Unterstützende Materialien zur Vorlesung

Verfahren zur Kanalcodierung – Teil 5

Prof. Dr. Bernd Friedrichs
KIT CEL

Inhalt

- Motivation Galoisfelder, Kurzeinführung Galoisfelder mit Beispielen GF(4), GF(8) und GF(16)
- Spektraltransformation auf Galoisfeldern
- RS-Codes (Definition und BER-Kurven)
- BCH-Codes (Definition und BER-Kurven)
- Anwendung Compact Disc Digital Audio (Vorbereitungen: Ausfallkorrektur, Verkürzung, Interleaving, Verkettung)
- Faltungscodes: Definition und Schieberegister-Beschreibung
- Polynombeschreibung
- Spezielle Codeklassen: Terminierte und punktierte Faltungscodes

Alle bisherigen Codes wurden als binär betrachtet, dazu reichten die bisherigen Kenntnisse zu Galoisfeldern GF(q) = F_q mit q = p = prim und insbesondere q=2 aus (simples Rechnen modulo p).

Leistungsfähige Codes (und insbesondere Codes zur Bündelfehler-Korrektur) erfordern jedoch q>2, typischerweise q = 2^8 = 256, d.h. alle Info- und Codesymbole sind 256-stufig (einem Byte entsprechend).

Außerdem sind weitere Konzepte wie Spektraltransformationen auf Galoisfeldern erforderlich um RS- und BCH-Codes übersichtlich einzuführen.

Bevor ein kurzer Einblick in die Welt der Galoisfelder erfolgt, wird an einem Beispiel erklärt warum höherstufiges q die Voraussetzung für leistungsfähige Bündelfehler-korrigierende Codes ist.

Während Galoisfelder und RS/BCH-Codes im Buch ausführlich behandelt werden, konzentriert sich die Vorlesung auf die Anwendung: welchen Schutz/Verbesserung kann man mit diesen Codes bei der digitalen Nachrichtenübertragung überhaupt erreichen und was ist dabei zu beachten?

- *Viele System-Ingenieure werden mit diesen prinzipiellen Fragen konfrontiert,*
- *aber nur ganz wenige werden einen RS/BCH-Decoder selber bauen/programmieren.*

Deshalb werden die Decoder-Algorithmen und Details der Galoisfelder nur im Buch aber nicht in der Vorlesung thematisiert.

Es existieren (Begründung in §7):

Korrekturfähigkeit (pro Wort)

BCH-Code $(63,45,7)_2$ $\Rightarrow$ t = 3 Bits

RS-Code $(15,11,5)_{16} \Leftrightarrow (60,44,5)_2$ $\Rightarrow$ t = 2 Symbole

Jedes 16-stufige RS-Symbol wird durch 4 Bits repräsentiert.
In der binären Interpretation verlängern sich n und k um den Faktor 4, aber d_{min} bleibt gleich

Beispiele für Fehlermuster:

e = 0000 1111 1111 0000 ... BCH versagt / RS korrigiert

e = 0001 0100 1000 0000 ... BCH korrigiert / RS versagt

Also

BCH korrgiert 3 Einzelfehler oder 1 Bündelfehler bis zur Länge 3

RS korrgiert 2 Einzelfehler oder 1 Bündelfehler bis zur Länge 5 (garantiert) oder 8 (max)

abhängig von der Position des Bündelfehlers relativ zu den 4er-Bitgruppen

Fazit

BCH besser als RS bei Einzelfehlern

RS besser als BCH bei Bündelfehlern

Der Vergleich beider Codes ist fair, denn Coderate und Blocklänge (in binärer Interpretation) sind ähnlich.

Ein Galoisfeld $\mathbb{F}_q$ ist nach Abschnitt 3.1 ein endlicher Körper mit q Elementen. Galoisfelder existieren nur für $q = p^m$ mit einer Primzahl p und einer natürlichen Zahl m. Praktisch ist fast nur der Fall $p = 2$ und $q = 2^m$ interessant.

Für $m = 1$ kann $\mathbb{F}_p$ als aus den natürlichen Zahlen 0 bis $p - 1$ bestehend aufgefaßt werden:

$$\mathbb{F}_p = \{0, 1, 2, \ldots, p-1\}$$
Addition und Multiplikation modulo p. (6.1.1)

In Beispiel 3.1 wurde schon gezeigt, daß $\mathbb{F}_4$ so nicht konstruiert werden kann. Für den Fall $m > 1$ ist eine andere Methode erforderlich, die in diesem Abschnitt am Beispiel $\mathbb{F}_4$ demonstriert wird.

Nach dem Einführungsbeispiel F_4=GF(4) folgt eine Kurzdarstellung des allgemeinen Falls F_q=GF(p^m) und danach wird das dann nochmal an den Beispielen F_8=GF(2^3) und F_{16}=GF(2^4) demonstriert.

Darstellung (1): Setze $\mathbb{F}_4 = \{0, 1, z, 1+z\}$. Dabei ist z eine abstrakte Größe mit der Eigenschaft $z^2 + z + 1 = 0$ und es wird modulo 2 gerechnet, d.h. es gilt beispielsweise

$$1 + 1 = 0, \quad z + z = 0, \quad z^2 + z^2 = 0, \quad z^2 = 1 + z.$$

Damit ergeben sich folgende Verknüpfungstafeln für Addition und Multiplikation:

$+$	0	1	z	$1+z$
0	0	1	z	$1+z$
1	1	0	$1+z$	z
z	z	$1+z$	0	1
$1+z$	$1+z$	z	1	0

$\cdot$	0	1	z	$1+z$
0	0	0	0	0
1	0	1	z	$1+z$
z	0	z	$1+z$	1
$1+z$	0	$1+z$	1	z

(6.1.2)

Hierbei gilt beispielsweise $(1 + z)^{-1} = z$ und $z^{-1} = 1 + z$. Man kann alle Eigenschaften eines Körpers aus Definition 3.1 verifizieren.

Darstellung (2): Setze $\mathbb{F}_4 = \{0, 1, z, z^2\}$, d.h. $\mathbb{F}_4$ besteht aus 0 und den z-Potenzen von z^0 bis z^{q-2}. Es ergeben sich die gleichen Verknüpfungstafeln wie bei Darstellung (1), wobei lediglich $1 + z$ durch z^2 ersetzt wird:

$+$	0	1	z	z^2
0	0	1	z	z^2
1	1	0	z^2	z
z	z	z^2	0	1
z^2	z^2	z	1	0

$\cdot$	0	1	z	z^2
0	0	0	0	0
1	0	1	z	z^2
z	0	z	z^2	1
z^2	0	z^2	1	z

(6.1.3)

Beispielsweise gilt $z^3 = z \cdot z^2 = z(z+1) = z^2 + z = 1$ und $z^{-2} = (z^2)^{-1} = z$. Allgemein entspricht die Multiplikation in $\mathbb{F}_4$ der Exponenten-Addition modulo 3 in den natürlichen Zahlen:

$$z^i \cdot z^r = z^{i+r \text{ modulo } (q-1)}. \tag{6.1.4}$$

Da alle z-Exponenten modulo $(q-1)$ betrachtet werden können, kann $\mathbb{F}_4$ auch als aus 0 und der Menge aller z-Potenzen bestehend aufgefaßt werden. ■

Darstellung (3): Mit der direkten Entsprechung $k_0 + k_1 z \cong (k_0, k_1)$ kann auch $\mathbb{F}_4 = \{00, 10, 01, 11\}$ gesetzt werden. Der Darstellung (1) entsprechen dann folgende Verknüpfungstafeln für binäre Vektoren der Länge 2:

$+$	00	10	01	11
00	00	10	01	11
10	10	00	11	01
01	01	11	00	10
11	11	01	10	00

$\cdot$	00	10	01	11
00	00	00	00	00
10	00	10	01	11
01	00	01	11	10
11	00	11	10	01

(6.1.5)

Die Addition in $\mathbb{F}_4$ entspricht hier der komponentenweisen Vektoraddition. ■

Es wird jetzt die Darstellung (1) oder (2) vorausgesetzt. Durch elementare Rechnung ergibt sich

$$(x - z)(x - z^2) \;=\; x^2 + x(z^2 + z) + z \cdot z^2 \;=\; x^2 + x + 1 \;=\; p(x). \qquad (6.1.9)$$

Das Polynom $p(x)$ ist irreduzibel über $\mathbb{F}_2$, aber in $\mathbb{F}_4$ zerfällt $p(x)$ vollständig in Linearfaktoren und mit der ersten Nullstelle z liegt auch die weitere Nullstelle z^2 in $\mathbb{F}_4$. Weiter gilt

$$(x - z^0)(x - z^1)(x - z^2) \;=\; (x + 1)(x^2 + x + 1) \;=\; x^3 - 1, \qquad (6.1.10)$$

d.h. das Produkt über alle Linearfaktoren zu den z-Potenzen ergibt das Polynom $x^{q-1} - 1$.

Satz 6.1. *Zu jeder Primzahl p und jeder natürlichen Zahl m existiert ein* primitives Polynom *$p(x) \in \mathbb{F}_p[x]_m$ vom Grad m mit Koeffizienten aus $\mathbb{F}_p$, das irreduzibel über $\mathbb{F}_p$ ist und folgende Eigenschaft erfüllt:*

Für jede abstrakte Nullstelle z von $p(x)$, also $p(z) = 0$, sind die $n = p^m - 1$ Elemente $z^1, z^2, \ldots, z^{n-1}, z^n$ alle verschieden mit $z^n = z^0 = 1$. Die Nullstelle z heißt primitives Element *bzw. erzeugendes Element für die multiplikative Gruppe von $\mathbb{F}_{p^m}$. Für $\mathbb{F}_{p^m}$ ist neben der bereits aus (6.2.2) bzw. (6.2.6) bekannten* Komponentendarstellung *auch die* Exponentendarstellung *(6.2.7) möglich:*

$$\mathbb{F}_{p^m} = \left\{ \sum_{i=0}^{m-1} a_i z^i \;\middle|\; a_0, \ldots, a_{m-1} \in \mathbb{F}_p \right\} \qquad \text{(6.2.6)}$$

Komponenten-darstellung

$$= \left\{ 0, z, z^2, z^3, \ldots, z^{n-1}, z^n = z^0 = 1 \right\}. \qquad \text{(6.2.7)}$$

Exponenten-darstellung

Dabei wird modulo p und modulo $p(z)$ gerechnet. Bis auf Isomorphien (Umbenennungen) gibt es zu jedem $q = p^m$ nur ein einziges eindeutig bestimmtes Galoisfeld und für alle anderen Werte von q kann prinzipiell kein endlicher Körper existieren.

Tabelle 6.1. Primitive Polynome über $\mathbb{F}_2$ bis zum Grad 24

m	$p(x)$	m	$p(x)$
1	$x+1$	13	$x^{13}+x^4+x^3+x+1$
2	x^2+x+1	14	$x^{14}+x^{10}+x^6+x+1$
3	x^3+x+1	15	$x^{15}+x+1$
4	x^4+x+1	16	$x^{16}+x^{12}+x^3+x+1$
5	x^5+x^2+1	17	$x^{17}+x^3+1$
6	x^6+x+1	18	$x^{18}+x^7+1$
7	x^7+x^3+1	19	$x^{19}+x^5+x^2+x+1$
8	$x^8+x^4+x^3+x^2+1$	20	$x^{20}+x^3+1$
9	x^9+x^4+1	21	$x^{21}+x^2+1$
10	$x^{10}+x^3+1$	22	$x^{22}+x+1$
11	$x^{11}+x^2+1$	23	$x^{23}+x^5+1$
12	$x^{12}+x^6+x^4+x+1$	24	$x^{24}+x^7+x^2+x+1$

Beispiele die hier behandelt werden:

F_{256}=GF(2^8) ist der wichtigste Fall:

Satz 6.3. *Es sei $p(x) \in \mathbb{F}_p[x]_m$ ein primitives Polynom vom Grad m und z ein primitives Element für $\mathbb{F}_{p^m}$ und $n = p^m - 1$. Dann sind die Nullstellen von $p(x)$ alle verschieden und explizit durch die z-Potenzen wie folgt gegeben:*

$$p(x) = \prod_{i=0}^{m-1} (x - z^{p^i}). \tag{6.2.12}$$

Über $\mathbb{F}_p$ ist $p(x)$ also irreduzibel, während es über $\mathbb{F}_{p^m}$ vollständig in Linearfaktoren zerfällt. Auch das Polynom $x^n - 1$ zerfällt über $\mathbb{F}_{p^m}$ vollständig in Linearfaktoren, wobei die n Nullstellen genau $\mathbb{F}_{p^m} \setminus \{0\}$ ergeben:

$$x^n - 1 = \prod_{i=0}^{n-1} (x - z^i). \tag{6.2.13}$$

Entsprechend ergeben die Nullstellen von $x^q - x$ genau $\mathbb{F}_{p^m}$. Die z^i werden als n-te Einheitswurzeln, $x^n - 1$ als Kreisteilungspolynom und $\mathbb{F}_{p^m}$ als Kreisteilungskörper *oder* Zerfällungskörper *(splitting field) bezeichnet. Das primitive Polynom ist ein Teiler des Kreisteilungspolynoms.*

Ausblick:

Bei RS-Codes wird g(x) genau aus bestimmten Linearfaktoren ($x-z^i$) bestehen und h(x) dann aus den restlichen Linearfaktoren.

Bei BCH-Codes sind die Linearfaktoren so zu wählen dass g(x) Koeffizienten in GF(p) hat während bei den RS-Codes die Koeffizienten in GF(p^m) liegen dürfen.

Analogie zu den komplexen Zahlen:

z sei abstrakte Nullstelle von $p(x) = x^2 + 1$, also $z^2 + 1 = 0$

Dann ergibt sich $\mathbb{C}$ aus $\mathbb{R}$ so: $\mathbb{C} = \{k_0 + k_1 z \mid k_0, k_1 \in \mathbb{R}\}$

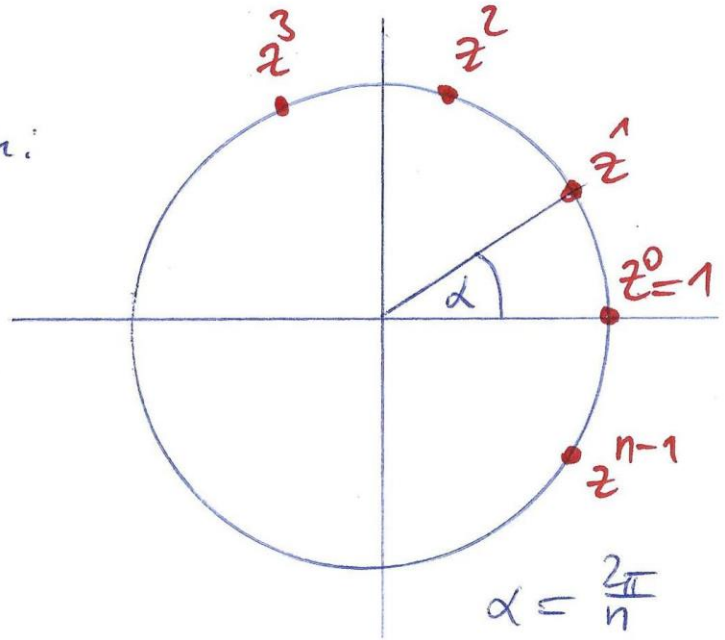
üblicherweise wird i oder j statt z verwendet

$x^n - 1$ zerfällt in $\mathbb{C}$ vollständig in Linearfaktoren:

mit $z = \exp(j\frac{2\pi}{n}) = \cos(\frac{2\pi}{n}) + j\sin(\frac{2\pi}{n})$ gilt

$$x^n - 1 = \prod_{r=0}^{n-1} (x - z^r)$$

Deshalb 'Kreisteilungspolynom'



Während hier für die beiden verschiedenen Terme j und exp(j2π/n) jeweils der gleiche Buchstabe verwendet wurde, hat z bei Galoisfeldern keine unterschiedlichen Bedeutungen und es gilt n=q-1.

Beispiel 6.4. $\mathbb{F}_8 = \mathbb{F}_{2^3}$ mit $n = 7$. Es gibt $\varphi(7) = 6$ primitive Elemente und $\varphi(7)/3 = 2$ primitive Polynome. Nach Tabelle 6.1 wird $p(x) = x^3+x+1$ gewählt. Für das primitives Element z gilt also $z^3 + z + 1 = 0$ bzw. $z^3 = z + 1$. Durch Potenzierung von z ergibt sich:

									k_0	k_1	k_2
z^1	$=$					$=$	z		0	1	0
z^2	$=$					$=$	z^2		0	0	1
z^3	$=$	$z+1$				$=$	$z+1$		1	1	0
z^4	$=$	z^2+z				$=$	z^2+z		0	1	1
z^5	$=$	z^3+z^2	$=$	$(z+1)+z^2$		$=$	z^2+z+1		1	1	1
z^6	$=$	z^3+z^2+z	$=$	$(z+1)+z^2+z$		$=$	z^2+1		1	0	1
z^7	$=$	z^3+z	$=$	$(z+1)+z$		$=$	1	$= z^0.$	1	0	0

Somit gilt für die Komponenten- und Exponentendarstellung:

$$\begin{aligned}\mathbb{F}_8 &= \{0, 1, z, 1+z, z^2, 1+z^2, z+z^2, 1+z+z^2\} \\ &= \{k_0 + k_1 z + k_2 z^2 \mid k_0, k_1, k_2 \in \mathbb{F}_2\} \\ &= \{0, 1, z, z^2, z^3, z^4, z^5, z^6\}.\end{aligned}$$

Verknüpfungstafeln für die Komponentendarstellung:

+	000	100	010	001	110	011	111	101
000	000	100	010	001	110	011	111	101
100	100	000	110	101	010	111	011	001
010	010	110	000	011	100	001	101	111
001	001	101	011	000	111	010	110	100
110	110	010	100	111	000	101	001	011
011	011	111	001	010	101	000	100	110
111	111	011	101	110	001	100	000	010
101	101	001	111	100	011	110	010	000

Einfaches Addieren: komponentenweise modulo 2

·	000	100	010	001	110	011	111	101
000	000	000	000	000	000	000	000	000
100	000	100	010	001	110	011	111	101
010	000	010	001	110	011	111	101	100
001	000	001	110	011	111	101	100	010
110	000	110	011	111	101	100	010	001
011	000	011	111	101	100	010	001	110
111	000	111	101	100	010	001	110	011
101	000	101	100	010	001	110	011	111

Verknüpfungstafeln für die Exponentendarstellung:

Einfaches Multiplizieren: addiere Exponenten modulo n-1=7

+	0	1	z	z^2	z^3	z^4	z^5	z^6
0	0	1	z	z^2	z^3	z^4	z^5	z^6
1	1	0	z^3	z^6	z	z^5	z^4	z^2
z	z	z^3	0	z^4	1	z^2	z^6	z^5
z^2	z^2	z^6	z^4	0	z^5	z	z^3	1
z^3	z^3	z	1	z^5	0	z^6	z^2	z^4
z^4	z^4	z^5	z^2	z	z^6	0	1	z^3
z^5	z^5	z^4	z^6	z^3	z^2	1	0	z
z^6	z^6	z^2	z^5	1	z^4	z^3	z	0

·	0	1	z	z^2	z^3	z^4	z^5	z^6
0	0	0	0	0	0	0	0	0
1	0	1	z	z^2	z^3	z^4	z^5	z^6
z	0	z	z^2	z^3	z^4	z^5	z^6	1
z^2	0	z^2	z^3	z^4	z^5	z^6	1	z
z^3	0	z^3	z^4	z^5	z^6	1	z	z^2
z^4	0	z^4	z^5	z^6	1	z	z^2	z^3
z^5	0	z^5	z^6	1	z	z^2	z^3	z^4
z^6	0	z^6	1	z	z^2	z^3	z^4	z^5

Nach (6.2.12) hat $p(x)$ die Nullstellen z, z^2, z^4 und somit gilt

$$(x-z)(x-z^2)(x-z^4) = x^3 + x + 1 = p(x).$$

Nach (6.2.13) gilt:

$$x^7 - 1 = (x-1)(x-z)(x-z^2)(x-z^3)(x-z^4)(x-z^5)(x-z^6).$$

Das ist einfacher in folgender Form nachzurechnen:

$$x^7 - 1 = \underbrace{\underbrace{(x-1)}_{x+1} \cdot \underbrace{(x-z)(x-z^2)(x-z^4)}_{x^3+x+1} \cdot \underbrace{(x-z^3)(x-z^5)(x-z^6)}_{x^3+x^2+1}}_{x^7+1}.$$

Damit erscheint das primitive Polynom als irreduzibler Faktor im Kreisteilungspolynom. Auch x^3+x^2+1 ist ein primitives Polynom.

