Untersuchung und Simulation von Low-Density Parity-Check-Codes

MASTERARBEIT

Michael Petter

eingereicht am
Universitäts-Masterstudiengang

Telematik

Institut für
Kommunikationsnetze und Satellitenkommunikation

Technische Universität Graz

Betreuer: Dipl.-Ing. Dr. Wilfried Gappmair
Betreuer: Dipl.-Ing. Michael Bergmann
Begutachter: Univ.-Prof. Dipl.-Ing. Dr. Otto Koudelka

Graz, im Februar 2012

Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Graz, im Februar 2012

Michael Petter

Inhaltsverzeichnis

Kurzfassung

Seit den 90er Jahren befindet sich die Codierungstheorie im Umbruch. Etwa gleichzeitig mit der Entdeckung von Turbo-Codes, Anfang der 90er Jahre, wurden auch Low-Density Parity-Check (LDPC)-Codes, welche im Jahre 1963 von Gallager in seiner Dissertation mathematisch erstmals beschrieben wurden, wiederentdeckt. Die Einführung dieser beiden Codierverfahren ermöglichte erstmals Ergebnisse nahe an der von Shannon angegebenen Grenze, die vorher mit vertretbarem Aufwand nicht erreicht werden konnte. Mit irregulären LDPC-Codes ist es möglich, bis auf einige Tausendstel Dezibel an diese Grenze heranzukommen.

LDPC-Codes werden über Kontrollmatrizen dargestellt, welche typischerweise mit Einsen schwach besetzt sind, um den Rechenaufwand gering zu halten. Die Einsen wiederum werden, je nach Konstruktionsprinzip, zufällig oder auch durch entsprechende Verfahren in der Matrix platziert. Letztere sind als sogenannte strukturierte LDPC-Codes bekannt und zeichnen sich durch eine effizientere Codierung aus.

Die Decodierung von mit LDPC-Codes codierten Daten geschieht iterativ, wobei zwischen Hard- (Bit-Flipping) bzw. Soft-Decision (Message Passing) Algorithmen unterschieden wird. Letztere sind rechenintensiver, jedoch gibt es für beide Verfahren Varianten mit reduzierter Komplexität. LDPC-Codes bieten für große Blocklängen bessere Leistungsparameter als vergleichbare Turbo-Codes und können einfacher parallelisiert werden.

Abstract

Since the discovery of turbo codes in the early 90s, coding theory has run through a radical change. Almost simultaneously with the discovery of turbo codes, published in the early 90s, also low-density parity-check (LDPC) codes, which were mathematically described 1963 by Gallager for the first time in his thesis, were rediscovered. LDPC codes are capacity-approaching codes. Irregular codes proved to be only a few thousandth of a decibel away from the Shannon limit.

LDPC codes are represented via parity-check matrices, with the additional constraint that they are sparsely populated with ones so as to reduce the computational complexity. These matrices might be constructed in a random-like or a structured way. Structured codes have the advantage of an efficient encoding process.

The methods for iterative decoding are known as hard decision (bit flipping) and soft decision (message passing) algorithms; the latter is computationally more intesive, but there are low-complexity variants for both of them. Compared with turbo codes, LDPC codes exhibit better performance for larger blocklengths. Furthermore, they can be implemented more efficiently due to their high parallelization capability.

Kapitel 1

Einleitung

Seit SHANNON versuchten Wissenschaftler und Ingenieure fehlerkorrigierende Verfahren zu entwerfen, deren Leistungsfähigkeit möglichst nahe an die nach ihm benannte Grenze herankommt. Die Ziel bei der Suche nach geeigneten Codes ist es, bei gegebener Coderate diejenigen zu finden, die eine möglichst große Mindest-HAMMING-Distanz aufweisen, zeitgleich aber einfach zu codieren und decodieren sind. Doch die Versteifung auf die Optimierung der Mindest-HAMMING-Distanz eines Codes stellte sich im Nachhinein als problematisch dar.

Bis in die 90er Jahre des letzten Jahrhunderts erwies sich diese Suche als äußerst schwierig. Erst mit der Erfindung von Turbo-Codes [1] und der Wiederentdeckung von Low-Density Parity-Check (LDPC)-Codes [2] gelingt es, bis auf 1 dB oder weniger an diese Grenze heranzukommen. Sowohl die Erfinder der Turbo- als auch der LDPC-Codes standen ein wenig abseits der Codierungstheorie, was ihnen einen etwas anderen Blick auf die Materie ermöglichte.

LDPC-Codes wurden 1962 von Robert G. GALLAGER im Rahmen seiner Dissertation [3] am Massachusetts Institute of Technology (MIT) entwickelt. Sie erwiesen sich aber zu jener Zeit als nicht praktikabel und wurden deswegen nicht weiter beachtet. Heutzutage jedoch handelt es sich bei LDPC-Codes um eines der aktuellsten Themen im Bereich der fehlerkorrigierenden Verfahren.

LDPC-Codes fallen unter die Gruppe der linearen Blockcodes und werden über Matrizen beschrieben. Der Schlüssel für die Leistungsfähigkeit liegt darin, dass die beschreibende Matrix nur sehr wenige von Null verschiedene Stellen aufweist. Die Konstruktion solcher Matrizen erfolgt nach unterschiedlichen Ansätzen und bringt Codes mit verschiedenen Eigenschaften hervor. Die Decodierung erfolgt über Verfahren, bei denen das Ergebnis durch einen *iterativen* Vorgang berechnet wird. Dieser lässt sich effizient mit Mitteln der Graphentheorie behandeln und wird anhand von sogenanntenTANNER-Graphen (TG) [4] beschrieben.

Aufgabe dieser Masterarbeit war es, einen Überblick zum Thema LDPC-Codes zu geben, was in erster Linie durch ein umfassendes Studium der einschlägigen Fachliteratur geschah und durch Softwaresimulationen ergänzt wurde. Die Simulationsergebnisse wurden in der Arbeit mit Ergebnissen aus der Literatur verglichen.

Der Rest dieser Arbeit ist folgendermaßen gegliedert: In Kapitel 2 werden die grundlegenden Konzepte der Informationsübertragung betrachtet. Das Kapitel 3 beschäftigt sich mit linearen Blockcodes, zu denen die LDPC-Codes gehören. Im Kapitel 4 werden TANNER-Graphen sowie der Codiervorgang behandelt, während der Decodiervorgang Gegenstand von Kapitel 5 ist. Kapitel 6 widmet sich den wichtigsten Konstruktionsprinzipien. Implementation und Simulation werden in Kapitel 7 diskutiert, Schlussbemerkungen und Ausblick finden sich in Kapitel 8.

Kapitel 2

Kanalcodierung

Die Kanalcodierung ist eines der zentralen Elemente der Nachrichtentechnik, um eine Nachricht (fast) fehlerfrei über einen störanfälligen Kanal zu übermitteln. In der Codierungstheorie wird dazu grundsätzlich zwischen *fehlererkennenden* und *fehlerkorrigierenden* Maßnahmen unterschieden. Bei der ersten Methode (Automatic Repeat reQuest, ARQ) werden empfangsseitig etwaige Fehler nur erkannt und über einen Rückwärtskanal eine nochmalige Übertragung der fehlerhaften Daten veranlasst. Die Hauptanwendung findet sich in modernen Datennetzen, in welchen die natürliche Fehleranfälligkeit bzw. die Fehlerwahrscheinlichkeit ohne Codierung bereits sehr gering ist. Durch fehlerkorrigierende Verfahren kann eine Bitfehlerrate (Bit Error Rate, BER) von $= 10^{-9}$ und noch kleiner erreicht werden [5, S.140]. Ein Vorteil von ARQ-Methoden ist die geringere Komplexität des Empfängers, da dieser Fehler nur detektieren muss. Die fehlerkorrigierende Methode kommt dagegen in Bereichen zum Einsatz, in welchen kein Rückwärtskanal vom Empfänger zum Sender zur Verfügung steht. Auftretende Fehler müssen beim Empfänger vor Ort erkannt und, wenn möglich, korrigiert werden. Fehlerkorrigierende Lösungen (Forward Error Correction, FEC) benötigen hierfür zusätzlich hinzugefügte Informationen in Form von redundanten Bits. LDPC-Codes fallen unter die Kategorie der fehlerkorrigierenden Codes.

2.1 Quellen- und Kanalcodierung

Abbildung 2.1 [5, S.134] zeigt im Prinzip das Modell eines digitalen Übertragungssystems. Die *Quellencodierung* verfolgt das Ziel, die Daten von unnötigen bzw. redundanten Informationen zu befreien, um so die zu übertragende Datenmenge auf ein Minimum zu reduzieren. Die Quellencodierung liefert idealerweise ein von Redundanz befreites, komprimiertes Datenwort $\mathbf{u}$, welches im Codierer nach bestimmten Regeln auf ein Codewort $\mathbf{c}$ abgebildet wird. Dieser Schritt wird als *Kanalcodierung* bezeichnet. Das Codewort wird nun mit Hilfe des Modulators in ein entsprechendes (übertragbares)

Signal umgewandelt und über den fehleranfälligen Kanal übertragen. Der Demodulator wandelt nun das empfangene Signal zurück in ein Codewort **r**. Durch den Decodierungsprozess wird das Codewort wieder in ein Datenwort **d** zurück transformiert.

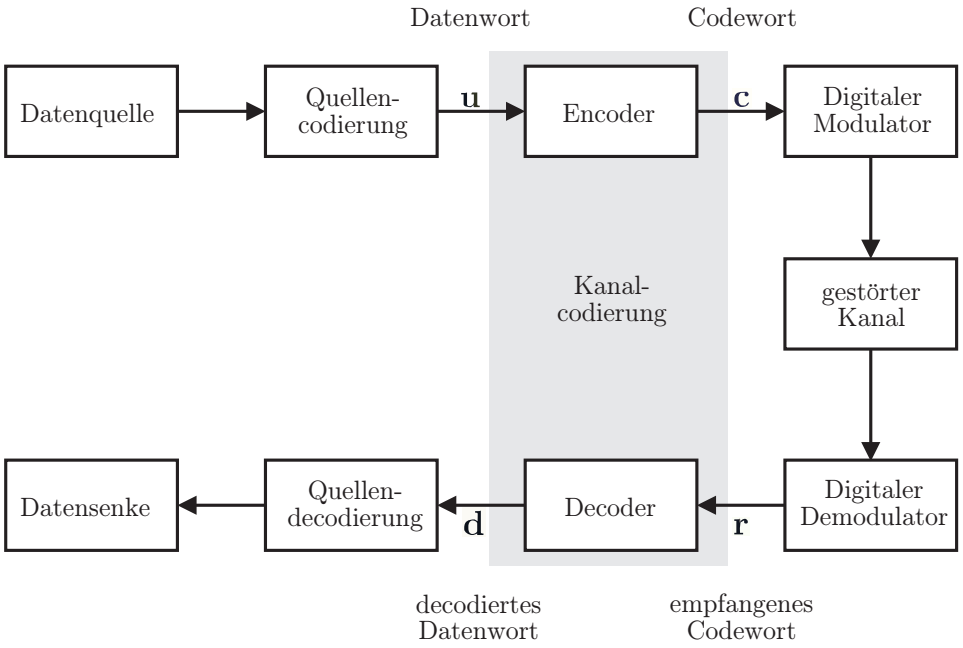

Abbildung 2.1: Vereinfachtes Modell eines digitalen Kommunikationssystemes mit hervorgehobener Kanalcodierung.

Durch die Kanalcodierung ist es theoretisch möglich, fast beliebig kleine Bitfehlerwahrscheinlichkeiten zu erreichen. Praktisch gesehen ist die Auswahl eines für die Applikation passenden Codes jedoch auch noch von anderen Faktoren abhängig. Dem Erreichen eines guten *Fehlererkennungs-* und *Fehlerkorrekturvermögens* steht die dazu benötigte Komplexität (Rechenaufwand, Kosten) sowie die vom Kanal zur Verfügung gestellten Bandbreite (Datendurchsatz) gegenüber [5, S.134].

2.2 Grundbegriffe der Kanalcodierung

Bei der Übertragung über einen Kanal wird es immer zu Störungen und in der Folge zu Fehlern kommen. Ziel ist die Beherrschung dieser Fehler durch den Einsatz fehlerkorrigierender Verfahren. Ein Kanal wird durch seine Bandbreite, das gewählte Modulationsverfahren und das betreffende Störmodell beschrieben. Aus diesen Beobachtungen leitete Shannon [6] den Begriff der sogenannten *Kanalkapazität* ab. Betrachten wir zunächst einen gestörten gedächtnislosen Kanal, wie ihn Abbildung 2.2 [7, S.51] zeigt. Die

Quelle besitzt eine Entropie H. durch die Störungen werden nicht alle Zeichen x_i am Empfänger richtig empfangen und sind daher von den zu erwarteten Werten y_i verschieden. Es gelangt nur ein gewisser Teil, Transinformation T genannt, zum Empfänger. Der Anteil $H(x|y)$, der verloren geht, wird *Äquivokation* oder auch Rückschlussentropie genannt. Die empfangene Entropie setzt sich aus der *Transinformation* und der Fehlinformation $H(y|x)$ oder *Irrelevanz*, welche durch Störungen entsteht zusammen:

$$T = H(x) - H(x|y) = H(y) - H(y|x) \tag{2.1}$$

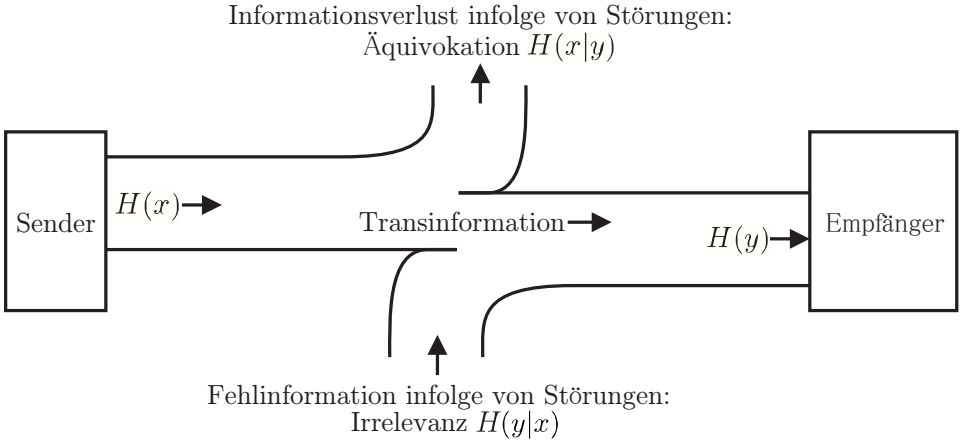

Abbildung 2.2: BERGERsches Kanaldiagramm.

SHANNON zeigt in seinem berühmten *Kanalcodierungstheorem*, dass für den Fall, dass die Übertragungsrate kleiner der Kanalkapazität ist, immer ein Code mit ausreichend großer Blocklänge gefunden werden kann, welcher nach der Decodierung eine beliebig kleine reelle Fehlerwahrscheinlichkeit aufweist [6, S.22]. Der Beweis basiert rein auf wahrscheinlichkeitstheoretischen Überlegungen ohne allgemeine Konstruktionsvorschriften für fehlerkorrigierende Verfahren.

Nachfolgend wird für den speziellen, aber wichtigen Fall eines Additive White Gaussian Noise (AWGN)-Kanals die Kanalkapazität behandelt. Als ein sogenannter AWGN-Kanal wird ein Kanal bezeichnet, bei welchem ein Nutzsignal durch ein Störsignal mit GAUSSverteilter Signalamplitude und konstanter spektraler Rauschleistungsdichte überlagert bzw. gestört wird. Daher setzt sich die Ausgangsgröße y additiv aus der eingangsseitigen Zufallsvariablen x und der GAUSSverteilten Zufallsvariablen n zusammen, wobei n die Varianz σ^2 und den Erwartungswert μ haben soll (siehe Abbildung 2.3 [7, S.67]).

Für einen AWGN-Kanal berechnete SHANNON die Kanalkapazität C in Abhängigkeit von der Signalleistung S, der Bandbreite W und der Rausch-

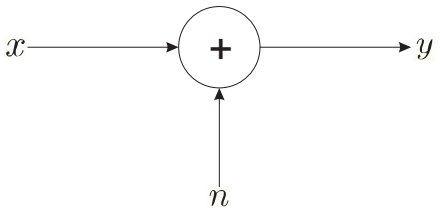

Abbildung 2.3: Modell eines AWGN-Kanals.

leistung N auf folgende weise:

$$C = W \operatorname{ld}(1 + \frac{S}{N}) \tag{2.2}$$

Wird die Rauschleistung durch das Produkt von Bandbreite und spektraler Rauschleistungsdichte N_0 als

$$N = N_0 W \tag{2.3}$$

dargestellt und die Signalleistung durch das Produkt aus Energie pro Bit E_b und Bitrate R_b gesehen, d.h.

$$S = E_b R_b, \tag{2.4}$$

dann bekommt man folgende Gleichung mit der für digitale Modulationsverfahren definierten spektralen Effizienz[1] $\eta_s = R_b/W$:

$$C = W \operatorname{ld}\left(1 + \frac{E_b}{N_0}\eta_s\right) \tag{2.5}$$

Für eine fehlerfreie Übertragung ist nun $R_b \leq C$ gefordert. Dividiert man nun durch die Bandbreite, bekommt man auf der linken Seite der Gleichung ebenfalls die spektrale Effizienz:

$$\eta_s \leq \operatorname{ld}\left(1 + \frac{E_b}{N_0} \cdot \eta_s\right) \tag{2.6}$$

Umgeformt wird dieser Zusammenhang als Shannon-Hartley-Gesetz bezeichnet [7, S.67ff].

$$\frac{E_b}{N_0} \geq \frac{2^{\eta_s} - 1}{\eta_s} \tag{2.7}$$

Würde man einen Kanal mit unendlicher Bandbreite betrachten, so ginge die spektrale Effizienz gegen Null und es kann auf das minimale E_b/N_0 geschlossen werden:

[1] Die spektrale Effizienz stellt ein Maß, für die bei einem bestimmten Modulationsverfahren erreichbaren Bitübertragungsrate, bei gegebener Bandbreite dar.

$$\lim_{\eta_s \to 0} \frac{E_b}{N_0} \geq \lim_{\eta_s \to 0} \frac{2^{\eta_s} - 1}{\eta_s} = \ln(2) = 0,693 \approx -1,6 \text{ dB} \tag{2.8}$$

Ein $E_b/N_0 = -1,6$ dB stellt somit die Grenze dar, ab welcher eine fehlerfreie Übertragung im Prinzip gewährleistet ist. Diese Grenze ist allgemein als SHANNON-Grenze bekannt und wird in Abbildung 2.4 skizziert. Für $\eta_s \ll 1$ befindet man sich im leistungsbegrenzten Bereich für $\eta_s \gg 1$ im bandbreitenbegrenzten Bereich.

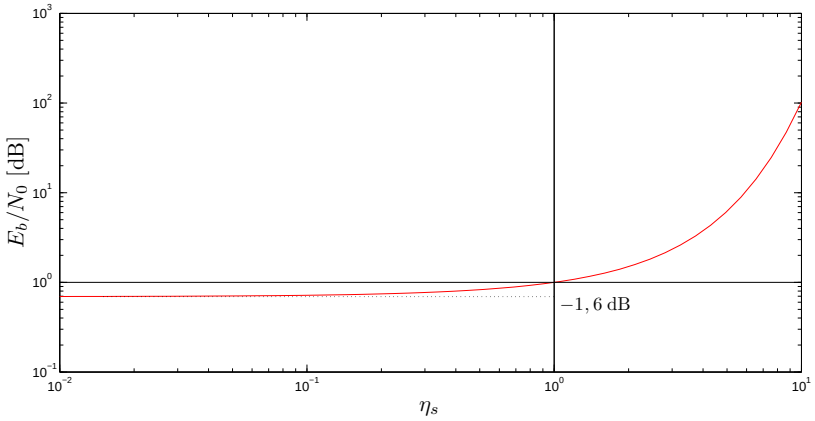

Abbildung 2.4: Verlauf der SHANNON-Grenze.

Kapitel 3

Lineare Blockcodes

Binär-lineare Blockcodes sind solche, die den Gesetzen und Regeln der linearen Algebra gehorchen, wobei der Koeffizientenkörper aus nur zwei Elementen besteht.

3.1 Mathematische Grundlagen

3.1.1 Restklassenarithmetik und Vektorräume

Die einzelnen Symbole eines Daten- oder Codewortes können q verschiedene Werte annehmen. Sämtliche arithmetische Operationen finden über einen endlichen Körper, auch Galois-Feld GF(q) genannt, und werden mit der Hilfe der entsprechenden Restklassenarithmetik ausgeführt. Diese Arbeit beschäftigt sich zur Gänze mit binären Codes d.h. $q = 2$, ein Symbol entspricht somit einer Binärstelle. Die Gesamtheit der eindimensionalen Vektoren über einen Koeffizientenkörper GF(q) spannt einen Vektorraum $\mathcal{V}$ über den Elementen 0 und 1 auf. Ein Unterraum $\mathcal{V}_\mathbf{u} \subseteq \mathcal{V}$ eines Vektorraum $\mathcal{V}_\mathbf{u}$ ist dann gegeben, wenn folgende zwei Bedingungen erfüllt sind [8, S.329].

