzelnen Bits zum Einsatz mit der auch nachfolgende Blöcke arbeiten. Der Zusammenhang zwischen NRZ Bits und Null/Eins kodierten Bits ist durch

$$b_{\mathrm{NRZ}}(i) = 1 - 2 \cdot b_{0,1}(i) \tag{3.7}$$

gegeben. Das Vorzeichen bestimmt den Wert des Bits. Darüberhinaus wird durch die NRZ-Kodierung der Abstand der verschiedenen Symbole bei der Demodulation nicht verringert, wodurch die Sicherheit der Entscheidung auf einzelne Bits höher ist.

$$\begin{aligned} \tilde{b}(2i) &= \Re(x(i)) \\ \tilde{b}(2i+1) &= \Im(x(i)) \end{aligned}$$

Tabelle 3.4.: QPSK Demodulationsschema

3.5.5. Descrambling

Das Descrambling stellt die inverse Operation zum Scrambling im Sender dar. Dabei wird die Scramblingsequenz $c(i)$, die schon im Sender für das Scrambling genutzt wurde, auch zum Descrambling genutzt. Da sie nur von der Cell ID (N_{ID}) abhängig ist, wird dieser Parameter über den Block *Cell ID Daemon* an den Block *Descrambling* übergeben. Das Descrambling wurde in Abschnitt 2.6.2 erklärt. Für eine vollständige Sequenz müsste der Eingangsvektor eine Länge von 1920 Elementen haben. Da der PBCH jedoch über vier Frames verteilt gesendet wird, hat der Eingangsvektor nur eine Länge von 480 Elementen. Der Eingangsvektor wird viermal aneinandergereiht, sodass er eine Länge von 1920 Elementen hat, und danach durch $\tilde{b}_d(i) = \tilde{b}(i) * c(i)$ mit der Scramblingsequenz verrechnet.

Im Block *Descrambling* ist nicht bekannt welches Viertel der Sequenz $\tilde{b}_d(i)$ nach dem Descrambling gültige Daten enthält. Erst nach dem Block *CRC Calculator* kann dies überprüft werden. Die Position des Viertels mit gültigen Daten innerhalb der Sequenz $\tilde{b}_d(i)$ bestimmt die beiden niedrigwertigsten Bits der System Frame Number (SFN). Die berechnete Sequenz $\tilde{b}_d(i)$ wird vom Block *Descrambling* in sechzehn Teilsequenzen mit jeweils 120 Elementen aufgeteilt. Eine Teilsequenz enthält alle Informationen, die durch den MIB übertragen werden, also sind in einem Viertel der Sequenz $\tilde{b}_d(i)$ die Informationen bereits vierfach redundant enthalten. Diese Eigenschaft der Teilsequenzen wird ausgenutzt um ein Softcombining durchzuführen. Die Teilsequenzen werden einmal mit und einmal ohne Softcombining an nachfolgende Blöcke weitergegeben, dadurch werden aus Eingangsvektoren mit 480 Elementen des Blocks *Descrambling* insgesamt jeweils 32 Ausgangsvektoren mit 120 Elementen erzeugt. Um die Position der Teilsequenzen innerhalb der Sequenz

$\tilde{b}_d(i)$ in den folgenden Blöcken identifizieren zu können, wird jedem Vektor mit einer Teilsequenz, dessen Position als Tag übergeben.

3.6. BCH-Dekodierung

An die PBCH-Dekodierung schließt sich direkt die Broadcast Channel (BCH)-Dekodierung an. Diese besteht aus dem Rate Unmatching, der Viterbiberechnung und der CRC-Überprüfung. In Abschnitt 2.6.1 wurden die einzelnen Verarbeitungsschritte des BCH bereits eingeführt. Einen Überblick über die einzelnen Schritte bietet Abbildung 3.11. Im Folgenden wird auf die Implementierung der Dekodierung eingegangen.

Abbildung 3.11.: Innere Struktur des Blocks *BCH Decoding*

3.6.1. Rate Unmatching

Der Block *Rate unmatch* führt das *Subblock Deinterleaving* durch und gruppiert die so entstandenen Sequenzen für den Block *BCH Viterbi*. In Abschnitt 2.6.1 wurde der verwendete Faltungscode vorgestellt. Dabei werden drei Bit lange Codeworte berechnet. Der Block *BCH Viterbi* benötigt diese drei Bits als direkt aufeinanderfolgende Werte. Beim *Subblock Interleaving* im Sender werden diese jedoch jeweils unterschiedlichen Subblocks zugeordnet. Der Block *Rate unmatch* führt deshalb das *Subblock Deinterleaving* durch und gruppiert diese Ausgangselemente für den Block *BCH Viterbi*. Insgesamt werden so die 120 Elemente der Eingangsvektoren des Blocks *Rate unmatch* umgruppiert und in Ausgangsvektoren mit 120 Elementen weitergegeben.