Beispiel 6.5. $\mathbb{F}_{16} = \mathbb{F}_{2^4}$ mit $n = 15$. Es gibt $\varphi(15) = 8$ primitive Elemente und $\varphi(15)/4 = 2$ primitive Polynome. Nach Tabelle 6.1 wird $p(x) = x^4 + x + 1$ gewählt. Für das primitive Element z gilt also $z^4 + z + 1 = 0$ bzw. $z^4 = z + 1$. Durch Potenzierung von z ergibt sich:

$$
\begin{array}{lllll}
z^1 & = & & & = z \\
z^2 & = & & & = z^2 \\
z^3 & = & & & = z^3 \\
z^4 & = & z+1 & & = z+1 \\
z^5 & = & z^2+z & & = z^2+z \\
z^6 & = & z^3+z^2 & & = z^3+z^2 \\
z^7 & = & z^4+z^3 & = (z+1)+z^3 & = z^3+z+1 \\
z^8 & = & z^4+z^2+z & = (z+1)+z^2+z & = z^2+1 \\
z^9 & = & z^3+z & & = z^3+z \\
z^{10} & = & z^4+z^2 & = (z+1)+z^2 & = z^2+z+1 \\
z^{11} & = & z^3+z^2+z & & = z^3+z^2+z \\
z^{12} & = & z^4+z^3+z^2 & = (z+1)+z^3+z^2 & = z^3+z^2+z+1 \\
z^{13} & = & z^4+z^3+z^2+z & = (z+1)+z^3+z^2+z & = z^3+z^2+1 \\
z^{14} & = & z^4+z^3+z & = (z+1)+z^3+z & = z^3+1 \\
z^{15} & = & z^4+z & = (z+1)+z & = 1 = z^0.
\end{array}
$$

Somit gilt für die Komponenten- und Exponentendarstellung:

$$\begin{aligned}\mathbb{F}_{16} &= \{k_0 + k_1 z + k_2 z^2 + k_3 z^3 \mid k_0, k_1, k_2, k_3 \in \mathbb{F}_2\} \\ &= \{0, 1, z, z^2, z^3, z^4, z^5, z^6, z^7, z^8, z^9, z^{10}, z^{11}, z^{12}, z^{13}, z^{14}\}.\end{aligned}$$

Nach (6.2.12) hat $p(x)$ die Nullstellen z, z^2, z^4, z^8 und somit gilt

$$(x - z)(x - z^2)(x - z^4)(x - z^8) = x^4 + x + 1 = p(x).$$

Nach (6.2.13) gilt

$$x^{15} - 1 = (x - 1)(x - z)(x - z^2) \cdots (x - z^{12})(x - z^{13})(x - z^{14}).$$

Das Produkt von speziellen Gruppen von Linearfaktoren ergibt Polynome mit Koeffizienten aus GF(2).
Diese Polynome ergeben dann die eindeutige Zerlegung des Kreisteilungspolynoms über GF(2) in irreduzible Faktoren.
Über GF(16) zerfällt das Kreisteilungspolynom jedoch in 16 Linearfaktoren.
Diese Zusammenhänge sind wichtig für BCH-Codes, werden aber nur im Buch und nicht in der Vorlesung behandelt.

RS- und BCH-Codes können mit der diskreten Fouriertransformation (DFT) kompakt und übersichtlich beschrieben werden. Zur Zeit der Erfindung dieser Codes (um 1960) war das noch nicht üblich.

Vor Einführung der DFT auf GF(q) hier eine Erinnerung an die DFT auf den komplexen Zahlen

$$a = (a_0, \ldots, a_{n-1}) \quad \circ\!\!-\!\!\bullet \quad (A_0, \ldots, A_{n-1}) = A$$

(Zeit) (Frequenz)

$$A_i = \sum_{\mu=0}^{n-1} a_\mu \exp(j 2\pi \mu i/n) = \sum_{\mu=0}^{n-1} a_\mu z^{\mu i} = a(z^i)$$

$$a_i = \frac{1}{n} \sum_{\mu=0}^{n-1} A_\mu \exp(-j 2\pi \mu i/n) = \frac{1}{n} \sum_{\mu=0}^{n-1} A_\mu z^{-\mu i} = \frac{1}{n} A(z^{-i})$$

wobei $z = \exp(j 2\pi/n)$

Vorausgesetzt wird weiterhin ein Galoisfeld $\mathbb{F}_{p^m}$ mit einem primitiven Element z. Die nachfolgend definierte Transformation auf $\mathbb{F}_{p^m}$ mit der dadurch festgelegten Länge $n = p^m - 1 = q - 1$ entspricht sowohl formal wie von ihren Eigenschaften her der gewöhnlichen diskreten Fouriertransformation im Bereich der komplexen Zahlen.

Definition 6.4. *Betrachte zwei Polynome vom Grad* $\leq n-1 = p^m - 2$ *mit Koeffizienten aus* $\mathbb{F}_{p^m}$*:*

$$\boldsymbol{a} = (a_0, \ldots, a_{n-1}) \leftrightarrow a(x) = \sum_{\mu=0}^{n-1} a_\mu x^\mu,$$

$$\boldsymbol{A} = (A_0, \ldots, A_{n-1}) \leftrightarrow A(x) = \sum_{\mu=0}^{n-1} A_\mu x^\mu.$$

$A(x)$ *heißt* diskrete Fouriertransformierte (DFT) *von* $a(x)$ *mit der Schreibweise* $A = \mathrm{DFT}(a)$ *bzw.* $a(x)$ *heißt* inverse diskrete Fouriertransformierte (IDFT) *von* $A(x)$ *mit der Schreibweise* $\boldsymbol{a} = \mathrm{IDFT}(\boldsymbol{A})$*, wenn gilt* $(0 \leq i \leq n-1)$*:*

$$A_i = a(z^i) = \sum_{\mu=0}^{n-1} a_\mu z^{i\mu} \quad , \quad a_i = -A(z^{-i}) = -\sum_{\mu=0}^{n-1} A_\mu z^{-i\mu}. \tag{6.5.1}$$

Schreibweise:

$$\text{“}\underline{\text{Zeitbereich}}\text{”} \quad a(x) \quad \circ\!\!-\!\!\bullet \quad A(x) \quad \text{“}\underline{\text{Frequenzbereich}}\text{”}.$$

Es ist zwischen den Nullstellen der Polynome und den Nullkomponenten der Vektoren zu unterscheiden. Für den Zusammenhang gilt:

$$\begin{aligned} z^i \text{ ist Nullstelle von } a(x) &\iff \text{die } i\text{-te Komponente von } \boldsymbol{A} \text{ ist Null} \\ z^{-i} \text{ ist Nullstelle von } A(x) &\iff \text{die } i\text{-te Komponente von } \boldsymbol{a} \text{ ist Null.} \end{aligned}$$

Weiter gilt (mit $\circ$ ist die Hintereinanderausführung gemeint):

$$\text{IDFT} \circ \text{DFT} = \text{Identität} \tag{6.5.2}$$

$$\text{DFT} \circ \text{DFT} : (a_0, a_1, \ldots, a_{n-1}) \mapsto -(a_0, a_{n-1}, \ldots, a_1) \tag{6.5.3}$$

$$\text{DFT} \circ \text{DFT} \circ \text{DFT} \circ \text{DFT} = \text{Identität.} \tag{6.5.4}$$

Die Schieberegister-Implementierung der DFT zeigt Bild 6.1: Zu Beginn ist das Register mit den Zeitkomponenten $a_0, \ldots, a_{n-1}$ vorbelegt. Im i-ten Schritt enthält der μ-te Speicher den Wert $a_\mu z^{i\mu}$ und die Summe dieser Werte über μ ergibt die Frequenzkomponente A_i.

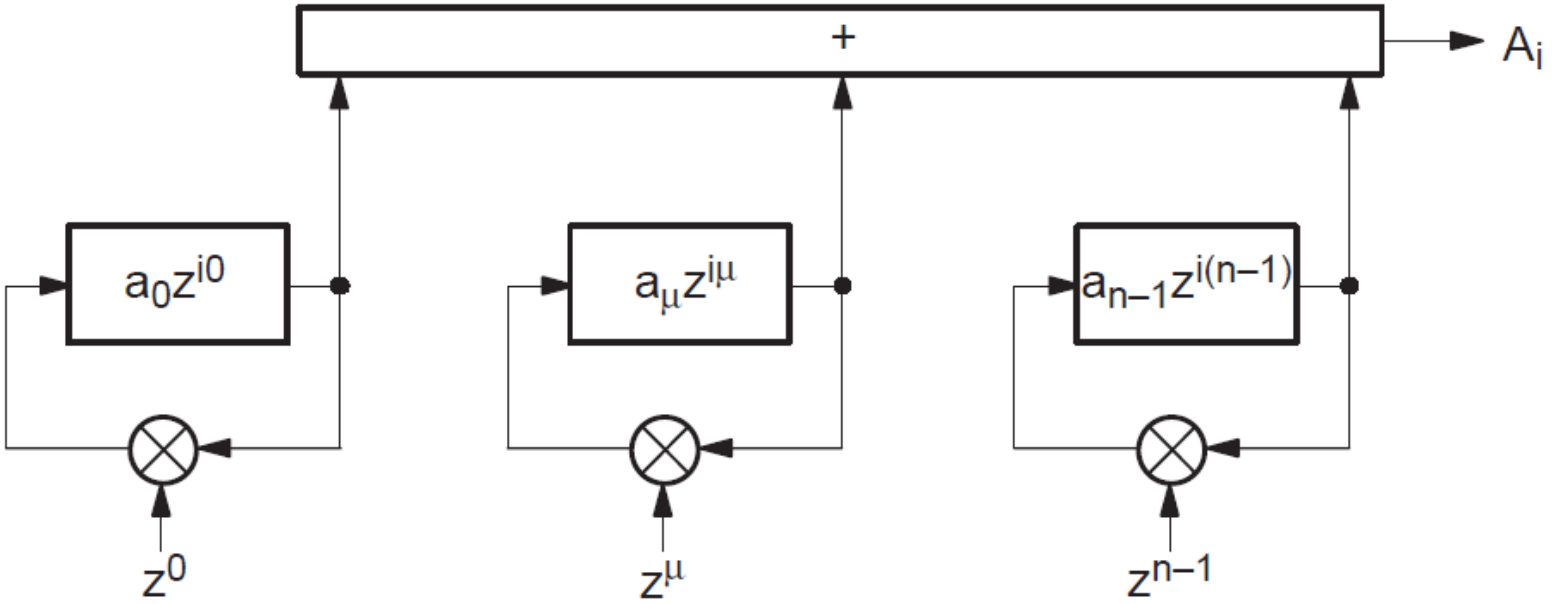


Bild 6.1. Schieberegister-Implementierung der DFT

Die Transformationen können auch mit Matrizen beschrieben werden:

$$\begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ \\ \vdots \\ \\ A_{n-1} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & z^1 & z^2 & \cdots & z^{n-1} \\ 1 & z^2 & z^4 & \cdots & z^{2(n-1)} \\ & & & & \\ \vdots & \vdots & \vdots & & \vdots \\ & & & & \\ 1 & z^{n-1} & z^{2(n-1)} & \cdots & z^{(n-1)^2} \end{pmatrix}}_{\boldsymbol{T}_{\mathrm{DFT}}} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \\ \vdots \\ \\ a_{n-1} \end{pmatrix}, \qquad (6.5.5)$$

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \\ \vdots \\ \\ a_{n-1} \end{pmatrix} = - \underbrace{\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & z^{-1} & z^{-2} & \cdots & z^{-(n-1)} \\ 1 & z^{-2} & z^{-4} & \cdots & z^{-2(n-1)} \\ & & & & \\ \vdots & \vdots & \vdots & & \vdots \\ & & & & \\ 1 & z^{-(n-1)} & z^{-2(n-1)} & \cdots & z^{-(n-1)^2} \end{pmatrix}}_{\boldsymbol{T}_{\mathrm{IDFT}}} \cdot \begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ \\ \vdots \\ \\ A_{n-1} \end{pmatrix}. \qquad (6.5.6)$$

Satz 6.8. *Die zyklische Faltung geht durch Fourier-Transformation in die komponentenweise Multiplikation über. Zur genaueren Formulierung werden die Zeit-Frequenz-Paare* $a(x) \circ\!\!-\!\!\bullet A(x)$, $b(x) \circ\!\!-\!\!\bullet B(x)$, $c(x) \circ\!\!-\!\!\bullet C(x)$ *betrachtet. Dann gilt:*

$$c(x) = R_{x^n-1}[a(x)b(x)] \iff C_i = A_i B_i, \tag{6.5.7}$$

$$C(x) = R_{x^n-1}[A(x)B(x)] \iff c_i = -a_i b_i.$$

Speziell gilt für die zyklische Verschiebung:

Verschiebung im Zeit-Bereich
= Phasendrehung im Freq.Bereich

$$c(x) = R_{x^n-1}[xa(x)] \iff C_i = z^i A_i, \tag{6.5.8}$$

Modulation = Verschiebung im Freq.Bereich
= Multiplikation im Zeit-Bereich

$$C(x) = R_{x^n-1}[xA(x)] \iff c_i = z^{-i} a_i.$$

Die Polynom-Multiplikation modulo $x^n - 1$ in (6.5.7) entspricht natürlich der zyklischen Faltung der Polynomkoeffizienten:

$$c(x) = R_{x^n-1}[a(x)b(x)] \iff c_i = \sum_{\mu=0}^{n-1} a_\mu b_{R_n[i-\mu]}. \tag{6.5.9}$$

Vorteile der Reed-Solomon (RS) und Bose-Chaudhuri-Hocquenghem (BCH) Codes

- Die beiden Codeklassen können analytisch geschlossen konstruiert werden. Die RS-Codes sind MDS-Codes, d.h. es gilt Gleichheit in der Singleton-Schranke.
- Die Minimaldistanz ist bekannt bzw. wird als Entwurfsparameter vorgegeben. Bei den RS-Codes ist sogar die gesamte Gewichtsverteilung exakt bekannt.
- Sowohl die RS- wie die BCH-Codes sind sehr leistungsfähig, sofern die Blocklänge nicht sehr groß wird.
- Es ist eine Anpassung an die Fehlerstruktur des Kanals möglich: Die RS-Codes sind besonders für Bündelfehler und die BCH-Codes sind besonders für Einzelfehler geeignet.
- Die Decodierung nach dem BMD-Prinzip kann rechentechnisch sehr einfach erfolgen. Zwar gibt es etliche Codes mit einfacheren Decodern, die aber die anderen Vorteile der RS- und BCH-Codes nicht aufweisen.
- Ohne wesentlichen Mehraufwand kann simple Soft-Decision-Information (Ausfallstellen-Information) im Decoder verarbeitet werden.
- Die RS- und BCH-Codes bilden die Grundlage für das Verständnis vieler anderer Codeklassen, die hier nicht behandelt werden.

Vorgabe: p, m beliebig $\Rightarrow$ $n = p^m-1 = q-1$ = primitive Blocklänge

Vorgabe: $d = 2t+1$ = Entwurfsdistanz (üblicherweise ungerade)

RS-Codes: $(n, k, d_{min})_q = (p^m-1, p^m-d, d)_q$ $\Rightarrow$ $n-k = d-1 = 2t$, $d_{min} = d$

(RS ist also MDS-Code)

$a(x) \in GF(q)[x]_{n-1}$

BCH-Codes: $(n, k, d_{min})_p = (p^m-1, k, d)_p$ mit $n-k \geq d_{min}-1 \geq d-1 = 2t$, $d_{min} \geq d$

$a(x) \in GF(p)[x]_{n-1}$

(Für die wirklich notwendige Anzahl von Prüfstellen bei BCH-Codes und die tatsächliche Minimaldistanz gibt es Tabellen. Der Zusammenhang ist komplex wird bestimmt durch die Hinzunahme weiterer Linearfaktoren damit g(x) GF(p)-wertig wird und damit letztlich von der quasi-zufälligen Verteilung der Primzahlen).