- Der Nullvektor ist Element von $\mathcal{V}_\mathbf{u}$.
- Die Summe zweier beliebiger Vektoren aus $\mathcal{V}_\mathbf{u}$ ergibt wieder einen Vektor aus $\mathcal{V}_\mathbf{u}$.

Auf diesen Eigenschaften beruht das Prinzip von linearen Blockcodes. Ein Codewort eines Blockcodes besteht insgesamt aus n Bits, wobei $k < n$ Bits als Informationsbits gelten. Aufgrund dessen wird ein solcher Code als *(n,k) Blockcode* bezeichnet, mit dem 2^k verschiedene Codewörter gebildet werden können. Der Codierungsprozess bildet nun jedes Datenwort auf genau ein Codewort ab.

Ziel dieser Codierung ist es, einerseits den Vektorraum mit so vielen Codewörtern wie möglich zu füllen, um so die Redundanz gering zu halten und Bandbreite zu sparen, und andererseits einen möglichst großen Abstand zwischen den Codewörtern zu erreichen, um so eine möglichst große Fehler-

korrigierbarkeit zu erreichen [8, S.330].

Das Prinzip von Blockcodes wird nun anhand des bekannten (7,4) HAMMING-Codes erklärt. Ein (7,4) HAMMING-Code hat eine Blocklänge von $n = 7$, davon sind $k = 4$ Stellen dem uncodierten Datenwort zuzurechnen. Dieser Code besteht nun aus $2^4 = 16$ verschiedenen Codewörtern. In diesem speziellen Fall werden nun den Datenwörtern folgende Codewörter zugeordnet:

Nachrichtenwort	Codewort	Nachrichtenwort	Codewort
0000	0000 000	1000	1000 110
0001	0001 011	1001	1001 101
0010	0010 111	1010	1010 001
0011	0011 100	1011	1011 010
0100	0100 101	1100	1100 011
0101	0101 110	1101	1101 000
0110	0110 010	1110	1110 100
0111	0111 001	1111	1111 111

Tabelle 3.1: Codetabelle eines (7,4) HAMMING-Codes.

Man erkennt, dass das Datenwort in den ersten vier Stellen des Codeworts enthalten ist, d.h. man hat es mit einem *systematischen* Code zu tun.

3.1.2 Erweiterte Paritätskontrolle

Bei einem (7,4) HAMMING-Code sind $m = n - k$ Prüfstellen vorhanden: p_1, p_2, p_3. Diese drei Prüfstellen werden nun nicht mit allen übrigen vier Datenbitstellen d_1, d_2, d_3, d_4 verknüpft, sondern nur mit einzelnen ausgewählten. Der Algorithmus zur Auswahl dieser Stellen für HAMMING-Codes ist in folgenden Paritätsgleichungen festgelegt [9]:

$$p_1 = d_1 \oplus d_2 \oplus d_3 \tag{3.1}$$

$$p_2 = d_1 \oplus d_3 \oplus d_4 \tag{3.2}$$

$$p_3 = d_2 \oplus d_3 \oplus d_4 \tag{3.3}$$

Hier kommt zum Ausdruck, dass die Information mehrfach übertragen wird: direkt durch das Datenwort (*intrinsisch*) und indirekt über die zusätzlichen Informationen aus den Paritätsgleichungen (*extrinsisch*).

3.2 Codierung

Linearität in Bezug auf Blockcodes bedeutet, dass ein Codewort eine Linearkombination aus zwei oder mehrereren Codewörtern darstellt. Man muss sich nur auf einige Basiscodewörter einigen, die den gesamten Coderaum aufspannen. Die logischen Basiscodewörter sind jene, bei welchen jeweils nur ein Einser im Datenwort vorkommt, wie in Tabelle 3.1 gekennzeichnet. Aus diesen Basiscodewörtern setzt sich nun die Generatormatrix $\mathbf{G}$ zusammen, die aus der Einheitsmatrix $\mathbf{E}$ und der Prüfmatrix $\mathbf{P}$ besteht.

$$\mathbf{G} = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right] = [\mathbf{E} \mid \mathbf{P}] \tag{3.4}$$

Die Codierung eines Datenworts erfolgt so durch die Generatormatrix $\mathbf{G}$ über folgende Codierungsvorschrift:

$$\mathbf{c} = \mathbf{u} \odot \mathbf{G} = [\mathbf{u} \mid \mathbf{u} \odot \mathbf{P}] \tag{3.5}$$

3.3 Decodierung

Die Aufgabe des Decoders ist nun zu erkennen, welches Codewort ursprünglich gesendet wurde. Bei binären Blockcodes geschieht die Decodierung durch die sogenannte Paritäts- oder Kontrollmatrix $\mathbf{H}$ welche gegeben ist durch

$$\mathbf{H} = \left[\mathbf{P^T} \mid \mathbf{E}\right] \tag{3.6}$$

oder

$$\mathbf{H^T} = \begin{bmatrix} \mathbf{P} \\ - \\ \mathbf{E} \end{bmatrix} \tag{3.7}$$

Jede Reihe von $\mathbf{H}$ entspricht einer Paritätsgleichung und jede Spalte ist mit einem Bit des Codewortes verknüpft. Die transponierte Kontrollmatrix bildet, multipliziert mit der Generatormatrix, den sogenannten Nullraum:

$$\mathbf{G} \odot \mathbf{H^T} = \mathbf{0} \tag{3.8}$$

Durch diese Eigenschaft wird jedes gültige Codewort, multipliziert mit $\mathbf{H^T}$, als Ergebnis immer den Nullvektor ergeben, welcher für eine fehlerfreie Übertragung steht, d.h.

$$\mathbf{c} \odot \mathbf{H^T} = \mathbf{0} \tag{3.9}$$

Die Zeilen von $\mathbf{H}$ bzw. die Spalten von $\mathbf{H^T}$ sind orthogonal zu allen Codewörtern. Ein ungültiges Codewort erfüllt obige Gleichung nicht und man bekommt einen Vektor $\mathbf{s}$ als Resultat.

$$\mathbf{r} \odot \mathbf{H^T} = \mathbf{s} \neq \mathbf{0} \tag{3.10}$$

Dieser Vektor wird Syndrom genannt. Ein gestörtes Codewort kann als Modulo-2-Addition des Codewortes mit einem Fehlerwort betrachtet werden $\mathbf{r} = \mathbf{c} \oplus \mathbf{e}$. Ein Fehlerwort kennzeichnet die fehlerhafte Bitstelle mit einer Eins in seiner Bitsequenz. Mit dieser Beziehung ergibt sich nun:

$$\mathbf{r} \odot \mathbf{H^T} = (\mathbf{c} \oplus \mathbf{e}) \odot \mathbf{H^T} = \mathbf{e} \odot \mathbf{H^T} = \mathbf{s} \tag{3.11}$$

Wird nun am Empfänger ein *Syndrom* $\mathbf{s} \neq \mathbf{0}$ gebildet, ist während der Übertragung ein Fehler aufgetreten und man kann über die Syndromtabelle Fehlerstellen finden und korrigieren, da das Fehlerwort und das empfangene ungültige Codewort das selbe Syndrom aufweisen.

Fehlerwörter								Syndrome			
$e_1 =$	1	0	0	0	0	0	0	$s_1 =$	1	1	0
$e_2 =$	0	1	0	0	0	0	0	$s_2 =$	1	0	1
$e_3 =$	0	0	1	0	0	0	0	$s_3 =$	1	1	1
$e_4 =$	0	0	0	1	0	0	0	$s_4 =$	0	1	1
$e_5 =$	0	0	0	0	1	0	0	$s_5 =$	1	0	0
$e_6 =$	0	0	0	0	0	1	0	$s_6 =$	0	1	0
$e_7 =$	0	0	0	0	0	0	1	$s_7 =$	0	0	1

Tabelle 3.2: Fehlerwörter mit zugehörigen Syndromen.

Wird bei der Übertragung über den Kanal ein Codewort so gestört, dass es im Decodierer auf ein anderes gültiges Codewort zurück transformiert wird, so kann dies nicht erkannt werden und man spricht in diesem Fall von einem *Decodierversagen*. Für die Syndroma lassen sich somit folgende Fälle erkennen:

- Fall 1: $\mathbf{s} = \mathbf{0} \leftrightarrow \mathbf{e} \in \mathcal{C}$
  - für $\mathbf{e} = \mathbf{0}$ fehlerfreie Übertragung
  - für $\mathbf{e} \neq \mathbf{0}$ Decodierversagen, nicht erkennbar!
- Fall 2: $\mathbf{s} \neq \mathbf{0} \leftrightarrow \mathbf{e} \notin \mathcal{C}$ Fehler wird detektiert

3.4 Zyklische Codes

Ein Code ist zyklisch, wenn eine zyklische Vertauschung des Codewortes um x Stellen ein gültiges Codewort ergibt. Der große technische Nutzen entfaltet sich dadurch, dass sich zyklische Codes durch einfache rückgekoppelte Schieberegisterschaltungen codieren und decodieren lassen. Bose Chaudhuri Hocquenghem (BCH)-Codes etwa sind mehrfehlerkorrigierende zyklische Codes. Im Digital Video Broadcasting via Satellit Standard der zweiten Generation (DVB-S2) etwa sind sie Teil eines verketteten Systems (äußeres Schema: BCH-Codes, inneres Schema: LDPC-Codes).

3.5 Eigenschaften von Blockcodes

3.5.1 Coderate

Bei der Codierung wird ein k Bit langes Datenwort auf ein n Bit langes Codewort abgebildet. Die zusätzlichen m Bits stellen die Absicherung gegen Fehler während der Übertragung auf dem Kanal dar (Redundanz). Das Verhältnis wird als Coderate R_c bezeichnet:

$$R_c = \frac{k}{n} = \frac{k}{m+k} \tag{3.12}$$

Durch Hinzufügen von Bits im Codewort steigt die Redundanz, zugleich aber auch der benötigte Übertragungsaufwand, da z. B. bei einer Coderate von 1/4 nur 25% Nutzdaten im Codewort enthalten sind.

3.5.2 Hamming-Distanz und Korrekturvermögen

Das Hamming-Gewicht eines Codes ist gleich der Anzahl der im Codewort von Null verschiedenen Stellen. Die Hamming-Distanz ist definiert durch die Anzahl der unterschiedlich belegten Binärstellen zweier beliebiger Codewörter eines Codes. Ein Maß für die Fehlertoleranz ist die Mindest-Hamming-Distanz d_{min}. Die Anzahl der korrigierbaren Fehler bei einer Mindest-Hamming-Distanz kann mit folgender Formel bestimmt werden:

$$t = \lfloor \frac{d_{min} - 1}{2} \rfloor \tag{3.13}$$

Als Verständnishilfe für die Wichtigkeit der Hamming-Distanz bei der Fehlerkorrektur dient Abbildung 3.1 [5, S.149]. Ein gesendetes Codewort $\mathbf{v}$ kann durch eine Störung verfälscht werden. Dabei können drei Fälle auftreten [5, S.148]:

1. Die Störung wird durch $\mathbf{e_1}$ beschrieben. Zusammen mit dem ursprünglichen Codewort $\mathbf{v_1}$ ergibt sich der Empfangsvektor $\mathbf{r_1}$. Dieses befindet sich noch innerhalb der Hamming-Sphäre und kann somit erkannt und korrigiert werden.

2. Der Empfangsvektor liegt innerhalb der HAMMING-Sphäre von $\mathbf{v_2}$. Somit ist der Fehler erkennbar, aber nicht mehr korrigierbar.
3. Hier wird das Codewort $\mathbf{v_1}$ durch $\mathbf{e_3}$ zum Codewort $\mathbf{v_2}$. Dieser Fall wird als Decodierversagen bezeichnet.

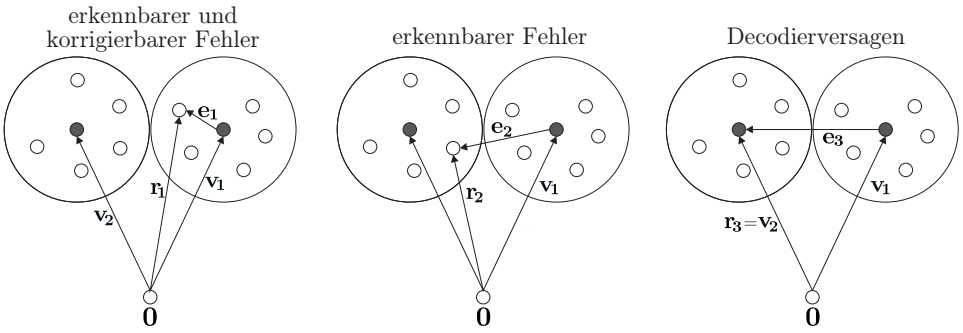

Abbildung 3.1: Vektorraum mit Codewörtern und möglichen Empfangswörtern.

Die Mindest-HAMMING-Distanz eines Codes wird maximiert, indem man die Ähnlichkeiten der Spalten in der Kontrollmatrix minimiert [10, S.16].

3.5.3 HAMMING-Schranke

Es stellt sich bei der Codekonstruktion die Frage, wie viele Paritätsbits m benötigt werden, um einen t-Fehler korrigierbaren Code konstruieren zu können. Die Antwort liefert die sogenannte HAMMING-Schranke [11]:

$$1 + \underbrace{n}_{1-Fehler} + \underbrace{\binom{n}{2}}_{2-Fehler} + \ldots + \underbrace{\binom{n}{t}}_{t-Fehler} = \sum_{i=0}^{t} \binom{n}{i} \tag{3.14}$$

Möglichkeiten, wie sich i Fehler auf n Bitstellen verteilen lassen. Mit 2^k Datenwörtern bekommt man

$$2^k \sum_{i=0}^{t} \binom{n}{i} \tag{3.15}$$

verschiedene Varianten für Fehlerstellen, die in Summe nicht größer sein können als der Raum, der mit n Binärstellen aufgespannt werden kann. Also gilt:

$$2^k \sum_{i=0}^{t} \binom{n}{i} \leq 2^n \tag{3.16}$$

Ist diese Beziehung erfüllt, so spricht man von perfekten oder dichtgepackten Codes. Die HAMMING-Schranke wird deswegen auch als Kugelpackungsgrenze bezeichnet.

3.5.4 Modifikationen linearer Codes

Mit Hilfe einfacher Modifikationen ist es möglich, einen Blockcode an spezielle Anwendungen anzupassen:

- Expandieren: Zusätzliche Prüfstellen werden hinzugefügt.
- Punktieren: Einige Prüfstellen werden entfernt.
- Verlängern: Zusätzliche Informationsstellen werden hinzugefügt.
- Verkürzen: Einige Informationsstellen werden entfernt.

3.6 Leistungsfähigkeit eines Codes

Die Leistungsfähigkeit eines Nachrichtensystems wird anhand seiner Bit Error Rate (BER)-Kurve im Vergleich zum uncodierten Fall bestimmt (Codierungsgewinn, siehe Abbildung 3.2 [12, S.159]).

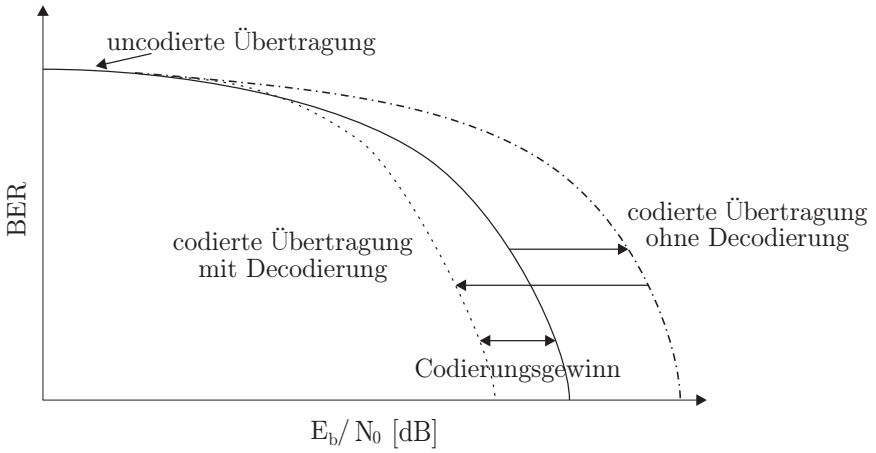

Abbildung 3.2: Prinzip des Codiergewinns.

In diesem Kontext ist der Zusammenhang von Signal-Rauschleistungs-Verhältnis (Signal-to-Noise Ratio, SNR), E_s/N_0 und E_b/N_0 von Bedeutung:

$$SNR = \frac{E_s}{N_0} = m\, R_c\, \frac{E_b}{N_0} \tag{3.17}$$

wobei $E_s = m\, E_b$ die Energie pro Symbol und m die Anzahl der Bits pro Symbol (durch das gewählte Modulationsverfahren gegebenen) bezeichnet.

Kapitel 4

LDPC-Codes

Publiziert wurden Low-Density Parity-Check (LDPC)-Codes das erste Mal von Gallager [3] in den 1960er Jahren. Aufgrund ihrer Komplexität, die auf damaligen Hardwarearchitekturen nicht realisierbar war, blieben LDPC-Codes bis zu ihrer Wiederentdeckung durch MacKay und Neal [13, 14] Mitte der 1990er vergessen. Als eine der Ausnahmen ist jedoch Tanner zu nennen, welcher mit seiner Arbeit [4] den später nach ihm benannten Tanner-Graphen (TG) zur Untersuchung und Design von FEC-Verfahren einführte. Im Folgenden sollen die Begriffe *Low-Density* und *Parity-Check*, welche die LDPC-Codes charakterisieren, genauer erläutert werden.

Low-Density: Die Einsen in den Zeilen der Kontrollmatrix legen fest, welche Bits der jeweiligen Zeile an der Paritätsgleichung beteiligt sind. Bei LDPC-Codes ist charakteristisch, dass in der Kontrollmatrix nur sehr wenige Einsen, im Vergleich zur Dimension der Matrix, vorhanden sind. Sie ist also dünn besetzt (*low density*), wobei der Term *dünn* ziemlich vage ist. Eine solche dünn besetzte Matrix verringert deutlich den benötigten Rechenaufwand beim Decodieren.

Parity-Check: Wie bei allen Blockcodes werden auch bei LDPC-Codes zusätzliche Bits den informationstragenden Bits angefügt. Diese zusätzlichen Bits (Paritätsbits) bilden zusammen mit den Informationsbits Paritätsgleichungen, die bei einer fehlerlosen Übertragung Null ergeben müssen (*parity check*).

LDPC-Codes sind im Gegensatz zu herkömmlichen Blockcodes wie Hamming-Codes, welche nach bestimmten Regeln konstruiert werden und eine bekannte Fehlerkorrektur aufweisen, meist pseudo-zufälliger Natur, sodass sich ihre Fähigkeit zur Fehlerkorrektur nur durch Wahrscheinlichkeiten ausdrücken lässt. Dadurch kommt ein weiterer Unterschied zu normalen Blockcodes zu Tage, nämlich jener der unterschiedlichen Decodierung. Während

klassische Blockcodes in einem Durchlauf decodiert werden, kommen bei LDPC-Codes iterative Verfahren zum Einsatz [15, S.34]. Ein weiterer Unterschied besteht darin, dass LDPC Codes durch ihre Kontrollmatrix charakterisiert sind und nicht, wie bei Blockcodes üblich, durch ihre Generatormatrix. Grundsätzlich unterscheidet man zwei Archetypen von LDPC Codes, *reguläre Codes* und *irreguläre Codes*.

4.1 Tanner-Graphen

Zusätzlich zur Beschreibung von Codes über Matrizen gibt es noch eine graphische Repräsentationsform über sogenannte *bipartite Graphen*. Da bei LDPC-Codes nur sehr wenige Bits beteiligt sind, ergibt sich auf diese Weise eine einfache und elegante Darstellung. Diese Form der Beschreibung geht auf Tanner zurück und man bezeichnet deshalb diese Art von Graphen in der Codierungstheorie als einen TANNER-Graph (TG) [4]. Bipartite Graphen eignen sich hervorragend, um Beziehungen zwischen den Elementen zweier Mengen zu kennzeichnen. Dabei sind die Elemente der zwei Mengen in unterschiedliche Knotenklassen einzuteilen, wobei Verbindungen (Kanten) nur zwischen Knoten aus unterschiedlichen Klassen vorkommen können. Diese zwei Klassen von Knoten werden in einem TG als Bitknoten (variable nodes; stellvertretend für die übertragenden Bits) und Gleichungs- oder Kontrollknoten (check nodes; stellvertretend für Bits, welche die Paritätsgleichungen bestimmen), bezeichnet. Von einem Kontrollknoten führen Kanten, entsprechen der zugehörigen Prüfgleichung, zu allen Bitknoten. Es gibt also n Bitknoten und m Kontrollknoten im TG, wobei n die Blocklänge und $m = n - k$ die Anzahl der Paritätsbits ist.