Subblock Deinterleaving

Als Teil des Rate Unmatching wird ein *Subblock Deinterleaving* durchgeführt. Dazu wird die Interleaversequenz, wie in Abschnitt 2.6.1 angegeben, einmal berechnet und dann anhand der berechneten Indizes das Deinterleaving vorgenommen. Dieses ist für alle Subblocks gleich und wird deshalb für alle Subblocks parallel durchgeführt.

3.6.2. Viterbidekodierer

Der in Abbildung 3.11 zu sehende Block *BCH Viterbi* repräsentiert einen hierarchischen GNU Radio Block. Seine einzelnen Blöcke und die Vektorgrößen zwischen den Blöcken sind in Abbildung 3.12 dargestellt. Wie in Abschnitt 2.6.1 beschrieben wird zunächst der Vektor dupliziert, um den Viterbidekodierer in den richtigen Anfangszustand zu bringen.

Da der von GNU Radio bereitgestellte Viterbidekodierblock einzelne Abtastwerte und keine Vektoren verlangt, müssen diese entsprechend in einen Datenfluss einzelner Abtastwerte umgewandelt werden. Danach werden dem Viterbidekodierer diese Abtastwerte übergeben, dessen Ausgangsdaten wieder in Vektoren transformiert und die jeweils erste Hälfte davon entfernt. Die erste Hälfte entspricht dem Vektor, der den Viterbidekodierer in den gewünschten Initialzustand gebracht hat, und ist deshalb ansonsten unbrauchbar. Da es sich um einen Coderate ein drittel Faltungscode handelt, ist das Verhältnis von Eingangs- zu Ausgangsdatenrate des Blocks *BCH Viterbi* ebenfalls ein Drittel.

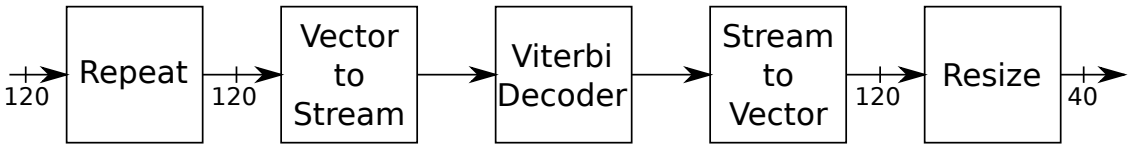


Abbildung 3.12.: Schematische Darstellung des Blocks *BCH Viterbi*

Parametrisierung des Viterbiblocks

In diesem Abschnitt wird auf die Parametrisierung des von GNU Radio bereitgestellten Block *Viterbi Combo* eingegangen, da diese Parametrisierung sehr umfangreich ist. In Tabelle 3.5 werden die einzelnen Parameter zusammengefasst. Der Faltungscode wird immer blockweise für 40 Bit – 24 Bit MIB, 16 Bit CRC-Prüfwort – berechnet. Daraus folgt, dass ein Codeblock K$=40$ Elemente hat. Der Initialzustand wird zu SO$=0$ gesetzt. Dies könnte aber in diesem Fall jeder beliebige zulässige Wert sein, da durch die beschriebene Wiederholung erst der Initialzustand eingestellt wird. Der Endzustand wird auf SK$=-1 \simeq$ *nicht verfügbar* gesetzt. Dies resultiert aus der Verwendung des *tail-biting* in der Basisstation (BS). Die Dimension der Konstellationsvektoren wird auf D$=3$ gesetzt.

Parameter	Erläuterung
FSM	Finite State Machine
K	Anzahl Werte für einen Codeblock
SO	Initialzustand des Dekodierers
SK	Endzustand des Dekodierers
D	Dimension der Konstellationsvektoren
Konstellation	Mögliche Eingangswertkombinationen
Berechnungsart	beispielsweise euklidische Distanz

Tabelle 3.5.: Parameter des GNU Radio Blocks *Viterbi Combo*

Die Finite State Machine (FSM) beschreibt den Faltungscode, der bei der Kodierung verwendet wurde. Sie wird durch die Eingangs- und Ausgangsdimension (1, 3), sowie einem Vektor mit den dezimalen Werten von G_0, G_1, G_2 aus Abbildung 2.12, parametrisiert. Aus diesen Werten ergibt sich auch, dass die FSM zwei Eingangszustände, 64 innere Zustände und acht Ausgangszustände annehmen kann.