Definition 7.1 (RS-Code). *Für beliebiges p und m und eine beliebige sogenannte* Entwurfsdistanz $d = d_{\min}$ *ist ein* Reed-Solomon-Code *mit primitiver Blocklänge definiert als ein $(n, k, d_{\min})_q = (p^m - 1, p^m - d, d)_q$-Code. Üblicherweise wird $d = 2t + 1$ als ungerade vorgegeben und dann gilt*

$$n - k = d - 1 = 2t \tag{7.1.1}$$

für die Anzahl der Prüfstellen. Der Code besteht aus allen Zeitwörtern $(a_0, \ldots, a_{n-1}) \leftrightarrow a(x)$ mit Koeffizienten aus $\mathbb{F}_{p^m}$, so daß die zugehörigen Frequenzwörter $(A_0, \ldots, A_{n-1}) \leftrightarrow A(x)$ an $n - k = d - 1$ zyklisch aufeinanderfolgenden Stellen Null sind. Diese Stellen werden auch als Paritätsfrequenzen *bezeichnet. Zur exakten Beschreibung ist ein weiterer Parameter l vorzugeben*

Speziell für $l = 1$ gilt

$$\mathcal{C} = \Big\{ a(x) \;\Big|\; a(z^1) = a(z^2) = \cdots = a(z^{d-1}) = 0 \Big\} \tag{7.1.3}$$

und für $l = n + 1 - d$ gilt

$$\mathcal{C} = \Big\{ a(x) \;\Big|\; \text{Grad } A(x) \leq n - d \Big\}. \tag{7.1.4}$$

Die Performance des RS-Codes ist vom Parameter l unabhängig, jedoch kann dadurch die Implementierung beeinflusst werden.

Für die Anzahl der Codewörter gilt:

$$|\mathcal{C}| = q^k = q^{q-d} = p^{m(p^m-d)}. \tag{7.1.5}$$

Beispiel 7.1. Sei $q = 2^m$ und $n = 2^m - 1$. Speziell für $d = 2^{m-1}$ ergibt sich die Coderate als

$$R = \frac{k}{n} = \frac{2^m - 2^{m-1}}{2^m - 1} \approx \frac{1}{2}.$$

Für $m = 8$ resultiert ein $(255, 128, 128)_{256}$-Code, der auch als ein binärer $(2040, 1024, 128)_2$-Code aufgefaßt werden kann. Die Anzahl der Codewörter beträgt $256^{128} = 2^{1024} \approx 10^{308}$, d.h. schon bei einem primitiven Polynom vom Grad 8 ergibt sich in dieser Weise eine astronomische Anzahl von Codewörtern. Jeweils zwei Byte-Codewörter unterscheiden sich für mindestens 128 Bytes (Symbole). Bei der binären Interpretation folgt daraus aber nur ein Unterschied von 128 Bits, denn zwei Bytes unterscheiden sich schon dann, wenn die beiden 8-Bit-Gruppen sich nur bei einem einzigen Bit unterscheiden. ■

Satz 7.1. *RS-Codes sind zyklische MDS-Codes, d.h. die Entwurfsdistanz entspricht der Minimaldistanz und es gilt* $d_{\min} = d = n - k + 1 = p^m - k$.

Beweis: "Linearität": ist offensichtlich.

"Zyklisch": Sei $a(x) \in \mathcal{C}$ und $c(x) = R_{x^n-1}[xa(x)]$ die zyklische Verschiebung. Nach Satz 6.8 gilt $C_i = z^i A_i$ und somit $C_i = 0 \Leftrightarrow A_i = 0$. Also folgt $c(x) \in \mathcal{C}$.

"MDS": Sei $d_{\min}$ die tatsächliche Minimaldistanz und $d = n - k + 1$ die Entwurfsdistanz. Zu zeigen ist $d_{\min} = d$: Nach Satz 3.7 gilt $d_{\min} \leq n - k + 1 = d$ (Singleton-Schranke). Betrachte nun $\mathcal{C}$ in der Form (7.1.4) mit Grad $A(x) \leq n - d$. Somit hat $A(x) \neq 0$ höchstens $n - d$ Nullstellen, d.h. es gibt höchstens $n - d$ Werte z^{-i} mit $a_i = -A(z^{-i}) = 0$. Also folgt $w_H(a(x)) \geq d$ und somit $d_{\min} \geq d$. Auch für jeden anderen Wert von l gilt $d_{\min} = d$, da die "Modulation" im Zeitbereich die Gewichte nicht verändert. ■

Der eigentliche Beweis für das Hauptergebnis $d=d_{min}$ ist nach allen Vorbereitungen nun erstaunlich simpel.

Satz 7.2. *Für den RS-Code in der Form (7.1.3) werden das Generatorpolynom und das Prüfpolynom gegeben durch:*

$$g(x) = \prod_{i=1}^{d-1}(x - z^i) \quad , \quad h(x) = \prod_{i=d}^{n}(x - z^i). \tag{7.1.6}$$

Beweis: Für jedes Codewort gilt $a(z^i) = 0$ für $1 \leq i \leq d-1$. Also ist $(x - z^i)$ ein Teiler von $a(x)$. Damit erweist sich $a(x)$ als Vielfaches von $g(x)$. Da $g(x)$ den Grad $d - 1 = n - k$ hat, ist $g(x)$ das Generatorpolynom. Für das Prüfpolynom $h(x)$ muß $g(x)h(x) = x^n - 1$ gelten, was nach (6.2.13) erfüllt ist. ■

Auch dieser Beweis ist simpel.

Beispiel 7.2. Sei $p = 2$, $m = 3$ und somit $n = 7$. Die Arithmetik in $\mathbb{F}_{2^3}$ erfolgt gemäß Beispiel 6.4. Zur Entwurfsdistanz $d = 3$ resultiert ein $(7, 5, 3)_8$-RS-Code $\mathcal{C} = \{(a_0, \ldots, a_6) \mid A_1 = A_2 = 0\}$. Nach Satz 7.2 gilt:

$$\begin{aligned} g(x) &= (x - z)(x - z^2) = x^2 + z^4 x + z^3, \\ h(x) &= (x - z^3)(x - z^4)(x - z^5)(x - z^6)(x - z^7) \\ &= x^5 + z^4 x^4 + x^3 + z^5 x^2 + z^5 x + z^4. \end{aligned}$$

Obwohl RS-Codes besonders für Bündelfehler geeignet sind, wird auf der nächsten Seite die Leistungsfähigkeit der RS-Codes bei statistisch unabhängigen Einzelfehlern gezeigt.

Unterstellt wird ein AWGN mit binärer Modulation und Hard-Decision. Die beiden Grafiken unterscheiden sich nur bei der Parametrisierung der Abszisse: links über E_b/N_0, rechts über der Decoder-Input-SER.

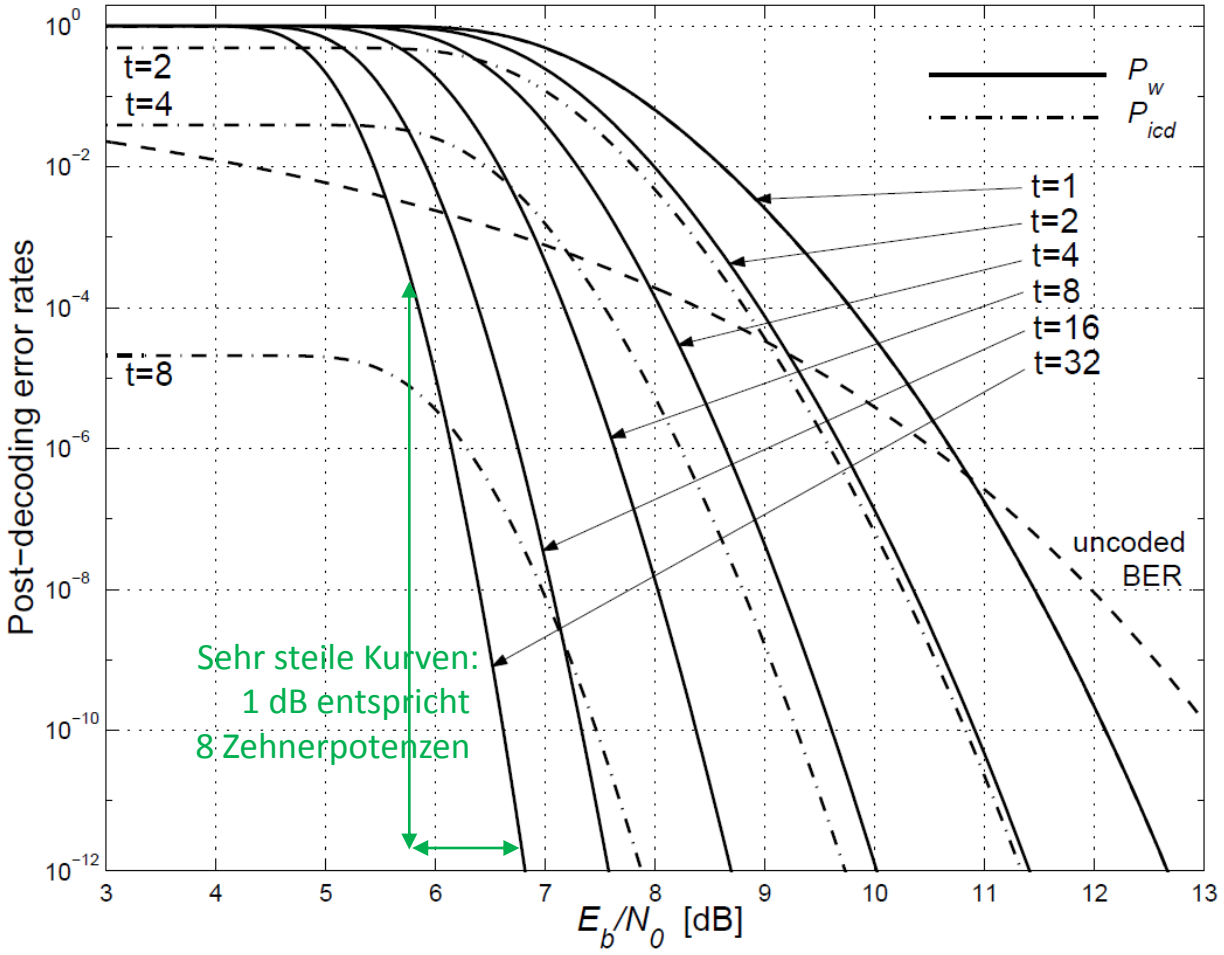
Weitere Details:

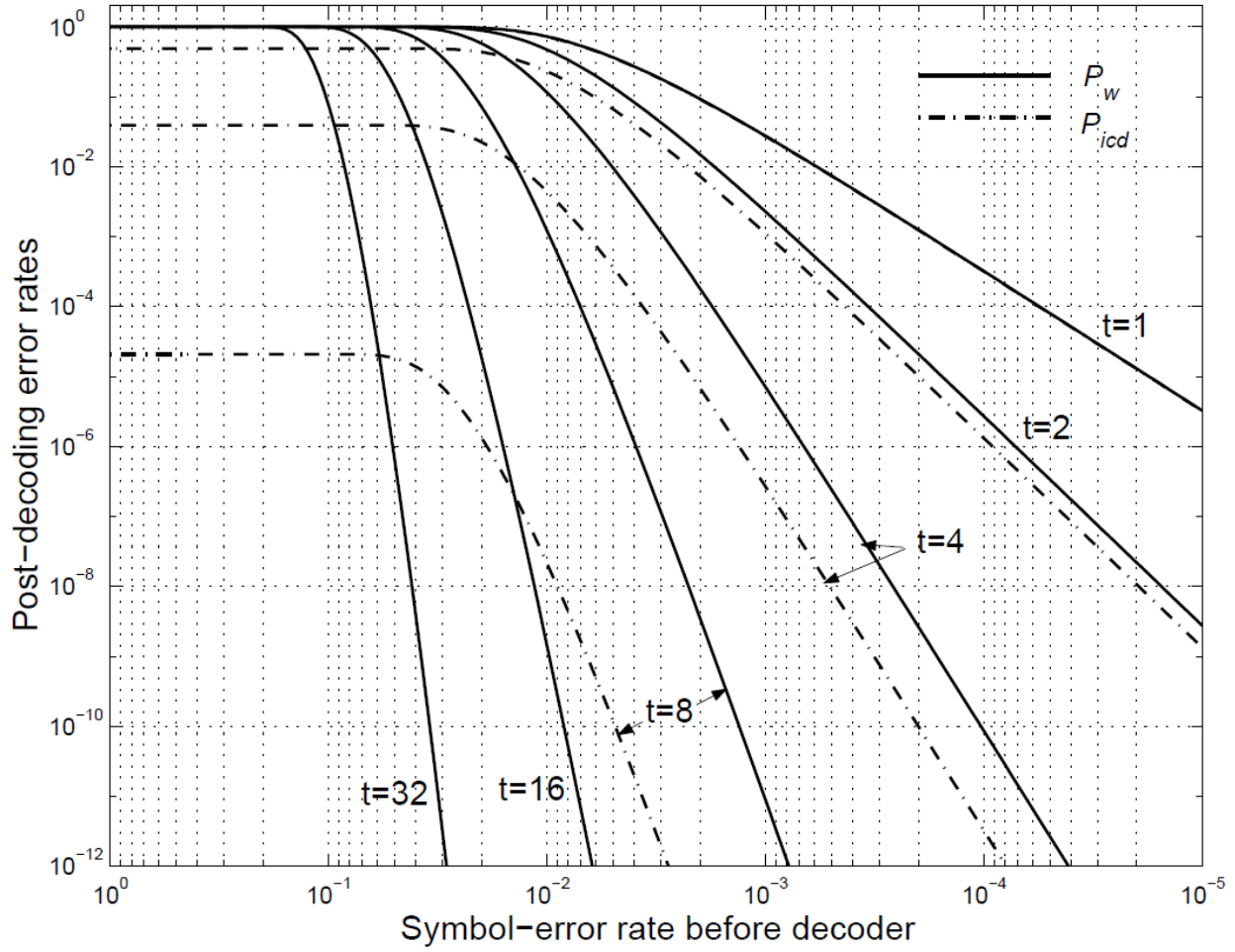
The following figures show the performance of RS codes over $q = 2^m$ for various parameter constellations, where the post-decoding error rates are displayed over E_b/N_0 and p_e in the upper and lower subfigures, respectively. Generally, the word-error rate P_w is always given as in (8.1.11) and the probability of incorrect decisions P_{icd} as in (8.1.18), usually over E_b/N_0 in the upper subfigures as well as over the q-ary symbol error rate prior to the decoder p_e in the bottom subfigures. For the binary AWGN channel, i.e., for BPSK modulation, p_e is determined by E_b/N_0 as

$$p_e = 1 - \left(1 - Q\left(\sqrt{\frac{2RE_b}{N_0}}\right)\right)^m , \qquad (8.1.25)$$

using (1.3.16), with $m = \log_2 q$ and $M = 1$, as well as (1.3.23).

P_w und P_{icd} der $(255, 255\text{-}2t, 2t\text{+}1)_{256}$-RS-Codes für verschiedene t





P_w = Wort-Fehlerwahrscheinlichkeit (BMDD) = P(Empfangswort liegt außerhalb der richtigen Kugel)

d.h. zwischen den Kugeln oder in einer falschen Kugel

P_{icd} = Wahrscheinlichkeit inkorrekter Decodierung = P(Empfangswort liegt in einer falschen Kugel)

BCH-Codes entstehen aus den RS-Codes dadurch, daß zwar weiterhin die Fourier-Transformation auf $\mathbb{F}_{p^m}$ betrachtet wird und die Blocklänge weiterhin $n = p^m - 1$ beträgt, aber die Codesymbole müssen im Primkörper $\mathbb{F}_p$ statt in $\mathbb{F}_{p^m}$ liegen. Für den Normalfall $p = 2$ sind die Codesymbole also Bits und keine Bitgruppen, d.h. BCH-Codes sind normale Binärcodes.

Definition 7.2 (BCH-Codes). *Für beliebiges p und m und eine beliebige Entwurfsdistanz d ist ein* Bose-Chaudhuri-Hocquenghem Code *definiert als ein $(n, k, d_{\min})_p$-Code mit der primitiven Blocklänge $n = p^m - 1 = q - 1$ und weiter gilt*

$$d_{\min} \geq d \quad , \quad k \leq n + 1 - d_{\min} \leq n + 1 - d. \tag{7.2.1}$$

Bei $d = 2t + 1$ gilt somit $n - k \geq 2t$. Der Code besteht aus allen Zeitwörtern $(a_0, \ldots, a_{n-1}) \leftrightarrow a(x)$ mit Koeffizienten aus $\mathbb{F}_p$, so daß die zugehörigen Frequenzwörter $(A_0, \ldots, A_{n-1}) \leftrightarrow A(x)$ an mindestens $d-1$ zyklisch aufeinanderfolgenden Stellen Null sind. Üblicherweise wird folgende Lage der Paritätsfrequenzen vorausgesetzt, was als BCH-Code im engeren Sinn *(narrow sense) bezeichnet wird:*

$$\mathcal{C} = \Big\{ a(x) \in \mathbb{F}_p[x] \;\Big|\; a(z^1) = \cdots = a(z^{d-1}) = 0 \Big\}. \tag{7.2.2}$$

Aus der Vorgabe der Entwurfsdistanz d ergibt sich die Infoblocklänge bzw. die Coderate sowie die tatsächliche Minimaldistanz nicht direkt, sondern muß erst noch berechnet werden.

Die Herleitung der Generator- und Prüfpolynome erfordert mehr Kenntnisse als bisher vermittelt wurden, siehe dazu das Buch.

Tabelle 7.1. $(n,k)_2$-BCH-Codes zur Korrektur von t Fehlern (nach [53])

n	k	t	n	k	t	n	k	t	n	k	t
7	4	1	127	85	6	255	123	19	511	403	12
				78	7		115	21		394	13
15	11	1		71	9		107	22		385	14
	7	2		64	10		99	23		.	.
	5	3		57	11		91	25		259	30
				50	13		87	26		.	.
31	26	1		43	14		79	27		130	55
	21	2		36	15		71	29		.	.
	16	3		29	21		63	30		.	.
	11	5		22	23		55	31	1023	1013	1
	6	7		15	27		47	42		1003	2
				8	31		45	43		993	3
63	57	1					37	45		983	4
	51	2	255	247	1		29	47		973	5
	45	3		239	2		21	55		963	6
	39	4		231	3		13	59		953	7
	36	5		223	4		9	63		943	8
	30	6		215	5					933	9
	24	7		207	6	511	502	1		923	10
	18	10		199	7		493	2		913	11
	16	11		191	8		484	3		903	12
	10	13		187	9		475	4		.	.
	7	15		179	10		466	5		768	26
				171	11		457	6		.	.
127	120	1		163	12		448	7		513	57
	113	2		155	13		439	8		.	.
	106	3		147	14		430	9		258	106
	99	4		139	15		421	10		.	.
	92	5		131	18		412	11		.	.