Bitknoten werden in Anlehnung an [15] mit Rechtecken und Kontrollknoten mit Kreisen gekennzeichnet. Zur anschaulichen Erklärung dient die Kontrollmatrix aus (4.1), die als TG in Abbildung 4.1 zu sehen ist.

$$\mathbf{H} = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{array} \begin{array}{c} \begin{array}{cccccc} n_1 & n_2 & n_3 & n_4 & n_5 & n_6 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \end{array} \tag{4.1}$$

Aus dieser Kontrollmatrix lassen sich nun folgende Paritätsgleichungen ableiten:

$$\begin{aligned} c_1 &= n_1 \oplus n_2 \oplus n_4 \\ c_2 &= n_2 \oplus n_3 \oplus n_5 \\ c_3 &= n_1 \oplus n_5 \oplus n_6 \\ c_4 &= n_3 \oplus n_4 \oplus n_6 \end{aligned} \tag{4.2}$$

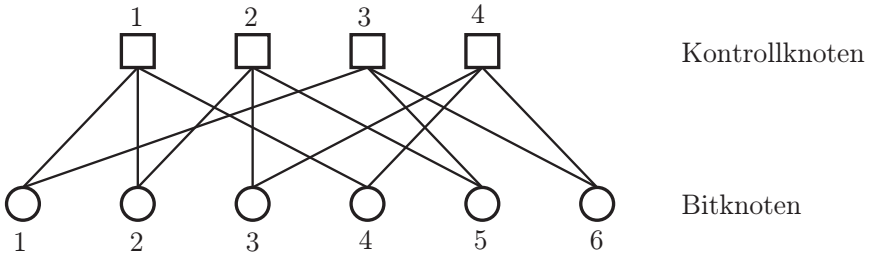

Abbildung 4.1: TG des Codes aus (4.1).

Jeder Bitknoten repräsentiert also eine Bitstelle im Codewort, während jeder Kontrollknoten für eine Paritätsgleichung steht.

Die Verbindung eines Knotens im TG mit einem beliebigen anderen Knoten wird *Kante* genannt. Mehrere Kanten können einen Pfad in einem TG bilden, welcher auf sich zurückführt. Solche Knoten- bzw. Kantenfolgen, die wieder zum Ausgangspunkt zurückführen, werden als Zyklus bezeichnet. Die Länge des kürzesten Zyklus im TG ist der Umfang (*girth*). Bedingt durch den Aufbau des TG ist der Umfang immer gerade und mindestens gleich vier. Generell aber gilt, dass sich kurze Zyklen negativ auf den iterativen Decodierungsprozess auswirken. Man ist daher bemüht kurze Zyklen, im speziellen jene der Länge vier, zu vermeiden. In der Kontrollmatrix ist ein Zyklus der Länge vier durch vier Einsen, welche in den Eckpunkten einer Submatrix liegen, erkennbar:

$$\mathbf{H} = \begin{bmatrix} \dots & & \dots & & \dots \\ & 1 & & 1 & \\ \dots & & \dots & & \dots \\ & 1 & & 1 & \\ \dots & & \dots & & \dots \end{bmatrix} \tag{4.3}$$

Ein Zyklus der Länge vier kann also durch zwei sich überlappende mit Einsen besetzte Einträge in einer Zeile oder Spalte identifiziert werden. Die Nebenbedingung, dass sich keine Zyklen der Länge vier im TG befinden dürfen, wird dementsprechend *row-colum* Nebenbedingung genannt [16, S.243].

4.2 Reguläre LDPC-Codes

Ein regulärer LDPC-Code ist über die Kontrollmatrix **H** definiert, die folgenden Eigenschaften aufweist:

1. Jede Zeile besteht aus ρ Einsen.
2. Jede Spalte besteht aus γ Einsen.
3. Das Vektorprodukt λ zweier beliebiger Spalten ist nicht größer als 1.
4. Es gilt $\rho \ll n$ und $\gamma \ll m$.

Reguläre LDPC-Codes sind schon von GALLAGER beschrieben worden [3]. Ihre Kontrollmatrix besteht in jeder Spalte aus γ Einsen und in jeder Reihe aus exakt $\rho = \gamma \frac{n}{m}$ Einsen [17, S.2]. Demnach gilt:

$$\gamma n = \rho m \tag{4.4}$$

Reguläre Codes weisen demnach eine Dichte von

$$r = \frac{\rho}{n} = \frac{\gamma}{m} \tag{4.5}$$

Einsen auf. Die Coderate ergibt sich daraus wie folgt:

$$R_c = \frac{k}{n} = 1 - \frac{\gamma}{\rho} \tag{4.6}$$

Ein (γ, ρ)-regulärer Code erreicht, wie in Abbildung 4.2 [18, S.58] ersichtlich, mit größer werdender Blocklänge eine gewisse Schwelle (Threshold) die allerdings nicht der SHANNON-Grenze entspricht.

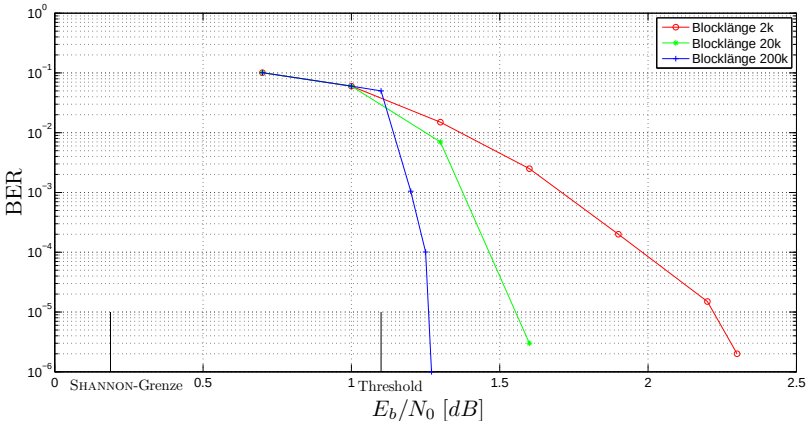

Abbildung 4.2: Vergleich von (3,6)-regulären LDPC Codes mit $R_c = 1/2$ und verschiedener Blocklängen auf einem BPSK AWGN-Kanal.

Dieser Umstand führt uns direkt zu irregulären LDPC-Codes, mit welchen man viel näher an die SHANNON-Grenze herankommt.

4.3 Irreguläre LDPC-Codes

Wenn die Anzahl der Einsen in den Reihen oder Spalten nicht konstant ist, spricht man von einem irregulären LDPC-Code. Dieser Ansatz geht auf [19] und in weiterer Folge auf [20, 21] zurück, wo gezeigt wurde, dass die Fehlerkorrektur von sogenannten Gradverteilungen abhängig ist. Da die beiden Parameter γ und ρ variieren können, ist es sinnvoll, den Grad eines Knoten

durch die mit ihm verbundenen Kanten darzustellen. Generell gilt, dass Bitknoten *hochgradig* sein sollten, da mehr Information ein genaueres Ergebnis bedeutet. Kontrollknoten hingegen sollten *niedergradig* sein, da sich falsche Informationen sonst zu sehr verbreiten. Zugleich ist aber auch darauf zu achten, dass die Dichte der Kontrollmatrix nicht zu hoch wird.

4.4 Codierung

Wie Blockcodes über die Generatormatrix $\mathbf{G}$ codiert werden ist in Kapitel 3 erklärt. Ein Nachteil dieser Vorgangsweise ist jedoch, dass LDPC-Codes üblicherweise durch ihre Kontrollmatrix und nicht über die Generatormatrix definiert werden. Mit Hilfe des Gauss-Jordan-Algorithmus ist es jedoch möglich, eine Kontrollmatrix, welche in der Form

$$\mathbf{H} = [\mathbf{P^T} \mid \mathbf{E}] \tag{4.7}$$

vorliegt, in eine Generatormatrix der Form

$$\mathbf{G} = [\mathbf{E} \mid \mathbf{P}] \tag{4.8}$$

umzuwandeln. Durch diese Transformation mit der Komplexität $O(n^3)$, welche als Vorverarbeitungsschritt zur eigentlichen Codierung zu sehen wäre, bekommt man im Allgemeinen eine nicht mehr dünn besetzte Generatormatrix. Oder mit anderen Worten: Die Matrixmultiplikation $\mathbf{c} = \mathbf{u} \odot \mathbf{G}$ während der Codierung hätte die Komplexität $O(n^2)$, was in Anbetracht von Codewortlängen mit einigen zehntausend bis hunderttausend Bits einen erheblichen Rechenaufwand ausmacht.

Annähernd lineare Codierung Ein anderer Weg zur Codierung von LDPC-Codes wird in [22] beschrieben. Die Idee dahinter besteht darin, die Kontrollmatrix durch einfache Zeilen- und Spalten-Vertauschungen annähernd auf eine untere Dreiecksform zu transformieren und dabei die Kontrollmatrix so dünn wie möglich zu belassen. Die Erklärung der Vorgehensweise folgt [15, S.44ff], wo auch ein Beispiel zu finden ist. Unter einer ungefähren unteren Dreiecksmatrix versteht man eine Matrix mit der Form:

$$\mathbf{H_t} = \begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{T} \\ \mathbf{C} & \mathbf{D} & \mathbf{Z} \end{bmatrix} \tag{4.9}$$

wobei die Matrix $\mathbf{T}$ eine untere Dreiecksmatrix der Dimension $(m-g) \times (m-g)$ ist (siehe Abbildung 4.3 [22, S.641]). Falls $\mathbf{H_t}$ vollen Rang besitzt, ergibt sich die Matrix $\mathbf{B}$ mit Dimension $(m-g) \times g$ und die Matrix $\mathbf{A}$ mit $(m-g) \times k$. Die g Reihen der ursprünglichen $\mathbf{H}$ Matrix, welche durch die Matrizen $\mathbf{C}$, $\mathbf{D}$ und $\mathbf{Z}$ gebildet werden, nennt man Lücke (gap), wobei g den Abstand der Kontrollmatrix zu einer echten unteren Dreiecksmatrix

angibt. Je kleiner diese Lücke, desto geringer ist der Rechenaufwand für den Codierungsprozess [22, S.641].

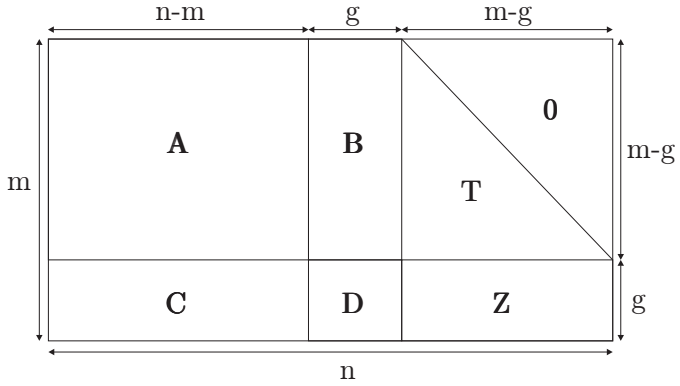

Abbildung 4.3: Kontrollmatrix in annähernder unterer Dreiecksform.

Wurde $\mathbf{H}$ in die untere Dreiecksmatrix gebracht, muss nun die Matrix $\mathbf{Z}$ durch den Gauss-Jordan-Algorithmus in eine Nullmatrix übergeführt werden. Da es sich bei der Matrix $\mathbf{T}$ um eine untere Dreiecksmatrix handelt, werden durch den Gauss-Jordan-Algorithmus nur die Matrizen $\mathbf{C}$ und $\mathbf{D}$ beeinflusst und zu $\widetilde{\mathbf{C}}$ und $\widetilde{\mathbf{D}}$ verändert. Um nun eine Nachricht zu codieren ist es nötig, die Nachricht in drei Teile $\mathbf{c} = [\mathbf{u}, \mathbf{p^{(1)}}, \mathbf{p^{(2)}}]$ aufzuspalten: $\mathbf{u} = [u_1, u_2, \dots u_k]$ stellt die zu codierende Nachricht dar, $\mathbf{p^{(1)}} = [p_1^{(1)}, p_2^{(1)}, \dots p_g^{(1)}]$ die ersten g Paritätsbits und $\mathbf{p^{(2)}} = [p_1^{(2)}, p_2^{(2)}, \dots p_g^{(2)}]$ die restlichen Paritätsbits. Das Codewort muss somit der Paritätsgleichungen $\mathbf{c} \odot \widetilde{\mathbf{H}}^{\mathbf{T}}$ entsprechen. Diese Paritätsgleichungen aufgelöst ergeben somit:

$$\mathbf{u} \odot \mathbf{A} + \mathbf{p^{(1)}} \odot \mathbf{B} + \mathbf{p^{(2)}} \odot \mathbf{T} = \mathbf{0} \tag{4.10}$$

$$\mathbf{u} \odot \widetilde{\mathbf{C}} + \mathbf{p^{(1)}} \odot \widetilde{\mathbf{D}} + \mathbf{p^{(2)}} \odot \mathbf{0} = \mathbf{0} \tag{4.11}$$

Da nun $\mathbf{Z}$ eine Nullmatrix ist, kann unabhängig $p^{(2)}$ berechnet werden.

$$\mathbf{p^{(1)}} = \mathbf{u} \odot \widetilde{\mathbf{D}}^{-\mathbf{1}} \widetilde{\mathbf{C}} \tag{4.12}$$

Von $\mathbf{p^{(1)}}$ lässt sich direkt auf $\mathbf{p^{(2)}}$ schließen:

$$\mathbf{p^{(2)}} = (\mathbf{u} \odot \mathbf{A} + \mathbf{p^{(1)}} \odot \mathbf{B}) \odot -\mathbf{T}^{-\mathbf{1}} \tag{4.13}$$

Mit dieser Variante ist es möglich eine obere Grenze für die Komplexität mit $O(n + g^2)$ anzugeben [22].

Es gibt zwei Möglichkeiten, wie man den Aufwand der Codierung weiter senken kann:

1. Schieberegisterschaltungen für zyklische oder quasi-zyklische Codes [23, S.858ff].

2. Einsatz von extendend Irregular Repeat-Accumulate (eIRA)-Codes [17].

4.5 Leistungsfähigkeit von LDPC-Codes

Unter der Leistungsfähigkeit eines Codes versteht man den BER-Verlauf als Funktion des Signalrauschabstandes (Signal-to-Noise Ratio, SNR). In Abbildung 4.4 [10, S.16] ist das für einen LDPC-Code zu sehen: in einen engen SNR-Bereich (*Wasserfall-Region*) sinkt die BER rasch und geht dann in eine flach verlaufende Fehlerhorizontale (*error floor*) über.

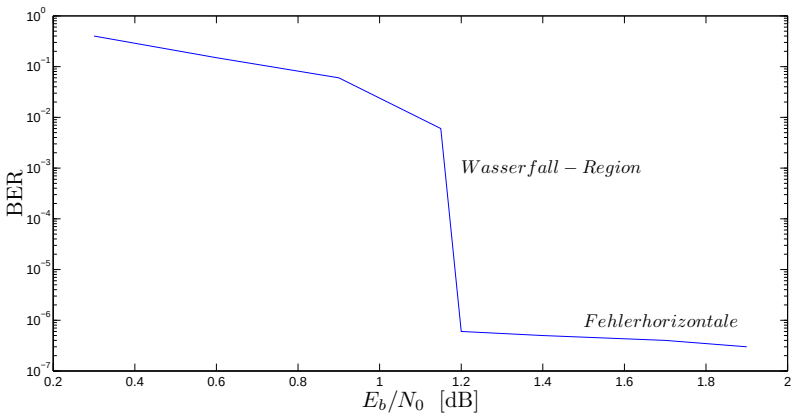

Abbildung 4.4: Charakteristische BER für iterativ decodierte LDPC-Codes.

Im Gegensatz zu Turbo-Codes ist dieses Phänomen nicht ausschließlich Codewörtern mit geringem Gewicht anzulasten [24, S.1]. Ursache für ein solches Verhalten liegt bei Bits, welche nur an eine Paritätsgleichung gebunden sind, sodass diese eine erhöhte Wahrscheinlichkeit haben nicht korrigiert zu werden [10, S.16]. Ein weiterer Grund sind Schwachstellen der iterativen Algorithmen [17, S.4], sowie die Anwesenheit von vielen Codewörtern mit zu geringem Gewicht [25, S.1479].

4.6 Vergleich mit Turbo-Codes

Turbo- sowie LDPC-Codes haben gemeinsam, dass sie sehr nahe an die Shannon-Grenze herankommen. Beide Codes sind im Aufbau unterschiedlich und weisen daher verschiedene Eigenschaften auf. Turbo-Codes haben seit ihrer Publikation im Jahr 1993 den Weg in zahlreiche Anwendungen gefunden. Als die zwei größten Nachteile von Turbo-Codes sind jedoch die patentrechtliche Lage (und die damit verbundenen Kosten) und das frühzei-

tige Auftreten von Fehlerhorizontalen bei Anwendungen mit sehr niedrigen BER-Anforderungen zu nennen.

Vorteile von LDPC-Codes

- Keine patentrechtlichen Probleme.
- Es gibt zahlreiche Decodervarianten mit verminderter Komplexität.
- Ein LDPC-Decoder kann theoretisch soweit parallelisiert werden, dass jede Paritätsgleichung für sich betrachtet werden kann (benötigt aber eine größere Chipfläche).
- Reguläre LDPC-Codes sind im Vergleich zu ähnlichen Turbo-Codes etwas schlechter, irreguläre hingegen sind leistungsfähiger als Turbo-Codes.
- Der korrekte Abschluss des Decodierprozesses ist leicht detektierbar.

Als weitere Vorteile gegenüber Turbo-Codes sind laut [23, S.851] zu nennen:

- Sie brauchen keinen langen *Interleaver*, um ein gutes Fehlerkorrekturvermögen zu erreichen.
- Sie besitzen ein besseres Korrekturverhalten gegenüber Bündelfehler.
- Zur Decodierung ist kein *Trellis* notwendig.
- Ihre Fehlerhorizontalen treten bei viel kleineren BER-Werten auf.

Nachteile von LDPC-Codes

- Der Codierungsprozess kann unter Umständen sehr komplex sein.
- Gute irreguläre LDPC-Codes zu entwerfen ist eine schwierige Aufgabe.
- Bei kleinen bis mittleren Blocklängen besitzen LDPC-Codes nicht ganz das Leistungsvermögen von Turbo-Codes.

Kapitel 5

Iterative Decodierung von LDPC-Codes

LDPC-Codes zählen neben den *Turbo-* oder *Produkt-Turbo-Codes* zu den iterativ decodierbaren Verfahren. Das Ziel hierbei ist die näherungsweise Verwirklichung einer echten Maximum-Likelihood (ML)-Decodierung durch mehrere Iterationen. Es werden verschiedene Decodierungsverfahren unterschieden, welche folgend nach ihrer Komplexität und ihrem Fehlerkorrekturvermögen aufsteigend gereiht aufgelistet werden [23, S.871]:

1. Majority-Logic (MLG) Decodierung
2. Bit-Flipping (BF) Decodierung
3. Weighted Bit-Flipping (WBF) Decodierung
4. Sum-Product Alogrithm (SPA) Decodierung
5. Maximum-a-Posteriori (MAP) Decodierung

MLG und BF sind Algorithmen, die Hard Decisions verwenden, während SPA und MAP auf sogenannten Soft Decisions beruhen. Eine hybride Form stellt der WBF dar. Empfängerseitig stehen nach dem Demodulationsprozess am Ausgang des signalangepassten (matched) Filters die entsprechenden Abtastwerte (Samples) zur Verfügung. Arbeitet der Decoder mit diesen wert-kontinuierlichen-Zahlen, so wird er als Soft-Decision Decoder bezeichnet. Arbeitet dagegen der Decoder mit wert-diskreten Symbolen (Bits), die ihrerseits aus den Samples abgeleitet werden, dann spricht man von einem Hard-Decision Decoder.

Soft-Decision Decodierung (SDD) benötigt naturgemäß einen höheren Rechenaufwand als Hard-Decision Decodierung (HDD), was einerseits an den Rechenoperationen im reellen Raum und andererseits an der Einbeziehung von *a posteriori* Wahrscheinlichkeiten liegt. Durch den erhöhten Rechenaufwand kann aber ein Codiergewinn von $2 - 3$ dB erzielt werden [26, S.18].

LDPC-Codes, welche über euklidische Geometrie (EG) und projektive Geometrie (PG) konstruiert werden, stellen eine Untergruppe von MLG-Codes dar. Dieses nicht iterative Decodierungsverfahren beruht auf der Eigenschaft, dass orthogonale Paritätprüfungen für jede fehlerhafte Bitstelle geformt werden können. Dies ist für spezielle Hardwareanwendung von Interesse, da der jeweilige Rechenaufwand sehr gering ist.

5.1 Decodierung über einen TANNER-Graph

Ein weiterer Vorteil der Darstellungsweise von LDPC-Codes in Form von TG ist die einfache Veranschaulichung des iterativen Decodierverfahrens. Zu Beginn der Decodierung werden die Bitknoten mit der vom Kanal gewonnenen *intrinsischen* a-posteriori *Information* initialisiert. In den Kontrollknoten wird danach für jeden Bitknoten eine *extrinsische Information* berechnet. Die so gewonnenen *extrinsischen* Wahrscheinlichkeiten werden nun mit den in den Bitknoten vorhandenen *intrinsischen* Wahrscheinlichkeiten zusammengefasst und ergeben so nach der ersten Iteration die neuen *intrinsischen* Wahrscheinlichkeit des zu bestimmenden Zustandes. Weitere Iterationen werden folglich nach dem selben Prinzip abgearbeitet. Der Algorithmus kommt zu einer optimalen Decodierung im Sinne einer MAP- bzw. ML-Decodierung, solange es keine Schleifen im Graphen gibt. Betrachtet man einen schleifenfreien TG, so kann dieser wie in Abbildung 5.1 [27, S.122] als zwei an ihren Wurzeln zusammenhängende Baumdiagramme dargestellt werden.