Die Konstellation wird durch Vektoren der Dimension D beschrieben. Sie enthält alle acht möglichen, NRZ kodierten Vektoren, die man aus $\{-1, 1\}$ bilden kann. Diese müssen in der richtigen Reihenfolge, wie in Tabelle 3.6, angeordnet sein um die acht Ausgangszustände des Faltungscodes richtig abzubilden. Die Eingangswerte des Block *Viterbi Combo* sind NRZ-kodierte Bits mit ihren Zuverlässigkeitswerten, die gruppiert werden zu Vektoren der Dimension D. Diese werden mit den Konstellationsvektoren, die durch die Konstellation gegeben sind, verglichen. Die genaue Berechnungsvorschrift wird für diesen Block zu *euklidische Distanz* durch den Parameter Berechnungsart gesetzt. Der Block *Viterbi Combo* entscheidet dann auf das Bit, welches durch den Konstellationsvektor mit dem geringsten euklidischen Abstand repräsentiert wird.

Zustand	0	1	2	3	4	5	6	7
Vektor	$\begin{pmatrix}1\\1\\1\end{pmatrix}$	$\begin{pmatrix}1\\1\\-1\end{pmatrix}$	$\begin{pmatrix}1\\-1\\1\end{pmatrix}$	$\begin{pmatrix}1\\-1\\-1\end{pmatrix}$	$\begin{pmatrix}-1\\1\\1\end{pmatrix}$	$\begin{pmatrix}-1\\1\\-1\end{pmatrix}$	$\begin{pmatrix}-1\\-1\\1\end{pmatrix}$	$\begin{pmatrix}-1\\-1\\-1\end{pmatrix}$

Tabelle 3.6.: Vektoren für die verwendete Konstellation

3.6.3. CRC-Berechnung

Nach dem Block *Viterbi Combo* wird das Ergebnis durch den CRC Block überprüft. GNU Radio verwendet die Bibliothek *C++ Boost* für verschiedene Aufgaben, [Boo12]. Diese stellt auch eine CRC-Berechnungsfunktion bereit und wird daher hier im Block *CRC Calculator* verwendet.

Zunächst wird der Eingangsvektor aufgeteilt in Datenbits und Prüfbits. Da der Eingangsvektor noch aus 40 Bytes, die jeweils ein Bit repräsentieren besteht, müssen diese noch gepackt werden, sodass jedes Byte acht Bit repräsentiert. Danach wird für jede Antennenkonfiguration mit der CRC-Berechnungsfunktion ein Prüfwort berechnet und mit dem Empfangenen verglichen. Die Parametrisierung der *Boost*-Funktion ist in Tabelle 3.7 angegeben. Stimmt das empfangene Prüfwort mit einem der berechneten Prüfwörter überein, so wird entsprechend des Prüfworts auf die Antennenkonfiguration entschieden und die Datenbits als Vektor auf dem Datenausgang und die Antennenkonfiguration auf dem Antennenkonfigurationsausgang weitergegeben. Kommt es zu keiner Übereinstimmung, wird $N_{\text{ant}} = 0$ weitergegeben. Dies indiziert eine fehlgeschlagene CRC Prüfung.

3.7. Master Information Block Dekodierung

Der Block *MIB unpack* stellt im implementierten LTE-Flowgraph die Senke dar. Er dekodiert den MIB immer dann, wenn $N_{\text{ant}} > 0$ ist. Die Bedeutung und Position der einzelnen Teile des MIB wurde in Abschnitt 2.6.3 vorgestellt. Der Block *Descrambling* vergibt für jeden Ausgangsvektor einen Tag, der im Block *MIB unpack* zur Berechnung der niedrig-

Parameter	Wert
Anzahl Bits	24
CRC Polynom	1012_H
Initialzustand	0000_H
Final XOR	Hexadezimalzahl entsprechend Tabelle 2.2
Reflect Input	FALSE
Reflect REM	FALSE

Tabelle 3.7.: Parametrisierung der CRC-Berechnungsfunktion

wertigsten zwei Bits der SFN genutzt wird. Auf diese Weise liegt in diesem Block die komplette SFN vor.

4. Simulation und Messung

In diesem Kapitel werden einige der Ergebnisse aus der Matlab Simulation mit denen aus den Messdaten in GNU Radio verglichen. Dazu wird zunächst der Messaufbau vorgestellt und anschließend auf die Ergebnisse und deren Bewertung eingegangen.

4.1. Simulation

Für das Verständnis des LTE-Standards werden zunächst die einzelnen betrachteten Teile des Standards in Matlab implementiert. Dazu werden jeweils die Signalverarbeitungsschritte im Sender und deren Gegenstück im Empfänger implementiert um die verwendeten Algorithmen zu testen. Die mit GNU Radio aufgenommenen Messdaten werden dann für weitere Tests in Matlab herangezogen. Nachdem die Implementierung in Matlab mit Messdaten korrekt arbeitet, wird die Implementierung nach GNU Radio portiert.