In Tabelle 7.1 werden die binären BCH-Codes im engeren Sinn zur Korrektur von t Fehlern aufgelistet, und zwar vollständig für die primitiven Blocklängen $n = 7, 15, 31, 63, 127, 255$ und teilweise für die Blocklängen $n = 511, 1023$. Dabei gilt natürlich $2t + 1 \leq d \leq d_{\min} \leq n - k + 1$. Komplette Listen finden sich beispielsweise in [49, 53, 75, 80].

Den Tabellen 7.1 und 7.2 liegt die Entwurfsdistanz $d = 2t + 1$ zugrunde. Die tatsächliche Minimaldistanz ist unbekannt und kann eventuell wesentlich größer sein. Bei den meisten Decodierverfahren für BCH-Codes kann allerdings eine größere Minimaldistanz gar nicht ausgenutzt werden, da die Decodierung auf den aufeinanderfolgenden Paritätsfrequenzen basiert.

Satz 7.6. *Für einen binären $(n,k)_2$-BCH-Code mit der primitiven Blocklänge $n = 2^m - 1$, der mindestens t Fehler korrigieren kann, gilt stets*

$$n - k \leq m \cdot t. \tag{7.3.1}$$

Wenn t erhöht wird um 1, so muss k reduziert werden um m, aber das gilt nur meistens, ab und zu gibt es Ausnahmen und dann springt t gleich um mehr als 1 nach oben. Ursache: die Verteilung der Primzahlen. Damit ergibt sich eine seltsame Mischung aus Regelmäßigkeit und Zufälligkeit.

Tabelle 7.2. Generatorpolynome für einige $(n, k)_2$-BCH-Codes (nach [135])

n	k	t	$g(x)$ oktal
7	4	1	13
15	11	1	23
	7	2	721
	5	3	2467
31	26	1	45
	21	2	3551
	16	3	107657
	11	5	5423325
	6	7	313365047
63	57	1	103
	51	2	12471
	45	3	1701317
	39	4	166623567
	36	5	1033500423
	30	6	157464165547
	24	7	17323260404441
	18	10	1363026512351725
127	120	1	211
	113	2	41567
	106	3	11554743
	99	4	3447023271
	92	5	624730022327
	85	6	130704476322273
	78	7	26230002166130115
	71	9	6255010713253127753
	64	10	1206534025570773100045
	57	11	335265252505705053517721
	50	13	54446512523314012421501421
255	247	1	435
	239	2	267543
	231	3	156720665
	223	4	75626641375
	215	5	23157564726421
	207	6	16176560567636227
	199	7	7633031270420722341
	191	8	2663470176115333714567
	187	9	5275531354000132223635 1
	179	10	226247107173404324163004 55
	171	11	154162142123423560770616306 37
	163	12	750041551007560255157472451 4601
	155	13	375751300540766501572250646467 7633
	147	14	164213017353716552530416530544101 1711
	139	15	461401732060175561570722730247453567445
	131	18	215713331471510151261250277442142024165471

Tabelle 7.2 enthält für einige der BCH-Codes aus Tabelle 7.1 die Generatorpolynome in oktaler Codierung. Für den $(15, 7)_2$-BCH-Code mit $t = 2$ gilt beispielsweise $721_{\text{oktal}} = 111\ 010\ 001_{\text{dual}} = x^8 + x^7 + x^6 + x^4 + 1$.

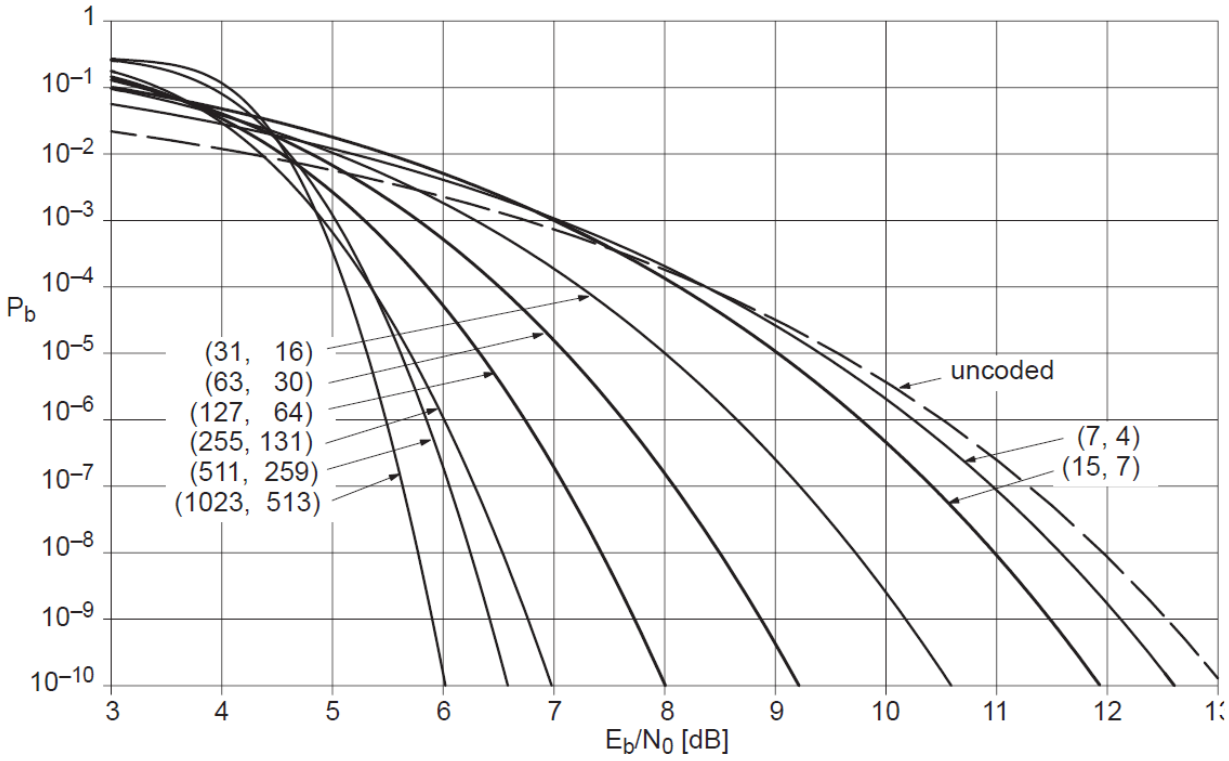


Bild 7.1. Fehlerwahrscheinlichkeit der $R = 1/2$-BCH-Codes

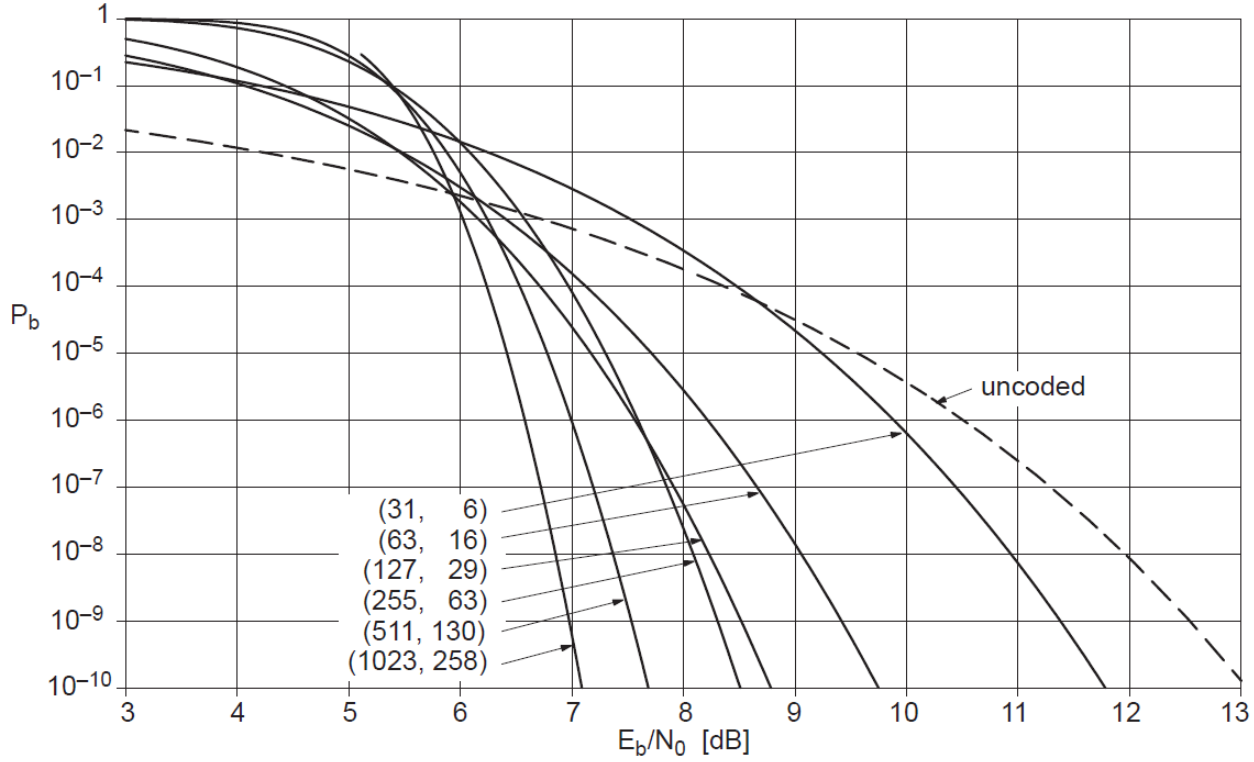


Bild 7.2. Fehlerwahrscheinlichkeit der $R = 1/4$-BCH-Codes

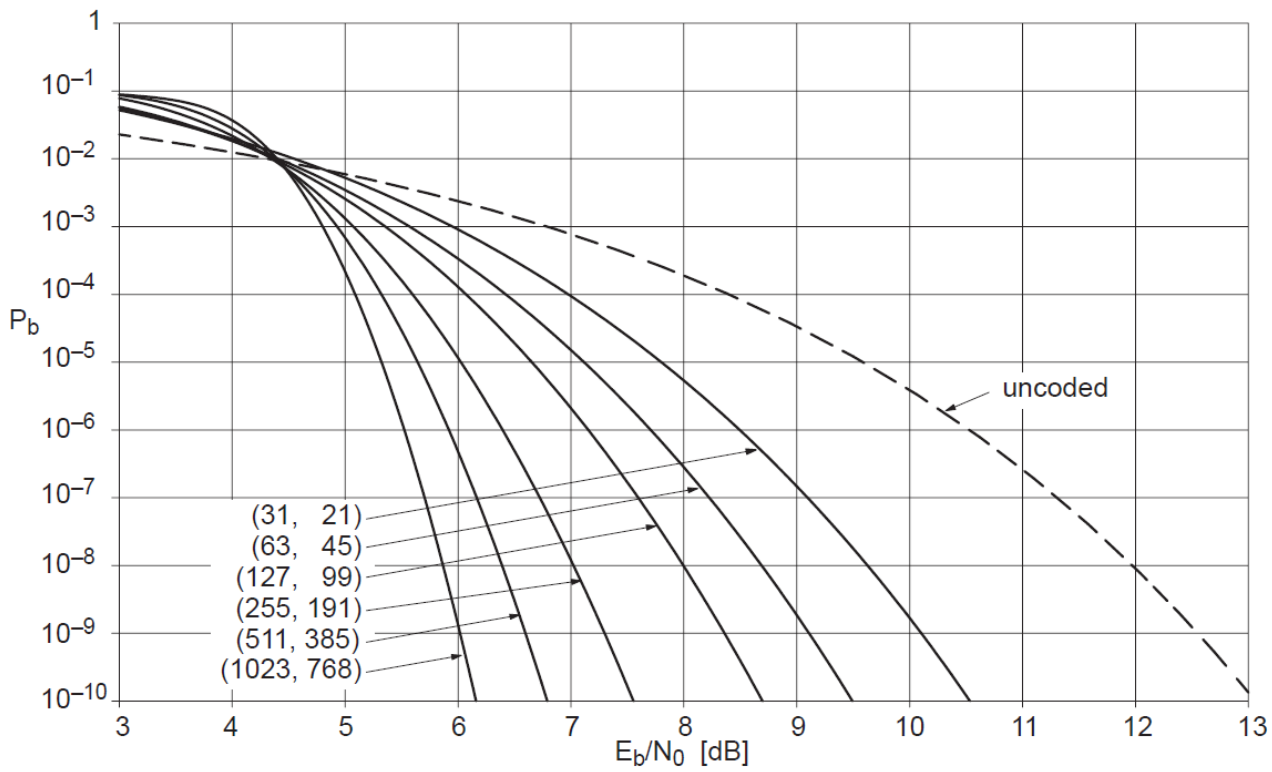


Bild 7.3. Fehlerwahrscheinlichkeit der $R = 3/4$-BCH-Codes

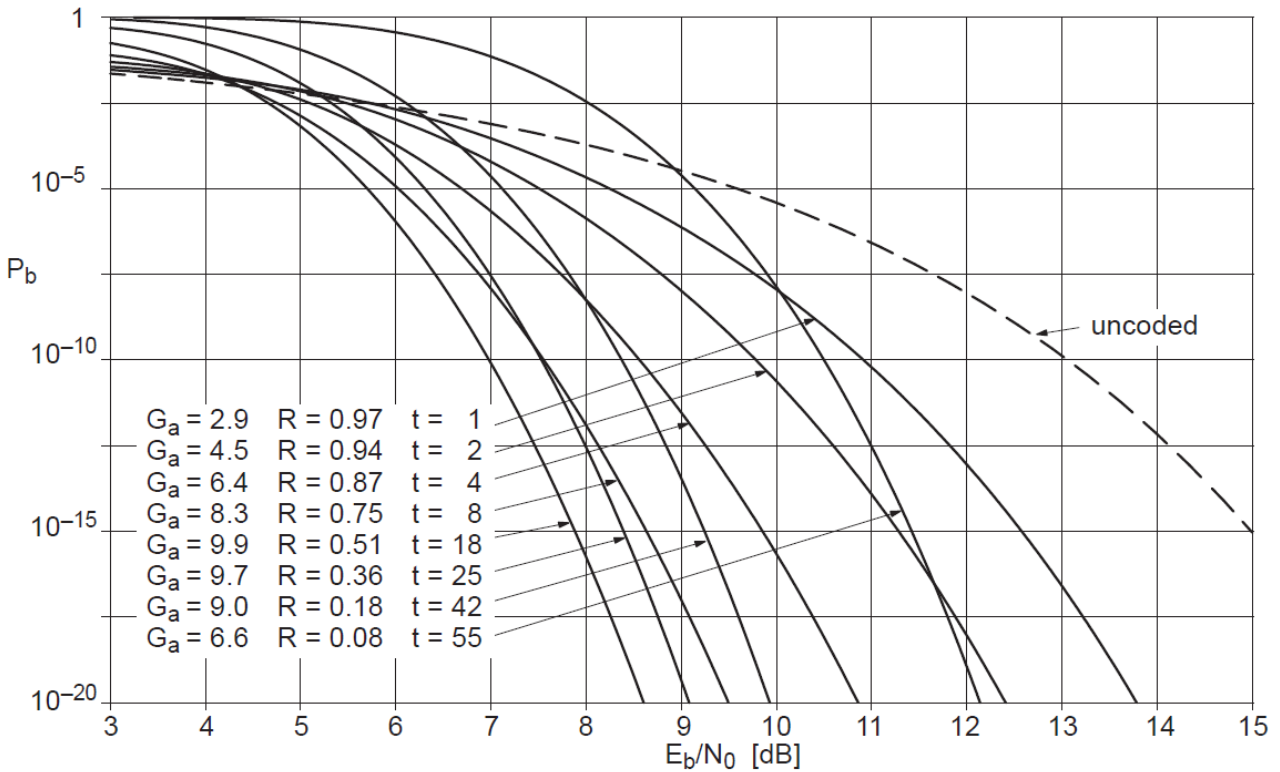
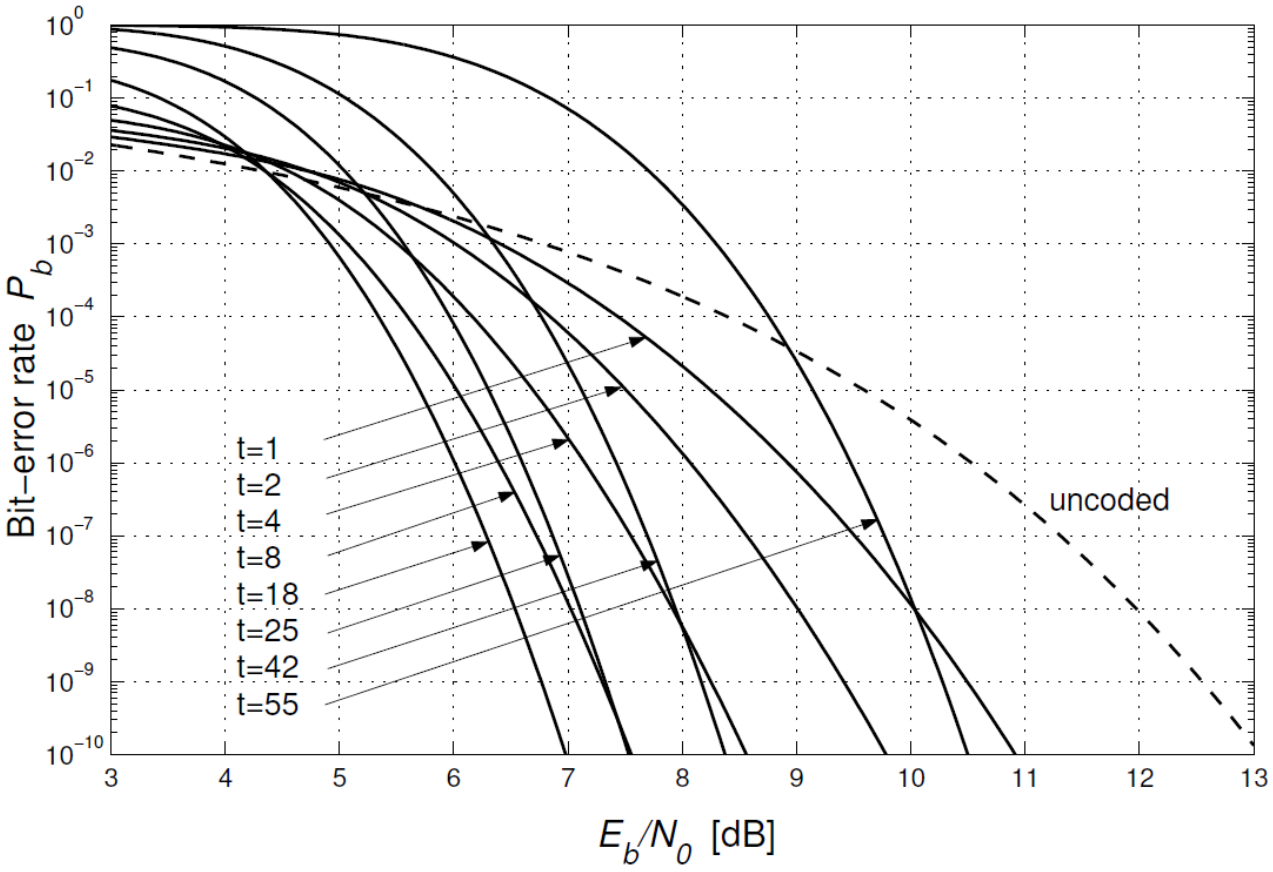
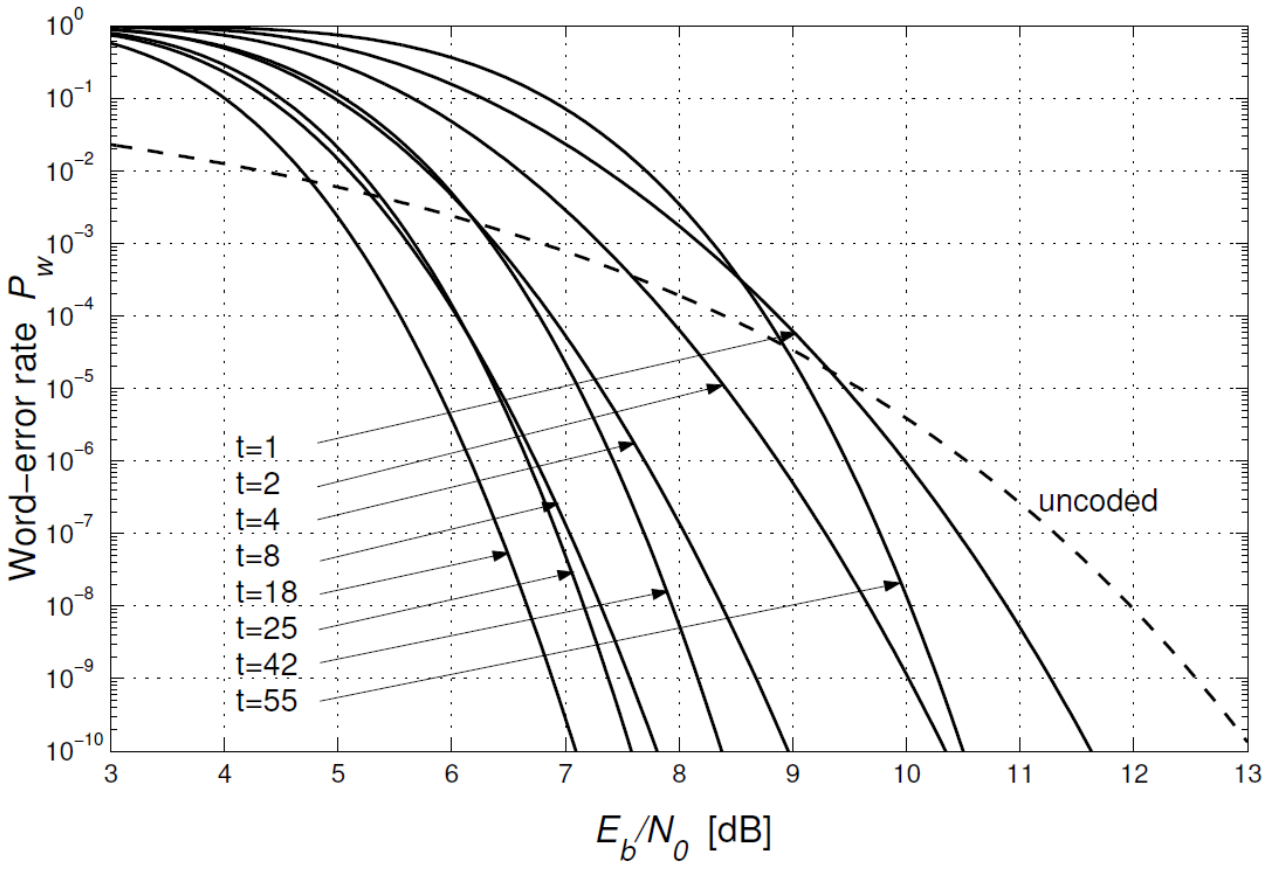