Eine Schleife im TG bewirkt, dass Wahrscheinlichkeiten, welche von einem bestimmten Bitknoten ausgehen, früher oder später zu diesem zurückkehren und so zu einer Mitkopplung führen. Eine statistische Unabhängigkeit zwischen den Bitknoten ist somit nicht mehr gegeben. Nichtsdestotrotz ist die Decodierleistung im TG mit Schleifen überraschend gut. Codes, welche auf Graphen ohne Schleifen beruhen, besitzen leider keine guten Distanzeigenschaften und so ist man bemüht TGen zu finden, bei welchen die kürzeste geschlossene Schleife (Umfang) maximiert wird. Dies führt direkt zu dünn besetzten Kontrollmatrizen von LDPC-Codes, welche die Wahrscheinlichkeit, dass die selben Bits in mehreren Paritätsgleichungen vorkommen, minimiert [28, S.20f].

Die Anwesenheit von Zyklen im Graphen führt zu einem Kreisen von Wahrscheinlichkeiten im Graphen, was bedeutet, dass es keinen natürlichen Haltemechanismus für diesen Algorithmus gibt. Demnach sind Stopp-Kriterien zu wählen, um den Algorithmus zu beenden, entweder indem ein gültiges Codeword erkannt wird oder indem eine vorgegebene Anzahl von Iterationen durchlaufen wurde. Letztere Abbruchbedingung spart vor allem Verarbeitungszeit und ist dann von Vorteil, wenn das empfangene Codewort zu stark gestört wurde, als dass es noch rekonstruierbar wäre. Was im

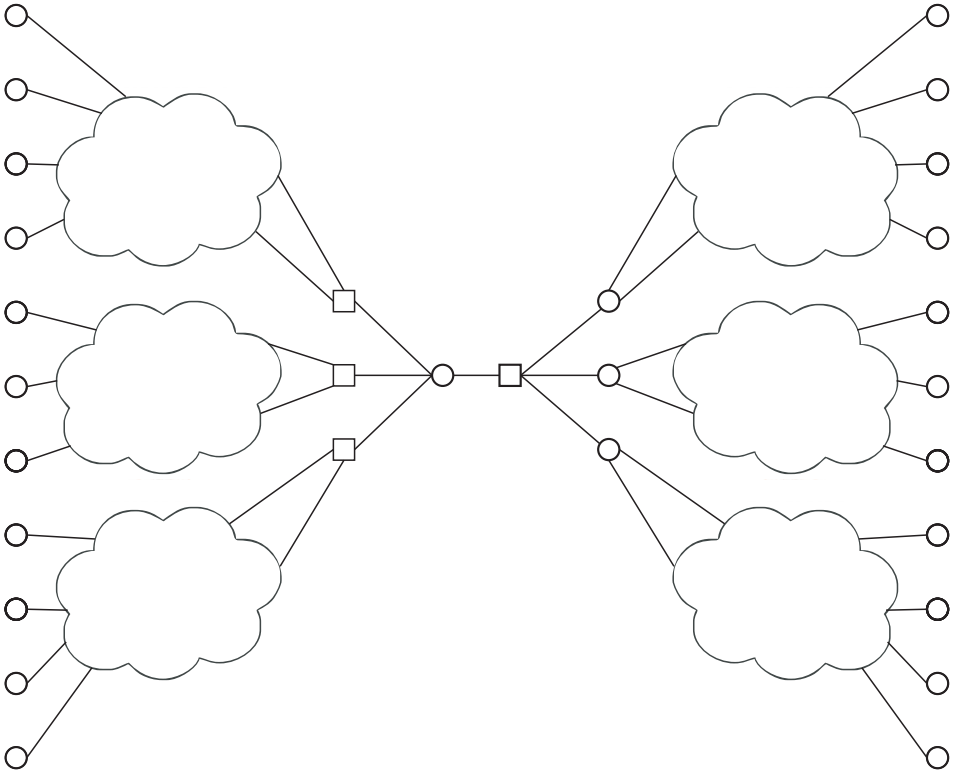

Abbildung 5.1: Baumstruktur eines in zwei Subgraphen aufgeteilten zyklusfreien TG. Es ist gut zu erkennen, dass es nur an den Wurzeln zu einem Informationsaustausch kommt.

Falle eines Decodierversagens geschieht, ist von der jeweiligen Anwendung abhängig. Die Verarbeitung von probabilistischen Zusammenhängen in Graphen oder graphenähnlichen Strukturen ist auch als *Belief Propagation* in der Fachliteratur bekannt. [29, S.43ff].

5.2 Sum-Product Algorithmus

Der SPA geht auf Gallager [3] zurück. Folgend wird jedoch die leichter zu berechnende Variante von MacKay und Neal [2] vorgestellt. Beide Varianten sind jedoch als äquivalent anzusehen [30, S.301]. Als Eingangswerte dienen dem Decoder die reellen Ausgangswerte des Kanals.

Initialisierung Im ersten Schritt werden die Wahrscheinlichkeiten der Werte $y_1, y_2, \ldots, y_n$ vom AWGN-Kanal kommend berechnet. Je nachdem, ob es sich um Einsen oder Nullen handelt, erhält man für $j = 1, 2, \ldots, n$:

$$f_j^1 = \frac{1}{1 + e^{\frac{-2y_j}{\sigma^2}}} \tag{5.1}$$

$$f_j^0 = 1 - f_j^1 \tag{5.2}$$

Dabei ist σ^2 die Varianz des thermischen Rauschens, dass den AWGN-Kanal beschreibt. Mit diesen Wahrscheinlichkeiten werden nun mehrere Variablen initialisiert, wobei $i = 1, 2, \ldots, m$:

$$q_{i,j}^1 = f_j^1 \tag{5.3}$$

$$q_{i,j}^0 = f_j^0 \tag{5.4}$$

$$\delta q_{i,j} = q_{i,j}^0 - q_{i,j}^1 \tag{5.5}$$

Der weitere Ablauf des Algorithmus lässt sich als wechselseitig iterative Abarbeitung eines *horizontalen* und *vertikalen* Schrittes beschreiben.

Horizontaler Schritt Dabei wird für jede Paritätsgleichung das Produkt aller mit ihm über Kanten verbundenen Wahrscheinlichkeiten $\delta q_{i,j}$ berechnet, mit Ausnahme der aktuell betrachteten Spalte. Die Menge B_i bezeichnet demnach alle Bits in der $j$ten Paritätsgleichung, also alle mit Einsen besetzten Stellen in der Kontrollmatrix. $B_{i\backslash j}$ stellt die durch den aktuellen Index bereinigte Menge dar.

$$\delta r_{i,j} = \prod_{j' \in B_{i\backslash j}} \delta q_{i,j'} \tag{5.6}$$

$$r_{i,j}^0 = (1 + \delta r_{i,j})/2 \tag{5.7}$$

$$r_{i,j}^1 = (1 - \delta r_{i,j})/2 \tag{5.8}$$

Vertikaler Schritt Die Variable $\alpha_{i,j}$ kennzeichnet eine Ausgleichsvariable, mit welcher die Gleichung $q_{i,j}^1 \alpha_{i,j} + q_{i,j}^0 \alpha_{i,j} = 1$ erfüllt werden soll. Die Menge $A_{j\backslash i}$ bezeichnet analog zu $B_{i\backslash j}$ die Menge aller, mit dem aktuell betrachtet Bit, verknüpften Paritätsgleichungen, mit Ausnahme des derzeitigen Index.

$$q_{i,j}^0 = \alpha_{i,j} f_j^0 \prod_{i' \in A_{j\backslash i}} r_{i',j}^0 \tag{5.9}$$

$$q_{i,j}^1 = \alpha_{i,j} f_j^1 \prod_{i' \in A_{j\backslash i}} r_{i',j}^1 \tag{5.10}$$

Die Ausgleichsvariable $\alpha_{i,j}$ wird zuerst mit 1 angenommen und erst nachträglich berechnet.

$$\alpha_{i,j} = \frac{1}{q^1_{i,j} + q^0_{i,j}} \tag{5.11}$$

Das so berechnete $\alpha_{i,j}$ wird dann rückwirkend in Gleichungen 5.9 und 5.10 eingesetzt, welche dann neu ermittelt werden.

Auswertung Die Berechnung gleicht dem horizontalen Schritt, außer dass für die Auswertung alle Wahrscheinlichkeiten in die Formel miteinfließen.

$$q^0_j = \alpha_j f^0_j \prod_{i' \in A_j} r^0_{i',j} \tag{5.12}$$

$$q^1_j = \alpha_j f^1_j \prod_{i' \in A_j} r^1_{i',j} \tag{5.13}$$

Die Berechnung der Variable α_j geschieht analog zu jener im *vertikalen* Schritt. Die Entscheidung über den Wert des Bits liefert ein einfacher Größenvergleich der Variablen:

$$f(n) = \begin{cases} 1 & \text{wenn } q^1_j > q^0_j \text{ bzw. } q^0_j < 0.5 \\ 0 & \text{sonst} \end{cases} \tag{5.14}$$

Mit der so gewonnenen Bitfolge kann ein Syndrom berechnet werden. Ergibt das Syndrom Null, ist die Decodierung abgeschlossen und der Algorithmus stoppt. Andernfalls wird mit den nächsten *horizontalen* Schritt des Algorithmus fortgefahren. Der Algorithmus iteriert so lange, bis er entweder zu einem korrekten Ergebnis konvergiert oder eine festgelegte Anzahl von Iterationen erreicht wird.

5.3 Logarithmische Version des SPA

Ansätze zur Erklärung des logarithmischen SPA halten sich zu großen Teilen an [15]. Der Vorteil der logarithmischen Version entsteht dadurch, dass sich Wahrscheinlichkeiten als Log-Likelihood Ratios (LLRs) einfacher multiplizieren, also addieren lassen. Die extrinsische Information $E_{i,j}$ vom Kontrollknoten i zum Bitknoten spiegelt die Wahrscheinlichkeit wider, dass das am $j$ten Bitknoten anliegende Bit r_j gleich 1 ist und somit die $i$te Paritätsgleichung erfüllt. Dies ist gleich der Wahrscheinlichkeit dafür, dass eine ungerade Anzahl von Bits in der Paritätsgleichung vorkommt. Folgende Gleichung beschreibt diesen Zusammenhang:

$$P^{ext}_{i,j} = \frac{1}{2} - \frac{1}{2} \prod_{j' \in B_{i \setminus j}} (1 - 2P_{i,j'}) \tag{5.15}$$

wobei $P_{i,j'}$ die Wahrscheinlichkeit von Kontrollknoten i ist, ob nun $r_{j'} = 1$. Somit ergibt sich die Wahrscheinlichkeit, dass mit $r_{j'} = 0$ die Paritätsgleichung erfüllt ist mit $1 - P_{i,j}^{ext}$. In LLR Werten ausgedrückt berechnet, sich $E_{i,j}$ mit

$$E_{i,j} = \mathrm{LLR}(P_{i,j}^{ext}) = \ln\left(\frac{1 - P_{i,j}^{ext}}{P_{i,j}^{ext}}\right) \tag{5.16}$$

Die Werte $E_{i,j}$ beschreiben somit die Nachrichtenqualität von den Kontrollknoten zu den Bitknoten. Setzt man nun (5.15) in (5.16) ein, erhält man

$$E_{i,j} = \ln\left(\frac{\frac{1}{2} + \frac{1}{2}\prod_{j' \in B_{i\setminus j}}(1 - 2P_{i,j'})}{\frac{1}{2} - \frac{1}{2}\prod_{j' \in B_{i\setminus j}}(1 - 2P_{i,j'})}\right) \tag{5.17}$$

Nachfolgend bezeichnet $M_{i,j}$ die Nachrichtenqualität, welche von den Bitknoten zurück zu den Kontrollknoten gesandt wird:

$$M_{i,j'} = \mathrm{LLR}(P_{i,j'}) = \ln\left(\frac{1 - P_{i,j'}}{P_{i,j'}}\right) \tag{5.18}$$

Integriert in (5.17) ergibt sich $E_{i,j}$ zu

$$\begin{aligned} E_{i,j} &= \ln\left(\frac{1 + \prod_{j' \in B_{i\setminus j}}\left(1 - 2\frac{e^{-M_{i,j'}}}{1+e^{-M_{i,j'}}}\right)}{1 - \prod_{j' \in B_{i\setminus j}}\left(1 - 2\frac{e^{-M_{i,j'}}}{1+e^{-M_{i,j'}}}\right)}\right) \\ &= \ln\left(\frac{1 + \prod_{j' \in B_{i\setminus j}}\frac{1-e^{-M_{i,j'}}}{1+e^{-M_{i,j'}}}}{1 - \prod_{j' \in B_{i\setminus j}}\frac{1-e^{-M_{i,j'}}}{1+e^{-M_{i,j'}}}}\right) \end{aligned} \tag{5.19}$$

Unter Verwendung der Beziehung

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{5.20}$$

ergibt sich $E_{i,j}$ zu

$$E_{i,j} = \ln\left(\frac{1 + \prod_{j' \in B_{i\setminus j}}\tanh\left(\frac{M_{i,j'}}{2}\right)}{1 - \prod_{j' \in B_{i\setminus j}}\tanh\left(\frac{M_{i,j'}}{2}\right)}\right). \tag{5.21}$$

Eine alternative Schreibweise ist mit Beziehung

$$\tanh^{-1}(x) = \frac{1}{2}\ln\left(\frac{1+x}{1-x}\right) \tag{5.22}$$

des öfteren in der Literatur zu finden:

$$E_{i,j} = 2\tanh^{-1}\left(\prod_{j' \in B_{i\setminus j}} \tanh\left(\frac{M_{i,j'}}{2}\right)\right). \tag{5.23}$$

Die an den Bitknoten anliegenden LLR-Werte der *a-priori*-Wahrscheinlichkeiten des Kanals werden als R_j bezeichnet. Zusammen mit den LLR-Werten, welche von den verknüpften Kontrollknoten eintreffen, ergibt sich die Summe dieser LLRs zu:

$$M_{i,j} = \mathrm{LLR}(P_j) = R_j + \sum_{i' \in A_{j\setminus i}} E_{i',j} \tag{5.24}$$

(5.23) kann noch weiter verändert werden, um die Komplexität des Decoders zu verringern. Dabei wird der Produktteil der Gleichung durch eine Summe ersetzt. Die Nachrichten von den Kontroll- zu den Bitknoten werden dabei folgend dargestellt: $M_{i,j'} = \alpha_{i,j'}\beta_{i,j'}$ mit $\alpha_{i,j'} = \mathrm{sgn}(M_{i,j'})$ und $\beta_{i,j'} = |M_{i,j'}|$. Über die Hilfsfunktion

$$\phi(x) = -\ln\tanh\left(\frac{x}{2}\right) = \ln\left(\frac{e^x+1}{e^x-1}\right) \tag{5.25}$$

mit der Eigenschaft $\phi(\phi(x)) = x$, also $\phi(x) = \phi^{-1}(x)$ ergibt sich:

$$E_{i,j} = \left(\prod_{j' \in B_{i\setminus j}} \alpha_{i,j'}\right) \phi\left(\sum_{j' \in B_{i\setminus j}} \phi(\beta_{i,j'})\right) \tag{5.26}$$

Das Produkt der Vorzeichen kann über eine Modulo-2 Addition der Hard Decisions von $M_{i,j'}$ berechnet werden, während man die Funktion $\phi(x)$ über eine Lookup-Tabelle implementiert.

Eine weitere Vereinfachung stellt der *Min-Sum* Algorithmus dar. Die Überlegung hierbei ist, dass der Produktterm durch seinen kleinsten Wert approximiert werden kann.

$$E_{i,j} \approx \prod_{j' \in B_{i\setminus j}} \mathrm{sgn}(M_{i,j'}) \underbrace{\min_{j' \in B_{i\setminus j}}} |M_{i,j'}| \tag{5.27}$$

Das Produkt der Vorzeichen berechnet sich wie oben, sodass der *Min-Sum* Algorithmus nur Additionen und die Berechnung von Minima beinhaltet. Diese Komplexitätsreduktion wirkt sich aber auch auf das Fehlerkorrekturvermögen aus, welches sich um einige Zehntel dB verschlechtert [31, S.153]. Durch die Beigabe eines Normalisierungsfaktors $\kappa > 0$ kann man diesem Effekt entgegenwirken:

$$E_{i,j} \approx \frac{1}{\kappa} \prod_{j' \in B_{i\setminus j}} \mathrm{sgn}(M_{i,j'}) \underbrace{\min_{j' \in B_{i\setminus j}}} |M_{i,j'}| \tag{5.28}$$

Eine genauere Betrachtung verschiedener Algorithmen mit verringerter Komplexität ist in [31] und [32] zu finden.

5.3.1 Analyse des SPA

Ohne vorhergegangene Simulation ist es nicht möglich vorherzusagen, wie sich ein gegebener TG während des Decodierungsprozesses verhält. In [21, S.600] wurde jedoch Folgendes gezeigt:

1. Das durchschnittliche Verhalten eines *Ensembles* von Codes ist charakteristisch für (fast) alle Codes des *Ensembles*.
2. Für große Blocklängen ist das durchschnittliche Verhalten gleichzusetzen mit dem eines zyklusfreien TG. Dies ermöglicht wiederum die rechnerische Bestimmung dieses Verhaltens durch Density Evolution (DE).
3. Es existiert ein Kanalparameter τ, für welchen es dem Decodierer nicht möglich ist, korrekt zu arbeiten, egal wie viele Iterationen in diesem Fall durchlaufen werden.

Als Ensemble werden dabei alle Codes mit der selben Blocklänge verstanden, für die die dazugehörigen Kontrollmatrizen die selben Parameter γ und ρ aufweisen, siehe Abbildung 5.2.

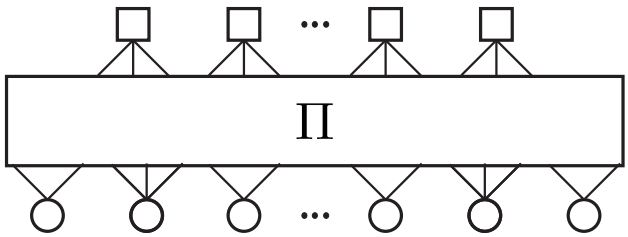

Abbildung 5.2: Darstellung eines Ensembles von Codes, welche nur durch ihre unterschiedliche Kantenanordnungen bei gleichen Gradverteilungen gekennzeichnet sind.

In diesem Kontext konnte bereits Gallager zeigten, dass die durchschnittliche Anzahl der benötigten Iterationen proportional $\log(\log(n))$ ist [3, S.24].

Density Evolution Die Wahrscheinlichkeiten, welche sich über den TG verbreiten, werden dabei als Zufallsvariablen betrachtet, die sich wiederum über Wahrscheinlichkeitsdichtefunktionen beschreiben lassen. Es wird dann beobachtet, wie sich diese Wahrscheinlichkeitsdichtefunktionen mit fortlaufenden Iterationen entwickeln. Dabei hat sich herausgestellt, dass ab einem bestimmten, vom Kanal abhängigen SNR-Wert, es dem Decodierungsprozess nicht möglich ist, zu einem korrekten Ergebnis zu konvergieren. Dieser

Schwellenwert τ gibt also an, ab welchen Kanalbedingungen eine korrekte Decodierung im Prinzip möglich ist.

5.3.2 Hybride SPAs

Ein hybrider SPA besteht in der ersten Stufe aus einem konventionellen SPA und hat als zweite Stufe einen MLG Decoder. Ein solcher hybrider Algorithmus ist natürlich nur bei MLG-decodierbaren LDPC-Codes sinnvoll. Er ist besonders nützlich, wenn der konventionelle SPA nur langsam konvergiert. In der Literatur wird eine Umstellung auf die zweite Decoderstufe nach etwa fünf Iterationen empfohlen [33, S.2730].

5.4 Bit-Flipping Algorithmus

Das Prinzip des BF ist dasselbe wie beim SPA, nur dass nicht mit Wahrscheinlichkeiten oder LLR-Werten gerechnet wird, sondern mit einfachen Ja/Nein-Entscheidungen, d.h. BF-Lösungen arbeiten demnach mit Hard Decisions. Nachdem die Paritätsgleichungen in den Kontrollknoten berechnet und das Ergebnis an die Bitknoten zurück übermittelt wurde, wird eine Entscheidung getroffen, ob nun das anliegende Bit korrekt ist oder nicht. Ist das anliegende Bit nicht korrekt, wird es einfach umgedreht (flipping), daher auch der Name[1]. Die einfachste Variante dieses Schrittes ist eine Mehrheitsentscheidung. Die nächste Iteration wird nun mit geflippten Bits durchgeführt. Der BF-Algorithmus stoppt sofort, wenn durch die Paritätsgleichungen ein gültiges Codewort $\mathbf{r} \odot \mathbf{H^T} = \mathbf{0}$ gefunden wurde.

Wie schon der SPA geht auch der BF-Algorithmus auf Gallager [3] zurück. Zusammenfassend gilt:

1. Berechne $\mathbf{r} \odot \mathbf{H^T}$ und bestimme die nicht erfüllten Paritätsgleichungen.
2. Berechne für jedes Bit des empfangen Codeworts die Anzahl der mit diesem Bit verknüpften, nicht erfüllten Paritätsgleichungen.
3. Invertiere jenes Bit oder jene Bits im empfangenen Codewort, für welche die größte Anzahl von nicht erfüllten Paritätsgleichungen errechnet wurde.
4. Durchlaufe mit dem veränderten Codewort erneut den Algorithmus, bis $\mathbf{r} \odot \mathbf{H^T} = \mathbf{0}$ oder sich nach einer bestimmten Anzahl von Iteration kein Ergebnis einstellt.