4.2. Messaufbau

Der Ausbau des Long Term Evolution (LTE) Funknetzes wird in Deutschland zunächst in den sogenannten *weißen Flecken* durchgeführt, also jenen Gebieten, in denen bisher noch kein schnelles Internet verfügbar ist. Dies führte dazu das LTE am Tag der Messung noch nicht in direkter Umgebung des Instituts verfügbar war. Die Messung wurde daher in Gondelsheim durchgeführt. Der Messaufbau ist in Abbildung 4.1 abgebildet und wird in Tabelle 4.1 beschrieben. In Abbildung 4.3 ist die Basisstation (BS) auf einem Turm in Gondelsheim zu sehen. Der Messaufbau ist so platziert, dass eine direkte Sichtverbindung zwischen diesem und der BS besteht. Die Messdaten wurden zunächst als Rohdaten gespeichert um sie später für Tests nutzen zu können.

USRP	N210
Daughterboard	WBX Rev. 2 mit WBX-FE-Simple Rev. 2.0
Abtastrate	10 MS/s
Frequenz	806 MHz
Datum	23.5.2012
Ort	Gondelsheim

Tabelle 4.1.: Daten des Messaufbaus

Mithilfe eines tragbaren Spektrumanalysators wurde zunächst eine möglichst ideale Position gesucht. Entscheidungskriterium für einen Standort war dabei die höhe des SNR.

(a) **(b)**

Abbildung 4.1.: Aufnahmen des Messaufbaus

In Abbildung 4.2 ist das aufgenommene Spektrum dargestellt.

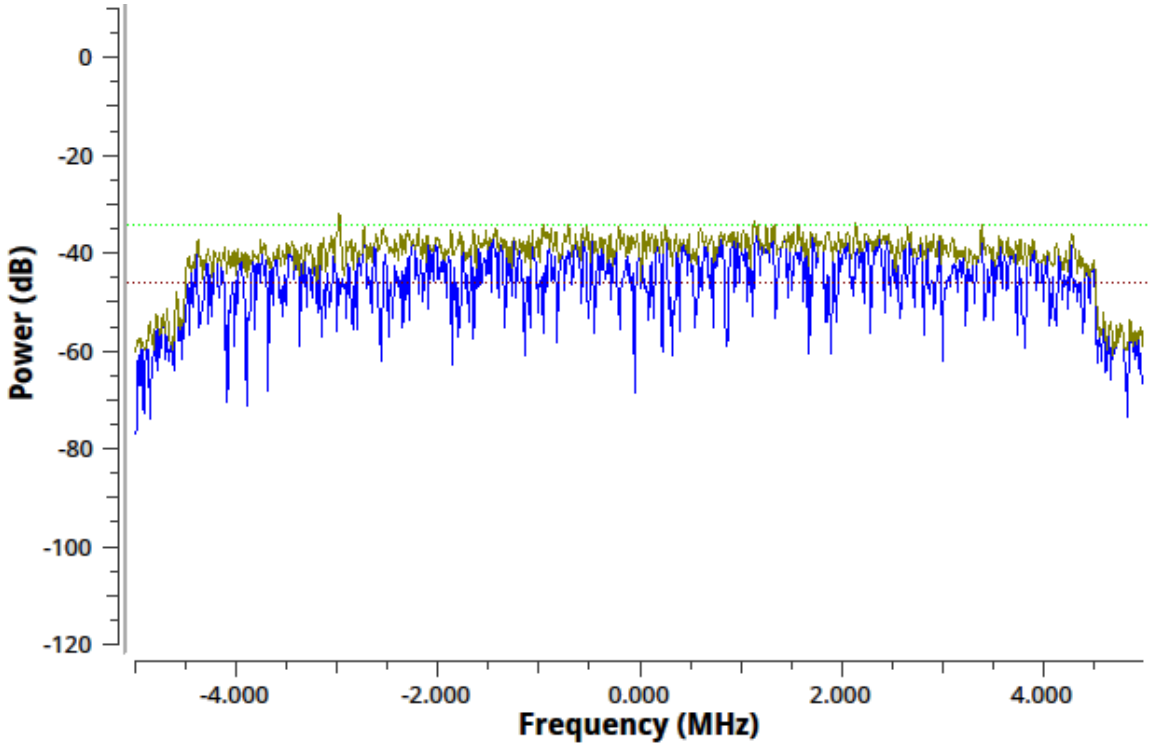


Abbildung 4.2.: Spektrum der Messdaten (blau: Momentaufnahme, gelb: Maximum)

4.3. Vergleich von Simulation und Messung

Simulation und Messung teilt sich in zwei Unterbereiche, die Synchronisation und die Dekodierung. Zunächst wird auf die Ergebnisse der Synchronisationsblöcke eingegangen und danach auf die Dekodierblöcke.