Bild 7.4. Fehlerwahrscheinlichkeit von BCH-Codes verschiedener Raten bei $n = 255$



Gleiche Kurven wie zuvor in Bild 7.4
aber jetzt auf 10^{-10} limitiert



P_w anstelle von P_b wie in den anderen Grafiken
Nur geringe Unterschiede zwischen Wort- und Bit-Fehlerw.

Die Bilder 7.1 bis 7.3 zeigen die Bit-Fehlerwahrscheinlichkeit von BCH-Codes bei Raten von näherungsweise $1/2$, $1/4$ und $3/4$ bei verschiedenen Blocklängen zwischen 15 und 1023. Dabei wird eine BMD-Decodierung unterstellt, so daß die Kurven nach (3.7.2) berechnet werden können. Offensichtlich führt eine größere Blocklänge zu kleinerer Fehlerrate – zumindest für Blocklängen bis 1023. Der Vergleich der Bilder 7.1 bis 7.3 legt $R \approx 1/2$ als scheinbar günstigste Coderate nahe. Um dies ganz deutlich zu machen, zeigt Bild 7.4 einige BCH-Codes verschiedener Raten mit der konstanten Blocklänge $n = 255$. Nach der Theorie sind zwar kleine Coderaten am besten (nach Bild 2.4 ist beispielsweise beim Übergang von $R = 1/2$ auf $R = 1/10$ ein Gewinn von etwa 1 dB zu erwarten), aber offensichtlich liefert das Konstruktionsprinzip der BCH-Codes bei mittleren Raten die besten Ergebnisse.

Tabelle 7.3. Asymptotischer Codierungsgewinn $G_{\text{a,hard}}$ [dB] einiger BCH-Codes

n	$R \approx 1/2$	$R \approx 1/4$	$R \approx 3/4$
31	3,1	1,9	3,1
63	5,2	4,8	4,6
127	7,4	7,0	5,9
255	9,9	8,8	8,3
511	12,0	11,5	10,5
1023	14,6	14,3	13,1

Der asymptotische Codierungsgewinn wächst mit zunehmender Blocklänge.

Tabelle 7.4. Distanzrate d/n einiger BCH-Codes

n	$R \approx 1/2$	$R \approx 1/4$	$R \approx 3/4$
31	0,23	0,48	0,16
63	0,21	0,37	0,11
127	0,17	0,34	0,07
255	0,15	0,24	0,07
511	0,12	0,22	0,06
1023	0,11	0,21	0,05
asympt.GV	0,11	0,21	0,04

Im Gegensatz dazu sinkt die Distanzrate mit zunehmender Blocklänge.

Siehe dazu auch die nächste Seite.

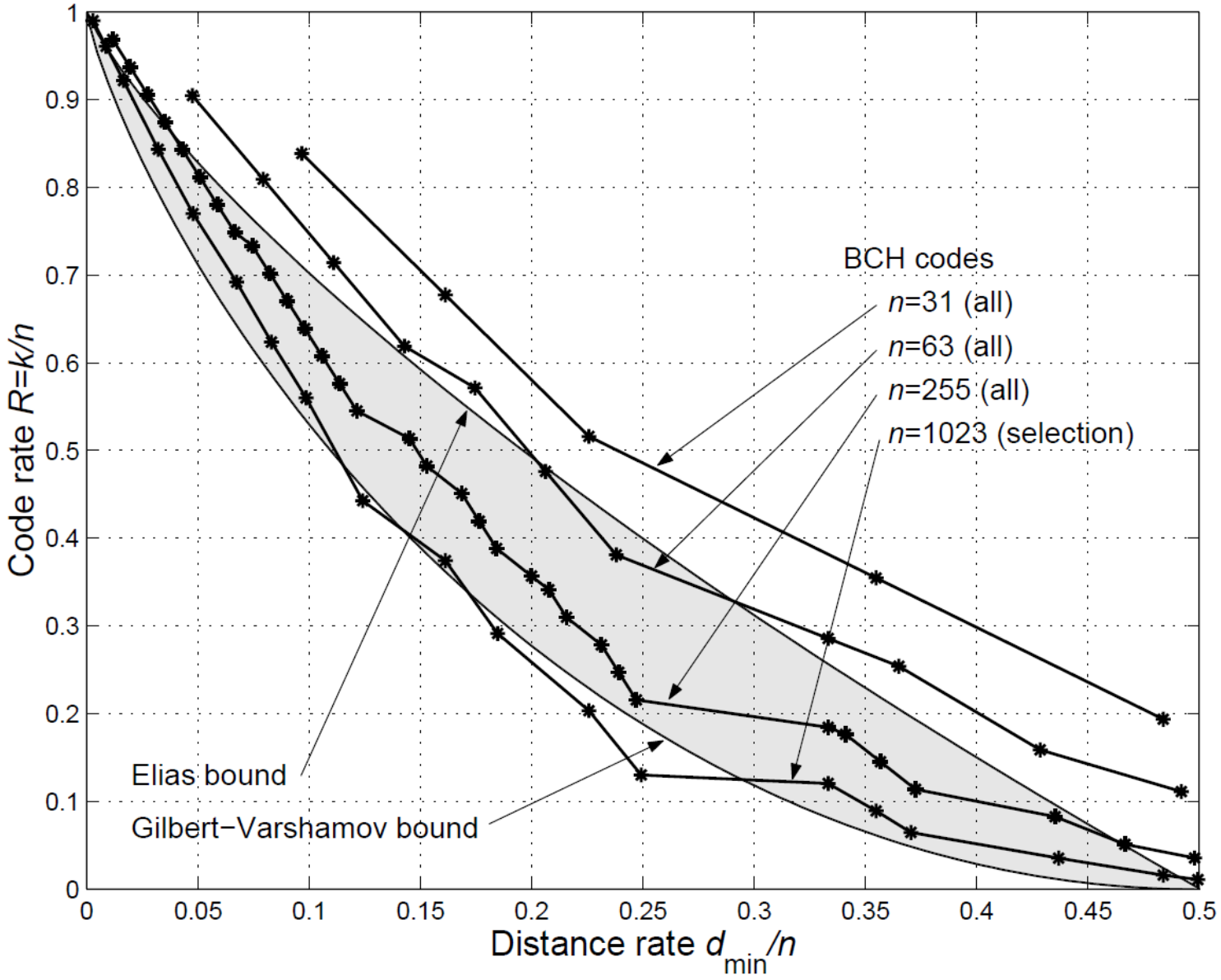


Figure 8.16. Comparison of some BCH codes with upper and lower asymptotic bounds

Kurze BCH-Codes liegen noch oberhalb der oberen Schranken,
Lange BCH-Codes liegen unterhalb der unteren GV-Schranke und konvergieren gegen den linken Rand.

In dieser Betrachtung sind BCH-Codes also asymptotisch schlecht – aber unter praktischen Gesichtspunkten dennoch attraktiv und viel verwendet.

Vergleich von BCH- und RS-Codes für Einzelfehler (AWGN).

Voraussetzung für einen fairen Vergleich sind ähnliche Coderate (erfüllt) und ähnliche bzw. vernünftige Blocklänge:

Auf den ersten Blick überrascht dass der RS(255) besser als der BCH(255) ist. Aber in der binären Interpretation hat der RS(255) auch die viel größere Blocklänge von 2040.

Der faire Vergleich von BCH(255) sollte mit den beiden oberen RS-Codes erfolgen die binäre Blocklängen von 115 und 282 haben und diese RS-Codes sind erwartungsgemäß bei Einzelfehlern schlechter.

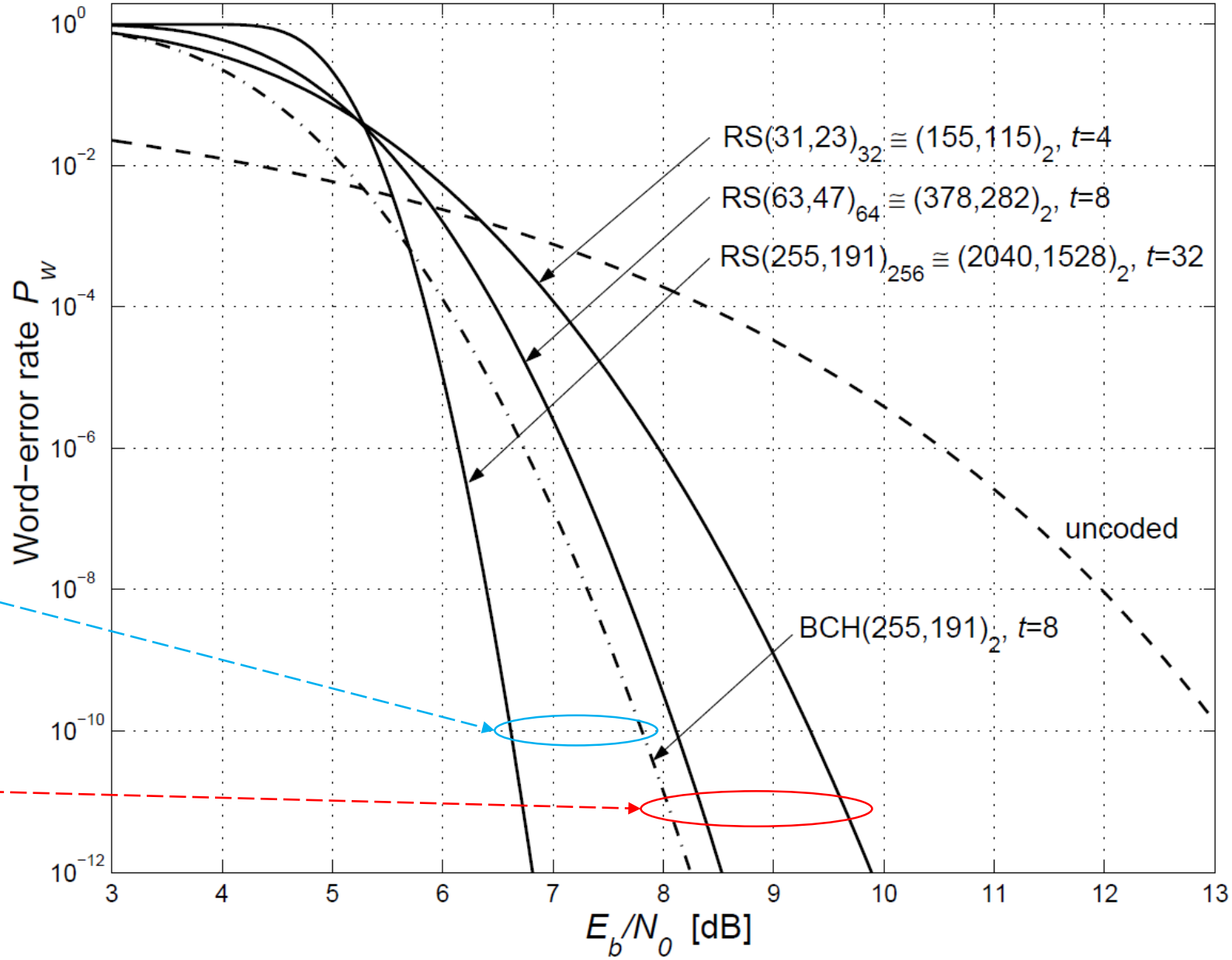


Figure 8.15. Comparison of a BCH code with some RS codes (all with $R \approx 3/4$)

Anwendung der Kanalcodierung bei der Musik-CD

Ziele
- Korrektur von Bündelfehlern (Kratzer)
- Korrektur von Einzelfehlern (Materialverunreinigungen)
- Robuste Meachnik (Spur-Synchronisation)
- Implementierbarkeit mit der Technologie von 1979 (Spezifikation) / 1982 (Geräteverfügbarkeit)

Vorbereitungen (verwendete Techniken)
- Ausfallkorrektur bei RS-Codes (§ 7.9)
- Gekürzte RS-Codes (§ 4.5)
- Interleaving (§ 11.1)
- Verkette Codes mit äußerem und innerem Code (hier sowie § 9.7, § 12.1 und weitere Anwendungen)

Diese Techniken sind für sich gesehen relativ simpel, aber so wirkungsvoll in der Gesamt-Kombination dass Bündelfehler mit Tausenden von Fehlern korrigiert werden können obwohl die RS-Codes mit nur 4 Prüfstellen betrieben werden.

Es wird jetzt ein q-närer symmetrischer Kanal mit Ausfällen als Verallgemeinerung des BSEC aus Bild 1.3 vorausgesetzt:

$$\mathcal{A}_{\text{in}} = \mathbb{F}_q \quad , \quad \mathcal{A}_{\text{out}} = \mathbb{F}_q \cup \{?\}. \tag{7.9.1}$$

Der Demodulator entscheidet auf $y = ?$ (Ausfall), wenn die Entscheidung für ein bestimmtes $y \in \mathbb{F}_q$ sehr unsicher wäre. Der Vorteil dieser Methode liegt darin, daß die Korrektur eines derartigen Ausfalls (Position bekannt, Sendesymbol unbekannt) nur eine Prüfstelle erfordert, während zur Korrektur eines Fehlers (Position unbekannt, Sendesymbol bzw. Fehlersymbol unbekannt) zwei Prüfstellen erforderlich sind.