Ein erklärendes Beispiel ist in Abbildung 5.3 [15, S.57] angeführt. Ein Codewort 001011 wird bei der Übertragung gestört und als 011011 empfangen. Im ersten Schritt übermitteln die Bitknoten ihre Werte an die Kontrollknoten. Die für die einzelnen Bitknoten in den Kontrollknoten errechneten Werte

[1] Pseudocode für den BF Algorithmus findet man beispielsweise in [15, S.56].

werden im nächsten Schritt an die Bitknoten zurückgesandt. Aufgrund dieser neuen Informationen wird entschieden das zweite Bit zu invertieren. Im dritten Schritt werden die aktualisierten Werte wieder an die Kontrollknoten übermittelt, die Paritätsgleichungen errechnet und bei Erfüllung aller der Decodierungsprozess beendet.

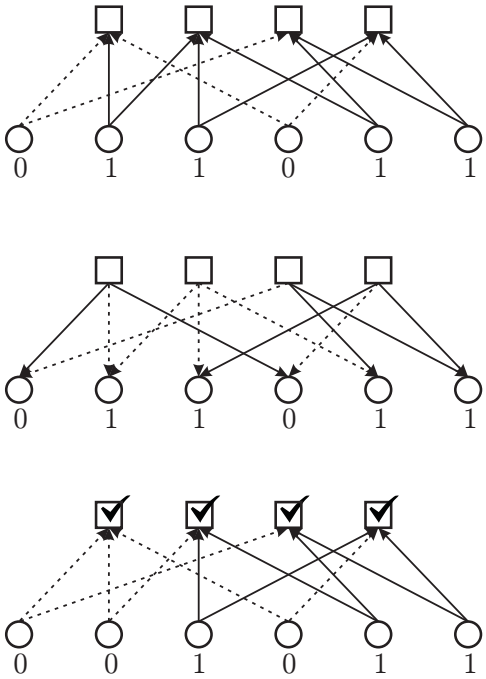

Abbildung 5.3: BF Decodierung der empfangenen Bitsequenz 011011 über den TG aus Abbildung 4.1.

Modifikationen des Bit-Flipping Algorithmus HDD-Algorithmen sind nicht so leistungsstark wie SDD-Varianten, dafür aber auch weniger komplex und rechenintensiv. Um die Leistungslücke zwischen SDD und HDD zu verkleinern, gibt es einige verbesserte Varianten des BF-Algorithmus.

Weighted Bit-Flipping (WBF) [33] berücksichtigt zusätzlich Wahrscheinlichkeiten über die Vertrauenswürdigkeit der berechneten Syndrome. Ähnliches gilt für Modified Weighted Bit-Flipping (MWBF) [34], nur dass die Qualität der empfangenen Symbole ebenfalls berücksichtigt wird.

Eine nochmalige Verbesserung ist in [35] zu finden. Dabei wird ähnlich wie bei SDD-Algorithmen die Wahrscheinlichkeit über die Richtigkeit eines bestimmten Bits nur aus den Wahrscheinlichkeiten der anderen Bits, welche an der Paritätsgleichung beteiligt sind, berechnet. Das geschieht aber nur, wenn die Vertrauenswürdigkeit des Bit selbst minimal ist. Dieser Algorithmus ist unter dem Namen Improved Modified Weighted Bit-Flipping

(IMWBF) bekannt. Der Rechenaufwand für WBF, MWBF und IMWBF ist proportional zu der Anzahl der Einsen in **H**, abgesehen von unterschiedlichen konstanten Faktoren, wie sie für die jeweiligen Varianten typisch sind.

5.5 Stopping Sets und Stoppkriterien

Die Aufgabe des Decoders ist es, wenn möglich zu einem korrekten Codewort zu konvergieren. Unter bestimmten Umständen mündet der Algorithmus aber zu einem ungültigen Codewort statt zu einem gültigen. Abhängig von der empfangenen Information kann es in Folge des Decodierungsprozesses zu Konstellationen zusammenhängender Knoten (Subgraphen) kommen, was es dem Decoder nicht mehr möglich macht, zu einem korrekten Ergebnis zu gelangen. In der Literatur sind solche Konstellationen unter den Namen *near codewords*, *trapping bzw. stopping sets*, *pseudo-codewords* und *instantons* bekannt [36, S.1].

Da der Decodierungsaufwand linear mit der Zahl der Iterationen ansteigt, ist es von Vorteil, ein mögliches Decodierversagen möglichst rasch durch geeignete Stoppkriterien zu prognostizieren. Nachfolgend sind einige dieser Kriterien aufgelistet.

Convergence of Mean Magnitude (CMM) Kriterium [37]: Durch die Beobachtung des Mittelwertes des Betrags der LLR-Werte ist es möglich, ein Decodierversagen vorherzusagen. In [38] wird vorgeschlagen, die Beobachtung nur auf die Bitknoten mit dem geringsten Grad zu beschränken, da diese für den Decodiererfolg repräsentativ sind. Durch diese Einschränkung wird bei gleichzeitig vernachlässigbarem Leistungsverlust an Komplexität gespart.

Number of Satisfied Parity-Check Constraints (NSPC) Kriterium [39]: Erfasst Veränderungen in der Anzahl der erfüllten Paritätsgleichungen, um so mögliche Trapping Sets zu identifizieren und die Decodierung abzubrechen.

Variable Node Reliability (VNR) Kriterium [40]: Stellt sich über mehrere Iterationen keine bemerkenswerte Veränderung in der Summe der Wahrscheinlichkeiten der Bitknoten ein, so steuert die Decodierung einem Decodierversagen entgegen und kann frühzeitig abgebrochen werden.

Kapitel 6

Konstruktionsprinzipien

Das Design von LDPC-Codes ist gleichzusetzen mit der Suche nach Ordnung und Struktur bei gleichzeitiger Erhaltung der zufälligen Eigenschaften der Kontrollmatrix. Für die Suche nach einer solchen Matrix gibt es viele Methoden und Prozeduren. LDPC-Codes können daher auch nach ihrem Konstruktionsprinzip unterschieden werden. Die Verringerung der Kanten ist gleichbedeutend mit der Dichte der Einsen in der Kontrollmatrix [41, S.162]. Daher ist eine Anforderung an alle Methoden, die Anzahl der Kanten im TG gering zu halten sowie keine Zyklen der Länge vier zu produzieren, welche den iterativen Decodierungsprozess negativ beeinflussen.

Man unterscheidet allgemein zwischen pseudo-zufällig konstruierten Codes und strukturierten Codes. Dabei erzeugen strukturierte Ansätze keine Codes mit besseren *Thresholds*, wirken sich jedoch positiv auf die Implementationskomplexität sowie auf das frühzeitige Auftreten von Fehlerhorizontalen aus [15, S.105]. Die prominentesten Vertreter dieser Methoden und Ansätze werden in diesem Kapitel zusammengefasst und kurz beschrieben. Da das Ergebnis dieser Methoden jeweils eine dünn besetzte Matrix ist, kommt es daher zu Überschneidungen unter den Methoden und ein und derselbe Code kann mehrfache Darstellungsmöglichkeiten besitzen. Irreguläre Codes als logische Weiterentwicklung bauen entweder auf diesen Konstruktionen auf oder die optimierten Gradverteilungen werden über DE gefunden.

6.1 Pseudo-zufällige Ansätze

6.1.1 Codes nach Gallager

Gallager-Codes sind die ursprünglichsten Erscheinungsformen von LDPC-Codes. Sie sind regulär und fallen auch unter die Gruppe der MLG-decodierbaren Codes. Eine nach Gallager definierte Kontrollmatrix besitzt folgende Form:

$$\mathbf{H} = \begin{Bmatrix} \mathbf{H_1} \\ \mathbf{H_2} \\ \vdots \\ \mathbf{H}_\gamma \end{Bmatrix} \tag{6.1}$$

Sie lässt sich folgenderweise bestimmen [23, S.852f]:

- Für ein gegebenes γ und ρ kann eine $k\gamma \times k\rho$ große Kontrollmatrix erstellt werden, wobei k eine positive ganze Zahl größer 1 ist.
- Die Matrix $\mathbf{H}$ besteht aus γ Teilmatrizen $\mathbf{H_1}, \ldots, \mathbf{H}_\gamma$, die alle die Dimension $k \times k\rho$ aufweisen. Jede Zeile einer Teilmatrix hat ρ Einsen und jede Spalte einer Teilmatrix hat eine einzige Eins.
- Die Zeilen $1 \leq i \leq k$ der ersten Teilmatrix $\mathbf{H_1}$ besitzen alle Einsen in den Spalten $(i-1)\rho + 1$ bis $i\rho$.
- Die restlichen Teilmatrizen werden aus Spaltenvertauschung der ersten Teilmatrix gebildet, wobei darauf zu achten ist, dass keine zwei Spalten und keine zwei Zeilen der resultierenden Kontrollmatrix mehr als eine gemeinsam mit Eins besetzte Stelle aufweisen.

Diese Vertauschungen sollten dazu führen, dass der resultierende Code gute Distanzeigenschaften besitzt, sowie keine kurzen Zyklen enthält. Unglücklicherweise gibt Gallager nicht vor, wie diese Vertauschungen auszusehen haben. Eine Möglichkeit, Gallager-Codes mit diesen Eigenschaften strukturiert zu konstruieren, besteht durch die Verwendung Euklidisch-Geometrischer Methoden [23, S.902f]. Derartige Codes besitzten keine Zyklen der Länge vier. Des Weiteren lässt er sich effizient codieren, da es sich um einen zyklischen Code handelt.

6.1.2 Codes nach MacKay

MacKay erweiterte die Codes von Gallager und entwarf mehrere Methoden zur Konstruktion von regulären als auch irregulären LDPC-Codes[1]. Diese Methoden beruhen auf einer computergestützten Suche, welche einer Reihe von Bedingungen zugrunde gelegt wurden. In [14, S.402] werden mehrere Wege, angeführt, um auf diese Weise Kontrollmatrizen $\mathbf{H}$ zu erzeugen:

1. Anfängliche Nullmatrix, in welcher zufällig γ Stellen pro Spalte invertiert werden. Dies ergibt einen irregulären Code.
2. Zufällig generierte Spalten mit Gewicht γ.
3. Spalten mit Gewicht γ und Zeilen einem Gewicht das so nahe wie möglich an einem vorgegebenen Wert ρ liegt.
4. Spalten mit Gewicht γ und Zeilen mit einem Gewicht ρ. Zwei Spalten dürfen sich nur an einer Position überlappen.

[1] Unter http://www.inference.phy.cam.ac.uk/mackay/ ist eine Online-Datenbank von verschiedenen iterativ-decodierbaren Codes zusammengestellt von MacKay, zu finden.

5. Wie in 4, nur dass kurze Zyklen vermieden werden.
6. $\mathbf{H} = [\mathbf{H_1}\mathbf{H_2}]$ erzeugt wie in 5, nur dass $\mathbf{H_2}$ invertierbar oder zumindest vollen Rang besitzt.

Diese Algorithmen könnten prinzipiell auch längere Zyklen vermeiden. Hierbei besteht jedoch die Gefahr, dass der Algorithmus keine brauchbare Kontrollmatrix mehr liefert [15, S.100]. Weiterführende Informationen zu MacKay-Codes sind in [23, S.99] und [15, S.99] zu finden. Ein großer Nachteil von MacKay-Codes ist die komplexe Codierung auf Grund der fehlenden Struktur in den Kontrollmatrizen. Abhilfe schafft aber die in Sektion 4.4 vorgestellte Matrixtransformation.

6.1.3 Ansätze über Tanner-Graphen

Über Algorithmen wie Bit Filling (BF) [42] oder Progressive Edge Growth (PEG) [43] ist es möglich, beginnend bei einem minimalen TG sukzessive diesen Graphen zu erweitern. Mittels dieser Methoden sind sowohl reguläre als auch irreguläre LDPC-Codes erzeugbar.

Ausgehend von einem bereits erstellten TG fügt der BF Algorithmus nacheinander zusätzliche Bit- und Kontrollkonten entsprechend hinzu, sodass Zyklen einer bestimmten Länge vermieden werden können. Diese Codes sind von ihrer Leistungsfähigkeit vergleichbar mit Mackay-Codes mit ähnlichen Parametern. Vereinfachte BF-Versionen erlauben einen TG-Aufbau mit einer Komplexität $O(m^2)$.

Ähnlich funktioniert der PEG Algorithmus, nur werden die Kanten dem TG so angefügt, dass ein möglichst großer Umfang erreicht wird. Über diese Methode lassen sich sehr effiziente Codes mit kurzer und mittlerer Blocklänge in $O(n\,m)$ konstruieren. Durch einige Veränderungen kann man den Algorithmus so modifizieren, dass sich eine Matrix in annähernder Dreiecksform ergibt, die eine Codierung in annähernd linearer Zeit (siehe Abschnitt 4.4) ermöglicht [43, S.392].

6.2 Strukturierte Ansätze

6.2.1 Geometrischer Ansatz

Geometrische Designs gehen aus den Beziehungen zwischen Punkten und Kanten, Ebenen oder Hyperebenen hervor [15, S.108]. Sie entstammen der euklidischen (EG) oder projektiven Geometrie (PG). Eine endliche Geometrie $\mathcal{Q}$, bestehend aus n Punkten und m Linien, besitzt folgende strukturelle Eigenschaften [23, S.859]:

- Jede Linie besteht aus ρ Punkten.
- Jeder Punkt liegt genau auf γ Linien.
- Zwei Punkte sind maximal durch eine Linie verbunden.

- Zwei Linien schneiden sich genau in einem Punkt oder sie schneiden sich überhaupt nicht.

Es wird nun eine $m \times n$ Matrix $\mathbf{H_Q^{(1)}}$ geformt, in der die einzelne Zeilen Inzidenzvektoren[2] darstellen. Vertauscht man die Zeilen mit den Spalten, erhält man die Matrix $\mathbf{H_Q^{(2)}} = [\mathbf{H_Q^{(1)}}]^{\mathbf{T}}$. Je nach Geometrie lassen sich daher zwei Typen von Codes konstruieren [33]: LDPC-EG-Codes Typ I und Typ II und LDPC-PG-Codes Typ I und Typ II. Bei beiden Typen handelt es sich um reguläre Codes mit guten Distanzeigenschaften und keinen Zyklen der Länge vier. EG- und PG-Codes sind entweder zyklisch (Typ I) oder quasi-zyklisch (Typ II) und lassen sich daher auch durch einfache Schieberegisterschaltungen in Hardware realisieren. Ein weiterer großer Vorteil ist, dass sich geometrische Codes analytisch besser behandeln lassen als Codes anderer Klassen [41, S.162]. Solche Codes können leider nicht mit jeder beliebigen Blocklänge und Coderate konstruiert werden. Ein weiterer Nachteil entsteht dadurch, dass die Spalten- und Zeilengewichte relativ groß und daher nicht optimal für eine iterative Decodierung sind [17, S.7]. Durch Punktieren oder Verkürzen lassen sich jedoch diese Nachteile teilweise beheben [17, S.7].

6.2.2 Kombinatorischer Ansatz

Die Theorie der kombinatorischen Designs beschäftigt sich mit der Existenz und Konstruktion von Mengensystemen, in denen die Schnittmengen genau festgelegte Eigenschaften aufweisen. Ein kombinatorisches Design wird auch Blockdesign oder Blockplan genannt. Um aufzuzeigen, wie mit Hilfe von kombinatorischen Fragestellungen LDPC-Codes konstruiert werden können, dient folgendes entnommenes Beispiel [15, S.105f]:

Es sollen *sieben* Politiker $\mathcal{P} = [1, 2, 3, 4, 5, 6, 7]$ auf *sieben* Ausschüsse genau so aufgeteilt werden, dass sich in jedem Ausschuss genau *drei* Politiker befinden und sich jedes Paar von Politikern höchstens in einem Ausschuss treffen darf. Eine mögliche Lösung für dieses Problem und ein Beispiel für ein kombinatorisches Design wäre, die Menge der Politiker $\mathcal{P}$ folgendermaßen auf eine Menge von Ausschüsse $\mathcal{B}$ aufzuteilen:

$$\mathcal{B} = [1, 3, 5], [2, 3, 7], [4, 5, 7], [1, 6, 7], [1, 2, 4], [3, 4, 6], [2, 5, 6] \tag{6.2}$$

Ein kombinatorisches Design besteht aus einer endlichen, nicht leeren Menge von $\mathcal{P}$ Punkten (Politikern) und einer endlichen, nicht leeren Menge $\mathcal{B}$, bestehend aus Untermengen von $\mathcal{P}$, die auch als Blöcke bezeichnet werden (Ausschüsse). Das Design ist regulär, wenn in jedem Block gleich viele Punkte γ sind und jeder Punkt in gleich vielen Blöcken ρ verkommt. Jedes kombinatorische Design kann auch als eine $v \times b$ Inzidenzmatrix $\mathcal{H}$

[2] Ein Punkt heißt inzident mit einer Linie, wenn er von dieser Linie geschnitten wird, oder anders ausgerückt: wenn der Punkt in der Linie enthalten ist [23, S.860].

mit $v = |\mathcal{A}|$ und $b = |\mathcal{C}|$ repräsentiert werden. Jede Zeile entspricht einem Punkt (Politiker) und jede Spalte einem Block (Ausschuss). Interessant für Konstruktionen von LDPC-Codes ist nun das sogenannte *t-Design* für eine Menge von 2 Punkten, in denen sich jedes Punktepaar nur in λ Blöcken überschneidet. Für $\lambda = 1$ werden diese Systeme *Steiner-2-Designs* genannt. Ein *2-Design* wird auch als Balanced Incomplete Block Designs (BIBD) mit den Parametern $2-(v, b, \rho, \gamma, \lambda)$ bezeichnet. Obiges Beispiel ist demnach ein 2 - (7, 7, 3, 3, 1) BIBD.

$$\mathcal{H} = \mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{6.3}$$

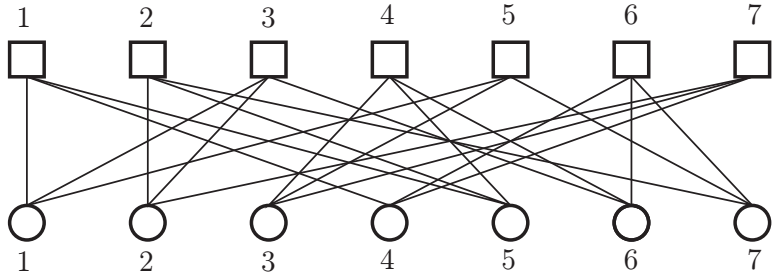

Abbildung 6.1: TG für die Matrix aus Gleichung 6.3 für ein 2-(7, 7, 3, 3, 1) BIBD [15].

Die Inzidenzmatrix dieses Graphen entspricht der Kontrollmatrix eines LDPC-Codes. Reguläre Designs ergeben einen (γ, ρ)-regulären LDPC-Code wobei auf $\gamma \ll v$ und $\rho \ll b$ zu achten ist. Durch die Verwendung von *Steiner-2-Designs* entstehen keine Zyklen der Länge vier, jedoch welche der Länge sechs [44, S.236].

Weiterführende kombinatorische Designs beruhen auf *Krikman-Tripel-Designs*, *Steiner-Tripel-Designs*, *difference families* [15, S.114] und Lateinischen Quadraten [45]. Codes auf der Basis von *difference families* sind One-Step Majority-Logic Decodeable (OSMLD) und besitzen ebenfalls keine Zyklen der Länge vier [23, S.929].