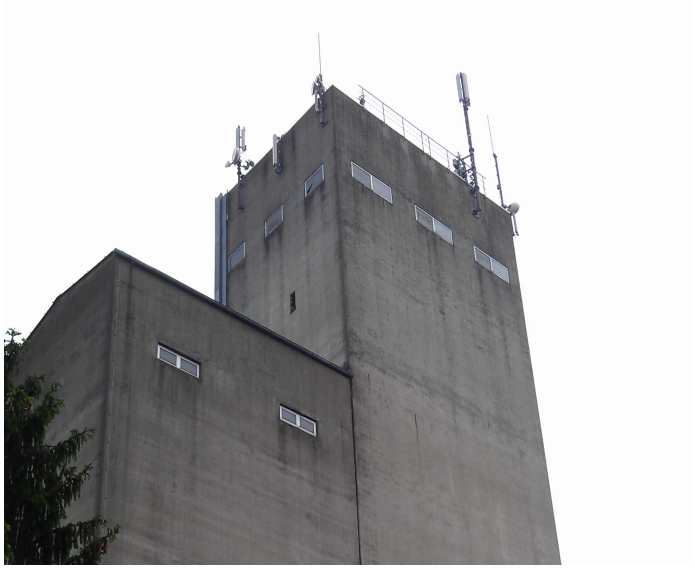


Abbildung 4.3.: Aufnahme der Basisstation auf einem Turm

4.3.1. Synchronisation

Orthogonal Frequency Division Multiplex (OFDM) Systeme sind generell sehr empfindlich gegenüber Synchronisationsfehlern. Wurde während der Synchronisation auf einen zu frühen oder zu späten Beginn der OFDM-Symbol entschieden, kann schon eine Verschiebung um wenige Abtastwerte dazu führen, dass das System keine Daten mehr dekodieren kann.

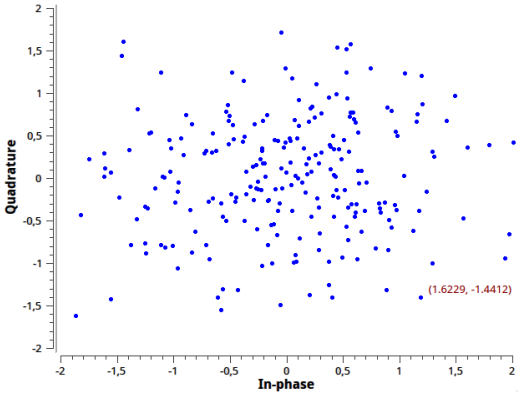
Die Cyclic Prefix (CP) Synchronisation liefert innerhalb weniger OFDM-Symbole gute Ergebnisse für die Zeitsynchronisation, muss aber immer durch die Primary Synchronization Symbol (PSS) Synchronisation unterstützt werden. Diese liefert eine sehr genaue Zeitsynchronisation. Beide Synchronisationsschritte in Kombination können die Zeitsynchronisation komplett durchführen. Die Secondary Synchronization Symbol (SSS) benötigt dann meist nur einen Dekodierversuch.

Im Vergleich zu Zeitsynchronisationfehlern sind Frequenzsynchronisationsfehler weniger kritisch. Der ursprüngliche Ansatz innerhalb der CP-Synchronisation, auch eine Fractional Frequency Offset (FFO) Korrektur durchzuführen, wurde jedoch verworfen, da in Tests meist das Gegenteil – ein zusätzlicher FFO – eintrat.

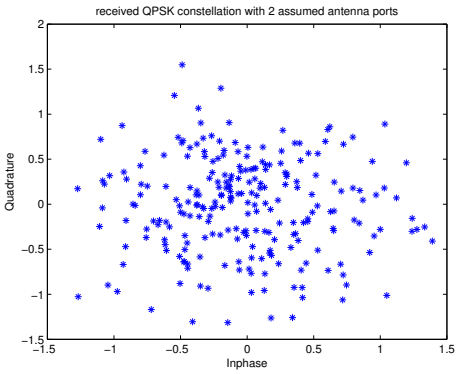
4.3.2. Dekodierung

Auf Grund der direkten Sichtverbindung wurde zunächst angenommen, dass Mehrwegeausbreitung keine Rolle spielen wird. In Abbildung 4.4 sind die Ergebnisse aus Simulation und Messung gegenüber gestellt. Die erwartete Häufung der Modulationssymbole um deren ideale Werte, tritt in den Messdaten nicht auf, was zahlreiche Ursachen haben kann. Die erste These, dass die Synchronisation schlechte Ergebnisse liefert, konnte durch Tests, die die Synchronisation in Zeit- oder Frequenzrichtung verschieben, als unwahrscheinlich