Satz 7.9 (Fehler- und Ausfallkorrektur). *Ein $(n, n-2t, 2t+1)_q$-RS-Code korrigiert τ Fehler und τ_v Ausfälle, sofern gilt:*

$$2\tau + \tau_v \;\leq\; 2t = d_{\min} - 1 = n - k. \tag{7.9.6}$$

Der RS-Code kann also im Extremfall entweder

- t Fehler (Fehlerposition unbekannt und Fehlerwert unbekannt) korrigieren oder
- 2t Ausfälle (Fehlerposition bekannt und Fehlerwert unbekannt) korrigieren

Definition 4.6. *Ein* $(n, k, d_{\min})_q$*-Code kann wie folgt zu einem* $(n', k', d'_{\min})_q$*-Code verändert werden:*

(4) Beim Verkürzen (shortening) werden Infobits unterdrückt:

$$n' < n, \quad k' < k, \quad n' - k' = n - k, \quad R' < R, \quad d'_{\min} \geq d_{\min}.$$

RS-Code mit m=8, t=2:

Ausgangscode $(255, 251, 5)_{256}$

Gekürzte Codes: $(28, 24, 5)_{256}$

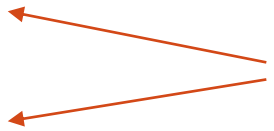
$(32, 28, 5)_{256}$

Alle Codes können entweder

- 2 Symbolfehler korrigieren & 0 Ausfälle korrigieren
- 1 Symbolfehler korrigieren & 2 Ausfälle korrigieren

- 0 Symbolfehler korrigieren & 4 Ausfälle korrigieren

- 4 Symbolfehler erkennen



Anwendung bei CD

Interleaving:

- Unterstützt die Korrektur von Bündelfehlern
  - wenn nur ein Einzelfehler-korrigierender Code verfügbar ist oder
  - wenn die Korrekturkapazität eines Bündelfehler-korrigierenden Codes nicht ausreicht

- Interleaving basiert auf einer Umsortierung der Symbolfolge:
  - empfangsseitige Umsortierung, damit ein Bündelfehler auf mehrere Codewörter verteilt (verspreizt) wird sowie
  - sendeseitige Umsortierung damit es zusammen mit der empfangsseitigen Umsortierung transparent wird

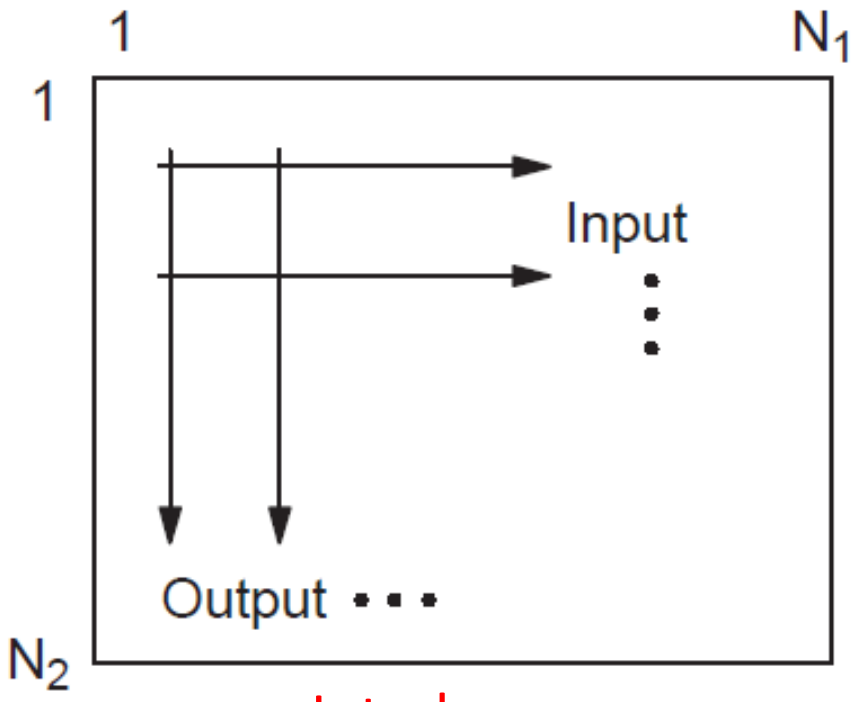
- Umsortieren
  - erfordert sende- und empfangsseitig einen Speicher für mindestens zwei oder mehr Codewörter
  - impliziert eine Verzögerung bei der Übertragung entsprechen der Speichergröße
    (d.h. transparent aber mit zusätzlicher Verzögerung)

- Dimensionierungs-Tradeoff:
  - Großer Speicher um auch lange Bündelfehler zu zerhacken
  - Kleiner Speicher um Verzögerungszeit und Aufwand zu begrenzen

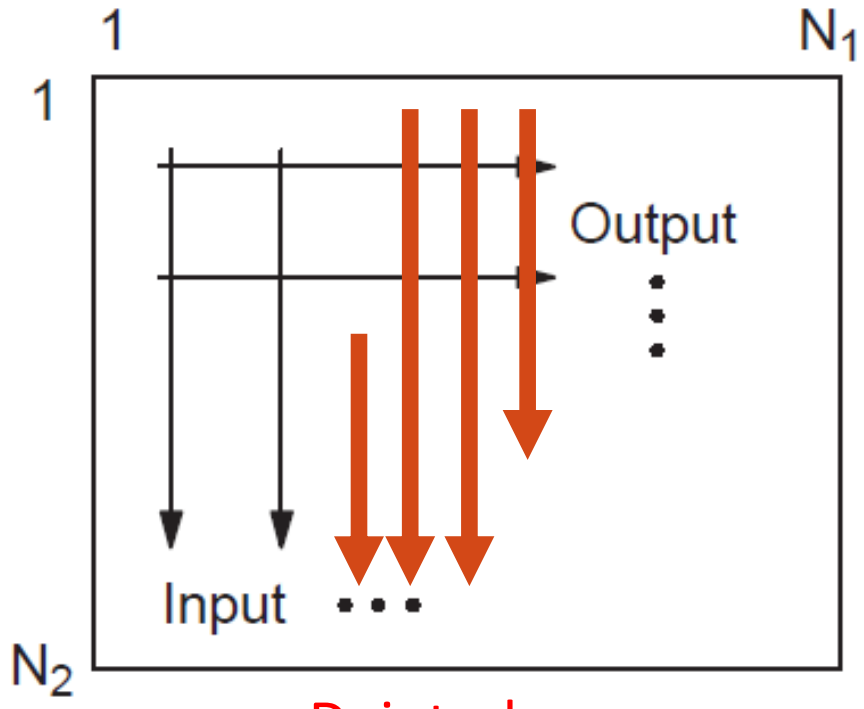
- Sonstiges
  - Neben dem nachfolgend vorgestellten Block-Interleaving gibt es auch ein Faltungs-Interleaving mit halbiertem Speicher bei gleicher Wirkung
  - In Kombination mit RS-Codes werden die 2^m-stufigen Symbole umsortiert und nicht etwa die Bits

Block-Interleaving: N_1 = Blocklänge
N_2 = Interleavingtiefe = Anzahl der Codewörter
(mindestens $N_2 \geq 2$, typisch $N_2 \leq 8$)

Achtung: Darstellung im Buch ist falsch
Interleaver und Deinterleaver vertauscht



Interleaver



Deinterleaver

Interleaving und Deinterleaving sind in der Zusammenschaltung insgesamt transparent, von der Verzögerungszeit abgesehen.

Nochmals: Es werden Symbole umsortiert, nicht etwa Bits!
Bei einem hochstufigen Code wäre es völlig kontraprodukt, die Bitfehler eines Symbolfehlers erst zu verspreizen und dann mühsam wieder zu lokalisieren.

Der auf dem Kanal entstehende Bündelfehler (hier beispielhaft mit einer Länge von etwa $3*N_2$ angenommen) wird vertikal in den Deinterleaver geschrieben. Beim horizontalem Auslesen ist dann jedes Codewort mit nur wenigen Fehlern (hier etwa 3) betroffen.

Ohne Deinterleaving wäre nur ein Codewort mit vielen Fehlern betroffen.

Das Prinzip des Faltungs-Interleavings zeigt Bild 11.2. Interleaver und Deinterleaver bestehen jeweils aus N Speichern in Form von Vektoren der Längen $0, J, 2J, 3J, \ldots, (N-1)J$. Es wird $M = J \cdot N$ gesetzt. Mit jedem neuen Symbol wird eingangsseitig ein Symbol links eingeschoben und ein Symbol rechts herausgeschoben. Anschließend werden die vier Multiplexer bzw. Demultiplexer weitergeschaltet, wobei wieder eine Synchronisation zwischen Interleaver und Deinterleaver erforderlich ist.

Bei gleicher Verspreizungs-Wirkung haben Faltungs-Interleaver gegenüber Block-Interleavern eine halbierte Verzögerungszeit und halbierte Speichergröße.

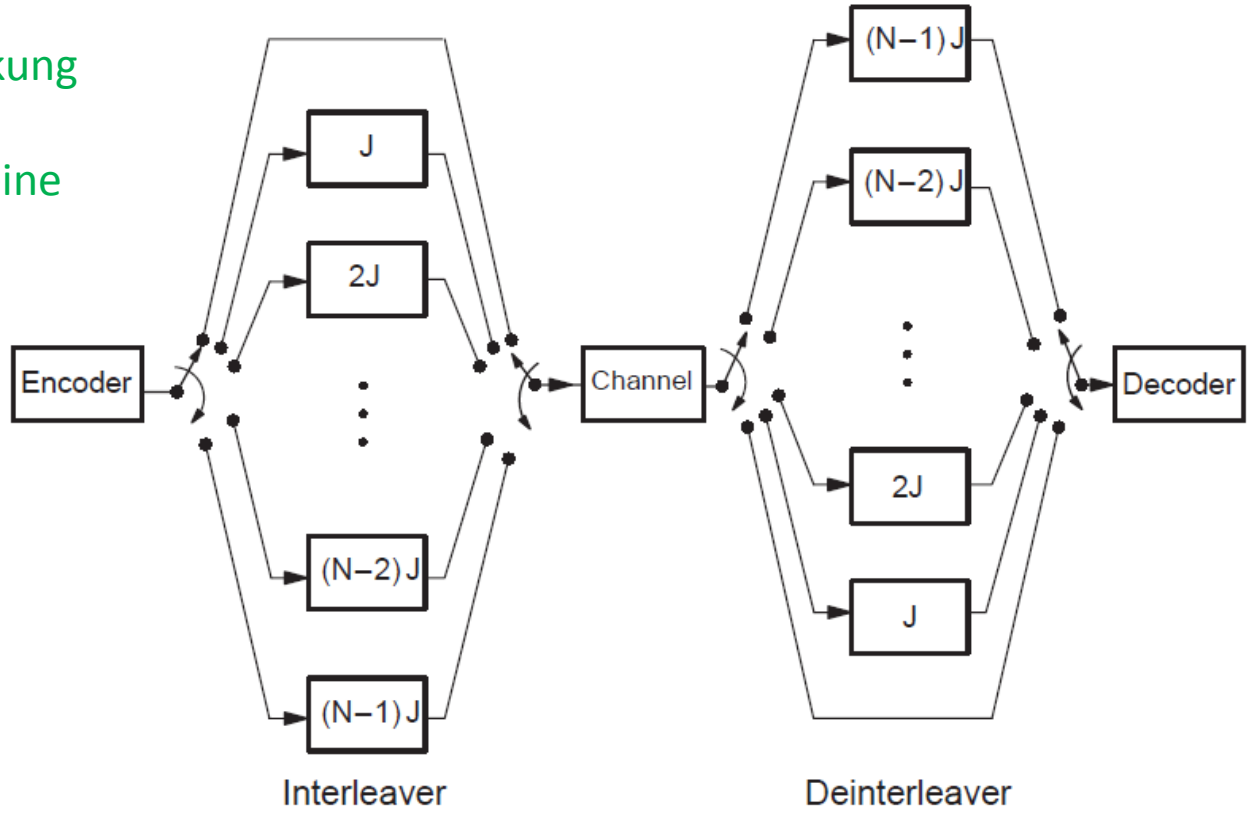


Bild 11.2. Faltungs-Interleaving

Beispiel 11.2. Es sei $N = 4, J = 1$ und somit $M = 4$. Die Symbolfolge $\ldots, -1, 0, 1, 2, 3, 4, \ldots$ führt zu folgender Belegung der Interleaver-Speicher:

nach Input 14				nach Input 18				nach Input 22			
11				15				19			
12	8			16	12			20	16		
13	9	5		17	13	9		21	17	13	
14	10	6	2	18	14	10	6	22	18	14	10

Somit entsteht die Folge ... 11,8,5,2, 15,12,9,6, 19,16,13,10 ... nach dem Interleaver. Im Deinterleaver führt das zu folgender Belegung der Speicher:

nach Input 2				nach Input 6				nach Input 10			
11	7	3	−1	15	11	7	3	19	15	11	7
8	4	0		12	8	4		16	12	8	
5	1			9	5			13	9		
2				6				10			

Somit ergibt sich die Folge ... −1,0,1,2, 3,4,5,6, 7,8,9,10 ... nach dem Deinterleaver. Die Interleaver-Operation kann auch so veranschaulicht werden:

...	23	19	15	11	...		
...	24	20	16	12	8	...	
...	25	21	17	13	9	5	...
...	26	22	18	14	10	6	2	...

Hierbei wird von oben nach unten und von rechts nach links eingeschrieben. Das Auslesen erfolgt diagonal von links-oben nach rechts-unten und von rechts nach links. ■

Ein Codierungssystem mit *verketteten Codes* (concatenated coding) zeigt Bild 9.11. Dabei werden zwei Codes in Reihe geschaltet, d.h. die von einem ersten (äußeren) Encoder erzeugte Symbolfolge wird in einem zweiten (inneren) Encoder mit zusätzlicher Redundanz versehen.

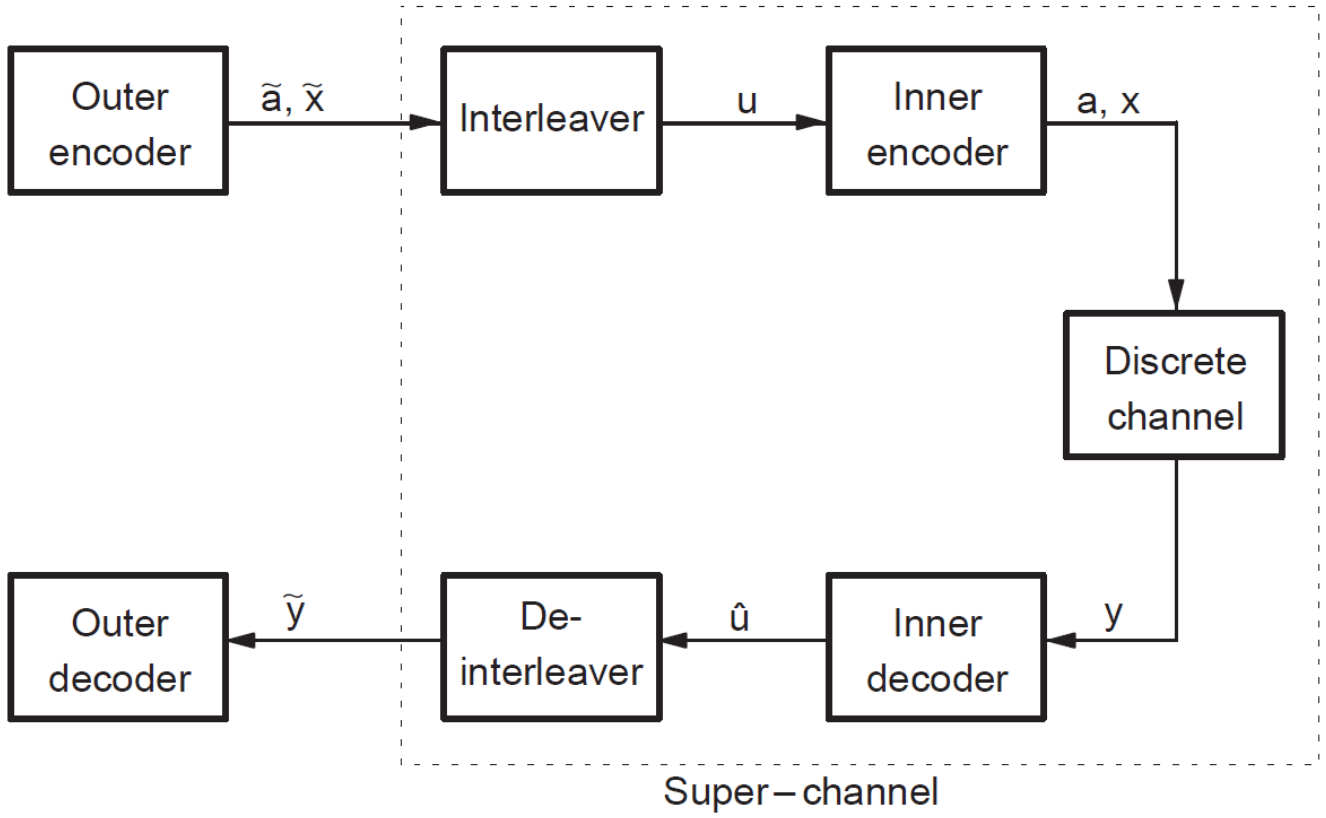
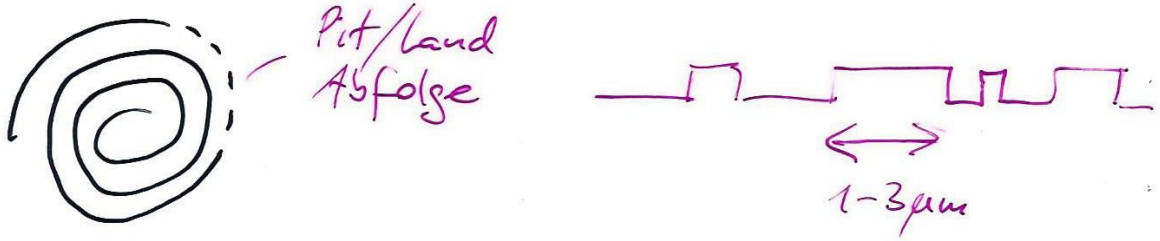


Bild 9.11. Verkettete Codierung

Ziele: einfache Herstellung (Einzelfehler)
robust gegen Beschädigungen (Bündelfehler)
einfache CD-Player (Synchronisation)



Daten: Spurlänge = 5 km
Spurbreite = 0.6 µm
Spurabstand = 1.0 µm
Geschwindigkeit = 1.2 m/s (76 min)
"Kanalbit"länge = 0.3 µm

$\Rightarrow$ $13 \cdot 10^9$ Kanalbits auf einer CD
$4.3 \cdot 10^6$ Kanalbits/s

Bündelfehler von 1 mm betrifft ≈ 3000 falsche Kanalbits
. . . . 7 . . . . 21 000 . . . .