6.2.3 Zirkulante Permutationsmatrizen

Codes, die auf zirkulanten Permutationsmatrizen[3] beruhen, werden auch binäre *Array-Codes* genannt. Sie besitzen eine dünn besetzte Kontrollmatrix und können daher als LDPC-Codes betrachtet und auch so decodiert werden. Die Kontrollmatrix von Array-Codes wird durch drei Parameter (eine Primzahl p und zwei natürliche Zahlen k und j, die beide kleiner oder gleich p sind) beschrieben und besitzt folgende Form [46, S.1753]:

$$\mathbf{H}_{[\mathbf{jp}\times\mathbf{kp}]} = \begin{bmatrix} \mathbf{E} & \mathbf{E} & \mathbf{E} & \dots & \mathbf{E} \\ \mathbf{E} & \mathbf{A} & \mathbf{A^2} & \dots & \mathbf{A^{k-1}} \\ \mathbf{E} & \mathbf{A^2} & \mathbf{A^4} & \dots & \mathbf{A^{2(k-1)}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{E} & \mathbf{A^{j-1}} & \mathbf{A^{2(j-1)}} & \dots & \mathbf{A^{(j-1)(k-1)}} \end{bmatrix} \tag{6.4}$$

Die Einträge $\mathbf{E}$ und $\mathbf{A}$ in der Kontrollmatrix stehen für eine $p \times p$ große Einheitsmatrix bzw. zirkulante Permutationsmatrix. Durch diese Konstruktion existieren keine Zyklen der Länge vier in der Kontrollmatrix. Um die Codierung effizient zu gestalten, ist es nötig, die Kontrollmatrix auf eine untere Dreiecksform zu bringen. Durch zyklische Vertauschungen der Zeilen und ersetzen derjenigen Elemente, die sich unter der aus Einheitsmatrizen bestehenden Diagonale befinden, entsteht eine Kontrollmatrix der Form:

$$\mathbf{H}_{[\mathbf{jp}\times\mathbf{kp}]} = \begin{bmatrix} \mathbf{E} & \mathbf{E} & \mathbf{E} & \dots & \mathbf{E} & \mathbf{E} & \dots & \mathbf{E} \\ \mathbf{0} & \mathbf{E} & \mathbf{A} & \dots & \mathbf{A^{j-2}} & \mathbf{A^{j-1}} & \dots & \mathbf{A^{k-2}} \\ \mathbf{0} & \mathbf{0} & \mathbf{E} & \dots & \mathbf{A^{2(j-3)}} & \mathbf{A^{2(j-2)}} & \dots & \mathbf{A^{2(k-3)}} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{E} & \mathbf{A^{j-1}} & \dots & \mathbf{A^{(j-1)(k-1)}} \end{bmatrix} \tag{6.5}$$

Ein Code basierend auf dieser Kontrollmatrix besitzt eine Blocklänge von $n = kp$ Bits mit $(k - j)p$ informationstragenden Bits. Array-Codes zeichnen sich durch sehr niedrige Fehlerhorizontale aus, und es können Codes mit niedriger als auch hoher Coderate entworfen werden. Es folgt aber aus obiger Konstruktionsvorschrift, dass nur ausgewählte Blocklängen und Coderaten möglich sind [17, S.7]. Zyklische und quasi zyklische Permutationsmatrix können über EG oder PG erzeugt werden. In [47] wurde gezeigt, dass Codes nach diesem Design maximal einen Umfang von 12 aufweisen können. Als ein Vorteil ist jedoch der verminderte Speicherbedarf einer $p \times p$ großen Ausgangsmatrix, um den Faktor $1/p$ zu nennen [48]. Der im Folgenden vorgestellte Ansatz über *Protographen* und *Superpositionen* ist ähnlich zur Methode mit zirkulanten Matrizen, bieten aber eine differenzierte Sichtweise.

[3] Als Permutationsmatrix wird eine quadratische Matrix bezeichnet, bei der das Zeilen- und Spaltengewicht konstant bei eins ist. Zirkulant ist somit gleichbedeutend mit zyklisch.

Protographen

Bei der Konstruktion über Protographen [49] wird ein kleiner TG, der Protograph, mehrfach dupliziert und die Kanten dieser Graphen zufällig vertauscht. Durch pseudo-zufällige Permuationen kann erreicht werden, dass kurze Zyklen im Endgraphen vermieden werden. Als erklärendes Beispiel dient Abbildung 6.2 [15, S.104]. Der Protograph wird dabei zwei Mal dupliziert und die Kanten nacheinander vertauscht. Dabei bleibt eine Kante immer an einem Kontrollknoten fixiert und wird zufällig mit dem gleichen Bitknoten eines anderen Graphen verbunden. Dieser Vorgang wird nun solange wiederholt, bis alle Kante vertauscht sind und sich ein Graph wie in der letzten Zeile ergibt [15, S.103].

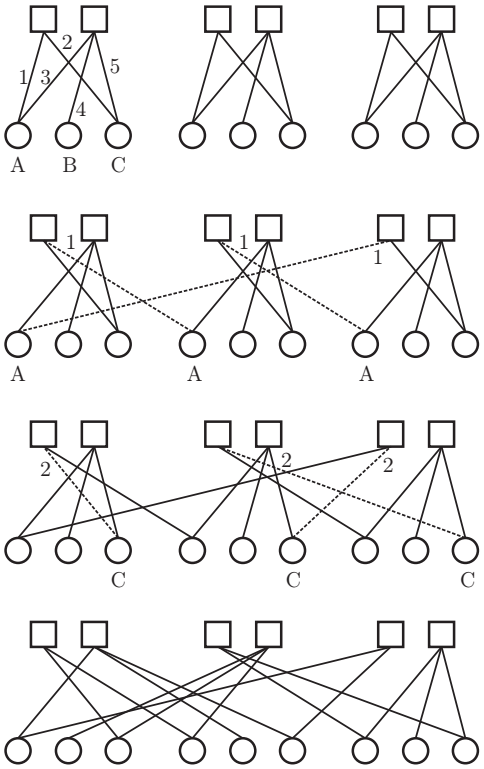

Abbildung 6.2: Protographische Konstruktion eines Codes.

Zyklische Permutationen erzeugen wiederum einen zyklischen Code mit den dazugehörigen Vorteilen (geringe Komplexität des Codierers). Die Methode ist gut geeignet, um lange irreguläre LDPC-Codes zu generieren. Ein so konstruierter Graph kann auch selbst wieder als ein Protograph verwendet werden. Unter bestimmten Umständen ist die Konstruktion über einen Protograph identisch mit der *Superpositions*-Methode.

Superposition

Bei dieser Technik werden die einzelnen Stellen einer Basismatrix durch eine $v \times v$ Matrix ersetzt. Diese Matrix wird Superpositionsmatrix genannt. Dabei werden die Stellen der Basismatrix, in der eine Null steht, durch Nullmatrizen und sonst durch immer verschiedene binäre Matrizen ersetzt. Wählt man zyklische Matrizen als überlagernde Matrizen, erhält man einen quasi-zyklischen Code. In [16] werden drei unterschiedliche Methoden zur Konstruktion von Superpositionsmatrizen genannt: Typ I konstruiert über EG oder PG, Typ II konstruiert über Reed-Solomon (RS) Codes mit zwei Informationsymbolen und Typ III konstruiert über HAMMING-Codes mit minimalem Gewicht. Auch hier gilt wie bei der Konstruktion über *Protographen*, dass über kurze und einfache LDPC Codes solche mit größerer Blocklänge konstruiert werden können.

6.2.4 Repeat-Accumulate Codes

Ein Repeat-Accumulate (RA) Code ist als eine Art Hybrid zwischen seriellen Turbo-Codes und LDPC-Codes anzusehen. Bei einem RA-Code werden die Eingangsbits mehrmals wiederholt und durch einen *Differentialencoder* (*Akkumulator*) verarbeitet. RA-Codes erbringen gute Leistungen auf AWGN-Kanälen [41, S.161] und kommen sehr nahe an die SHANNON-Grenze heran, weisen dabei aber nur Coderaten $R_c < 1/2$ auf. Eine Verbesserung stellen Irregular Repeat-Accumulate (IRA) Codes dar. Bei diesen Codes werden einige Bits öfters wiederholt als andere. Dies führt zu einer verbesserten Leistungsfähigkeit sowie zu höheren Coderaten. Die sogenannten extendend Irregular Repeat-Accumulate (eIRA)-Codes sind systematisch, was den Codierungsvorgang erleichtert, und erlauben niedrige sowie hohe Coderaten [17, S.8]. Die drei verschiedene Encoderschaltungen sind in Abbildung 6.3 zu sehen. Genaueres zum Aufbau von eIRA Codes und ihre Codierung sind in Abschnitt 7.1.1 zu finden.

6.3 Graphentheoretische Ansätze

Unter graphentheoretischen Ansätzen ist allgemein die Konstruktion von irregulären LDPC-Codes gemeint, die daraufhin optimiert wurden, um möglichst nahe an der SHANNON-Grenze zu operieren.

Der Vorteil von irregulären Codes liegt in der ungleichen Gradverteilung der Knoten im TG. Die höhergradigen Bitknoten sind weniger anfällig gegenüber fehlerhaften Wahrscheinlichkeiten im TG als Bitknoten mit weniger Kontrollknoten als Nachbarn. Durch die iterative Decodierung wird nun diese verstärkte Vertrauenswürdigkeit der hochgradigen Bitknoten über den TG zu den niedergradigen verteilt, vergleichbar mit dem Dominoeffekt.

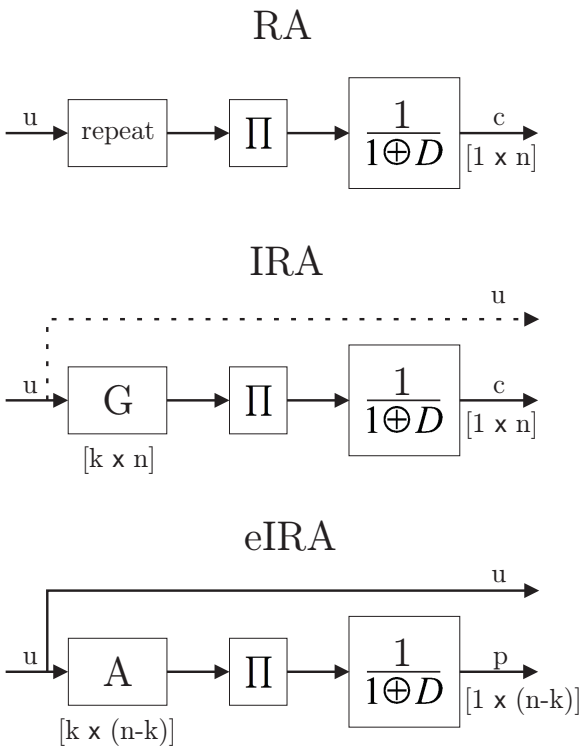

Abbildung 6.3: Encoderschaltungen für RA, IRA und eIRA Codes [17, S.7].

6.3.1 Irreguläre Codes aus regulären Codes

Eine Methode, um gewünschte kurze Zyklen aus dem TANNER-Graph zu entfernen, besteht darin, dass man die konkrete Reihe oder Spalte aufteilt. Die Einsen der zu ersetzenden Reihe oder Spalte werden auf die neuen aufgeteilt (siehe Abbildung 6.4). So entsteht eine noch dünner besetzte Matrix, da die Zahl der Einsen gleich bleibt, aber zusätzliche Nullen hinzukommen. Je nachdem, ob nun eine Spalte oder Reihe geteilt wurde, entsteht so ein Code mit vergrößerter Blocklänge bzw. verringerter Coderate [15, S.101].

Indem man die Knoten unterschiedlich aufteilt, ist es über diese Vorgehensweise möglich, aus einem regulären LDPC-Code einen irregulären zu erzeugen, welcher den Ursprungscode in seinem Fehlerkorrekturverhalten übertrifft [23, S.897].

6.3.2 Optimierung über Density Evolution

Über DE[4] in Kombination mit einer Optimierung der Gradverteilungen ist es möglich, den leistungbestimmenden *Threshold* in Richtung der SHAN-

[4]Unter http://sigpromu.org/ldpc/DE/index.php ist eine Internetseite zu erreichen, auf der Online für eine gegebene Gradverteilung der DE-Prozess durchgeführt werden kann.

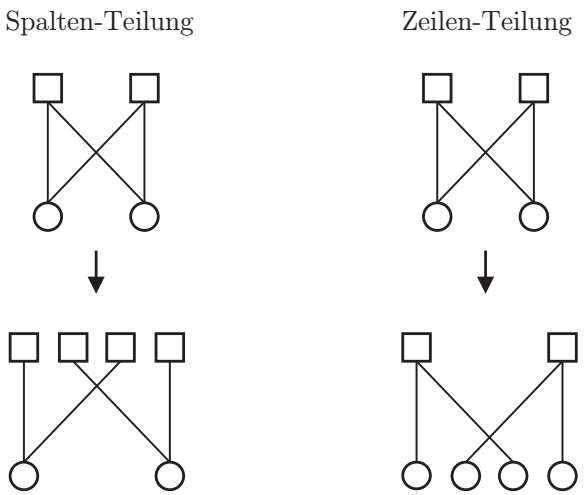

Abbildung 6.4: Aufbrechen eines Zykluses der Länge vier über Zeilen- bzw. Spalten-Teilung [23, S.893].

NON-Grenze zu verschieben. Die ersten Ansätze, um optimierte irreguläre Gradverteilungen zu finden, beruhen auf den Regeln der *linearen Programmierung* [19]. Folgende Bedingungen haben sich bei der Suche nach optimalen Gradverteilungen als produktiv erwiesen:

- Es dürfen keine Zyklen auftreten, die nur aus Bitknoten mit Grad zwei bestehen.
- Bitknoten mit Grad zwei dürfen nur mit Paritätsbits verknüpft werden.
- Es dürfen keine Zyklen der Länge vier im TG vorkommen.
- Die maximale Anzahl von Bitknoten darf nicht größer sein als die Gesamtzahl der Paritätsbits.

Codes mit gutem *Threshold* weisen eine Gradverteilung mit wenigen sehr hohen Graden und mit vielen sehr kleinen Graden auf. Die restlichen Grade verteilt man zwischen den Extremwerten [15, S.115]. Über eine Verbesserung des DE-Algorithmus war es in [50] möglich einen Code mit einer Blocklänge von $n = 10^7$ und Coderate von $R_c = 1/2$ zu finden, der bei BER von 10^{-6} nur 0.0045 dB von der SHANNON-Grenze entfernt ist (2000 Iterationen). Das Besondere bei diesem Code ist, dass einige Knoten einen extrem hohen Knotengrad, nämlich 8000, besitzen.

6.4 Design-Regeln

Laut [15, S.116] sollten beim Design von LDPC-Codes folgende Regeln in Betracht gezogen werden:

- **Umfang**: Da der SPA im Zusammenspiel mit einem TG ohne kurze Zyklen bessere Ergebnisse bringt als mit einem TG mit kurzen Zyklen, ist man bemüht, Zyklen der Länge vier zu vermeiden. Aber es gibt auch Codes, die *mit* Zyklen der Länge vier gute Eigenschaften in Bezug auf den SPA besitzen [51]. Dies gelingt durch eine große Anzahl von zusätzlichen linear unabhängigen Reihen (Paritätsgleichungen) in der Kontrollmatrix [15, S.82].
- **Mindest**-HAMMING-**Distanz**: Eine zu geringe Mindestdistanz verursacht bei großen SNR-Werten Fehlerhorizontalen, daher sollte die Mindestdistanz so hoch wie möglich sein.
- **Dichte der Einsen**: Eine zu geringe Anzahl von Einsen ist der Schlüssel zu einem SPA-Decoder mit geringer Komplexität. Zu wenig Einsen wiederum wirken sich negativ auf die Mindest-HAMMING-Distanz und die Größe der Stopping Sets aus.
- **Codierung**: Pseudo-zufällig erzeugte Codes besitzen den Nachteil der aufwendigen Codierung. Zyklische und RA-Codes können hingegen viel effizienter codiert werden.
- **Technische Umsetzung**: Für zufällig erstellte Matrizen ist es notwendig, die gesamte Matrix zu speichern, während sich ein deterministischer Code mit wenigen Parametern beschreiben lässt.

Kapitel 7

Implementation und Simulation

7.1 DVB-S2 Standard

Der Standard der zweiten Generation für breitbandinge Satelliten-Anwendungen (DVB-S2) wurde auf folgende Aspekte hin optimiert: Beste Übertragungsqualität, größtmögliche Flexibilität und maßvolle Komplexität [52, S.11]. Um einen Kompromiss zwischen der Qualität, der Übertragung und der Komplexität der eingesetzten Komponenten zu erreichen, setzt man auf LDPC-Codes in Kombination mit BCH-Codes, sowie auf multiple Modulationsverfahren und verschiedene Coderaten. Kleine Coderaten kommen bei niedrigen SNR-Werten zum Einsatz. Je nach Anwendung wird ein Block oder Frame mit 64800 Bits oder 16200 Bits übertragen. Die größeren Blocklängen sind zwar effizienter als kurze, weisen aber auch einen aufwändigeren Decodierungsprozess auf. Durch diese Verfahren und Methoden erreicht DVB-S2 gegenüber seinem Vorgänger Digital Video Broadcasting via Satellite (DVB-S) einen Kapazitätsgewinn von 20% bis 35% bei gleichen *Übertragungsbedingungen* [53, S.13].

Der FEC-Mechanismus bei DVB-S2 besteht aus einer Codeverkettung von BCH- und LDPC-Codes. Durch den äußeren BCH-Code wird ein Betrieb im Bereich von $0.6 - 1.2$ dB entfernt von der Shannon-Grenze ermöglicht [52, S.15]. Die Mindest-Hamming-Distanz eines verketteten Codes ergibt sich aus dem Produkt der Mindest-Hamming-Distanzen von innerem und äußerem Code, welches die verbesserte Korrekturleistung widerspiegelt [54, S.698]. Bis zur Einführung von iterativen Decodierungsverfahren stellten verkettete Codes die mächtigsten Werkzeuge zur Fehlererkennung und Fehlerkorrektur dar. BCH-Codes stellen eine leistungsfähige Klasse von zyklischen Codes dar, mit denen sich *Bündelfehler* [55, S.185], aber auch mehrfach auftretende Zufallsfehler [23, S.194] einfach korrigieren lassen. Im DVB-S2 Standard fällt dem BCH-Code die Aufgabe zu, die Fehlerhorizontale weiter

abzusenken [53, S.4].

7.1.1 Effiziente Codierung von eIRA-Codes

Als innerer Code kommt ein eIRA-LDPC-Code wie in Sektion 7.1.1 beschrieben, zum Einsatz. In [56] wird die Matrix $\mathbf{H_1^T}$ aus (7.3) in Tabellenform angegeben, in der die Indizes der mit Eins besetzen Stellen eingetragen sind. Um den Speicheraufwand dieser Matrix auf ein Minimum zu reduzieren, wurde eine gewisse Periodizität hinzugefügt. Dabei gelingt es, den Speicheraufwand um den Faktor 360 zu reduzieren. Details zum Codierungsprozess sind in [52, S.64] und [56, S.22] zu finden. Dieser spezielle Code wurde so gewählt, dass er komplett frei von Zyklen der Länge vier ist und Zyklen der Länge sechs auf ein Minimum reduziert sind [52, S.64]. Tabelle 7.1 liefert einen guten Überblick über den Aufbau des irregulären Codes, indem man die Anzahl der Bitknoten mit bestimmten Graden auflistet.

ID	Coderate	13	12	11	8	4	3	2	1
1/4	1/5		360				2800	12959	1
1/3	1/3		1800				3600	10799	1
2/5	2/5		2160				4320	9719	1
1/2	4/9				1800		5400	7999	1
3/5	3/5		3240				6480	6479	1
2/3	2/3	1080					9720	5399	1
3/4	11/15		360				11520	4319	1
4/5	7/9						12600	3599	1
5/6	37/45	360					12960	2879	1
8/9	8/9					1800	12600	1799	1

Tabelle 7.1: Anzahl der Bitknoten mit verschiedenen Graden für Codes mit einer Blocklänge von 16200 Bits [52, S.65].

Bei eIRA-Codes besitzt die Kontrollmatrix die Form

$$\mathbf{H} = [\mathbf{H_1}\ \mathbf{H_2}] \tag{7.1}$$

wobei $\mathbf{H_1}$ eine dünn besetzte $m \times k$ Matrix ist und $\mathbf{H_2}$ folgende Form besitzt:

$$\mathbf{H_2} = \begin{bmatrix} 1 & & & & & & \\ 1 & 1 & & & & & \\ & 1 & 1 & & & & \\ & & & \ddots & & & \\ & & & & 1 & 1 & \\ & & & & & 1 & 1 \end{bmatrix} \tag{7.2}$$

Die Generatormatrix

$$\mathbf{G} = [\mathbf{E}\ \mathbf{P}] = [\mathbf{E}\ \mathbf{H_1^T}\ \mathbf{H_2^{-T}}] \tag{7.3}$$

besteht in ihrer systematischen Form aus $\mathbf{P} = \mathbf{H_1^T}\ \mathbf{H_2^{-T}}$, wobei $\mathbf{H_2^{-T}}$ folgendermaßen aufgebaut ist:

$$\mathbf{H_2^{-T}} = \begin{bmatrix} 1 & 1 & \cdots & 1 & 1 \\ & 1 & & & \\ & & \ddots & \vdots & \vdots \\ & & & 1 & 1 \\ & & & & 1 \end{bmatrix} \tag{7.4}$$

Diese Matrix $\mathbf{H_2^{-T}}$ ist die Generatormatrix eines Differentialcodierers, der auch namensgebend als Akkumulator bekannt ist [57, S.14].

Ein Codewort eines eIRA-LDPC-Codes setzt sich aus den Datenbits und den Paritätsbits zusammen:

$$c = [u_1, \ldots, u_k, p_1, \ldots, p_{n-k}] \tag{7.5}$$

Über Kontrollmatrix und Datenbits können mit folgenden Gleichungen die Paritätsbits berechnet werden:

$$p_1 = \sum_{i=1}^{k} u_k h_{1,k} \tag{7.6}$$

$$p_m = p_{m-1} + \sum_{i=1}^{k} u_k h_{m,k}\ 2 \leq m \leq n-k \tag{7.7}$$

wobei $h_{i,j}$ dem (i,j)ten Element der Kontrollmatrix entspricht. (7.6) und (7.7) können auch als ein rekursives Gleichungssystem angesehen werden [58, S.1960]. Mit folgendem Gleichungssystem lässt sich ein Codewort ohne Zuhilfenahme der Generatormatrix erzeugen:

$$\begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{n-k} \end{bmatrix} = \begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,k} \\ h_{2,1} & h_{2,2} & \ddots & h_{2,k} \\ \vdots & \ddots & \ddots & \vdots \\ h_{(n-k),1} & \cdots & h_{(n-k),(k-1)} & h_{(n-k),k} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-k} \end{bmatrix} + \begin{bmatrix} 0 \\ p_1 \\ \vdots \\ p_{n-k-1} \end{bmatrix} \tag{7.8}$$

7.1.2 Decodierung

Ein Decodieralgorithmus, der auf Schleifen beruht [15, S.64], ist zwar anschaulicher, aber in einer MATLAB™-Implementation wenig effizient, da dieses Simulations-Tool gerade Vektor- und Matrixoperationen besonders effi-

zient verarbeitet. Die Vektorisierung[1] des SPA reduziert die Berechnungszeit drastisch. Der im Rahmen dieser Arbeit in MATLAB™programmierte SPA-Decoder liefert dieselben Ergebnisse wie der in der COMMUNICATIONS SYSTEM TOOLBOX™von MATLAB™vorhandene Decoder, ist aber immer noch zu langsam, um aufwändigere Simulationen in zufriedenstellender Zeit durchzuführen. Deshalb wurde, nach Verifikation der eigens implementierten Funktion, die in MATLAB™vorhandene Funktion verwendet, um die folgend abgebildeten Kurven aufzunehmen.