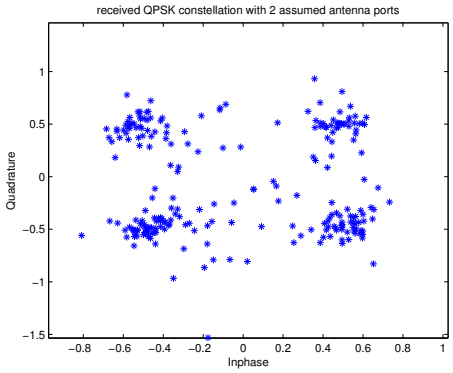
verworfen werden. Dies führt zu der in Abbildung 4.4 skizzierten These, dass Mehrwegeausbreitung oder Effekte im USRP zu dieser starken Verzerrung führt. Im Rahmen dieser Arbeit liegt der Fokus jedoch auf der Implementierung des Empfängers, deshalb wurden keine weiteren Untersuchungen in dieser Richtung vorgenommen.



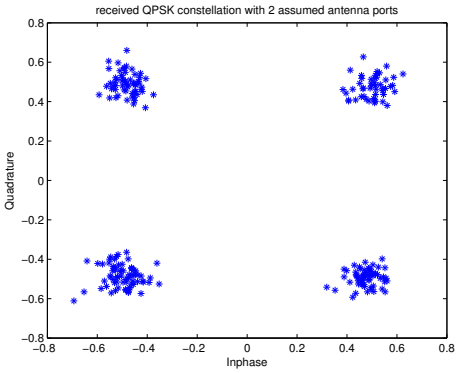
(a) Messwerte



(b) Simulierte Werte, konstanter Kanal (SNR=6dB, Verzögerung: 9,90 µs, 23,10 µs)



(c) Simulierte Werte, konstanter Kanal (SNR=6dB, Verzögerung: 1,32 µs, 2,64 µs)



(d) Simulierte Werte, konstanter Kanal (SNR=6dB, keine Verzögerung)

Abbildung 4.4.: Konstellationsdiagramme nach dem Layer Demapping

4.4. Ergebnisse

Ziel ist es, die grundlegenden LTE-Systeminformationen auszulesen. Für die aufgenommenen Testdaten wurden die folgenden Daten aus Tabelle 4.2 dekodiert. Ein kurzer Teil der dazugehörigen Textausgabe ist in Abbildung 4.5 wiedergegeben.

Es wird der Abschluss der PSS und SSS Synchronisationsblöcke und deren Ergebnis protokolliert, also ob die Synchronisation erfolgreich ist. Die Zeilen `remove_cp_cvc` und Folgende werden ausgegeben, sobald der Block *remove CP* beginnt Daten weiterzugeben. Das Attribut `frame_start` gibt die absolute Anzahl an Abtastwerten wieder, die bisher zu

Attribut	Wert
N_{ID}	124
N_{ant}	2
$N_{\mathrm{RB}}^{\mathrm{DL}}$	50
PHICH Duration	normal
PHICH Resources	1

Tabelle 4.2.: Ergebnisse der Messdaten

diesem Block propagiert wurden. Sie ist auch ein Maß für die Synchronisationsgeschwindigkeit. Ein niedrigerer Wert bezeichnet die Synchronisation in einer kürzeren Zeitspanne. In diesem Fall wurde mit 7,68 MS/s gearbeitet, also wurde innerhalb von 73,1 ms synchronisiert. `mod_start` gibt den Offset zwischen dem Beginn des ersten vollständig in den Messdaten enthaltenen Frame und dem ersten Abtastwert an.

Die Zeilen, die mit `Decoded` beginnen, indizieren, dass der Cyclic Redundancy Check (CRC) erfolgreich war und es wird ein Teil der dekodierten Daten ausgegeben. Besondere Bedeutung haben das erste und letzte Datum. Das erste Datum gibt die aktuelle System Frame Number (SFN) an und das auf `diff` folgende Datum ist ein Maß für die Anzahl erfolgreicher Dekodierungen. Optimal wäre hier immer eine 1, dies würde bedeuten das der Physical Broadcast Channel (PBCH), und damit der Broadcast Channel (BCH) und der Master Information Block (MIB), in jedem Frame erfolgreich dekodiert werden konnten.

```
pss_calc_vc is locked! half_frame_start = 23855
abs_pos = 187879 N_id_2 = 1 corr_val = 1518.589600
sss_calc_vci is locked! frame_start = 23855
abs_pos = 484655 cell_id = 124

remove_cp_cvc
frame_start = 561455
mod start   = 23855

Decoded 386 N_ant = 2 N_rb_dl = 50 diff = 387
Decoded 392 N_ant = 2 N_rb_dl = 50 diff = 6
Decoded 398 N_ant = 2 N_rb_dl = 50 diff = 6
Decoded 399 N_ant = 2 N_rb_dl = 50 diff = 1
Decoded 404 N_ant = 2 N_rb_dl = 50 diff = 5
Decoded 405 N_ant = 2 N_rb_dl = 50 diff = 1
```