Infobits insgesamt = 1.4 Mbit/s · 74 min = 6.2 Gbit
= 640 MByte

siehe Buch für weitere tecnische Details

Stereosignal (bis 20 kHz)

A/D ← 44.1 kHz, 16 bit

Infobits 1,4112 Mbit/s

1 Abtastwert = 32 bit

6 Abtastwerte = 192 bit $\in F_2^{192}$

= 24 · 8 bit $\in F_{256}^{24}$

CIRC Encoder R = 3/4 = $\frac{24}{28} \cdot \frac{28}{32}$

Codebits 1,8816 Mbit/s

6 Abtastwerte $\hat{=}$ 32 · 8 bit $\in F_{256}^{32}$

EFM-Modulator $R = \frac{256}{588}$ (addiert Synchronisation, vermeidet Dauerlagen)

Kanalbits 4,3218 Mbit/s

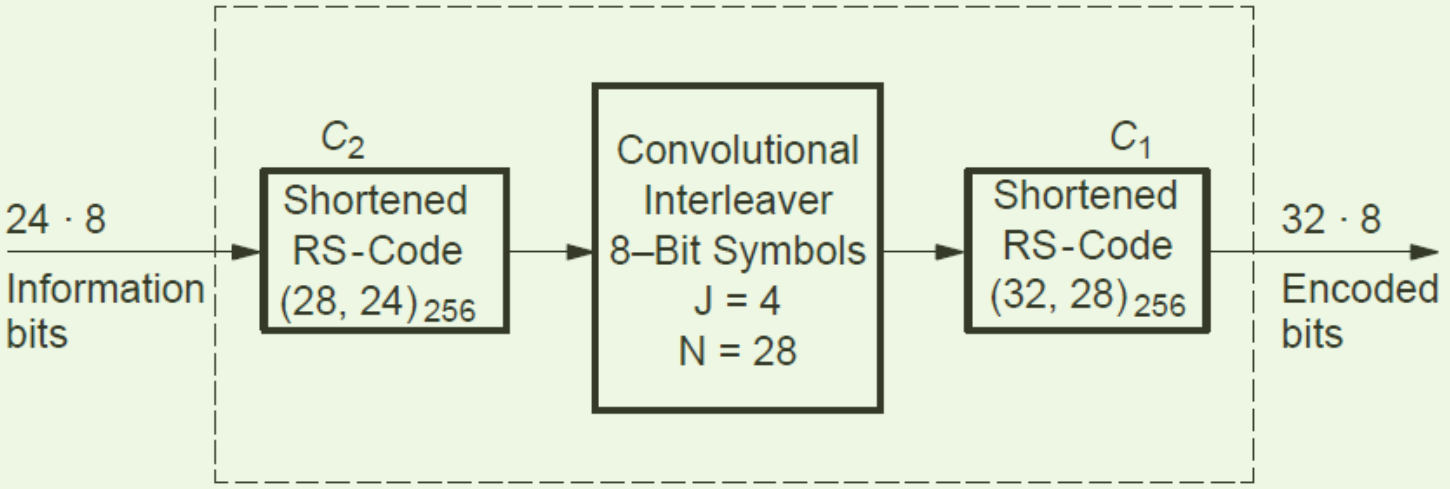
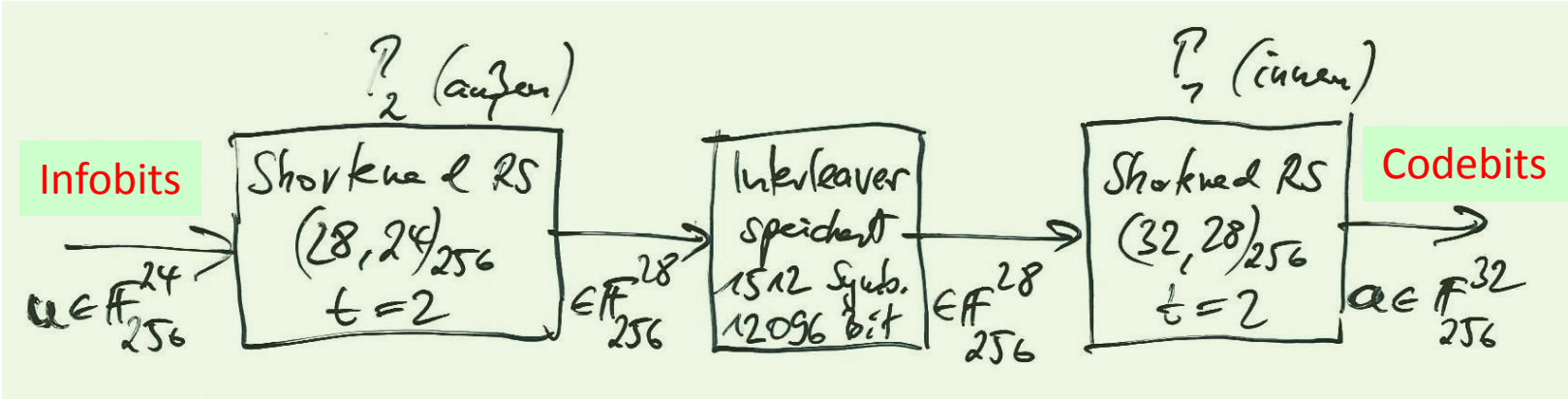


Bild 12.6. CIRC-Encoder

CIRC-Encoder (Cross-Interleaved RS-Code), R=3/4

(Pfeilrichtung = sendeseitig = Schreiben)



Korrektur von 4 Ausfällen

Nur Fehlererkennung

Korrigiert bis zu 9408 Kanalbits (≙ 3mm)
≙ 4096 Codebits (=16*32*8)
≙ 16 komplett falschen Γ_1-Codewörtern, aufeinanderfolgend

Nachweis der Korrekturfähigkeit (empfangsseitig = Lesen = entgegen der Pfeilrichtung, d.h. von rechts nach links in der Kette)

- In Γ_1 wird jedes verfälsche Wort mit hoher Wahrscheinlichkeit als falsch erkannt
- Jedes der 16 verfälschten Γ_1-Codewörter wird mit 28 Ausfällen klassifiziert, insgesamt also 16*28 =448 Symbol-Ausfälle
- Durch den Deinterleaver werden die 16*28 = 112*4 Symbole auf 112 Γ_2-Codewörter mit jeweils 4 Ausfällen verteilt
- Γ_2 korrigiert jedes der 112 Wörter korrekt.

Modifikationen

Korrektur eines Fehlers in C_1

$\Rightarrow$ Einzelfehler führen nicht zur Klassifizierung eines ganzen Wortes als ausgefallen

C_2 wird entlastet

Sicherheit der Bündelfehler-Erkennung sinkt

Größerer Interleaver

$\Rightarrow$ Noch längere Bündelfehler sind korrigierbar, setzt aber größere fehlerfreie Zone zwischen den Bündelfehlern voraus.

Standards für die Datenübertragung (hier Speicherung) enthalten grundsätzlich nur Spezifikationen für den Sender (hier CD Brennen), während der Empfänger (hier CD Lesen) durch den jeweiligen Hersteller ‚frei' dimensioniert werden kann.

In der Anfangsphase der Compact Disc wurde die „Überabtastung" beworben: das betrifft nur die Tiefpassfilterung beim Digital-Analog-Wandler, hier werden digitale Zwischenwerte berechnet so dass der analoge Wandler einfacher wird.

Faltungscodes bilden neben den Blockcodes die zweite große Klasse von Codes zur Kanalcodierung. In gewisser Weise können Faltungscodes und Blockcodes zwar als formal identisch angesehen werden, aber in der Beschreibung, in den Eigenschaften und in der Decodierung unterscheiden sich beide Codeklassen ganz erheblich. Als wesentliche Unterschiede zwischen Blockcodes und Faltungscodes sind folgende Punkte zu nennen:

- Faltungs-Encoder setzen nicht Infowörter blockweise in Codewörter um, sondern überführen eine ganze Sequenz von Infobits in eine Sequenz von Codebits, indem die Infobits mit einem Satz von Generatorkoeffizienten gefaltet werden.
- Faltungscodes werden nicht über analytische Verfahren konstruiert, sondern durch Probieren (Rechnersuche) gefunden.
- Von praktischem Interesse sind vorrangig nur ganz wenige sehr einfache Faltungscodes, deren Beschreibung und Verständnis wesentlich einfacher als bei den Blockcodes ist.
- Faltungs-Decoder können Soft-Decision-Input (Zuverlässigkeitsinformation des Demodulators) einfach verarbeiten sowie Soft-Decision-Output (Zuverlässigkeitsinformation über die geschätzten Infobits) einfach berechnen.
- Faltungscodes erfordern im Gegensatz zu den Blockcodes keine Blocksynchronisation.

Da Soft-Decision-Information ohne Mehraufwand verarbeitet werden kann und zu den in Abschnitt 1.7 dargestellten Gewinnen führt, sollten Faltungscodes immer im Zusammenhang mit Soft-Decision Demodulatoren eingesetzt werden. Dann sind Faltungscodes eigentlich nicht mehr als fehlerkorrigierende Codes zu bezeichnen, sondern besser als *Übertragungscodes*. Eine Bestimmung von Grenzen für die Korrektur oder Erkennung von Einzelfehlern oder Bündelfehlern wie bei den Blockcodes erübrigt sich dann.

Bei Faltungscodes sind die Infosymbole u und Codesymbole a nicht $q = p^m$-stufig, sondern üblicherweise binär: $u, a \in \mathbb{F}_2 = \{0, 1\}$. Alle Rechenoperationen erfolgen modulo 2. Wie beim Grundprinzip der Blockcodierung in Bild 1.7 wird der Datenstrom der Infobits bzw. Codebits unterteilt in Blöcke der Länge k bzw. n, die hier jedoch mit r indiziert werden:

k,n sind formal wie bei Blockcodes, aber de-facto viel kleiner:

$$\underline{u}_r = (u_{r,1}, \ldots, u_{r,k}) \qquad \underline{a}_r = (a_{r,1}, \ldots, a_{r,n}).$$

k=1 fast immer, k>1 nur in Ausnahmefällen

n=2,3 fast immer, n>3 nur in Ausnahmefällen

Die Zuordnung der Codeblöcke zu den Infoblöcken ist eindeutig und umkehrbar sowie zeitinvariant, aber im Gegensatz zu den Blockcodes nicht gedächtnislos:

Definition 8.1. *Ein Faltungscode der Rate $R = k/n$ wird durch einen Encoder mit Gedächtnis gegeben, indem der aktuelle Codeblock durch den aktuellen Infoblock und durch die m vorangehenden Infoblöcke bestimmt wird:*

$$\boldsymbol{a}_r = Encoder(\boldsymbol{u}_r, \boldsymbol{u}_{r-1}, \ldots, \boldsymbol{u}_{r-m}). \tag{8.1.1}$$

Der Encoder ist linear, d.h. die Codebits ergeben sich als Linearkombinationen der Infobits. Die formale Beschreibung erfolgt mit Generatorkoeffizienten $g_{\kappa,\nu,\mu} \in \mathbb{F}_2$ mit $1 \le \kappa \le k$, $1 \le \nu \le n$ und $0 \le \mu \le m$, so daß sich die Codebit-Teilfolgen als Faltungen der Infobitfolgen mit den Generatorkoeffizienten ergeben:

Faltungscodes werden primär an Beispielen erläutert, diese formale Definition ist weniger wichtig.

$$a_{r,\nu} = \sum_{\mu=0}^{m} \sum_{\kappa=1}^{k} g_{\kappa,\nu,\mu} u_{r-\mu,\kappa}. \tag{8.1.2}$$

Die Größe m wird als Gedächtnislänge und $m+1$ wird als Einflußlänge (constraint length) bezeichnet.

Der Parameter m prägt neben der Coderate sowohl die Leistungsfähigkeit des Codes wie auch den Aufwand bei der Decodierung. Die Einflußlänge wird oftmals mit k oder K bezeichnet,

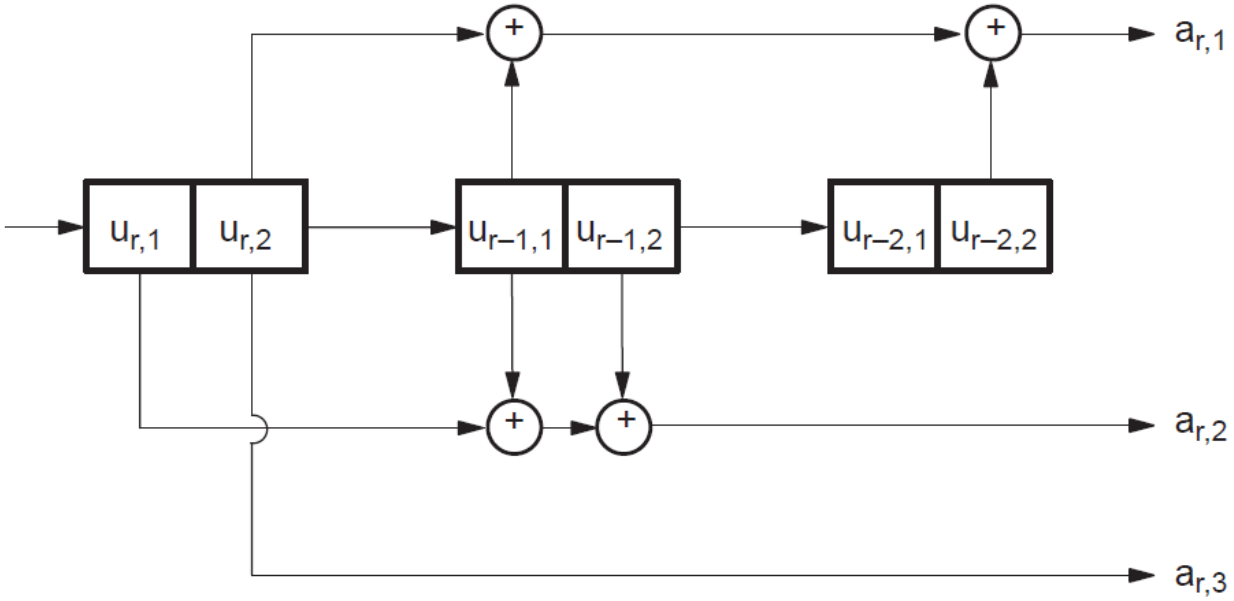


Bild 8.1. Beispiel eines $R = 2/3$-Faltungs-Encoders mit $m = 2$

Dieses ist ein eher exotisches Beispiel für den Fall k>1.

Aber im Normalfall ist k=1 und das Schieberegister wird dann einfacher als in diesem Beispiel. Siehe nächstes Beispiel zur prinzipiellen Funktionsweise.

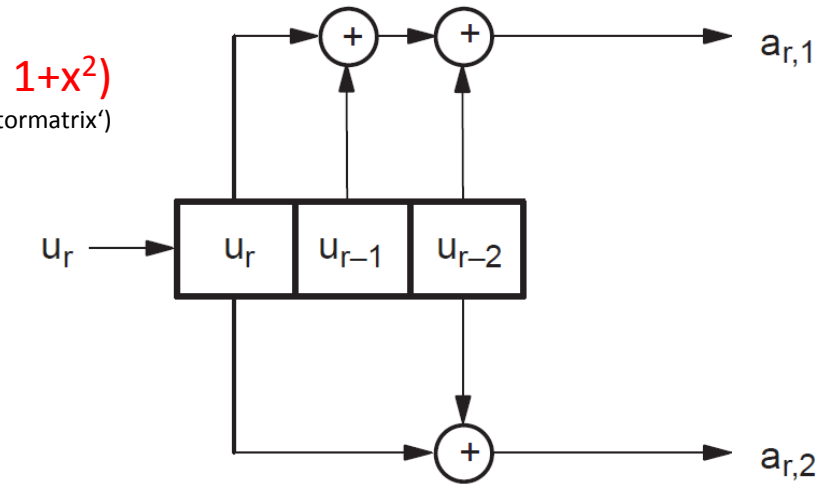
Ein R=2/3-Code wird normalerweise mit Punktierung (siehe später) aus einem R=1/2-Code erzeugt.

Beispiel 8.1. Betrachte einen $R = 2/3$-Code mit $m = 2$, wobei der Encoder durch das in Bild 8.1 dargestellte Schieberegister gegeben wird. Im r-ten Schritt werden die $k = 2$ Infobits $\boldsymbol{u}_r = (u_{r,1}, u_{r,2})$ eingeschoben und aus dem Inhalt $(\boldsymbol{u}_r, \boldsymbol{u}_{r-1}, \boldsymbol{u}_{r-2})$ des Schieberegisters wird $\boldsymbol{a}_r = (a_{r,1}, a_{r,2}, a_{r,3})$ gemäß

$$a_{r,1} = u_{r,2} + u_{r-1,1} + u_{r-2,2}$$
$$a_{r,2} = u_{r,1} + u_{r-1,1} + u_{r-1,2}$$
$$a_{r,3} = u_{r,2}$$

berechnet. Jeweils 2 Infobits werden 3 Codebits zugeordnet und somit gilt für die Coderate $R = 2/3$. ■

G(x) = (1+x+x^2, 1+x^2)
(siehe nachfolgend ‚Generatormatrix')



Anfangs ist das Register mit Nullen vorbelegt und dann wird die Folge der Infobits von links eingeschoben und rechts entstehen die Codebitpaare als Linearkombinationen der Schieberegister-Belegung.

Bild 8.2. Standardbeispiel $R = 1/2$-Faltungs-Encoder mit $m = 2$

Beispiel 8.2 (Standardbeispiel). Nachfolgend wird oftmals der $R = 1/2$-Code mit $m = 2$ und dem in Bild 8.2 dargestellten Encoder verwendet. Im r-ten Schritt wird das Infobit u_r in das Schieberegister eingeschoben und daraus wird $\boldsymbol{a}_r = (a_{r,1}, a_{r,2}) = (u_r + u_{r-1} + u_{r-2}, u_r + u_{r-2})$ berechnet. Zur Infobitfolge

$$(u_0, u_1, u_2, u_3, u_4, u_5) = (1, 1, 0, 1, 0, 0)$$

gehört die Codebitfolge

$$(a_0, a_1, a_2, a_3, a_4, a_5) = (11, 01, 01, 00, 10, 11).$$

Dieser simple Code ist durchaus von praktischer Bedeutung. Es wird sich noch zeigen, daß der asymptotische Codierungsgewinn $G_{\text{a,soft}}$ fast 4 dB beträgt. ■

Fast der gesamte Stoff zu Faltungscodes wird anhand dieses Beispiels erklärt.