7.2 Simulationen und Vergleiche

Simuliert wurden hauptsächlich Codes mit 16200 Bit Blocklänge[2], um so die Simulationszeiten kurz zu halten. Die Simulationen wurden mit denen aus [57] verglichen. Die erkennbaren Abweichungen erklären sich durch die halbierte Anzahl der höchstens erlaubten Iterationen. Die meisten der folgenden Simulationen wurden mit maximal 50 Iterationen durchgeführt, die in [57] hingegen mit 100. Die Werte zu den teilweise in den Diagrammen angegeben SHANNON-Grenzen für binäre Codes bei verschiedenen Coderaten sind aus [23, S.20] entnommen.

Coderate	16200 Bits	16200 Bits [57, S.16]
1/4	0,79 dB	0,25 dB
1/2	1,08 dB	0,93 dB
3/4	2,48 dB	2,33 dB
8/9	3,82 dB	3,78 dB

Tabelle 7.2: Vergleich der E_b/N_0-Werte, um eine FER $= 10^{-3}$ zu erreichen.

Um zu zeigen, dass die selbst geschriebene Decodierungsfunktion dieselben Ergebnisse erbringt wie die in MATLAB™, wurde ein eIRA-Code mit Coderate $R_c = 2/5$ mit beiden Funktionen simuliert und gegenübergestellt (siehe Abbildung 7.1).

In den Abbildungen 7.2 und 7.3 ist der BER- und FER-Verlauf in Abhängigkeit der erlaubten Iterationen dargestellt. Je mehr Iterationen durchlaufen werden, desto besser sind die Ergebnisse, wobei anzumerken ist, dass sich ab 100 Iterationen, eine zusätzliche Leistungsverbesserung nur durch sehr viele Iterationen bewerkstelligen lässt.

Abbildungen 7.4 und 7.5 stellen exemplarisch vier Codes mit unterschiedlichen Coderaten aus dem DVB-S2 Standard dar. Mit steigender Coderate

[1] Die Vektorisierung basiert auf dem SPA von *Igor V. Kozintsev*, der auf http://www.kozintsev.net/soft.html zu finden ist.

[2] Bei Codes mit kurzen Blocklängen ist zu beachten, dass ein Unterschied zwischen der bezeichneten Coderate und der echten Coderate besteht [57, S.13].

gelangen die BER-Kurven immer näher an die jeweiligen SHANNON-Grenzen heran. Bemerkenswert ist die geringere Steilheit der Wasserfallregion der FER-Kurve des Codes mit $CR = 1/4$. Dies liegt an der zu geringen Anzahl an Iterationen (vergleiche mit Abbildung 7.3).

Der Vergleich zwischen den beiden Frametypen in den Abbilungen 7.6 und 7.7, zeigt den zu erwartenden Vorteil des 64800 Bit langen Frames gegenüber des um dreiviertel kürzeren 16200 Bit langen Frames, bei niedrigen BER-Werten.

Eine Gegenüberstellung, eines $(128, 128)$ regulären LDPC-Codes [33, S.2738] und eines Quasi-Cyclic (QC)-LDPC-Codes [16, S.248] mit einem eIRA-DVB-S2-LDPC-Codes der selben Coderate und annähernd gleicher Blocklänge, ist in Abbildung 7.8 zu sehen.

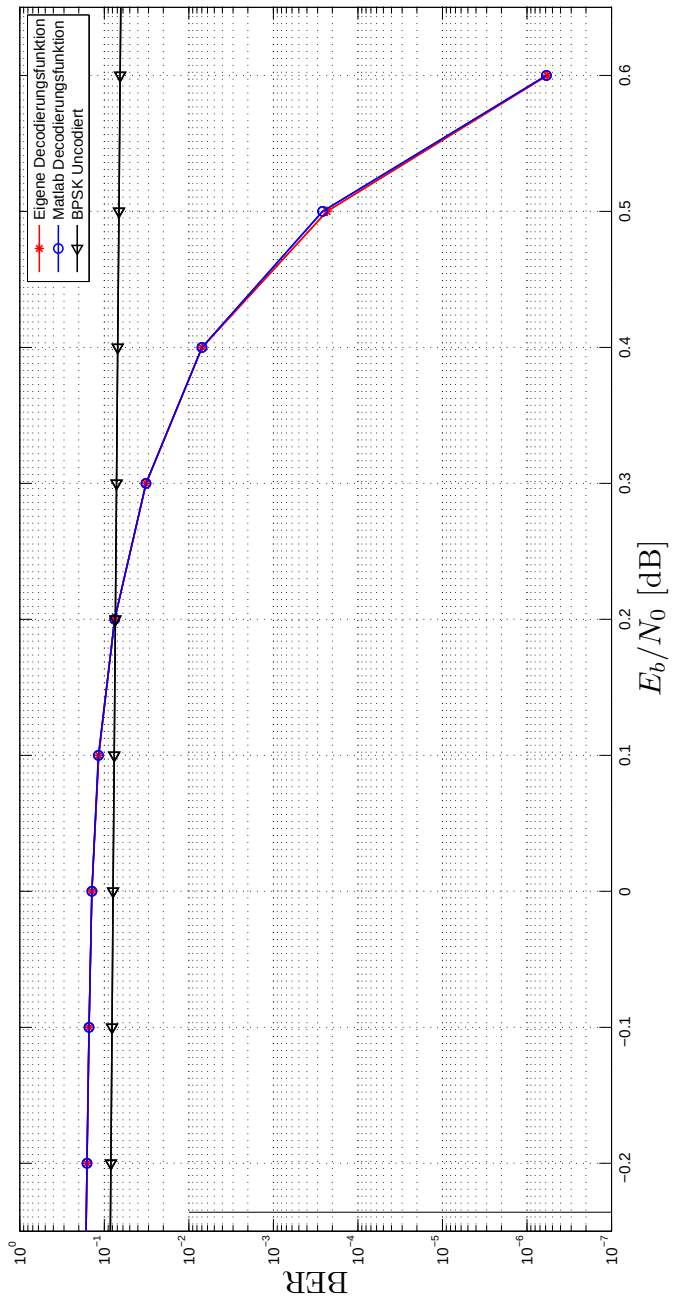

Abbildung 7.1: Gegenüberstellung der BER der eigenen und der in MATLAB™vorhandenen Decodierfunktion.

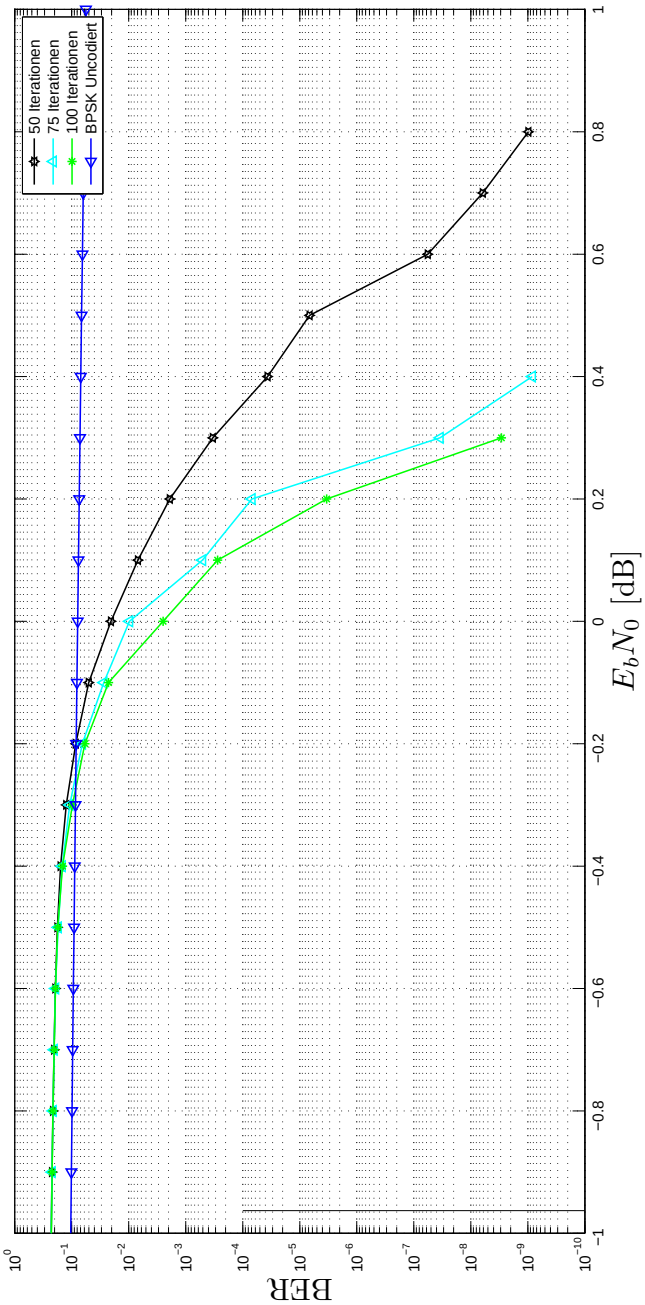

Abbildung 7.2: BER Vergleich für unterschiedliche Iterationen und $R_c = 1/4$.

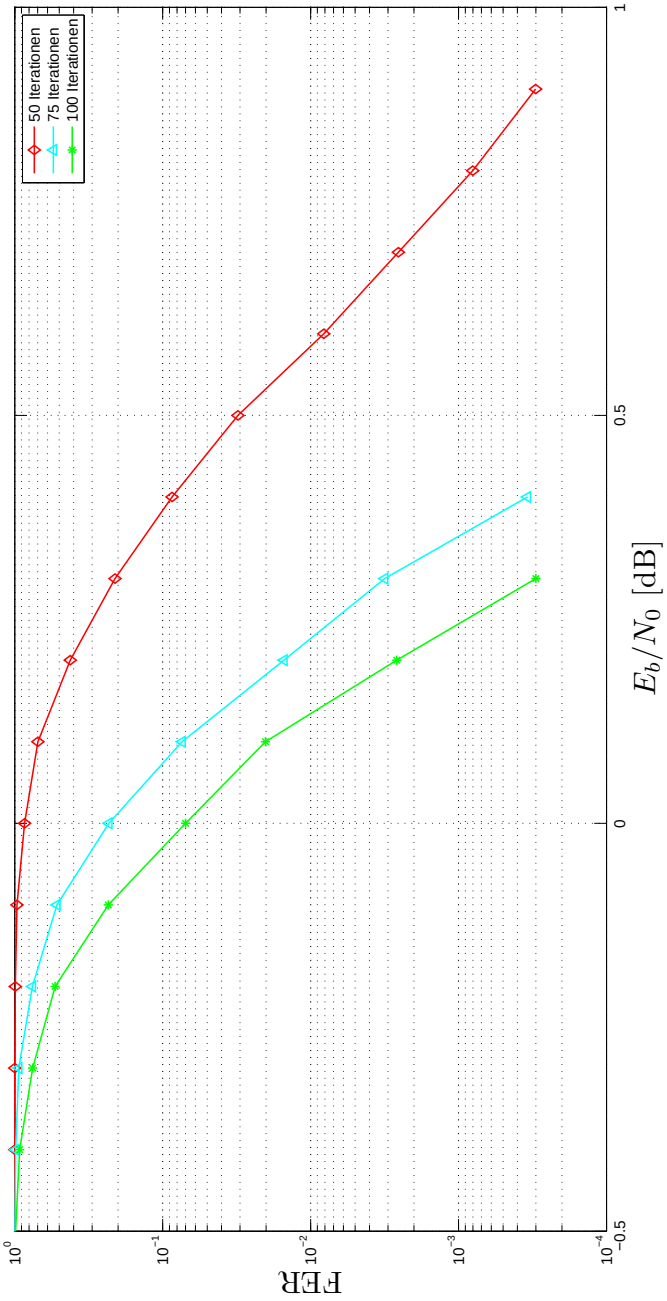

Abbildung 7.3: FER Vergleich für unterschiedliche Iterationen und $R_c = 1/4$.

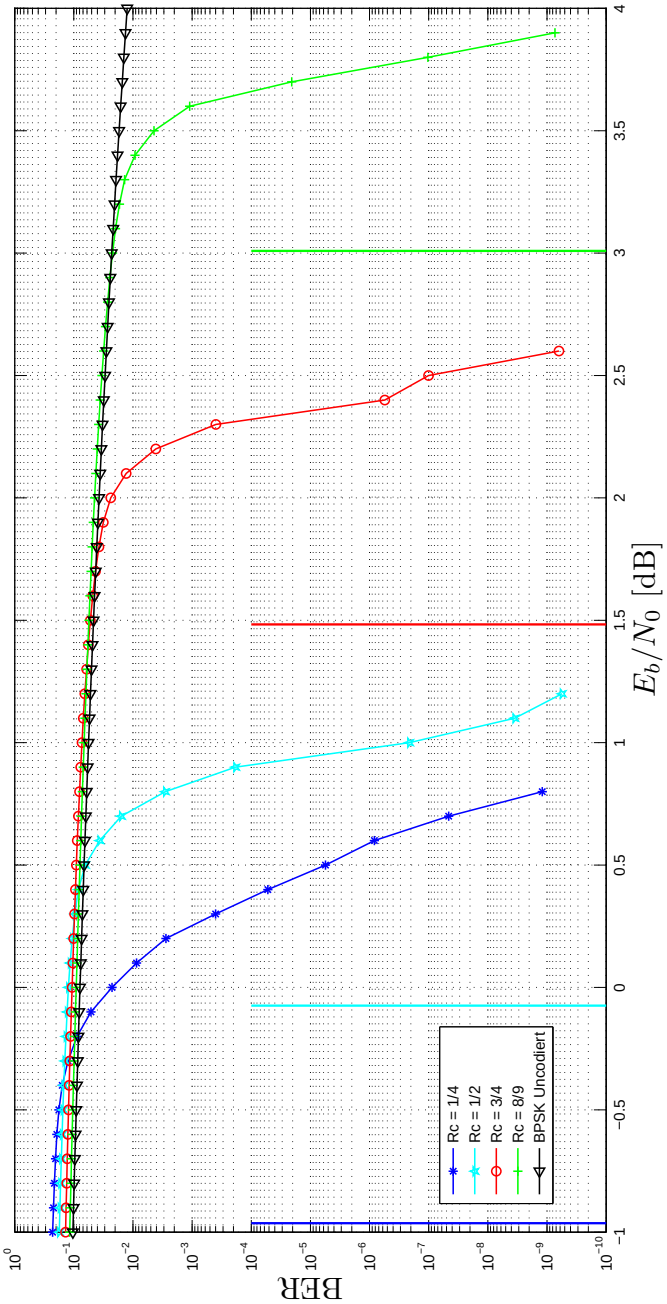

Abbildung 7.4: Darstellung von verschiedenen Coderaten und der dazugehörigen SHANNON-Grenzen.

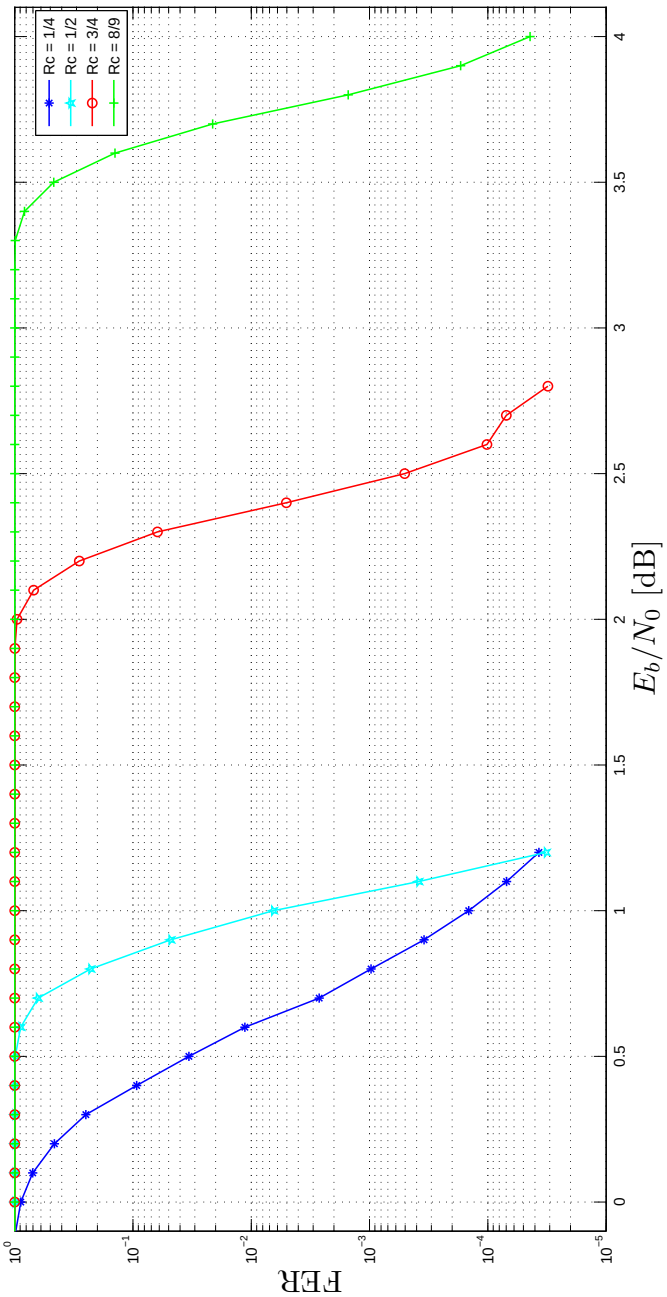

Abbildung 7.5: FER für verschiedene Coderaten.

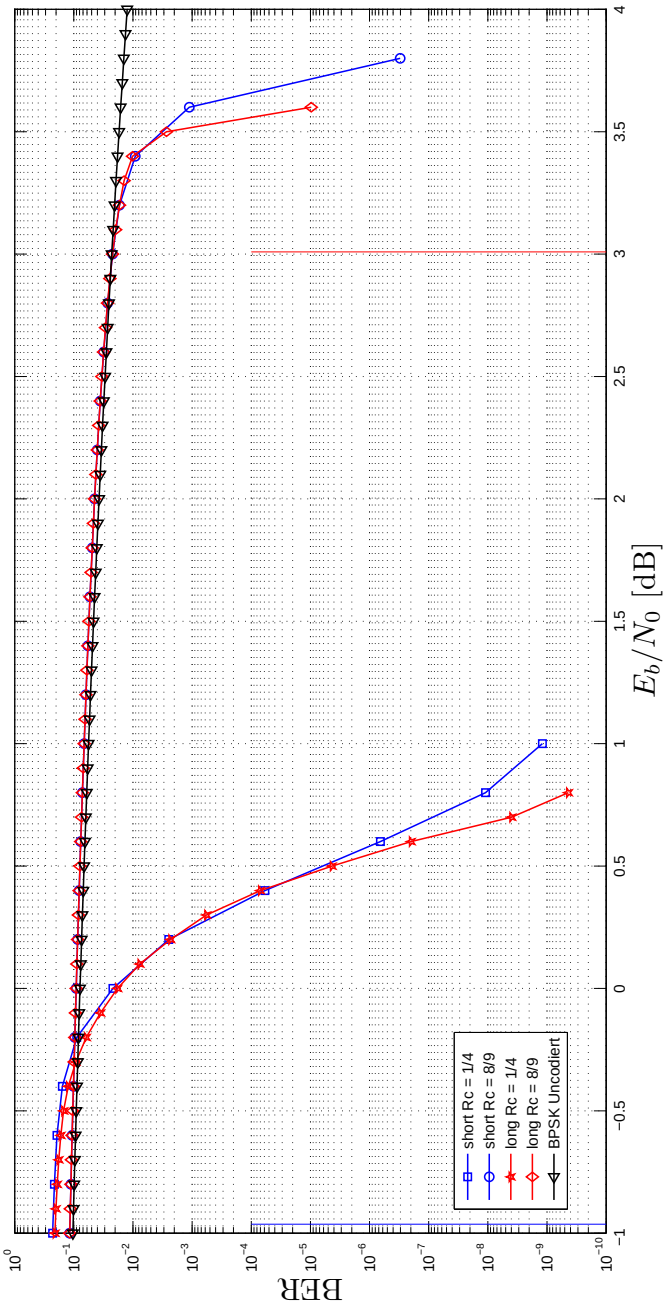

Abbildung 7.6: BER-Vergleich für Blocklängen von 16200 Bits bzw. 64800 Bits.