Abbildung 4.5.: Textausgabe des LTE-Flowgraphs

5. Zusammenfassung

In dieser Arbeit wurden die Grundlagen für den Empfang von Long Term Evolution (LTE)-Signalen in GNU Radio geschaffen. Dabei wurde die Synchronisation und die Orthogonal Frequency Division Multiplex (OFDM)-Operationen implementiert, so dass darauf zukünftige Erweiterungen der Implementierung aufbauen können. Zusätzlich wurden Funktionen zur Kanalschätzung implementiert. Abschließend wurde die Signalverarbeitung für den Empfang der grundlegenden LTE-Systeminformationen integriert. Es ist so möglich den Physical Broadcast Channel (PBCH), Broadcast Channel (BCH) und den Master Information Block (MIB) zu dekodieren.

Für die Entwicklung und Planung der Implementierung wurde mit dem Studium des LTE-Standards begonnen. Da dieser sehr umfangreich ist, wurde in dieser Arbeit die Spezifikation der Bitübertragungsschicht (PHY-Layer) sowie die für den MIB relevanten Teile betrachtet. Der LTE-Standard spezifiziert die Struktur der Sendesignale. Der Aufbau des Empfängers ist dann durch die entsprechenden inversen Operationen definiert. Dazu wurden auch mehrere wissenschaftliche Artikel studiert um aus diesen geeignete Methoden für den Empfänger auszuwählen.

Bei der Implementierung in GNU Radio wurde besonders auf Modularität geachtet, sodass für jeden Signalverarbeitungsschritt ein eigener Block bereitgestellt wird. Dies eröffnet die Möglichkeit in Zukunft zusätzliche Funktionen, wie beispielsweise der Empfang weiterer im LTE-Standard definierter Kanäle, zu integrieren. Die Blöcke für die Signalverarbeitung weiterer Kanäle kann einfach mit dem bestehenden Flowgraph verbunden werden. Außerdem wurde darauf geachtet, dass in GNU Radio implementierte Blöcke nach Möglichkeit generisch gehalten wurden, um deren Wiederverwendbarkeit zu erhöhen. Die bestehenden Blöcke können außerdem weiter optimiert und erweitert werden. Dies ist besonders interessant im Bereich der Synchronisationsblöcke und der Kanalschätzung. Die Integration von zusätzlichen in GNU Radio bereitgestellten Funktionen ist ebenfalls ein zukünftiges Gebiet für Erweiterungen. Hier ist explizit das *Message Passing*, welches in GNU Radio integriert ist, zu erwähnen.

Eine Anwendungsmöglichkeit der vorgestellten Arbeit besteht darin Belegungsmessungen von LTE-Funkzellen durchzuführen. Hierzu könnte man beispielsweise den Physical Hybrid-ARQ Indicator Channel (PHICH) betrachten, der Rückschlüsse auf die Anzahl aktiver Nutzer in der Funkzelle zulässt.

A. ausführliche Darstellung der Sequenzen für das SSS

$$\begin{aligned} d(2n) &= \begin{cases} s_0^{(m_0)}(n)c_0(n) & \text{in Subframe 0 / slot 0} \\ s_1^{(m_1)}(n)c_0(n) & \text{in Subframe 5 / slot 10} \end{cases} \\ d(2n+1) &= \begin{cases} s_1^{(m_1)}(n)c_1(n)z_1^{m_0}(n) & \text{in Subframe 0 / slot 0} \\ s_0^{(m_0)}(n)c_0(n)z_1^{m_1}(n) & \text{in Subframe 5 / slot 10} \end{cases} \end{aligned} \tag{A.1}$$

Die Indizes m_0 und m_1 werden mit der Cell Identity Group (N_{ID}^1) wie folgt berechnet:

$$\begin{aligned} m_0 &= m' \mod 31 \\ m_1 &= (m_0 + \lfloor m'/31 \rfloor + 1) \mod 31 \\ m' &= N_{\text{ID}}^1 + q(q+1)/2 \\ q &= \lfloor \tfrac{N_{\text{ID}}^1 + q'(q'+1)/2}{30} \rfloor \\ q' &= \lfloor N_{\text{ID}}^1/30 \rfloor \end{aligned} \tag{A.2}$$

$$\begin{aligned} s_0^{(m_0)}(n) &= \tilde{s}((n+m_0) \mod 31) \\ s_1^{(m_1)}(n) &= \tilde{s}((n+m_1) \mod 31) \end{aligned} \quad 0 \le n \le 30 \tag{A.3}$$

$$\tilde{s}(i) = 1 - 2x(i) \quad 0 \le i \le 30 \tag{A.4}$$

$$x(\bar{i}+5) = (x(\bar{i}+2) + x(\bar{i})) \mod 2 \quad 0 \le \bar{i} \le 25 \tag{A.5}$$

mit den Initialwerten $x(0) = 0, x(1) = 0, x(2) = 0, x(3) = 0, x(4) = 1$.