Den Generatorkoeffizienten $g_{\nu,\mu}$ aus Definition 8.1 (κ entfällt bei $k = 1$) werden die *Generatorpolynome*

$$g_\nu(x) = \sum_{\mu=0}^{m} g_{\nu,\mu} x^\mu \tag{8.2.1}$$

zugeordnet. Die Infobitfolge wird durch eine Potenzreihe und die Codeblockfolge wird durch einen Potenzreihenvektor charakterisiert:

$$u(x) = \sum_{r=0}^{\infty} u_r x^r \tag{8.2.2}$$

$$\boldsymbol{a}(x) = (a_1(x), \ldots, a_n(x)) \quad , \quad a_i(x) = \sum_{r=0}^{\infty} a_{r,i} x^r . \tag{8.2.3}$$

Dem Faltungs-Encoder entspricht die Polynom-Multiplikation

$$\underbrace{(a_1(x), \ldots, a_n(x))}_{= \boldsymbol{a}(x)} = u(x) \cdot \underbrace{(g_1(x), \ldots, g_n(x))}_{= \boldsymbol{G}(x)} \tag{8.2.4}$$

und das ist äquivalent zu

$$a_\nu(x) = u(x) g_\nu(x) \quad \text{für} \quad \nu = 1, \ldots, n. \tag{8.2.5}$$

Als *Generatormatrix* wird $\boldsymbol{G}(x) = (g_1(x), \ldots, g_n(x))$ bezeichnet, obwohl das unter der Voraussetzung $R = 1/n$ nur ein Generatorvektor ist. Die Menge aller Codefolgen, also der Code, kann als

$$\mathcal{C} = \left\{ u(x)\boldsymbol{G}(x) \;\middle|\; u(x) = \sum_{r=0}^{\infty} u_r x^r, \quad u_r \in \{0,1\} \right\} \tag{8.2.6}$$

geschrieben werden. Der Faltungscode ist linear und $\mathcal{C}$ ist wie bei den Blockcodes ein Vektorraum. Für die Gedächtnislänge gilt bei gegebener Generatormatrix

$$m = \max_{1 \le \nu \le n} \text{Grad } g_\nu(x). \tag{8.2.7}$$

Beispiel 8.3. Für das Standardbeispiel gilt $\boldsymbol{G}(x) = (1 + x + x^2, 1 + x^2)$. Zur Infobitfolge $(1,1,0,1,0,0) \leftrightarrow u(x) = 1 + x + x^3$ gehört die Codebitfolge

$$\begin{aligned}
\boldsymbol{a}(x) = (a_1(x), a_2(x)) &= u(x)\boldsymbol{G}(x) \\
&= \Big((1 + x + x^3)(1 + x + x^2), (1 + x + x^3)(1 + x^2) \Big) \\
&= (1 + x^4 + x^5, 1 + x + x^2 + x^5) \\
&\leftrightarrow (11, 01, 01, 00, 10, 11).
\end{aligned}$$

Eine endliche Infobitfolge der Länge L führt zu einer endlichen Codeblockfolge der Länge $L + m$. ■

Fast alle praktisch wichtigen Faltungscode sind in dieser Tabelle enthalten.

Standardbeispiel
R=1/2, m=2

„Industriestandard-Code“
R=1/2 oder 1/3, m=6

Tabelle 8.1. Optimale Codes der Raten 1/2 und 1/3 für $m = 2, 3, 4, 5, 6$

$g_1(x)$	$g_2(x)$	$g_3(x)$	d_f
$1 + x + x^2$	$1 + x^2$		5
$1 + x + x^3$	$1 + x + x^2 + x^3$		6
$1 + x^3 + x^4$	$1 + x + x^2 + x^4$		7
$1 + x^2 + x^4 + x^5$	$1 + x + x^2 + x^3 + x^5$		8
$1 + x^2 + x^3 + x^5 + x^6$	$1 + x + x^2 + x^3 + x^6$		10
$1 + x + x^2$	$1 + x^2$	$1 + x + x^2$	8
$1 + x + x^3$	$1 + x + x^2 + x^3$	$1 + x^2 + x^3$	10
$1 + x^2 + x^4$	$1 + x + x^3 + x^4$	$1 + x + x^2 + x^3 + x^4$	12
$1 + x^2 + x^4 + x^5$	$1 + x + x^2 + x^3 + x^5$	$1 + x^3 + x^4 + x^5$	13
$1 + x^2 + x^3 + x^5 + x^6$	$1 + x + x^4 + x^6$	$1 + x + x^2 + x^3 + x^4 + x^6$	15

In Tabelle 8.1 sind für die Gedächtnislängen $m = 2, 3, 4, 5, 6$ einige optimale Faltungscodes der Raten $R = 1/2$ und $R = 1/3$ aufgelistet. Der Begriff optimal und die freie Distanz d_f werden noch in Definition 8.2 erklärt.

Nach jeweils L Infobits werden m bekannte Bits (*tail bits*) in den Datenstrom der Infobits eingefügt, wobei dazu üblicherweise Nullen gewählt werden. Diese $L+m$ Infobits beeinflussen nur die $L+m$ Codeblöcke $\boldsymbol{a}_0, \ldots, \boldsymbol{a}_{L-1+m}$ und keine weiteren Codeblöcke. Das folgt auch aus (8.2.5) und (8.2.7):

$$\text{Grad } a_\nu(x) \leq \text{Grad } u(x) + m.$$

Somit resultiert ein Blockcode, der L Infobits auf $(L+m)n$ Codebits abbildet. Also reduziert sich die Coderate von $R = 1/n$ auf

$$R_{\text{terminiert}} = \frac{L}{L+m} \cdot \frac{1}{n} \qquad \left(< \frac{1}{n} = R \right). \tag{8.3.1}$$

Für großes L ist dieser Verlust in der Rate vernachlässigbar. Als Vorteil ergibt sich eine Blockstruktur, die beispielsweise Fehlerfortpflanzungen beim Decodieren verhindert. Insbesondere bei Daten mit Rahmenstruktur werden in der Praxis fast immer terminierte Faltungscodes verwendet. Rein formal sind terminierte Faltungscodes und Blockcodes identisch.

Jeweils P Codeblöcke werden zu einer Gruppe zusammengefaßt. Von den nP Codebits in dieser Gruppe werden l Codebits gestrichen (punktiert) und nicht übertragen. Somit werden P Infobits auf $nP-l$ Codebits abgebildet. Also erhöht sich die Coderate von $R = 1/n$ auf

$$R_{\text{punktiert}} = \frac{P}{nP - l} \qquad \left(> \frac{1}{n} = R \right). \tag{8.3.2}$$

Natürlich muß $l < (n-1)P$ sein, um eine Coderate kleiner als 1 zu gewährleisten, damit die Infobits aus den Codebits wiedergewonnen werden können.

Die Größe P wird als *Punktierungslänge* und der $R = 1/n$-Code wird als *Muttercode* bezeichnet. Die zu streichenden Codebits werden in einem *Punktierungsschema* festgelegt. Mit wachsendem P können nahezu alle Coderaten zwischen $1/n$ und 1 erreicht werden. Beispiele für $R = 1/2$ zeigt Tabelle 8.2. Speziell für $l = P - 1$ gilt $R_{\text{punktiert}} = P/(P + 1)$.

Tabelle 8.2. Beispiele für $R_{\text{punktiert}}$ bei $R = 1/2$

	$l=0$	$l=1$	$l=2$	$l=3$	$l=4$	$l=5$	$l=6$	$l=7$
$P=2$	2/4	2/3						
$P=3$	3/6	3/5	3/4					
$P=4$	4/8	4/7	4/6	4/5				
$P=8$	8/16	8/15	8/14	8/13	8/12	8/11	8/10	8/9

Von großer praktischer Bedeutung sind die von J.Hagenauer [106] eingeführten *RCPC-Codes* (Rate Compatible Punctured Convolutional Codes). Hierbei wird eine Codefamilie mit unterschiedlichen Raten aus einem einzigen Muttercode derart abgeleitet, daß die hochratigen Codes in die niederratigen Codes eingebettet werden. Codes höherer Rate entstehen dabei nur durch zusätzliche Punktierungen aus Codes niedrigerer Rate, d.h. alle bei höheren Raten benutzten Bits werden auch bei niedrigeren Raten benutzt.

Der besondere Vorteil der RCPC-Codes liegt darin, daß zu beliebigen Zeitpunkten die Coderate umgeschaltet werden kann, ohne daß sich im Bereich der Umschaltzeitpunkte negative Auswirkungen auf die Distanzeigenschaften des Codes ergeben.

Zwei Hauptanwendungen für RCPC-Codes

- Übertragungskanäle mit schwankender Güte, bei denen die Coderate über ein ARQ-Verfahren laufend an die Kanaleigenschaften angepaßt wird. Im normalen Fall bei gutem Kanal wird nur der punktierte Code benutzt und erst bei abnehmender Kanalqualität werden die punktierten Stellen nachträglich gesendet, um die Korrekturfähigkeit zu erhöhen.

- Datenquellen, bei denen die einzelnen Bits eine unterschiedliche Wichtigkeit haben und folglich einen abgestuften Fehlerschutz erfordern. Das wird auch als *UEP-Codierung* (Unequal Error Protection) bezeichnet. Eine interessante Anwendnung stellt die Quellencodierung von Sprache bei digitalen Mobilfunksystemen dar

Ein Beispiel für RCPC-Codes zeigt Tabelle 8.3 [75, 106], bei denen aus einem $R = 1/4$-Muttercode mit

$$G(x) = (\underbrace{1 + x^3 + x^4}_{g_1(x)}, \underbrace{1 + x + x^2 + x^4}_{g_2(x)}, \underbrace{1 + x^2 + x^3 + x^4}_{g_3(x)}, \underbrace{1 + x + x^3 + x^4}_{g_4(x)})$$

ratenkompatible punktierte Codes mit Raten von 8/9 bis 1/4 abgeleitet werden. Eine 0 im Punktierungsschema steht dabei für Punktierung. Wichtige Anwen-

Tabelle 8.3. RCPC-Codes zum $R = 1/4$-Muttercode mit $m = 4$ und $P = 8$

Punktierungsschema	$R_{\text{punktiert}}$
1111 0111 1000 1000 0000 0000 0000 0000	8/9
1111 1111 1000 1000 0000 0000 0000 0000	8/10
1111 1111 1010 1010 0000 0000 0000 0000	8/12
1111 1111 1110 1110 0000 0000 0000 0000	8/14
1111 1111 1111 1111 0000 0000 0000 0000	8/16
1111 1111 1111 1111 1000 1000 0000 0000	8/18
1111 1111 1111 1111 1100 1100 0000 0000	8/20
1111 1111 1111 1111 1110 1110 0000 0000	8/22
1111 1111 1111 1111 1111 1111 0000 0000	8/24
1111 1111 1111 1111 1111 1111 1000 1000	8/26
1111 1111 1111 1111 1111 1111 1010 1010	8/28
1111 1111 1111 1111 1111 1111 1110 1110	8/30
1111 1111 1111 1111 1111 1111 1111 1111	8/32
$g_1(x)$	$g_2(x)$ $g_3(x)$ $g_4(x)$

0 bedeutet Punktierung
1 bedeutet Übertragung

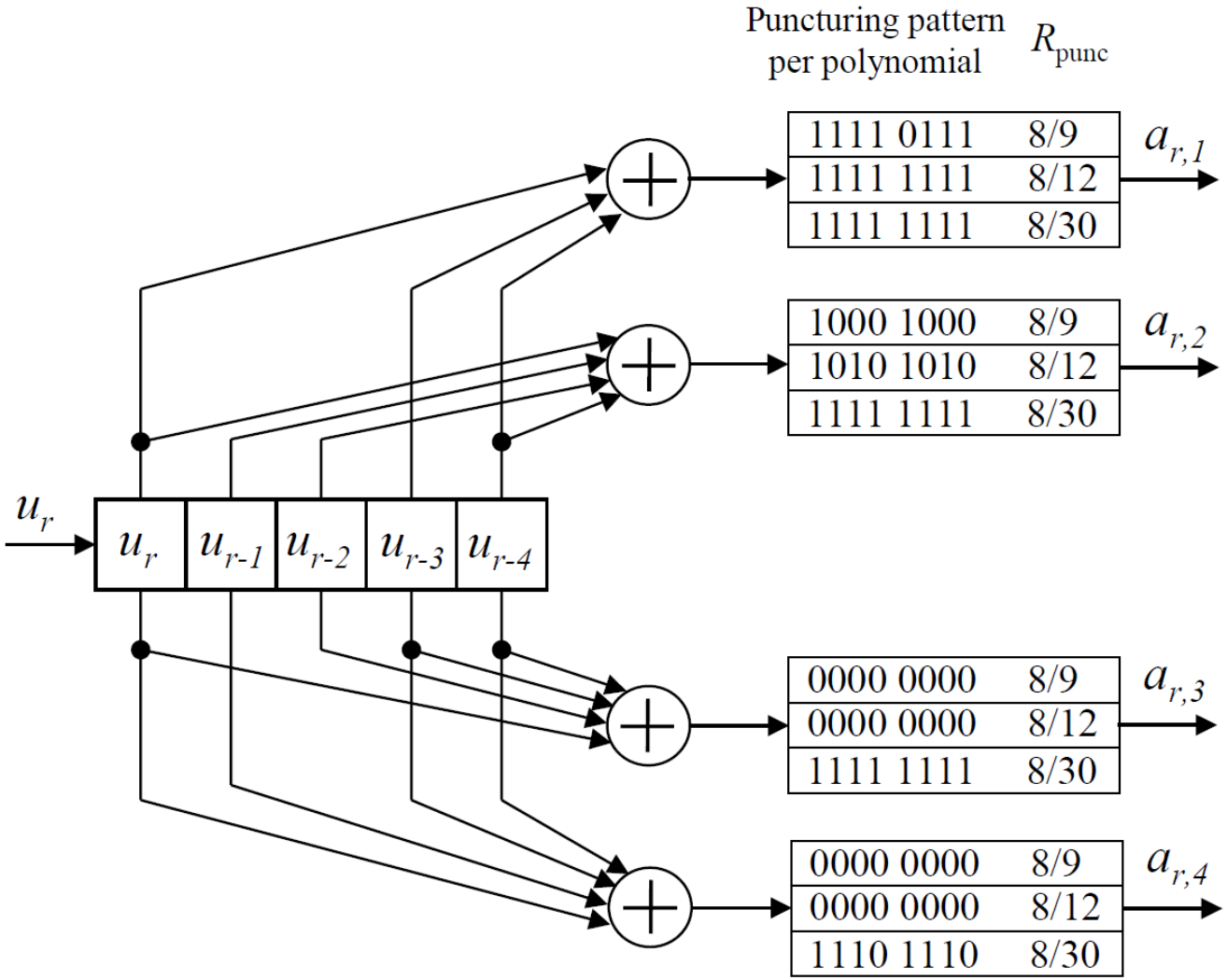


Figure 9.3. RCPC encoder with three exemplary code rates (rate-1/4 mother code with $m = 4$ and $P = 8$)

Darstellung des R=1/4-Encoders und der Punktierungsschemata für die punktierten Raten 8/9, 8/12, 8/30

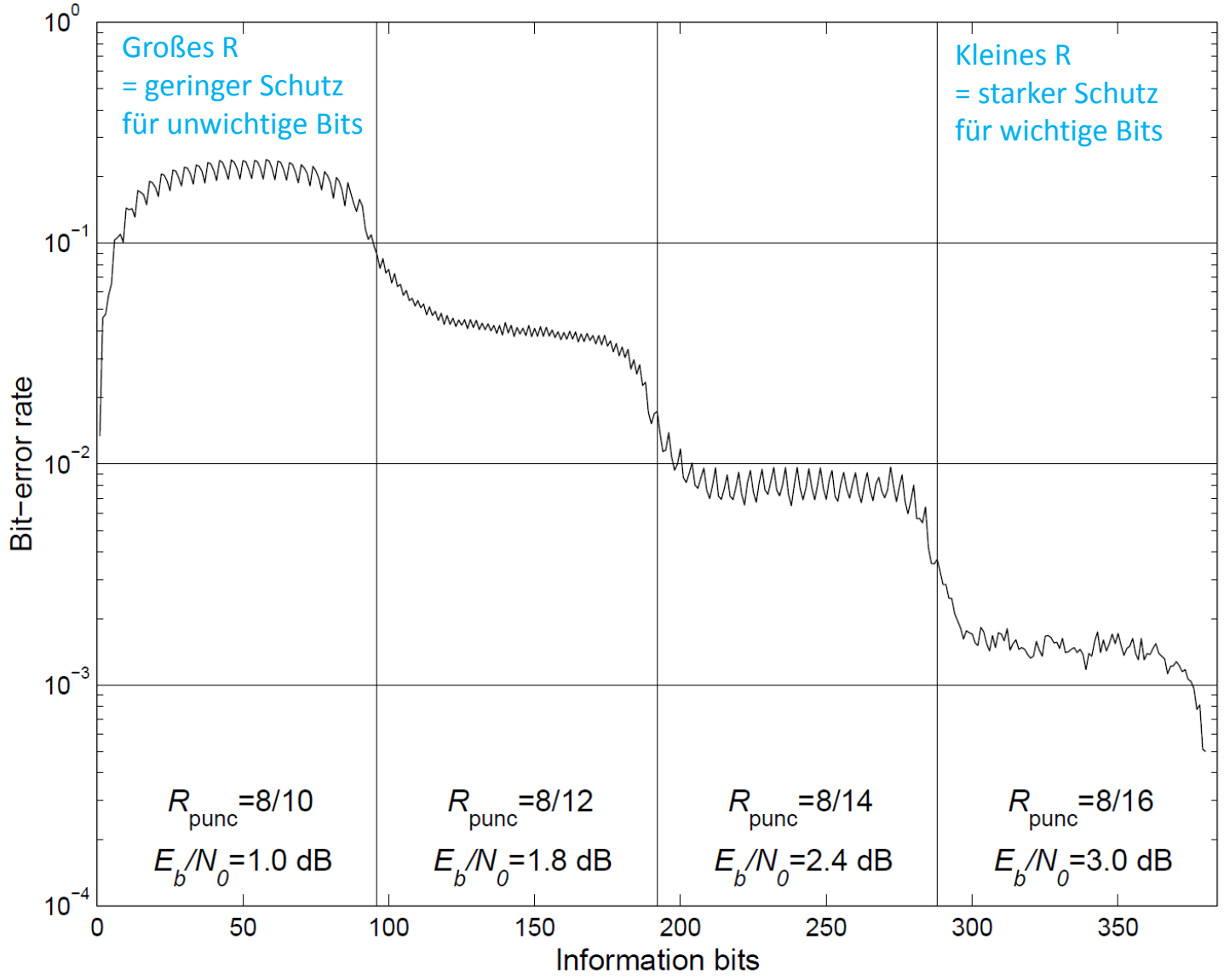


Figure 9.4. Performance of an RCPC oder with four code rates (at $E_c/N_0 = 0$ dB)

UEP-Anwendung:

Das RCPC-Prinzip garantiert dass es zu keinem Anstieg der Fehlerrate an den Umschaltpunkten der Coderate kommt (sondern einen gleichmäßigen Übergang).