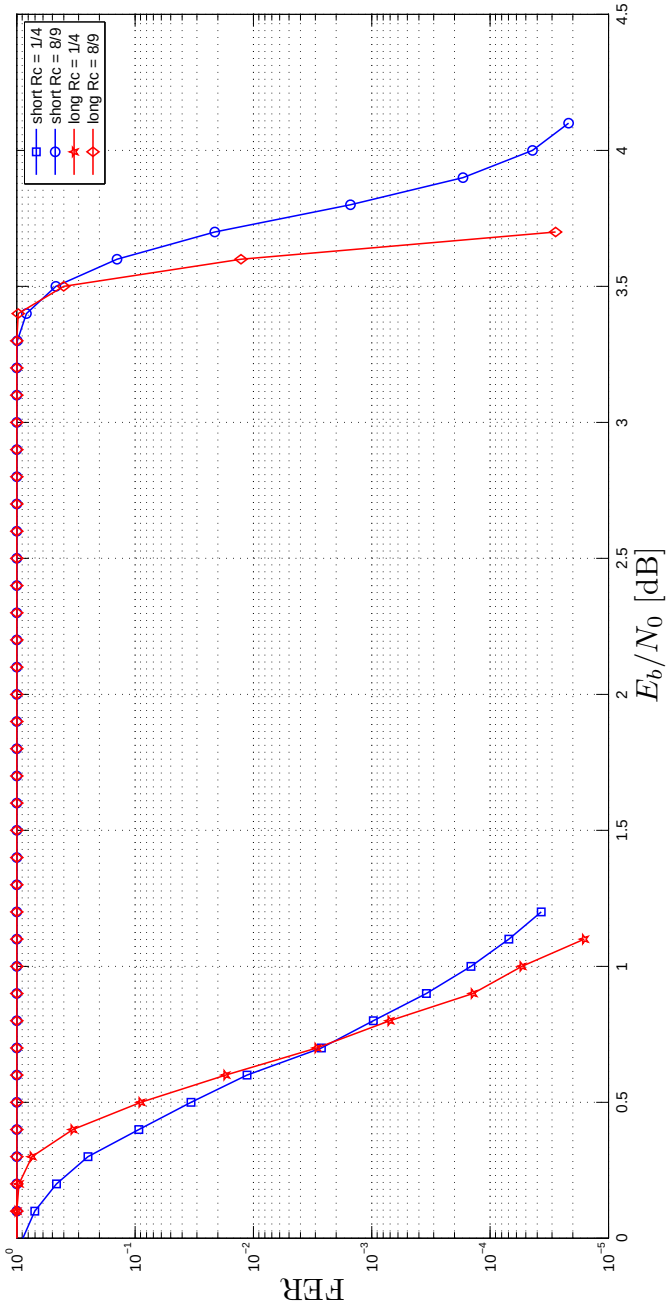

Abbildung 7.7: FER Vergleich mit einer Blocklänge von 16200 Bits bzw. 64800 Bits.

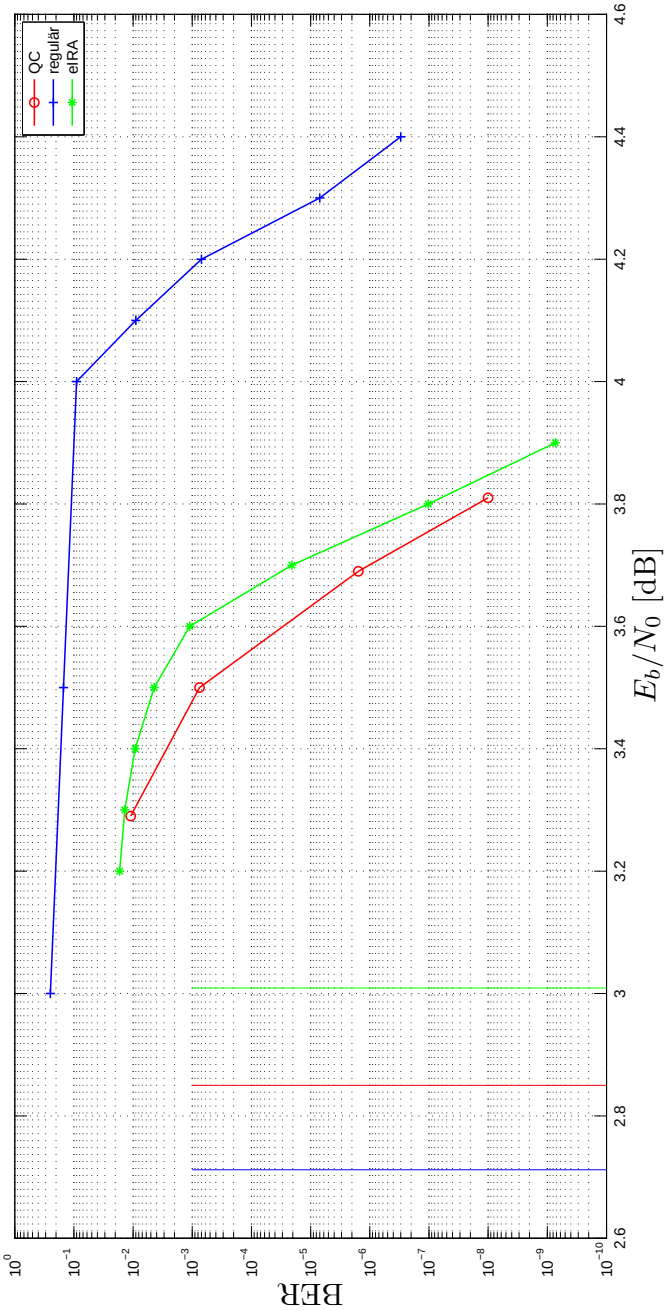

Abbildung 7.8: Gegenüberstellung unterschiedlicher LDPC-Codes.

Kapitel 8

Schlussbemerkungen

Durch Turbo-Codes und Low-Density Parity-Check (LDPC)-Codes wurde das Zeitalter iterativ decodierbarer Codes eingeläutet. Gleichzeitig sind LDPC-Codes inzwischen Teil einiger wichtiger Kommunikationsstandards geworden. In dieser Masterarbeit wurden die Hauptaspekte von LDPC-Codes genauer beleuchtet. Nach einem Überblick zur Kanalcodierung und der Betrachtung von linearen Blockcodes beschäftigte sich diese Arbeit allgemein mit LDPC-Codes, ihrer Repräsentation durch Tanner-Graphen (TG), ihrer regulären bzw. irregulären Natur, den ihren Vor- und Nachteilen gegenüber Turbo-Codes. LDPC-Codes können über Hard- sowie Soft-Decision Algorithmen decodiert werden. Von beiden Arten gibt es mehrere komplexitätsreduzierte Varianten, denen ein gewisses Maß an Leistungsverlust anhaftet. Die Untersuchung dieser Algorithmen geht von einem zyklusfreiem TG aus. Konstruktionsmethoden für reguläre Codes sind vielfältig und kommen aus vielen algebraischen Teilgebieten. Ziel dieser Konstruktionsmethoden ist es, den Kontrollmatrizen eine gewisse Struktur zu geben, um so den größten Nachteil von LDPC-Codes, nämlich den der aufwändigen Codierung, zu beseitigen. Irreguläre LDPC-Codes wurden aus regulären entwickelt und sind im Allgemeinen leistungsfähiger als die regulären Verfahren. Als praktischer Teil dieser Arbeit wurden die LDPC-Codes des DVB-S2 Standards implementiert, simuliert und mit der Literatur verglichen.

Abkürzungsverzeichnis

ARQ	Automatic Repeat reQuest.
AWGN	Additive White Gaussian Noise.
BCH	Bose Chaudhuri Hocquenghem.
BER	Bit Error Rate.
BF	Bit-Flipping.
BIBD	Balanced Incomplete Block Designs.
BPSK	Binary Phase-Shift Keying.
CMM	Convergence of Mean Magnitude.
DE	Density Evolution.
DVB-S	Digital Video Broadcast - Satellite.
DVB-S2	Digital Video Broadcast - Satellite Second Generation.
EG	euklidische Geometrie.
eIRA	extendend Irregular Repeat-Accumulate.
FEC	Forward Error Correction.
FER	Frame Error Rate.
HDD	Hard-Decision Decodierung.
IMWBF	Improved Modified Weighted Bit-Flipping.
IRA	Irregular Repeat-Accumulate.
LDPC	Low-Density Parity-Check.
LLR	Log-Likelihood Ratio.
MAP	Maximum-a-Posteriori.
MIT	Massachusetts Institute of Technology.
ML	Maximum-Likelihood.

MLG	Majority-Logic.
MWBF	Modified Weighted Bit-Flipping.
NSPC	Number of Satisfied Parity-Check Constraints.
OSMLD	One-Step Majority-Logic Decodeable.
PEG	Progressive Edge Growth.
PG	projektive Geometrie.
QC	Quasi-Cyclic.
RA	Repeat-Accumulate.
RS	Reed-Solomon.
SDD	Soft-Decision Decodierung.
SNR	Signalrauschabstande (Signal-to-Noise Ratio).
SPA	Sum-Product Alogrithm.
TG	TANNER-Graph.
VNR	Variable Node Reliability.
WBF	Weighted Bit-Flipping.

Symbolverzeichnis

C	Kanalkapazität.
E_b	Energie pro Bit.
$E_{i,j}$	Qualität der Nachrichten welche von Kontrollknoten ausgehen.
E_s	Energie pro Symbol.
H	Nachrichtengehalt oder Entropie.
$H(x\|y)$	Äquivokation.
$H(y\|x)$	Irrelevanz.
$M_{i,j}$	Qualität der Nachrichten welche von Bitknoten ausgehen.
N	Rauschleistung.
N_0	Spektralen Rauschleistungsdichte.
R_c	Coderate.
R_j	A-priori-Wahrscheinlichkeit des Kanals.
R_b	Bitrate.
S	Signalleistung.
T	Transinformation.
W	Bandbreite.
$\mathbf{A}$	Zirkulante Permutationsmatrix.
$\mathbf{E}$	Einheitsmatrix.
$\mathbf{G}$	Generatormatrix.
$\mathbf{H}$	Kontrollmatrix.
$\mathbf{P}$	Prüfmatrix.
η_s	Spektrale Effizienz.
γ	Spaltengewicht von $\mathbf{H}$.
$\mathrm{GF}(q)$	Galois-Feld.
ρ	Zeilengewicht von $\mathbf{H}$.
d_{min}	Mindest-HAMMING-Distanz.
k	Informationstellen.
m	Paritätsstellen.
n	Codewortstellen.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] *Near Shannon limit error-correcting coding and decoding: Turbo-codes*, vol. 2, 1993. [Online]. Available: http://dx.doi.org/10.1109/ICC.1993.397441

[2] D. J. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, pp. 1645–1646, 1996.

[3] R. G. Gallager, "Low-density parity-check codes," Ph.D. dissertation, Massachusetts Institute of Technology, 1963.

[4] R. Tanner, "A recursive approach to low complexity codes," *Information Theory, IEEE Transactions on*, vol. 27, no. 5, pp. 533 – 547, Sep. 1981.

[5] M. Werner, *Information und Codierung, Grundlagen und Anwendungen*, 2nd ed. Vieweg+Teubner, 2008.

[6] C. E. Shannon, "A mathematical theory of communication," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 5, pp. 3–55, Jan. 2001. [Online]. Available: http://doi.acm.org/10.1145/584091.584093

[7] A. Neubauer, *Informationstheorie und Quellencodierung*, 1st ed. J. Schlembach Fachverlag, 2006.

[8] B. Sklar, *Digital Communications: Fundamentals and Applications*, 2nd ed. Prentice-Hall, 1988.

[9] R. W. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[10] T. Strutz, "Low-Density-Parity-Check-Codes, Eine Einführung," 2010. [Online]. Available: http://www1.hft-leipzig.de/strutz/Kanalcodierung/ldpc_tutorial.pdf

[11] W. Reidler, G. Petter, and W. Werner, "Skriptum Informationstheorie und Codierung," 1999.

[12] M. Bossert, *Channel Coding for Telecommunications*, 1st ed. John Wiley & Sons, 1999.

[13] D. J. C. MacKay, "Good codes based on very sparse matrices," in *Cryptography and Coding, Proc. 5th IMA Conf. on.* London, UK: Springer-Verlag, 1995, pp. 100–111. [Online]. Available: http://portal.acm.org/citation.cfm?id=647992.742108

[14] ——, "Good error-correcting codes based on very sparse matrices," *Information Theory, IEEE Transactions on*, vol. 45, no. 2, pp. 399 –431, Mar. 1999.

[15] S. J. Johnson, *Iterative Error Correction: Turbo, Low-Density Parity-Check and Repeat-Accumulate Codes.* Cambridge University Press, Jan. 2010.

[16] J. Xu, L. Chen, L. Zeng, L. Lan, and S. Lin, "Construction of low-density parity-check codes by superposition," *Communications, IEEE Transactions on*, vol. 53, no. 2, pp. 243 – 251, Feb. 2005.

[17] W. E. Ryan, "An introduction to LDPC codes," 2003. [Online]. Available: http://www.telecom.tuc.gr/~alex/papers/ryan.pdf

[18] D. J. Costello, "An introduction to low-density parity check codes," 2009. [Online]. Available: http://www.itsoc.org/conferences/past-schools/2009-school-of-it/lecture-files/Costello-3.pdf

[19] M. Luby, M. Mitzenmacher, A. Shokrollah, and D. Spielman, "Analysis of low density codes and improved designs using irregular graphs," in *Theory of Computing, Proc. of the 30th annual ACM symposium on*, 1998, pp. 249–258. [Online]. Available: http://doi.acm.org/10.1145/276698.276756

[20] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, "Improved low-density parity-check codes using irregular graphs," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 585 –598, Feb. 2001.

[21] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 619 –637, Feb. 2001.

[22] T. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 638 –656, Feb. 2001.

[23] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Prentice-Hall, 2004.

[24] T. Richardson, "Error floors of ldpc codes," *Commun. Control, and Comput., in Proc. 41st Annual Allerton Conf. on*, pp. 1426–1435, 2003.

[25] J. Chen, R. Tanner, J. Zhang, and M. Fossorier, "Construction of irregular LDPC codes by quasi-cyclic extension," *Information Theory, IEEE Transactions on*, vol. 53, no. 4, pp. 1479 –1483, Apr. 2007.

[26] R. H. Morelos-Zaragoza, *The Art of Error Correcting Coding*, 2nd ed. John Wiley & Sons, 2006.

[27] F. R. Kschischang, "Codes defined on graphs," *Communications Magazine, IEEE*, vol. 41, no. 8, pp. 118–125, 2003. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1222727

[28] J. B. Huber, "Grundlagen der Wahrscheinlichkeitsrechnung für iterative Decodierverfahren," *Elektrotechnik und Informationstechnik*, vol. 119, no. 11, Nov. 2002. [Online]. Available: http://www.lit.lnt.de/papers/E_u_I_paper.pdf

[29] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *Information Theory, IEEE Transactions on*, vol. 47, pp. 498–519, 1998.

[30] J. C. Moreira and P. G. Farrell, *Essentials of Error-Control Coding*. Wiley, 2006.

[31] S. R. G. Papaharalobos, "Efficient iterative decoding algorithms for turbo and low-density parity-check codes," Ph.D. dissertation, School of Electronics and Physical Sciences University of Surrey, 2005.

[32] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *Communications, IEEE Transactions on*, vol. 53, no. 8, pp. 1288 – 1299, Aug. 2005.

[33] Y. Kou, S. Lin, and M. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovery and new results," *Information Theory, IEEE Transactions on*, vol. 47, no. 7, pp. 2711 –2736, Nov. 2001.

[34] J. Zhang and M. Fossorier, "A modified weighted bit-flipping decoding of low-density parity-check codes," *Communications Letters, IEEE*, vol. 8, no. 3, pp. 165 – 167, Mar. 2004.

[35] M. Jiang, C. Zhao, Z. Shi, and Y. Chen, "An improvement on the modified weighted bit flipping decoding algorithm for LDPC codes," *Communications Letters, IEEE*, vol. 9, no. 9, pp. 814 – 816, Sep. 2005.

[36] A. McGregor and O. Milenkovic, "On the hardness of approximating stopping and trapping sets in LDPC codes," in *Information Theory Workshop, 2007. ITW '07. IEEE*, Sep. 2007, pp. 248 –253.

[37] J. Li, X.-H. You, and J. Li, "Early stopping for LDPC decoding: convergence of mean magnitude (cmm)," *Communications Letters, IEEE*, vol. 10, no. 9, pp. 667 –669, Sep. 2006.

[38] C. Zhai, C. Xu, X. Zhang, and Y. Wang, "Research on stopping criterion for irregular LDPC codes decoding," in *Wireless Communications and Mobile Computing: Connecting the World Wirelessly, Proc. of the 2009 Int. Conf. on*, 2009, pp. 1321–1324. [Online]. Available: http://doi.acm.org/10.1145/1582379.1582668

[39] D. Shin, K. Heo, S. Oh, and J. Ha, "A stopping criterion for low-density parity-check codes," in *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th*, Apr. 2007, pp. 1529 –1533.

[40] F. Kienle and N. Wehn, "Low complexity stopping criterion for LDPC code decoders," in *Vehicular Technology Conference, 2005 IEEE 61st*, vol. 1, Jun. 2005, pp. 606 – 609 Vol. 1.

[41] G. Albertazzi, M. Chiani, G. E. Corazza, A. Duverdier, H. Ernst, W. Gappmair, G. Liva, and S. Papaharalabos, Eds., *Digital Satellite Communications*, 1st ed. Springer, 2007, ch. Forward Error Correction, pp. 116–174.

[42] J. Campello, D. Modha, and S. Rajagopalan, "Designing ldpc codes using bit-filling," in *Proc. IEEE Int. Conference Communications*, Jun. 2001, pp. 55 –59 vol.1.

[43] X.-Y. Hu, E. Eleftheriou, and D. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *Information Theory, IEEE Transactions on*, vol. 51, no. 1, pp. 386 –398, Jan. 2005.

[44] S. Johnson and S. Weller, "Codes for iterative decoding from partial geometries," *Communications, IEEE Transactions on*, vol. 52, no. 2, pp. 236 – 243, Feb. 2004.

[45] O. Milenkovic and S. Laendner, "Analysis of the cycle-structure of LDPC codes based on latin squares," in *Conference Communications, Proc. IEEE Int. on*, Jun. 2004, pp. 777 – 781 Vol.2.

[46] E. Eleftheriou and S. Olcer, "Low-density parity-check codes for digital subscriber lines," in *Communications, Proc. IEEE Int. Conf. on*, 2002, pp. 1752 – 1757 vol.3.

[47] M. Fossorier, "Quasicyclic low-density parity-check codes from circulant permutation matrices," *Information Theory, IEEE Transactions on*, vol. 50, no. 8, pp. 1788 – 1793, Aug. 2004.

[48] C.-M. Huang, J.-F. Huang, and C.-C. Yang, "Construction of quasicyclic LDPC codes from quadratic congruences," *Communications Letters, IEEE*, vol. 12, no. 4, pp. 313 –315, Apr. 2008.

[49] J. Thorpe, K. Andrews, and S. Dolinar, "Methodologies for designing ldpc codes using protographs and circulants," in *Information Theory, Proc. Int. Symp. on*, Jul. 2004, p. 238.

[50] C. Sae-Young, G. Forney, T. T., Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 db of the Shannon limit," *Communications Letters, IEEE*, vol. 5, no. 2, pp. 58 –60, Feb. 2001.

[51] L. Chen, J. Xu, I. Djurdjevic, and S. Lin, "Near-Shannon-limit quasicyclic low-density parity-check codes," *Communications, IEEE Transactions on*, vol. 52, no. 7, pp. 1038 – 1042, Jul. 2004.

[52] E.T.S.I., "Digital video broadcasting; user guidelines for the second generation system for broadcasting, interactive services, news gathering and other broadband satellite applications (DVB-S2)," 2005. [Online]. Available: http://www.etsi.org/deliver/etsi_tr/102300_102399/102376/01.01.01_60/tr_102376v010101p.pdf

[53] A. Morello and V. Mignone, "DVB-S2: The second generation standard for satellite broad-band services," *Proceedings of the IEEE*, vol. 94, no. 1, pp. 210 –227, Jan. 2006.

[54] J. G. Proakis and S. Masoud, *Grundlagen der Kommunikationstechnik*, 2nd ed. Pearson Studium, 2004.

[55] P. Sweeney, *Error Control Coding*, 1st ed. John Wiley & Sons, Ltd, 2002.

[56] E.T.S.I., "Digital video broadcasting; second generation framing structure, channel coding and modulation systems for broadcasting," 2009. [Online]. Available: http://www.etsi.org/deliver/etsi_en/302300_302399/302307/01.02.01_40/en_302307v010201o.pdf

[57] M. C. Valenti, S. Cheng, and R. I. Seshadri, "Digital video broadcasting," 2006. [Online]. Available: http://www.csee.wvu.edu/~mvalenti/documents/DVBChapter.pdf

[58] Y. Xiao and K. Kim, "Alternative good ldpc codes for DVB-S2," in *Signal Processing, Proc. 9th Int. Conf. on*, Oct. 2008, pp. 1959 –1962.