$$\begin{aligned} c_0(n) &= \tilde{c}((n + N_{\text{ID}}^2) \mod 31) \\ c_1(n) &= \tilde{c}((n + N_{\text{ID}}^2 + 3) \mod 31) \end{aligned} \quad 0 \le n \le 30 \tag{A.6}$$

$$\tilde{c}(i) = 1 - 2x(i) \quad 0 \le i \le 30 \tag{A.7}$$

$$x(\bar{i}+5) = (x(\bar{i}+3) + x(\bar{i})) \mod 2 \quad 0 \le \bar{i} \le 25 \tag{A.8}$$

mit den Initialwerten $x(0) = 0, x(1) = 0, x(2) = 0, x(3) = 0, x(4) = 1$.

$$\begin{aligned} z_1^{(m_0)}(n) &= \tilde{z}((n + (m_0 \mod 8)) \mod 31) \\ z_1^{(m_1)}(n) &= \tilde{z}((n + (m_1 \mod 8)) \mod 31) \end{aligned} \quad 0 \le n \le 30 \tag{A.9}$$

$$\tilde{z}(i) = 1 - 2x(i) \quad 0 \le i \le 30 \tag{A.10}$$

$$x(\bar{i}+5) = (x(\bar{i}+4) + x(\bar{i}+2) + x(\bar{i}+1) + x(\bar{i})) \mod 2 \quad 0 \le \bar{i} \le 25 \tag{A.11}$$

mit den Initialwerten $x(0) = 0, x(1) = 0, x(2) = 0, x(3) = 0, x(4) = 1$.

B. PN Sequenzgenerator

Der Pseudo-Noise (PN) Sequenzgenerator wird für verschiedene Aufgaben im LTE System genutzt und ist in [3GP11b] Abschnitt 7.2 definiert. Die erzeugte Sequenz $c(N)$ der Länge M_{PN} mit $n = 0, \ldots, M_{PN} - 1$ ist definiert durch

$$\begin{aligned} c(n) &= (x_1(n + N_c) + x_2(n + N_c)) \mod 2 \\ x_1(n + 31) &= (x_1(n + 3) + x_1(n)) \mod 2 \\ x_2(n + 31) &= (x_2(n + 3) + x_2(n + 2) + x_2(n + 1) + x_2(n)) \mod 2 \end{aligned} \tag{B.1}$$

mit $N_c = 1600$. x_1 wird initialisiert durch

$$x_1(0) = 1, \quad x_1(n) = 0 \text{ für } n = 1, \ldots, 30 \tag{B.2}$$

und x_2 durch die Gleichung

$$c_{init} = \sum_{i=0}^{30} x_2(i) \cdot 2^i \tag{B.3}$$

Die einzige Variable, die benötigt wird, ist also c_{init}.

Literaturverzeichnis

[3GP11a] 3GPP. LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding. TS 36.212 v10.0.0, 3rd Generation Partnership Project (3GPP), January 2011.

[3GP11b] 3GPP. LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation. TS 36.211 v10.0.0, 3rd Generation Partnership Project (3GPP), January 2011.

[3GP12] 3GPP. LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) radio transmission and reception. TS 36.101 v10.6.0, 3rd Generation Partnership Project (3GPP), March 2012.

[Ala98] S.M. Alamouti. A simple transmit diversity technique for wireless communications. *Selected Areas in Communications, IEEE Journal on*, 16(8):1451 –1458, oct 1998.

[Boo12] Boost C++ Libraries Website. `http://www.boost.org`, October 2012. accessed 25th October 2012.

[Chu72] D. Chu. Polyphase codes with good periodic correlation properties (corresp.). *Information Theory, IEEE Transactions on*, 18(4):531 – 532, jul 1972.

[GNU12] GNU Radio Website. `http://www.gnuradio.org`, October 2012. accessed 25th October 2012.

[GZMA10] A. Ghosh, J. Zhang, R. Muhamed, and J.G. Andrews. *Fundamentals of Lte*. Prentice Hall Communications Engineering and Emerging Technologies Series. Prentice Hall PTR, 2010.

[MEJ+09] Konstantinos Manolakis, David Manuel Gutiérrez Estévez, Volker Jungnickel, Wen Xu, and Christian Drewes. A closed concept for synchronization and cell search in 3gpp lte systems. In *WCNC*, pages 616–621, 2009.

[Mit92] Joseph Mitola. Software radios-survey, critical evaluation and future directions. *National Telesystems Conference*, May 1992.

[Sch11] Michael Schnell. Skript zu OFDM-basierte Übertragungstechniken, November 2011.